Università degli Studi di Roma "Tor Vergata"

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

DOTTORATO DI RICERCA
BIOLOGIA EVOLUZIONISTICA ED ECOLOGIA
CICLO XXI

Modelli di previsione dei popolamenti ittici nei fiumi: sviluppo e ottimizzazione mediante reti neurali artificiali

Tesi di Dottorato

Marco Quartararo

A.A. 2009/2010

Docente Guida: Prof. Michele Scardi

Coordinatore: Prof. Patrizia B. Albertano

Università degli Studi di Roma "Tor Vergata"

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

DOTTORATO DI RICERCA
BIOLOGIA EVOLUZIONISTICA ED ECOLOGIA
CICLO XXI

Modelli di previsione dei popolamenti ittici nei fiumi: sviluppo e ottimizzazione mediante reti neurali artificiali

Tesi di Dottorato

Marco Quartararo

A.A. 2009/2010

Docente Guida: Prof. Michele Scardi

Coordinatore: Prof. Patrizia B. Albertano

In copertina
M.C. Escher, *Fish and Scales (1959)*

“A te
che cammini”

RINGRAZIAMENTI

Grazie alla mia famiglia e a Greta, che mi hanno supportato e sopportato con amore e fiducia in questi anni.

Grazie al prof. Michele Scardi, che con disponibilità e pazienza mi ha guidato, consigliato e insegnato tante cose.

Grazie a tutti coloro (amici, colleghi, datori di lavoro) che in forme diverse hanno contribuito a rendere più leggero il mio cammino.

INDICE

ABSTRACT

Title: "Predictive models of fish assemblages in rivers: development and optimization by means of artificial neural networks"

Key words: predictive models, fish fauna, river ecology, artificial neural networks

The use of artificial intelligence methods for ecosystems modeling has had a considerable development in the last 20 years (Fielding, 1999; Lek, 2005; Olden et al., 2008). The work herein presented meets this research field, dealing particularly with river fish fauna prediction by artificial neural networks.

Artificial neural networks are empiric mathematical models aiming at global relations among ecosystem variables and based on general mathematical structures whose parameters need suit some given data. They typically receive one or more input variables (predictors, describing in this case abiotic environment: riverbed width, site elevation, water pH, catchment area surface, etc.) and yield the expected values of one or more output variables (presence/absence values of the species within the fish assemblage).

The relation between the ecosystems abiotic component and the structure of the connected communities is the best indicator of the ecosystems status/quality, as it represents the richest and most integrated expression of ecological processes: the ability of a physical system to support life is a direct and temporally consistent measure of the ecosystem health, more than any physical, chemical or biochemical indicator. This point of view is in agreement also with the most recent EC laws that, concerning particularly water matters, specify the adoption of biological communities as ecosystems quality indicators. Models capable of reconstructing the relations between the abiotic and biological components are then valuable work tools in ecological research and applications, supplying means for defining the characteristics and ecological status (*sensu* Water Framework Directive 2000/60/CE) of a site, estimating the impact of diverse strategies on the biological component, exploring the linking net of the variables involved, giving a handlable representation of the systems studied, etc. This is particularly true for the river ecosystems, today the centre of pressures potentially endangering their precious and delicate balance.

Both traditional methods (based, for example, on fish zoning or the concept of *continuum*) and the modern ones, based on indexes or multivariate statistic models, partially fail in giving reliable predictive models for river ecosystems, mostly due to simplifications or assumptions not always compatible with the real complexity of such systems and their deriving data (Fielding, 1999; Lek et al., 2005); many artificial intelligence or *machine learning* methods, instead, supported by suitable "learning" algorithms, are often able to build from the data effective representations of ecological systems (ivi).

The use of fish assemblage as representative elements of the biological component of ecosystems is justified by a series of considerations (Lek et al., 2005; Tancioni et al., 2005) linked to their biological activities (stay, shelter, exploration, feeding, reproduction) by which they strongly interact with both the biological and physical environment. Besides, fish are the most long-living and mobile organisms in a watercourse, therefore the information from the structure of their assemblage are spatially and temporally consistent.

The main purpose of the present work is an experimental examination of four hypotheses about as many potential strategies for model optimization (plus an additional hypothesis, ancillary with respect to the first one).

The basic model employed is a multilayer *perceptron*, i.e. a layered (input, internal and output layer) feed-forward neural network fully connected, with processing units (neurons) working as linear combiners with sigmoid activation function. The input layer consists of 20 neurons receiving data of as many environmental variables; each neuron of the output layer returns instead the prevision about presence or absence of one of the 32 species of the assemblage considered, on a continuous scale ranging from zero to one; this continuous value can be read as a probability of presence, but it can also be made binary by a threshold function (i.e., "absence" when the output is beneath the fixed threshold and "presence" when above); the traditional threshold is 0.5. The training was conducted by an EBP (Error Back Propagation) algorithm with *early stopping* and the performance assessment was based on a test set independent from the training data (obtained through a *split-sample* approach), so that the data were to be subdivided into three subsets (training, validation and test set) as much homogeneous as possible.

Following are the studied hypotheses:

1) regarding the threshold utilized to make the continuous output of the network binary, it is possible to determine an optimal threshold which improves (or, at most, leaves it unchanged) the performance of a model, compared with the base performance obtained with a traditional fixed threshold (0.5);
2) the optimal threshold value is linked to the species frequency within the data set through a non-decreasing function (this hypothesis is ancillary with respect to the first one);
3) with most species, the performance of the models predicting all species of the assemblage (multispecies models) is better than the one of the models predicting single species (monospecies models); with some species, however, the contrary is true;
4) models trained by a data subdivision equally distributing the presence occurrences among the subsets (training, validation and test) give a better performance than those with random distribution;
5) models trained by a data subdivision homogeneously distributing the cases relative to the elevation among the three subsets generally give a better performance than those with random distribution (the elevation, in fluvial systems, integrates a set of many covariant variables).

In order to simultaneously investigate the five hypotheses, an articulated experimental design has been arranged in order to train lots of models in the scheduled conditions; the software for the trials set up, the conduction of the experiments, the visualization of the raw outcome and the comparison of the results through statistical tests was specifically devised. A series of original algorithms and functions was developed to make data subdivisions, to find optimal thresholds, to descriptively synthesize results' distributions and to yield a measure of performance independent from the threshold.

Since the main target was the development of a method (with related software) to experimentally test the hypotheses, it was decided not to include in the research project an original activity of data collection, but to use an already available and well known data set (Scardi et al., 2005). The sampling sites the data come from are relative to the mountain and

piedmont ranges of the watercourses in the provinces of Vicenza and Belluno. The observation of the species within the various sites was conducted by using *electrofishing* techniques, supported with nets in cases of sites characterized by wide riverbeds.

Cohen's K statistics (1960), which compares observations and predictions by a *confusion matrix*, was adopted to evaluate models' performances.

The statistical comparisons were made through an original subroutine performing the Wilcoxon-Mann-Whitney test, also applying the necessary Bonferroni correction for multiple tests.

The total experimental results were 323,136.

Hypothesis 1 is fully confirmed, showing that the performance of models with an optimized threshold is systematically and remarkably greater than in traditional models. It's an important result, from both a practical point of view (the method is effective and economic and can also be applied to already trained models) and a methodological one, as it uncovers a portion of models' predictive ability that tends to remain latent with a rigid traditional threshold.

Hypothesis 2 confirms the theoretical reasons that led to the formulation of Hypothesis 1: the more a species is rare, the higher is the bias distorting the previsions (bias that it's possible to chase by searching for an optimal threshold).

Hypothesis 3 is well confirmed in its formulation. Although determining lower performance improvements than the optimized threshold strategy, the adoption of a multispecies or a monospecies model can be critical for the prevision of some species. As expected, most of species particularly benefit of a multispecies prevision, which allows the models to use also the additional information on species associations; some species, however, prefer a single prevision, probably because strongly linked to environmental descriptors and ambiguously associated with other species. This hypothesis proposes a theme of particular interest from an ecological point of view and it's worth further investigating.

Altogether, the strategies linked to Hypotheses 4 and 5, although generally useful, gave performance improvements less intense and significant than the ones observed for Hypotheses 1 and 3. However, the outcome encourages further studies.

In their whole, the results prove the practical and theoretical interest in working with predictive models, for both the opportunity of obtaining more effective models and the possibility of triggering a virtuous circle, where the reflections on the ecological reality produce modeling solutions that, in turn, propose new questions and ideas about the ecological reality itself. Beyond the results *per se*, the work done has further produced many ideas to better or to go thoroughly into some aspects or, still, to formulate new hypotheses.

The development of such an experimental work certainly implies a considerable time investment, even by only considering the necessity of writing specific software, but in the end a method and an informatic tool remain that, with the right adjustments, can test other optimization strategies and produce ecologically interesting results with any data set.

Some of the ideas herein illustrated are going to be developed in the near future, based on a broader data set (400 observations) relative to Central Italy watercourses.

1. INTRODUZIONE

L'uso di metodi di intelligenza artificiale per la modellizzazione degli ecosistemi ha avuto un considerevole sviluppo negli ultimi 20 anni, sollevando questioni teoriche, metodologiche, tecniche e applicative che sono state oggetto di vari studi (si vedano ad esempio le review proposte in Fielding, 1999; Lek, 2005; Olden et al., 2008). Il lavoro qui presentato si inserisce in quest'ambito di ricerca, occupandosi in particolare della previsione della fauna ittica nei fiumi tramite reti neurali artificiali (ANN[1]).

Le cinque ipotesi che si è scelto di studiare hanno favorito il confronto con un intreccio di tematiche tecniche (legate all'uso delle reti neurali artificiali come modelli di previsione), metodologiche (riguardanti il confronto tra misure sperimentali della performance di modelli alternativi), teoriche (legate all'ecologia dei fiumi e delle specie considerate) e applicative (connesse con gli usi potenziali dei modelli generati), che mostrano come la costruzione di modelli in ecologia sia un'attività utile e ricca di interesse.

Nei paragrafi che seguono, prima di illustrare le ipotesi oggetto di studio, si è ritenuto utile:

- introdurre il concetto generale di "modello";
- illustrare il ruolo dei modelli predittivi nella ricerca e nelle applicazioni in ecologia;
- fare uno zoom sul caso particolare degli ecosistemi fluviali e della valutazione della loro qualità ecologica tramite la fauna ittica;
- introdurre i metodi di intelligenza artificiale e, in particolare, le reti neurali;
- spiegare perché tali metodi possano essere più adeguati di altri, in alcune circostanze, per la costruzione di modelli di ecosistemi;
- far luce sul lavoro necessario per generare modelli di previsione accurati e attendibili tramite reti neurali artificiali.

[1] Da "Artificial Neural Network".

1.1 Inquadramento generale

1.1.1 Modelli predittivi in ecologia

Intesi in senso generale, i *modelli* sono rappresentazioni, cioè ricostruzioni di oggetti, sistemi, fenomeni o proprietà che ci permettono di visualizzarne, manipolarne, prevederne o imitarne alcune caratteristiche e comportamenti. Il concetto di modello è molto ampio: sono modelli le ricostruzioni in scala degli aerei o delle automobili, le carte geografiche, le sagome usate nelle realizzazioni artigianali, i racconti mitici che "spiegano" fatti naturali o umani, le regole culturalmente condivise, le equazioni matematiche che esprimono relazioni quantitative tra variabili, i diagrammi di flusso che descrivono procedure, le formule di struttura in chimica, gli spartiti musicali che rappresentano frequenze e durate di un insieme di suoni.

Nella scienza i modelli sono innanzitutto strumenti di conoscenza, perché permettono di formalizzare attraverso un codice che può essere grafico, linguistico, matematico, ecc. ciò che si è compreso della realtà; sono anche strumenti di previsione, perché permettono di anticipare il comportamento dei sistemi studiati sulla base di ciò che sappiamo su di essi; e sono infine strumenti di riflessione euristica, in quanto l'osservazione delle caratteristiche o del comportamento dei modelli può suggerire idee nuove sulle realtà che rappresentano, ispirando il processo di formulazione delle ipotesi. Non tutti i tipi di modello posseggono allo stesso modo queste tre caratteristiche, ma tutti le posseggono in qualche misura; pertanto, quando si parla di "modelli predittivi" o "modelli di previsione", non si fa riferimento ad un tipo speciale di modello, ma semplicemente a modelli che rispetto ad altri si prestano in modo particolare a dire "cosa succede se…", quale valore assumeranno alcune variabili del sistema rappresentato (variabili predette o di output) date certe condizioni di partenza (predittori o variabili di input).

I modelli possono essere molto diversi gli uni dagli altri e avere quindi ruoli diversi nella ricerca e nell'applicazione. Alcuni modelli sono più descrittivi mentre altri sono più esplicativi (si pensi alle leggi di Keplero e a quelle di Newton sul moto degli astri); alcuni si esprimono in linguaggio matematico mentre altri hanno una natura più concettuale (espressi verbalmente o tramite diagrammi) e altri ancora sono basati su relazioni geometriche (es. modelli di molecole); alcuni semplificano questioni complicate (come un grafo ad albero che rappresenta i rapporti evolutivi

tra le specie appartenenti al medesimo ordine), altri invece complicano questioni apparentemente semplici (come in meccanica quantistica); alcuni sono ben definiti, utili e dimostrabili (come quelli che descrivono il ruolo del DNA nella cellula), mentre altri possono mantenersi per anni a livello di ipotesi (come la teoria delle stringhe in fisica); alcuni sono eleganti e "trasparenti" nel loro modo di rappresentare e trattare gli oggetti studiati, ma magari si rivelano poco applicabili come strumenti di lavoro (come le equazioni differenziali di Lotka e Volterra), mentre altri sono molto più "opachi" (per questo si parla di *black box*) riguardo al loro modo di funzionare, ma possono rivelarsi molto utili in concreto (come le reti neurali artificiali).

Non ha molto senso chiedersi se ci siano in assoluto modelli migliori di altri, perché ognuno dialoga con particolari aspetti della capacità umana di comprendere e percepire (ad esempio, cogliere relazioni logiche tra eventi formalizzati, piuttosto che relazioni spaziali tra eventi rappresentati graficamente), costituendo un punto di vista potenzialmente utile sugli oggetti di studio. Piuttosto, quando si sceglie di sviluppare un certo tipo di modello è necessario essere consapevoli di cosa si vuol cogliere della realtà degli oggetti di studio, in modo da scegliere il canale migliore per veicolare specifici aspetti, e bisogna aver chiara la funzione che il modello sviluppato dovrà svolgere nel nostro lavoro (ad esempio, visualizzazione nitida dell'ingranaggio di un certo sistema o previsione affidabile dei suoi mutamenti di stato).

I modelli utilizzati in questo lavoro, le reti neurali artificiali, sono di tipo matematico.

Uno dei motivi per cui la formalizzazione matematica ha avuto così grande successo nel mondo della scienza è che i modelli matematici riescono spesso, rispetto ad altri, a fare bene molte cose insieme: rappresentare in modo chiaro ed elegante i rapporti tra grandezze, dar luogo a strumenti di lavoro oggettivi (cioè, condivisi e applicabili nello stesso modo da soggetti diversi), fornire previsioni precise e mostrare cosa può accadere al di là di ciò che è già stato osservato. Tra il XVII secolo e la fine del XIX la possibilità di costruire modelli matematici della realtà fu considerato un aspetto talmente caratterizzante dell'attività scientifica, da conferire una diversa dignità alle scienze *hard*, che erano in grado di farlo, rispetto a quelle *soft*, che non lo erano. Poi le cose si complicarono intorno ai primi del Novecento, quando una serie di terremoti culturali investirono il pensiero matematico (da Russell a Gödel) e quello fisico (da Planck ad Heisemberg), dando

avvio ad un percorso che rimescolerà le carte del fare scienza e valorizzerà l'uso di una pluralità di approcci e di metodi, sia in ragione della specificità degli oggetti di studio, che dell'opportunità di osservare le cose da più punti di vista[2]. Tuttavia, pur giustamente ridimensionato, l'approccio matematico trovò presto una nuova base epistemologica e fu in grado di esprimere rinnovate potenzialità applicative con lo sviluppo e la diffusione, tra gli anni '40 e '60, dei calcolatori elettronici.

L'ecologia nasce già, rispetto ad altre discipline biologiche, con una vocazione all'approccio matematico. Già alla fine del Settecento gli studi di Malthus sulla dinamica delle popolazioni sviluppavano con un linguaggio matematico temi che saranno poi propri dell'ecologia alcuni decenni più tardi, influenzando tra l'altro le idee di Darwin, precursore illustre del pensiero ecologico. Negli anni '20 del secolo scorso la formalizzazione matematica delle dinamiche di popolazione di prede e predatori, compiuta indipendentemente e quasi contemporaneamente da Lotka e Volterra, prelude all'evoluzione in senso quantitativo che di lì a qualche decennio l'ecologia avrebbe compiuto, anche grazie all'affermarsi dello studio dei processi ecologici in termini di flussi energetici. Intorno alla metà del Novecento, poi, nascono la teoria dei sistemi, la cibernetica, la teoria dell'informazione e tutta una serie di altre discipline il cui principale strumento di modellizzazione è proprio il formalismo matematico e che saranno destinate ad influenzare fortemente ogni ramo della scienza o della tecnologia legato a sistemi di qualche tipo.

I modelli matematici che vengono usati oggi in ecologia sono essenzialmente riconducibili a due categorie: *modelli analitici* e *modelli empirici* (Levins, 1966). I primi scompongono i processi ecosistemici in sottoprocessi e li traducono in equazioni che rendono conto del funzionamento dei vari "ingranaggi" alla base delle dinamiche ecosistemiche. I secondi cercano di cogliere in modo più complessivo le relazioni tra le variabili in gioco negli ecosistemi e si basano su modelli matematici generali nei quali volta per volta è necessario regolare in modo empirico alcuni parametri in base ai dati. Un esempio di modello analitico è il sistema di equazioni differenziali con cui Lotka e Volterra hanno modellizzato le dinamiche di popolazione di prede e predatori. Esempio di modello empirico è invece un General Linear Model (GLM): un'impalcatura teorica generale che postula l'esistenza di relazioni lineari tra n_i variabili di input e n_o variabili di output e richiede che una

[2] Il che permise alle discipline biologiche di affermare un proprio modo di fare scienza non meno valido di quello delle scienze *hard*.

serie di parametri vengano regolati in funzione dei dati, per poter definire la specifica natura delle diverse relazioni lineari e conferire al modello capacità predittive (i valori assunti dalle variabili di output possono essere calcolati in funzione di quelli delle variabili di input) ed esplicative (è possibile studiare i parametri determinati per comprendere meglio le relazioni tra le variabili in gioco).

Le reti neurali artificiali sono modelli matematici di tipo empirico; dunque può essere utile spendere qualche parola in più su di essi. Quando si sviluppa un modello matematico empirico è necessario passare attraverso tre fasi successive, che spesso vengono reiterate finché il modello non rappresenti in modo soddisfacente i fenomeni rappresentati:

1) *formulazione* del modello – in questa fase, in base agli obiettivi, si scelgono le variabili da considerare[3]; si traducono le relazioni rilevanti in una o più funzioni matematiche interconnesse che diano luogo ad una rappresentazione astratta e generalizzata dei processi ecosistemici da studiare; si definiscono le scale spazio-temporali a cui operare[4]; si scelgono i parametri che nella fase successiva andranno regolati per consentire al modello di cogliere le peculiarità del sistema modellizzato. In molti casi si preferisce non affrontare *ex novo* questa fase, ma usare modelli generali già predisposti, come ad esempio quello della regressione lineare semplice per la modellizzazione di una relazione causale lineare tra due variabili; in altri casi, soprattutto quelli in cui le relazioni tra le variabili interessate non sono facilmente riconducibili a relazioni di tipo semplice (si pensi ai complessi modelli di previsione climatica o della produzione primaria), è invece necessario trovare un modo originale ed efficace per combinare in modo funzionale le variabili in gioco;
2) *calibrazione* – è la fase di regolazione dei parametri, cioè di quelle componenti del modello che permettono di "personalizzare" il modello stesso in funzione delle caratteristiche peculiari del sistema rappresentato e dei dati da esso provenienti; nel caso della regressione lineare semplice l'intercetta *a* e il coefficiente angolare *b* della retta sono i parametri che, una volta definiti in funzione dei dati, permettono al modello lineare di specificarsi;

[3] Non tutte le variabili coinvolte nel funzionamento del sistema sono necessariamente significative per gli aspetti che si intende modellizzare.

[4] Questo spesso influisce non solo sulla struttura del modello, ma anche sulla scelta dei dati da usare per calibrarlo.

3) *validazione* – è la fase di controllo, in cui il modello viene fatto "funzionare" con dati che spesso sono diversi e indipendenti da quelli usati per la calibrazione, saggiandone così le capacità predittive e di generalizzazione; se il modello ha raggiunto un optimum, il processo può interrompersi, altrimenti si torna a calibrare e il processo viene reiterato fino a quando la performance non migliora più; nel caso del modello lineare, la retta parametrizzata durante la calibrazione può essere validata con coppie di valori (x,y) noti: confrontando i valori previsionali della y con quelli osservati, si misura la capacità del modello di predire correttamente la variabile dipendente in base alla variabile indipendente[5].

Indipendentemente dal tipo di modello utilizzato, i modelli predittivi hanno spesso la caratteristica di ricevere una o più variabili in ingresso (predittori) e di fornire i valori previsionali di una o più variabili in uscita. Se le variabili in uscita sono le medesime di quelle in ingresso, allora il modello può simulare il loro modificarsi nel tempo a causa delle interazioni tra di esse nel sistema e restituire il valore che assumeranno dopo un certo intervallo[6]. Se invece le variabili di output sono diverse da quelle di input, allora il modello potrà dirci qual è il valore assunto da queste ultime (variabili predette) in funzione del valore che hanno le variabili ricevute in ingresso (predittori).

Questo è un tipo di compito per cui le reti neurali artificiali sono particolarmente adatte e che può contribuire a risolvere problemi come quello di conoscere la componente biotica di un ecosistema in funzione di quella abiotica. Qui le variabili di input sono i descrittori dell'ambiente abiotico (nel caso degli ecosistemi fluviali, larghezza dell'alveo, altitudine del sito, pH dell'acqua, superficie del bacino versante, ecc.), mentre quelle di output potrebbero essere la biomassa, l'abbondanza, la presenza di singole specie o coorti d'età, la loro distribuzione lungo l'asta fluviale, la diversità espressa dall'intera comunità, ecc. Nella letteratura anglosassone questo approccio è spesso definito "habitat-association" (Fielding & Bell, 1997).

[5] A scopo esemplificativo si è proposto qui un metodo di regolazione dei parametri della retta di regressione che è utilizzabile in linea di principio, ma è diverso da quello canonico, basato sul calcolo della varianza e della covarianza.

[6] E' questo che si fa, ad esempio, in dinamica dei sistemi (System Dynamics), un approccio simile a quello che Lotka e Volterra adottarono per la costruzione del loro sistema di equazioni differenziali.

La relazione tra la componente abiotica dei sistemi ecologici e la struttura delle comunità che ne fanno parte è il miglior indicatore dello status/qualità degli ecosistemi, poiché costituisce l'espressione più ricca e integrata dei processi ecologici sottostanti alle dinamiche ecosistemiche: la capacità del sistema fisico di supportare la vita è una misura diretta e temporalmente consistente della salute dell'ecosistema, più di qualsiasi indicatore di tipo fisico, chimico o biochimico (Lek et al., 2005). Questa visione concorda peraltro con la più recente legislazione comunitaria, che in particolare in materia di acque[7] prevede l'uso delle comunità biologiche come indicatori della qualità degli ecosistemi.

I modelli che sono in grado di ricostruire i rapporti tra componente abiotica e componente biologica costituiscono dunque dei preziosi strumenti di lavoro nella ricerca e nelle applicazioni ecologiche.

Sul versante applicativo, un modello di previsione di questo tipo può:

1) caratterizzare un sito sulla base di una tipologia predefinita, identificandone la comunità potenziale in funzione dei descrittori ambientali;
2) definire lo "stato ecologico" (es. *sensu* Direttiva 2000/60/CE[8]) di un sito attraverso il confronto tra la componente biotica attesa (fornita dal modello calibrato su siti imperturbati) e quella riscontrata tramite campionamento;
3) stimare l'impatto sulla componente biologica di possibili interventi sulla componente ambientale o di azioni mirate al ripristino degli ecosistemi, attraverso la simulazione dei cambiamenti attesi nella componente biologica (supporto all'elaborazione di strategie tramite la costruzione di scenari potenziali).

Sul versante teorico, i modelli di previsione costituiscono, già nella fase di formulazione, dei potenti strumenti di riflessione sul funzionamento degli ecosistemi, sull'intreccio tra variabili al loro interno e sui processi che ne sostengono gli equilibri. Ma in modo più specifico essi permettono anche di:

1) esplorare la rete di relazioni tra variabili, identificando cosa influisce maggiormente, e come, su particolari variabili di interesse[9];

[7] Direttiva Quadro sulle Acque, 2000/60/CE del Parlamento Europeo e del Consiglio, del 23 ottobre 2000, che istituisce un quadro per l'azione comunitaria in materia di acque ed è nota anche con il suo acronimo inglese, WFD ("Water Framework Directive").

[8] La Direttiva prevede il raggiungimento dello status "Buono" per tutte le acque superficiali e sotterranee degli Stati aderenti entro il 2015.

[9] Uno degli strumenti più usati per fare questo è l'*analisi di sensibilità*, che variando uno per uno i valori dei predittori rileva il cambiamento atteso nelle diverse variabili predette e misura quanto ognuna

2) disporre di una rappresentazione manipolabile dei sistemi studiati, con la quale è possibile condurre "esperimenti simulati" in cui si osservano i comportamenti del sistema allorché alcune variabili vengono modificate in un certo modo; questa possibilità di "giocare" con i modelli dei sistemi ha una grande potenzialità euristica, perché permette di riprodurre condizioni mai osservate direttamente e di mettere in risalto meccanismi non ancora teorizzati, guidando così il processo di generazione di nuove ipotesi[10].

1.1.2 Ecosistemi fluviali, modelli di previsione e fauna ittica

La costruzione di modelli di previsione per gli ecosistemi fluviali è oggi un'attività di notevole interesse teorico e applicativo, sia per le ragioni descritte in generale nel paragrafo precedente, applicabili a tutti gli ecosistemi, che per alcune peculiarità legate in particolare a questi sistemi ecologici.

Innanzitutto, la crescita della popolazione umana e il conseguente incremento delle attività produttive e di uso del territorio determinano oggi nel mondo un continuo aumento sia del fabbisogno di acqua dolce, che della pressione antropica sui bacini idrogeologici[11] (Lek et al., 2005). Quest'ultima si esprime in forme estremamente varie, dall'artificializzazione degli alvei, alla costruzione di barriere necessarie per la produzione idroelettrica, al prelievo di acqua e materiali di vario genere, alla deforestazione, alla pressione edilizia o agricola sui versanti, alle attività estrattive, allo scarico di acque reflue da uso privato o industriale, alla pesca sportiva

di queste ultime dipenda da ognuno dei predittori. Si veda Olden et al. (2008), p. 181, per una veloce rassegna dei vari metodi utilizzati a questo scopo.

[10] Questo è particolarmente importante nelle scienze come l'ecologia o l'astronomia che non possono basarsi come altre su un vero e proprio approccio sperimentale.

[11] Il bacino idrogeologico è la porzione di ambiente fisico che ospita l'ecosistema fluviale, il quale non si limita alla zona inondata dalla massa d'acqua corrente, ma coinvolge l'intera superficie di dilavamento delle acque che naturalmente convergono verso l'alveo. Se si considera solo lo scorrimento superficiale delle acque, si parla di "bacino idrografico", mentre il concetto di bacino idrogeologico è più ampio e tiene conto della superficie impermeabile su cui può verificarsi uno scorrimento anche sotterraneo delle acque. Nel caso del bacino idrografico è possibile determinare in modo abbastanza preciso un "limite spartiacque", detto anche displuvio, che costituisce il margine entro cui le precipitazioni meteoriche restano confinate nel bacino stesso e dilavano verso il corso d'acqua. Nella maggior parte dei casi, comunque, il concetto di bacino idrografico è sufficiente a rappresentare in via generale il contesto fisico dell'ecosistema fluviale.

e commerciale, ecc[12]. Dunque è di primaria importanza comprendere e monitorare i vari aspetti del funzionamento di questi ecosistemi, non solo perché essi rappresentano un valore naturale in sé, ma anche per poter continuare ad interagire produttivamente con essi senza comprometterli. I fiumi costituiscono infatti un comparto fondamentale per il ciclo dell'acqua, interagiscono in modo profondo con il territorio, modellandolo e influenzandone le caratteristiche climatiche, e posseggono potenti sistemi di autodepurazione che li rendono fonti preziose di acqua dolce per tutti gli organismi. Per quanto molti studi nel secolo scorso abbiano contribuito ad identificare i processi ecosistemici operanti nelle acque dolci, l'approccio modellistico di tipo *top-down*[13] da essi proposto non ha condotto ai risultati attesi, cioè a modelli deterministici che in base ai determinanti ambientali e al disturbo antropico forniscono previsioni del comparto biologico ad un livello tassonomico o funzionale rilevante (ivi). La ricerca in questo campo resta dunque aperta e l'approccio *bottom-up*[14], legato a modelli come le reti neurali, è ancora tutto da esplorare.

La costruzione di modelli di previsione per gli ecosistemi fluviali è inoltre esplicitamente raccomandata dalla Direttiva Quadro sulle Acque (ivi) per completare il set di strumenti per la valutazione dello stato ecologico dei fiumi, attualmente rappresentati quasi esclusivamente da indici come l'IBE (Indice Biotico Esteso) e l'IFF (Indice di Funzionalità Fluviale). Questi hanno buone basi teoriche, praticità di utilizzo e una discreta capacità di integrare informazioni su variabili sia biotiche che abiotiche; non sempre però riescono a tenere conto in modo adeguato delle condizioni di riferimento specifiche di un particolare corso d'acqua o di una specifica tipologia fluviale (*sensu* Direttiva Acque), tendono a semplificare drasticamente il sistema rappresentato e non permettono simulazioni per proiettare nel futuro le conseguenze di potenziali strategie gestionali, né si prestano ad analisi di tipo euristico. Lo sviluppo di modelli di previsione può dunque essere affiancato agli indici per completarne le potenzialità.

[12] Alcune informazioni sono state tratte da documenti prodotti nell'ambito delle attività dell'Autorità di Bacino del fiume Magra (http://www.adbmagra.it/).

[13] La struttura del modello esprime le conoscenze teoriche circa il funzionamento del sistema rappresentato. In *System Dynamics*, ad esempio, si costruiscono modelli di questo tipo.

[14] La struttura del modello esprime le conoscenze empiriche, ricavate dai dati, circa il funzionamento del sistema rappresentato.

Infine i sistemi ecologici fluviali, nonostante la loro complessità di ecosistemi aperti, presentano una struttura particolare che ne favorisce la modellizzazione: con una ipersemplificazione si può dire che essi hanno una sagoma nastriforme, con i due capi estremi posti ad altitudini diverse e una sezione trasversale concava verso l'alto. Questo determina l'elemento più caratterizzante del sistema, lo scorrere unidirezionale della corrente d'acqua, e fa sì che molte variabili fondamentali di tipo geo-morfologico, chimico-fisico e trofico (Gelosi & Colombari, 2004), nonché biologico (comunità), varino in modo associato man mano che ci si sposta da monte a valle. Ad esempio, con l'allontanarsi dalla sorgente diminuiscono l'altitudine, la pendenza, la velocità della corrente, la turbolenza, la concentrazione di ossigeno, il trasporto solido, la granulometria media del sedimento, l'ombreggiatura dovuta alla vegetazione riparia, mentre aumentano la temperatura, la torbidità, l'ampiezza dell'alveo, la portata, il trasporto di materiale in sospensione, la concentrazione di sali e nutrienti, l'ordine dell'asta fluviale[15]; altre variabili seguono andamenti diversi ma comunque legati alla dimensione longitudinale del sistema, come il rapporto produzione/respirazione, che dalla foce al mare determina una U rovesciata. La polarità monte-valle costituisce una sorta di organizzatore attorno al quale molte variabili covariano in modo relativamente ordinato ed identificabile, determinandosi reciprocamente. Se un modello è in grado di integrare non solo la complessa rete di relazioni tra le diverse variabili, ma anche l'informazione implicita riguardo al loro variare congiunto da monte a valle, la dimensionalità (cioè il numero di fattori indipendenti da considerare) del problema predittivo risulterà notevolmente ridotta e con essa la sua complessità.

Su questa sorta di "informazione portante" che aggrega un gran numero di fattori, il modello potrà poi rappresentare le modulazioni prodotte da altre variabili, come i tipi di roccia che caratterizzano il bacino idrografico, la sinuosità e variabilità morfologica dell'alveo, la durezza dell'acqua, il pH, il genere e la quantità di copertura vegetale sui versanti, il disturbo antropico, ecc. Questi descrittori hanno a

[15] La complessità dei bacini idrografici, spesso articolati in sottobacini, dà luogo ad un reticolo fluviale costituito da "aste" che si strutturano in forme dendritiche. Queste sono caratterizzate da alcuni rami privi di affluenti (detti aste di primo ordine) e da altri generati dalla confluenza di due o più rami a monte, che saranno di ordine immediatamente superiore a quello dei rami che in essi confluiscono. Così, ad esempio, la confluenza di due rami del primo ordine dà luogo ad un ramo del secondo ordine e così via. L'ordine massimo viene raggiunto dall'asta fluviale che si getta direttamente in mare (il Po, ad esempio, è di settimo ordine). L'ordine di un tratto fluviale può fornire indirettamente molte informazioni sulle sue caratteristiche ecologiche, mentre l'articolazione dei rami di un reticolo idrografico nei vari ordini che li caratterizzano contribuisce a defnire la complessità del sistema idrografico stesso.

che fare con le altre tre dimensioni del sistema fluviale (Fig. 1)[16], che fino ad ora abbiamo considerato solo implicitamente; il fiume infatti non è caratterizzato solo da uno sviluppo longitudinale.

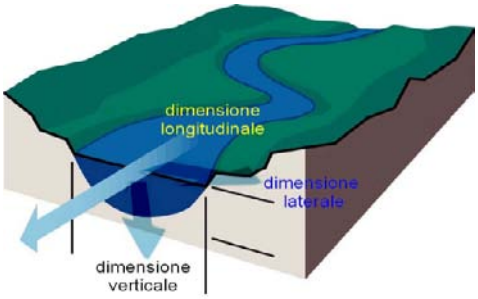


Fig. 1

Attraverso la dimensione trasversale (o laterale) l'acqua corrente entra in contatto con l'ambiente circostante, che viene bagnato e modellato dal fiume e riversa in esso detrito, nutrienti, particolato di vario genere, ostacoli solidi (tronchi, rocce), inquinamento e artefatti di origine antropica (Fig. 2)[17].

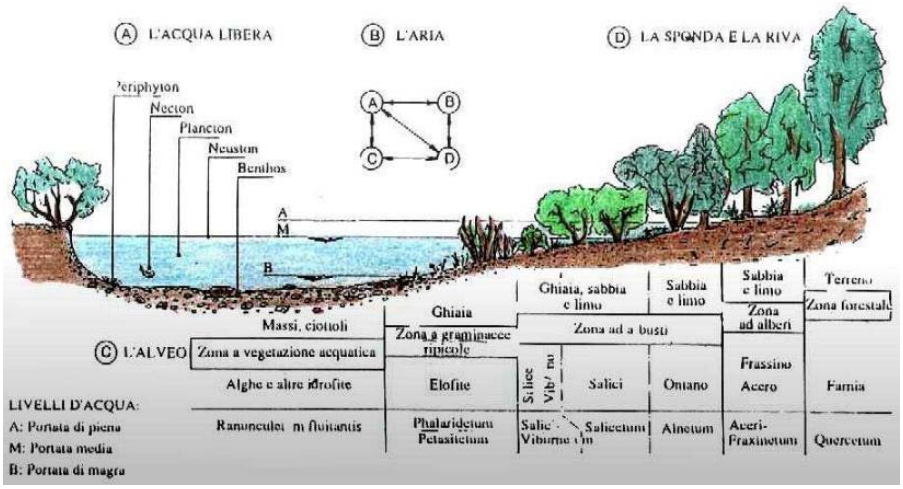


Fig. 2

[16] http://www.provincia.terni.it/caccia/laboratorio/didattica/fiumi.pdf
[17] http://www.provincia.ps.it/itis.mattei/chimica/tdistanza/images1/Piante1.jpg

Le interazioni tra la dimensione longitudinale e trasversale (alveo curvo) determinano inoltre variazioni nei pattern di dinamismo nella corrente.

Poi c'è la dimensione verticale, che a livello della superficie mette in contatto il fiume con l'atmosfera permettendo, ad esempio, l'ossigenazione delle acque; a contatto con l'alveo si osservano invece dinamiche di interramento ed emersione dell'acqua, nonché tutta una serie di fenomeni legati all'attrito, al trasporto solido e alle variazioni di profondità, che hanno importanti conseguenze sullo strutturarsi della corrente e del paesaggio.

Infine è necessario ricordare la dimensione temporale, che con periodo circannuale dà luogo a variazioni cicliche della portata, della temperatura, ecc.[18], ma che a medio e lungo termine permette lo sviluppo di una profonda azione di modellamento del territorio.

Sulla comunità biologica la direttrice monte-valle esercita un'azione strutturante che ad un livello macroscopico induce il configurarsi di tipologie associate al corso alto, medio o basso dei fiumi. Questo ha dato luogo, ad esempio, a modelli teorici che identificano una "zonazione ecologica" nei corsi d'acqua. Illies e Botosaneanu (1963) fanno corrispondere ad una serie di caratteristiche ambientali tipiche del corso superiore dei fiumi (detto Crenal), del corso medio (Rhithral) e del corso inferiore (Potamal) le relative comunità macrobentoniche (dette, rispettivamente, Crenon, Rhithron e Potamon). Ancora prima, Thieneman e poi Huet (1949) avevano proposto una zonazione basata sui popolamenti ittici (Stoch, 2002), che nella sua versione finale riconosce quattro zone, ognuna caratterizzata da una o più specie "guida" e da diverse "specie di corteggio" (Fig. 3)[19]:

1) zona a trota, con acque fredde, turbolente e ben ossigenate, substrato eterogeneo (roccia, massi, ciottoli, ghiaia, ecc.) e assenza di vegetazione acquatica; in questa zona la trota fario (*Salmo trutta trutta*) è tipicamente associata con lo scazzone (*Cottus gobio*), ma possono essere presenti anche ciprinidi reofili (bene adattati alla corrente) come la sanguinerola (*Phoxinus phoxinus*), il vairone (*Leuciscus souffia*) e il barbo canino (*Barbus meridionalis*);

[18] Non è difficile immaginare come l'ampiezza delle oscillazioni di queste variabili possa costituire un vincolo potente per i sistemi ecologici presenti in un certo contesto fluviale. Vi sono inoltre eventi particolarmente intensi e imprevedibili, come piene o secche di particolare entità, che possono condizionare notevolmente i sistemi ecologici.

[19] http://www.cridea.it/pesci/sistemiacquatici/zonazione.html

2) zona a temolo (detta anche "zona inferiore della trota"), caratterizzata ancora da acque fredde e ben ossigenate, ma da una minore velocità della corrente, substrato di ghiaia omogenea, alghe e poche macrofite; il temolo (*Thymallus thymallus*) è qui spesso associato alla trota marmorata (*Salmo trutta marmoratus*) e a ciprinidi reofili;
3) zona a barbo, in cui la corrente è ancora più lenta e la temperatura più elevata, ma l'ossigenazione dell'acqua ancora buona; la granulometria è più fine (sabbia e ghiaia) e la vegetazione si fa più consistente; insieme al barbo (*Barbus plebejus*) troviamo la lasca (*Chondrostoma genei*), il cavedano (*Leuciscus cephalus*), il gobione (*Gobio gobio*) e altri ciprinidi;
4) zona ad abramide[20], con acque lente, substrato fangoso e copertura macrofitica importante; qui si trovano, oltre all'abramide (*Abramis brama*), ciprinidi limnofili (o a deposizione fitofila) come la carpa (*Cyprinus carpio*), la scardola (*Scardinius erythrophtalmus*), il triotto (*Rutilus erythrophtalmus*), l'alborella (*Alburnus alburnus alborella*), la tinca (*Tinca tinca*), il carassio (*Carassius auratus*) e altre specie come il luccio (*Esox lucius*).

[20] In Italia l'abramide non è autoctono, anche se presente; la specie guida di questa zona diventa quindi la carpa, anch'essa non autoctona, ma introdotta oramai da molti secoli, probabilmente in epoca romana.

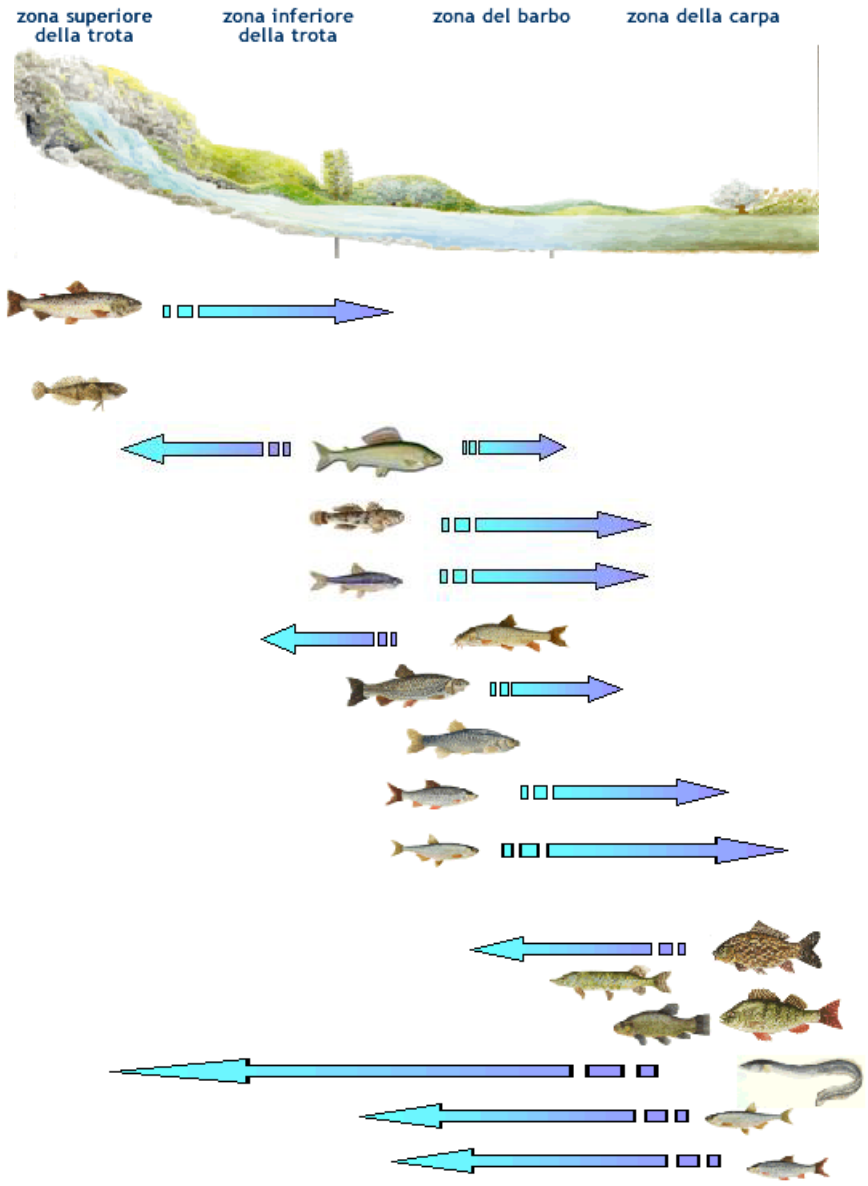


Fig. 3

Le prime due zone possono essere ricondotte ad un'unica "zona a salmonidi", caratteristica dei torrenti di montagna, mentre le ultime due identificano una "zona a ciprinidi", tipica delle zone pedemontane, collinari e planiziarie. Ad esse si può

aggiungere una “zona dei pesci eurialini” o “zona della foce”, caratteristica del tratto finale dei fiumi e in cui sono presenti specie che soprattutto per motivi trofici si spostano tra il mare e il fiume (cefali, spigole, passere, ecc.).

L’approccio zonale discretizza l’ecosistema fluviale riconoscendo al suo interno tipologie significative e utili. Tuttavia ignora gli aspetti di continuità presenti nell’andamento di molte variabili che si modificano da monte a valle e non tiene conto di tipologie zonali intermedie e di passaggio, né del fatto che molte specie presentano preferenze specifiche non del tutto sovrapponibili e distribuzioni che travalicano la zona tipica di appartenenza (si pensi ai ciprinidi reofili o ad *Anguilla anguilla*, che è possibile trovare tanto nelle zone a ciprinidi reofili, quanto in quelle a ciprinidi limnofili). Inoltre non rappresenta la connessione funzionale tra zone a monte e a valle e non è facilmente estendibile a contesti (come quello dei corsi d’acqua italiani) diversi da quelli per cui è stato elaborato (fiumi del centro Europa).

Una risposta teorica a questo approccio discreto è il “River Continuum Concept”, un modello che rappresenta l’ecosistema fluviale come una successione di ecosistemi determinata da una modificazione graduale dei fattori morfologici, idrodinamici, fisico-chimici, trofici e delle comunità ad essi associate. Elaborato da Verneaux, Vannote e Cushing tra la fine degli anni ’70 e l’inizio del decennio successivo, questo modello esprime una visione funzionale nelle variazioni della comunità da monte a valle (Fig. 4)[21].

[21] http://www.forestencyclopedia.net/p/p1495/i/i1176/quick_image_view

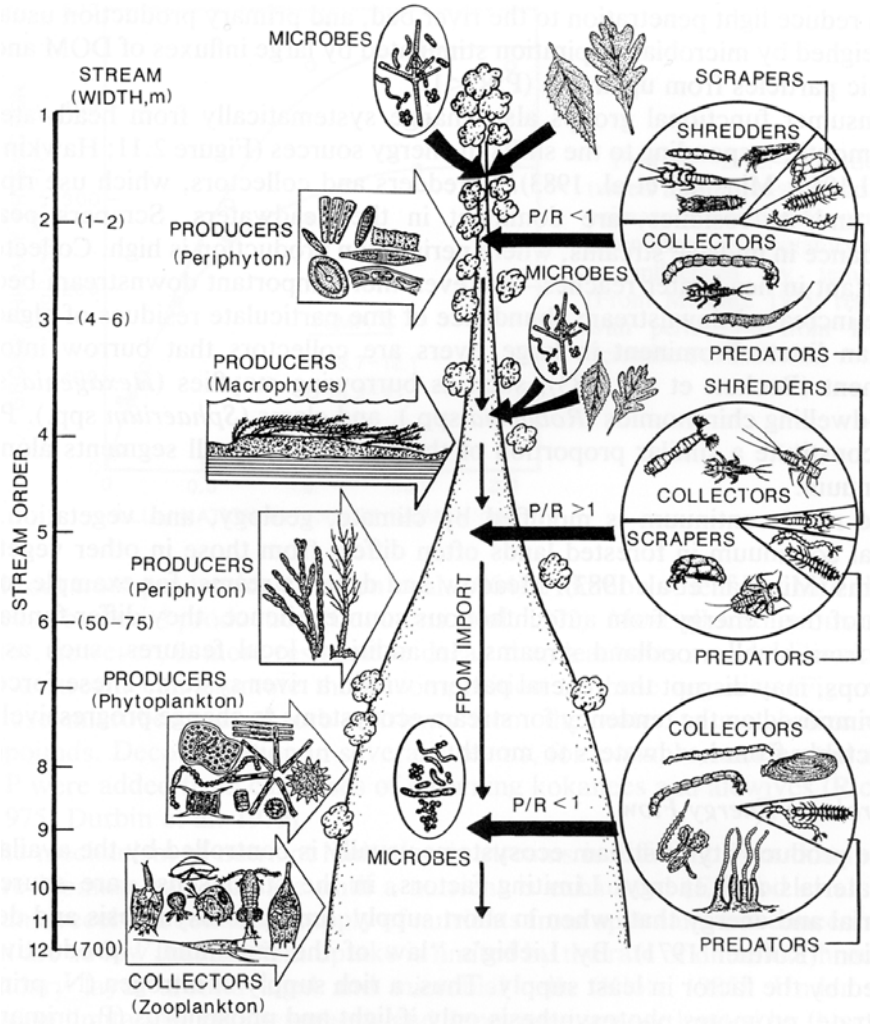


Fig. 4

Ad esempio, essendo i tratti alti dei fiumi caratterizzati da acque povere di nutrienti e da un'ampia copertura vegetale circostante, essi godono di discreti apporti di detrito da terra e di una consistente ombreggiatura; questo favorisce l'instaurarsi di un metabolismo di tipo eterotrofico, con una grande presenza di organismi raccoglitori e frammentatori e una limitata presenza di produttori autotrofi

(che necessiterebbero di luce solare e nutrienti) e pascolatori[22]. Scendendo verso valle queste caratteristiche cambiano gradualmente, con una riduzione dell'ombreggiatura dovuta alla vegetazione e l'apporto da monte di materiale organico relativamente ricco e fine; questo induce una progressiva riduzione dei frammentatori e un aumento di produttori primari e pascolatori, passando ad un metabolismo autotrofico. Avvicinandosi ancora alla foce, l'aumento del particolato organico (prodotto dalle attività dell'ecosistema a monte e trasportato dalla corrente a valle) incrementa la torbidità riducendo nuovamente il rapporto produzione/respirazione e favorisce una preponderante presenza di organismi raccoglitori (metabolismo nuovamente di tipo eterotrofico).

Il concetto di *continuum* è uno strumento teorico utile per la comprensione globale del funzionamento ecosistemico fluviale, ma la sua capacità di superare l'artificiosità della discretizzazione zonale è basata sull'introduzione di un nuovo elemento artificioso: quello di una gradualità che negli ecosistemi naturali si osserva solo parzialmente. Entrambi i modelli permettono di fare previsioni di massima sulla comunità attesa in un certo sito, ma nessuno di essi rende conto in modo adeguato di quel mosaico che è la struttura ecologica di questi sistemi, in cui macrozone o variazioni continue significative ad una certa scala (quella delle variazioni longitudinali di ampia portata), si fratturano in molteplici microambienti (*patch*) a scale di maggior dettaglio. In un unico sito può osservarsi un alternarsi di pozze (*pools*) e rapide (*riffles*), di aree di fondale con massi grossolani e altre con ghiaia fine o sabbia, di zone ampiamente vegetate e di altre spoglie, tratti rettilinei e anse pronunciate, sponde in erosione e sponde in accumulo. Una buona variabilità ambientale a questo livello di scala, ad esempio, è importante per i pesci, che necessitano, anche come singole specie, di un'ampia tipologia di microambienti per trovare rifugio, nutrirsi, accoppiarsi, deporre le uova, ecc. Se un modello non tiene conto della variabilità ambientale a questo livello di scala, rischia dunque di essere grossolano e poco utilizzabile praticamente, anche se a livelli di minor dettaglio può costituire un buon riferimento di massima a livello sia teorico che pratico.

A questi modelli tradizionali vengono oggi affiancati metodi matematici basati soprattutto su modelli di statistica multivariata, che permettono di superare le categorie di zona e di continuum e di tenere conto contemporaneamente di molte

[22] A parte i produttori autotrofi, costituiti da microrganismi bentonici, alghe e macrofite acquatiche, le categorie considerate dagli Autori sono riferite particolarmente alla fauna invertebrata.

variabili ambientali e biologiche a varie scale; lo scopo è trovare classificazioni o ordinamenti efficaci per i siti[23] studiati o approssimare relazioni funzionali tra variabili ad un livello appropriato di dettaglio. I metodi di classificazione e ordinamento non generano modelli predittivi, ma piuttosto descrittivi delle strutture presenti nei dati; sono usati soprattutto per "esplorare" le informazioni raccolte, cioè per far emergere somiglianze e differenze tra i siti campionati, che ci aiutino a caratterizzarli e a comprenderne le variazioni.

Il metodo di classificazione più rappresentativo è l'analisi dei cluster, che in base ai valori assunti dalle diverse variabili considerate assegna ogni sito ad una "classe", ad un "tipo", in modo da garantire la massima somiglianza (omogeneità) tra siti appartenenti allo stesso tipo e la massima differenza (eterogeneità) tra siti appartenenti a tipi diversi. Può essere usato con funzioni predittive, ad esempio analizzando in tempi diversi eventuali scostamenti dal "tipo atteso", ma non è questa la sua specifica vocazione.

I metodi di ordinamento[24] fanno una cosa simile, esprimendo però somiglianze e differenze tra i siti non in base all'assegnazione ad una classe, ma attribuendo ad ogni sito un valore su una o più variabili astratte; avendo misurato i valori di *n* descrittori in ogni sito, l'idea qui è quella di ridurre le dimensioni su cui rappresentare le differenze tra i vari siti ad una, due o, massimo, tre dimensioni astratte, tre fattori in grado di rendere immediatamente conto delle variazioni più significative tra i siti. Riguardo alle potenzialità predittive, vale quanto detto per i metodi di classificazione.

Considerazioni diverse vanno fatte invece per i metodi di tipo regressivo, come il GLM[25], che danno luogo a modelli che nascono già con una vocazione predittiva.

---

[23] Un "sito" è un tratto più o meno lungo di corso d'acqua (in genere si tratta di decine o poche centinaia di metri) nel quale sono stati rilevati i valori di tutti i descrittori abiotici e biotici ritenuti necessari per caratterizzare l'ecosistema.

[24] Tra di essi, l'Analisi delle Componenti Principali (PCA), l'Analisi delle Coordinate Principali (PCoA), l'Analisi delle Corrispondenze (CA), l'Analisi delle Corrispondenze Canoniche (CCA), il Multidimensional Scaling Non Metrico (NMDS).

[25] General Linear Model: è una generalizzazione della regressione lineare multipla ad un modello multivariato, che tiene conto di più di una variabile dipendente. Il modello matematico alla base di questo metodo può essere facilmente sintetizzato con l'equazione $Y=BX+A$, in cui Y è una matrice con *s* righe, tante quante le variabili dipendenti considerate, ed *n* colonne, tante quante le osservazioni condotte; X è una matrice con *t* righe, tante quante le variabili indipendenti considerate, ed *n* colonne, tante quante le osservazioni condotte; *B* è la matrice *s* x *t* con i coefficienti di regressione da trovare in base ai dati; *A* è la matrice *sxn*, anch'essa da determinare in base ai dati, che rappresenta Y quando la matrice X è la matrice nulla. L'equazione esprime la struttura generale del modello, che postula l'esistenza di relazioni lineari tra le variabili indipendenti e quelle dipendenti e si propone di regolare i parametri (elementi di *B* e *A*) in modo da adattare il modello alle caratteristiche delle relazioni lineari tra

Se le variabili indipendenti sono i descrittori ambientali e quelle dipendenti sono, ad esempio, le abbondanze delle varie specie che compongono la comunità, il modello sarà in grado di prevedere la struttura di comunità in base ai descrittori. Nel prossimo paragrafo vedremo che questi metodi hanno comunque dei limiti importanti per le applicazioni in ecologia e che la scelta del "tipo" di modello da utilizzare richiede una serie di riflessioni.

Ma in cosa consiste la "componente biotica" da prevedere? Idealmente sarebbe utile costruire modelli predittivi che a partire dai descrittori ambientali prevedano l'intera comunità! Tuttavia si incontrerebbero problemi sia tecnici che pratici nel realizzare un'impresa simile: la mole e la qualità dei dati che si dovrebbero ricavare da ogni sito sarebbero notevoli e richiederebbero una grande quantità di tempo e risorse; la complessità del sistema da rappresentare sarebbe elevatissima e potrebbe non essere facile regolare opportunamente i parametri del modello (quand'anche se ne avesse già una formulazione generale) per incorporare tale complessità; probabilmente sarebbe difficile anche andare a "leggere" la struttura del modello per ricavare informazioni sull'ecosistema modellizzato; inoltre si perderebbe ogni praticità nella valutazione dello stato ecologico, perché ogni volta, per conoscere la comunità effettiva e confrontarla con quella prevista, si dovrebbe fare un lavoro di campionamento molto impegnativo. Per queste ragioni è opportuno scegliere come target di previsione solo un gruppo limitato di organismi tra tutti quelli presenti, un gruppo la cui struttura costituisca un valido indicatore dello stato dell'ecosistema nel suo complesso.

In letteratura si evidenziano due tendenze generali: concentrarsi su alcune specie particolari (specie vulnerabili, specie indicatrici, specie ombrello, specie bandiera, ecc.) o su gruppi di specie che giocano nell'ecosistema un ruolo abbastanza integrato con il funzionamento generale, da poterne essere considerati indicatori. La classica visione dell'ambiente fluviale come un'organizzazione di sottosistemi depuranti, ad esempio, mette in evidenza l'esistenza di gruppi funzionali interdipendenti che spesso sono stati scelti come target di previsione.

Un primo gruppo funzionale di organismi è quello che costituisce il *perifiton*, quella pellicola scivolosa che riveste ciottoli e altri substrati duri presenti nei fiumi. Ne fanno parte batteri, funghi, ciliati, microalghe, amebe, rotiferi, nematodi,

---

le diverse variabili che si possono trarre dai dati. Una volta regolati i parametri (fase di calibrazione), il modello sarà in grado di calcolare (prevedere) i valori delle *s* variabili dipendenti in funzione dei valori delle *t* variabili indipendenti, sotto l'ipotesi che le relazioni tra di esse siano di tipo lineare.

gastrotrichi, tardigradi, ecc., ognuno con un proprio ruolo di demolizione, mineralizzazione o riutilizzo della sostanza organica prodotta in acqua o proveniente da terra (foglie, rami e tronchi, escrementi, resti animali, liquami di produzione antropica, ecc.). Come in ogni altro ecosistema, il loro ruolo è cruciale. Data la brevità del loro ciclo vitale e la varietà tassonomico-funzionale che esprimono, la loro risposta ai fattori ambientali può costituire un indicatore flessibile e rapido dello stato ecologico.

Poi ci sono i *macroinvertebrati bentonici*[26], tra cui troviamo crostacei, molluschi, vermi e larve di vari ordini di insetti. Le specializzazioni alimentari, anatomiche, fisiologiche e comportamentali di questi organismi fanno sì che essi siano presenti in tutte le nicchie disponibili a livello di microhabitat e che rispondano in modo flessibile e specifico a variazioni stagionali o occasionali di diverse variabili. Costituiscono una sorta di sistema di regolazione per il perifiton, sia perché se ne nutrono, mantenendone le popolazioni in stato di elevata attività, sia perché sminuzzano i detriti organici grossolani facilitandone il consumo da parte dei microrganismi.

Un terzo gruppo è costituito dai vertebrati, e in particolare dai pesci. Nutrendosi di organismi del macrozoobenthos e di vegetazione acquatica, i pesci contribuiscono anch'essi alla depurazione e al riciclo della sostanza organica, ma hanno la peculiarità di porsi complessivamente, come popolamento[27], a vari livelli nella rete trofica, integrandone una serie di variabili. Attraverso le loro attività biologiche (sosta, rifugio, esplorazione, alimentazione, riproduzione) essi interagiscono in modo molto ricco sia con la componente biologica che con quella fisica dell'ambiente. Le esigenze ambientali legate a ciascuna attività possono differire molto tra una specie e l'altra o tra stadi diversi dello sviluppo di una stessa specie; inoltre, ogni specie richiede la presenza di diversi elementi ambientali per compiere tutte le attività legate alla propria biologia. Dunque, eterogeneità del substrato, sequenze buche-raschi, sinuosità del tracciato e barre di meandro, rive dolcemente digradanti o ripide, ostacoli locali alla corrente (massi, rami incastrati, radici arboree sommerse), vegetazione acquatica rada o consistente, aree di infiltrazione dell'acqua nel sottosuolo e aree di risorgiva, ecc., costituiscono una

[26] Invertebrati di dimensioni non inferiori al millimetro, che passano almeno una parte della vita legati al substrato.

[27] Si usa il termine "popolamento" come traduzione del termine inglese "assemblage", che indica un sottoinsieme di una comunità determinato da un gruppo tassonomico (es. artropodi, diatomee, primati, pesci).

base di variabilità ambientale alla quale il popolamento ittico è complessivamente e specificamente sensibile. Inoltre i pesci sono gli organismi più longevi e mobili in un corso d'acqua per cui le informazioni tratte dalla struttura del popolamento sono molto consistenti a livello sia spaziale che temporale. Infine, il numero totale di specie da considerare (qualche decina) è abbastanza ampio da essere informativo, ma non tanto ampio da divenire ingestibile a livello di monitoraggio o nella costruzione del modello.

L'ultimo gruppo è costituito dall'insieme di organismi, soprattutto vegetali, che popolano l'ambiente circostante al corso d'acqua. Essi producono innanzitutto il detrito che costituisce l'unico apporto energetico almeno nel corso alto dei fiumi. Inoltre la vegetazione svolge una funzione di filtro meccanico e biologico: rallentando le acque di dilavamento e facilitando la sedimentazione del carico solido, impedisce il riempimento degli interstizi tra i ciottoli, microambiente di massima importanza per il perifiton e il macrozoobenthos; favorendo la sedimentazione o il riassorbimento di fosforo e azoto, protegge inoltre il sistema dall'eutrofizzazione. Le fasce riparie possono infine presentare un ricco mosaico ambientale che determina una transizione ecotonale complessa tra ambiente acquatico e zone emerse.

La struttura di ognuno di questi gruppi può costituire un valido indicatore dello stato ecologico del sistema fiume, così come possono farlo, almeno in parte, singole specie chiave; fare una scelta piuttosto che un'altra dipende dalle caratteristiche peculiari che le specie o i gruppi di specie posseggono in relazione agli scopi per cui il modello di previsione viene costruito. In questo lavoro abbiamo scelto di adottare il popolamento ittico a causa della complessità di interazioni che attraverso ogni specie esso esprime nei confronti dell'ambiente fisico e della comunità biologica ad una certa scala, a causa della consistenza spaziale e temporale delle informazioni che da esso si possono trarre e a causa della relativa facilità con cui è possibile catturare e riconoscere le varie specie (Tancioni et al., 2005)[28].

[28] Mirando ad un modello in grado di rappresentare lo stato dell'ecosistema in un sito estremamente localizzato o in risposta a rapidi mutamenti delle condizioni ambientali, si sarebbe probabilmente scelto un gruppo di microrganismi (come le diatomee); mentre se l'interesse si fosse concentrato su scale di maggiore dettaglio (ad esempio, di microhabitat) si sarebbero potuti scegliere i macroinvertebrati bentonici.

1.1.3 Reti neurali e modelli predittivi in ecologia

Creare un modello predittivo di una porzione di comunità in un sistema ecologico non è banale.

Innanzitutto c'è il problema della complessità. In un sistema ecologico c'è: 1) una grande quantità di entità biologiche a vari livelli (individui, specie, popolazioni, comunità) che interagiscono nei modi più diversi; 2) una grande eterogeneità dell'ambiente fisico a varie scale (nel caso dei fiumi, dai microambienti popolati dalla fauna interstiziale, all'articolazione dell'intero bacino idrogeologico); 3) una grande quantità di variabili in gioco (climatiche, fisiche, chimiche, idrogeologiche, etologiche, ecc.); 4) una fitta rete di relazioni tra le varie componenti; 5) confini spesso incerti o sovrapposizioni tra sottosistemi diversi. Questo esclude spesso che si possano formulare modelli analitici per descrivere i rapporti quantitativi tra i descrittori ambientali rilevanti e un popolamento target. Ma anche la formulazione di un modello empirico può rivelarsi macchinosa e improduttiva, perché la regolazione dei parametri in funzione dei dati è comunque subordinata alla possibilità di cogliere a sufficienza il modo in cui interagiscono le diverse variabili in gioco e di riuscire a tradurre questa intricata rete relazionale in un opportuno sistema di equazioni. Come si accennava nel primo paragrafo, un'alternativa è quella di non impegnarsi nella fase di formulazione, ma di adottare un modello generale già formulato e testato; in tal modo si potrebbe passare direttamente alla fase di calibrazione, che attraverso la regolazione dei parametri adatta il modello allo specifico problema predittivo di cui ci si sta occupando. Tuttavia questi modelli generali determinano a priori e in modo univoco la forma delle relazioni tra variabili indipendenti e dipendenti e spesso sono molto limitati nel tenere conto degli effetti di eventuali interazioni tra variabili. I General Linear Models, ad esempio, sono basati su relazioni di tipo lineare, soffrono della presenza di legami tra le variabili indipendenti (es. multicolinearità) e hanno una ridotta capacità di considerare le diverse interazioni tra i regressori nel determinare le variabili dipendenti. Se un sistema è caratterizzato, come nel caso degli ecosistemi, da relazioni di forme eterogenee tra variabili spesso intercorrelate, allora un modello generale è una soluzione insufficiente.

L'altro problema fondamentale è che l'approccio statistico tradizionale, sofisticato e flessibile quando si ha a che fare con dati ottenuti da campionamenti ed esperimenti ben controllati, perde in validità ed attendibilità con dati di natura più eterogenea e meno controllabile, come sono spesso quelli legati alle osservazioni in

ecologia (Fielding, 1999). I metodi statistici, infatti, si basano su una serie di assunzioni che sono necessarie per garantire che le distribuzioni teoriche di probabilità usate come riferimento siano valide. Anche i metodi non parametrici, meno esigenti di altri da questo punto di vista, restano tuttavia legati a modelli di riferimento che richiedono distribuzioni di frequenza di un certo tipo nei dati (ad esempio, distribuzioni unimodali e/o simmetriche), relazioni riconducibili a modelli lineari o di altre forme note, assenza o limitata presenza di valori nulli, indipendenza delle osservazioni, quantità di dati che non vadano al di sotto o al di sopra di una massa critica e che siano stati acquisiti in modo omogeneo e controllato. In ecologia, invece, ci si ritrova spesso ad avere distribuzioni di frequenza multimodali e asimmetriche; relazioni tra variabili molto complesse e soggette ad influenze esterne da parte di altri fattori; presenza di molti valori nulli che non è sempre facile attribuire ad una reale determinazione della variabile misurata o ad assenza di informazione (es. i valori di abbondanza relativi a specie rare); osservazioni non indipendenti a causa di pseudoreplicazione o autocorrelazione spaziale e temporale; squilibrio nella quantità di informazioni disponibili per diverse variabili o grande abbondanza generale di informazione in certi ambiti (si pensi ai dati telerilevati); presenza di dati non sempre omogenei rispetto ai metodi di acquisizione. Anche qui esiste un'alternativa, che tuttavia è da ritenere scorretta nella maggior parte dei casi: ignorare il fatto che i sistemi e i dati con cui si lavora in ecologia presentano caratteristiche formalmente incompatibili con i metodi statistici tradizionali e fidarsi della robustezza di questi ultimi, cioè della loro capacità di fornire analisi e modelli attendibili anche al di fuori delle condizioni di validità fissate dalla teoria in base alla quale sono stati formulati.

Negli ultimi vent'anni, però, alcuni ecologi hanno cominciato ad affrontare il problema in un modo diverso, legando alcuni aspetti numerici e metodologici del loro lavoro a tecniche elaborate nell'ambito dell'intelligenza artificiale.

L'intelligenza artificiale nasce tra gli anni '40 e gli anni '50 del Novecento, intercettando i percorsi di riflessione e ricerca provenienti da molte discipline: filosofia, fisiologia, psicologia, biologia, matematica, logica, tra quelle che esistevano già, e informatica, cibernetica, ingegneria dei calcolatori, teoria dell'informazione, tra quelle che si venivano sviluppando in quegli stessi anni. Il 1943, in particolare, fu un anno importante per varie ragioni, tra cui un fondamentale articolo di McCulloch e Pitts in cui veniva teorizzato per la prima volta il *neurone*

artificiale[29]. Warren Sturgis McCulloch era uno studioso di psicologia e neurofisiologia, mentre Walter Pitts era un logico che lavorava nell'ambito della psicologia cognitiva; essi si chiesero come le funzioni base del pensiero potevano essere determinate dall'attività nervosa. Lo sviluppo di un modello artificiale estremamente semplificato di cellula nervosa permise loro, tra l'altro, di dimostrare che già un'unica unità di elaborazione elementare era in grado di risolvere semplici problemi di logica, ricevendo come input alcune informazioni (premesse) e fornendo come output la soluzione del problema (conclusioni).

In Fig. 5 è mostrato un modello un po' più sofisticato di quello originario di McCulloch e Pitts, ma l'essenza è la stessa: il neurone artificiale è una funzione matematica del tipo $y(\bar{x}, \overline{w})$, che ha per variabili indipendenti il vettore $\bar{x}$, con i segnali provenienti dall'esterno del neurone, e il vettore $\overline{w}$, con i *pesi sinaptici* delle connessioni che portano i segnali al neurone; la variabile dipendente, cioè l'output del neurone, è lo *stato di attivazione* y di quest'ultimo, che può costituire l'output definitivo desiderato oppure essere usato a sua volta come segnale di ingresso per altri neuroni.

[29] La storia dell'intelligenza artificiale non si riduce a quella delle reti neurali, ma qui si farà riferimento soprattutto a quest'ultima, in modo da introdurre progressivamente tutti i concetti necessari alla comprensione degli aspetti tecnici dello studio presentato in questa tesi.

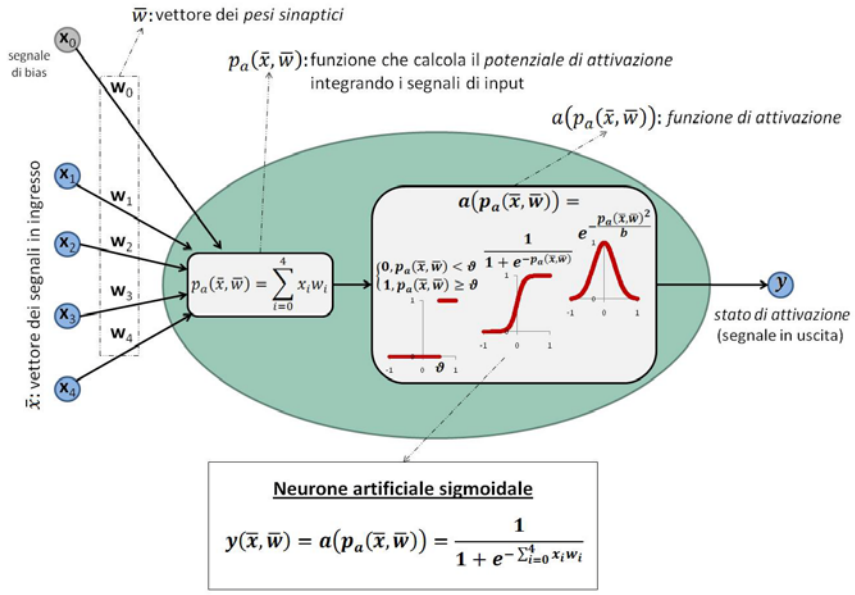


Fig. 5

La funzione $y(\bar{x},\bar{w})$ risulta dalla composizione di due diverse funzioni. Innanzitutto una funzione p_a che calcola il *potenziale di attivazione* del neurone combinando i segnali in ingresso e usando i pesi sinaptici per decidere l'effetto che il segnale trasmesso tramite una particolare connessione avrà nel determinare l'attivazione finale del neurone; quest'ultima operazione in genere viene compiuta calcolando il prodotto scalare (detto anche prodotto interno) tra il vettore dei segnali e quello dei pesi, cioè moltiplicando il generico segnale x_i (che varia tra 0 e 1) con il peso della connessione relativa w_i (che varia tra -1 e 1). Se il peso w_i della i-esima connessione è elevato, allora gran parte di x_i passerà alla funzione di attivazione e contribuirà all'attivazione finale del neurone; se invece w_i è vicino o uguale a zero, poco o nulla di x_i passerà; se w_i è negativo, avremo una connessione inibitoria e il valore di x_i contribuirà a diminuire l'attivazione del neurone artificiale. Tra le possibili funzioni per combinare i segnali in ingresso, la funzione p_a riportata in figura, che determina una combinazione lineare degli input, è quella più usata per calcolare il potenziale di attivazione[30]. La seconda funzione che interviene nel

[30] Le unità neurali di tipo RBF (Radial Basis Funtion) o quelle di tipo Sigma-Pi costituiscono esempi di neuroni artificiali che fanno uso di una diversa funzione di combinazione per il calcolo del potenziale di

modello matematico del neurone artificiale è la *funzione di attivazione*, che ha per argomento il potenziale di attivazione calcolato tramite la funzione p_a e permette il calcolo dello *stato di attivazione* del neurone. Vari tipi di funzione possono essere usati. La figura ne mostra tre: la funzione a gradino con soglia θ, usata da McCulloch e Pitts nel loro modello; la funzione sigmoide, molto comune nei modelli moderni; la funzione normale, usata solo per scopi specifici. Con ingressi di tipo binario (che potevano assumere solo i valori 1 e 0), pesi sinaptici opportunamente regolati e funzione di attivazione a gradino (che dà 0 per valori al di qua di una soglia θ e 1 per valori superiori), McCulloch e Pitts crearono un modello, detto anche *unità logica a soglia*, in grado di fornire i valori di verità di proposizioni generate tramite gli operatori logici AND, OR, NOT, cioè di simulare semplici ragionamenti a livello di logica proposizionale. Dimostrarono inoltre che più *unità* neurali artificiali di questo tipo, connesse tra loro secondo una particolare *architettura* e dotate di un'opportuna matrice di pesi sinaptici, erano in grado di svolgere calcoli di qualsiasi complessità.

La loro era una dimostrazione teorica, che fu determinante nel favorire lo sviluppo di ulteriori ricerche sui modelli di calcolo *connessi* (basati cioè su molti elaboratori semplici collegati in rete, piuttosto che su un unico elaboratore molto potente), *paralleli* (in cui i diversi elaboratori svolgono contemporaneamente molte operazioni, a differenza del modello seriale, in cui un unico elaboratore svolge una singola operazione alla volta) e *distribuiti* (in cui la memoria e il software che determinano i processi di calcolo sono parte integrante della struttura di elaborazione e si trovano distribuiti in essa, piuttosto che essere codificati e salvati all'esterno dell'elaboratore). Nei decenni successivi sarebbe divenuto via via sempre più chiaro che questo modo di elaborare le informazioni era particolarmente adatto ad alcuni problemi con cui gli elaboratori classici (di tipo seriale) avevano grosse difficoltà a confrontarsi; problemi di identificazione o di riconoscimento di strutture all'interno di configurazioni complesse. Un problema classico di questo tipo è quello del riconoscimento dei volti, un problema semplicissimo per scimmie e neonati di pochi mesi, ma proibitivo per un computer seriale anche molto potente! Si tratta di problemi formalmente riconducibili a situazioni in cui vi sono molte variabili che interagiscono esprimendo strutture di grande complessità... situazioni molto simili a quelle che si incontrano in ecologia! Il modello di McCulloch e Pitts diede un

attivazione. Costituiscono tuttavia soluzioni usate con reti neurali di tipo diverso da quelle qui considerate.

impulso notevole agli studi su questi metodi di elaborazione, ma presentava ancora alcuni importanti limiti pratici, legati al fatto che l'efficacia della rete è comunque determinata dalla scelta dell'architettura, della funzione di attivazione usata nelle singole unità e dell'insieme dei pesi sinaptici delle connessioni: definire tutto questo a tavolino, su una base teorica, comporta tutti i problemi relativi alla formulazione di un modello complesso ex novo (come si diceva nel paragrafo precedente) ed è praticamente impossibile (soprattutto per quanto riguarda i pesi); altrettanto arduo sarebbe d'altra parte costruire il modello per prove ed errori, poiché le combinazioni delle varie scelte possibili sono troppo numerose perché un processo di questo tipo possa essere efficiente.

In un lavoro di Donald Hebb del 1949, considerato di fondamentale importanza per tutta una serie di discipline, dalle neuroscienze alla psicologia, all'intelligenza artificiale, si cominciò a studiare il modo in cui le connessioni sinaptiche potevano incrementare o diminuire la propria efficienza in base all'esperienza: era un primo tentativo di descrivere i fenomeni elementari che stavano alla base dell'*apprendimento* nei sistemi biologici. Si cominciò dunque a studiare algoritmi che permettessero di fare qualcosa di simile con le reti neurali artificiali, in modo da produrre una regolazione automatica dei pesi sinaptici in base ad una sorta di "esperienza artificiale" fornita alla rete tramite "esempi".

Si immagini di aver condotto su alcune unità statistiche la misura della variabile X e della variabile Y, ritenuta dipendente da X; gli "esempi" saranno dunque costituiti da coppie di valori. Una rete neurale potrebbe essere costruita in modo che un neurone riceva i segnali in ingresso (variabile X) e che un altro fornisca le previsioni in uscita (variabile Y); tra di essi possono esservi altri neuroni, combinati in modi che per adesso non sono importanti. Le connessioni tra i neuroni sono dotate di pesi sinaptici che inizialmente vengono fissati in modo casuale. Un algoritmo di apprendimento automatico funziona in modo da proporre alla rete un valore di X come input e da rilevare poi se il valore di Y previsto dalla rete Y_{OUT} è più o meno vicino al valore di Y osservato sul campo Y_{OBS}; basandosi su una regola (che, si vedrà, non è banale trovare), l'algoritmo agisce poi in modo reiterato sui pesi sinaptici con l'obiettivo di far avvicinare il più possibile le successive previsioni alla realtà misurata, cioè di ridurre il più possibile il valore di $|Y_{OUT} - Y_{OBS}|$. Tali

algoritmi, detti di *machine learning*[31], aggirano dunque il problema di trovare su base teorica o per prove ed errori i pesi sinaptici e rendono possibile l'apprendimento automatico basato su esempi. Questo è particolarmente importante quando le variabili in gioco sono molte e il numero e la complessità strutturale dei pesi sinaptici sono elevati.

I primi algoritmi basati sul lavoro di Hebb non erano però ancora abbastanza efficaci e restava da risolvere volta per volta anche il problema dell'architettura della rete e della funzione di attivazione da adoperare. Nel 1957 fu pubblicato un lavoro di Andrej Kolmogorov che dimostrava la possibilità di approssimare una qualunque funzione di più variabili, con una qualsiasi accuratezza, tramite una serie di funzioni organizzate secondo una struttura riproducibile in una rete neurale con certe caratteristiche. Tra il 1958 e il 1962, Frank Rosenblatt e colleghi realizzarono, sulla base del lavoro di Kolmogorov e di Hebb, il *perceptron*, una rete neurale dotata di un'architettura e di proprietà matematiche che permettevano per la prima volta l'uso di un algoritmo di apprendimento che, attraverso una regolazione reiterata dei pesi sinaptici, permetteva alla rete di convergere in modo stabile verso l'approssimazione desiderata.

[31] Il *Machine Learning* è un ramo dell'intelligenza artificiale che si occupa specificamente dell'apprendimento automatico nei modelli artificiali, non solo basati su reti neurali.

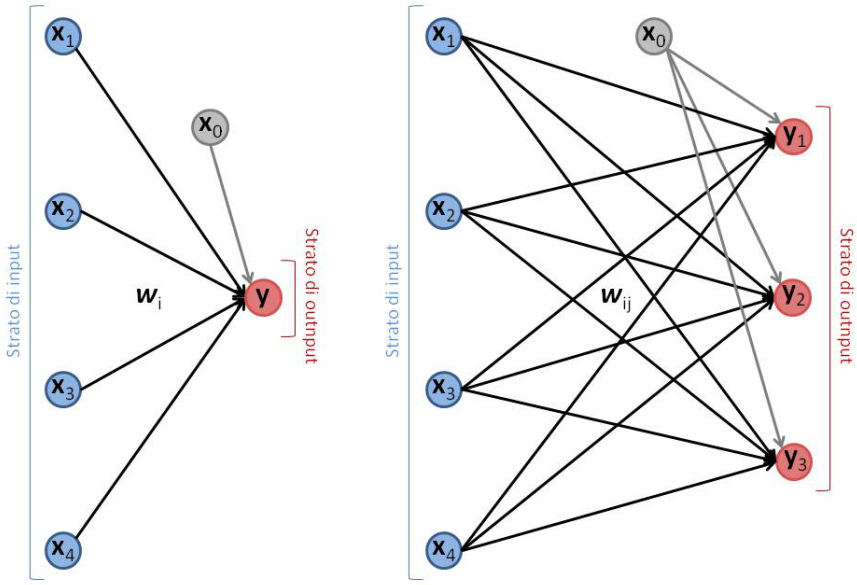


Fig. 6

Quando si parla di "architettura" di una rete neurale, si fa riferimento ad una serie di caratteristiche: il numero di unità neurali presenti, quante di esse ricevono i segnali di input e quante forniscono l'output, quali neuroni sono connessi tra loro, in che direzione vanno le connessioni, ecc. L'architettura del più semplice modello di percettrone (Fig. 6a) ha una serie di neuroni di input, tutti connessi ad un unico neurone di output; è una rete *feedforward*, in quanto le connessioni determinano solo un flusso "in avanti" dell'informazione, senza connessioni che tornino da neuroni successivi a neuroni precedenti (le reti in cui accade questo sono dette *ricorrenti* o *a feedback*[32] e possono presentare anche delle *autoconnessioni*, cioè delle connessioni di un nodo con se stesso); i neuroni di input formano qui uno s*trato*, cioè un insieme di nodi connessi con porzioni precedenti o successive della rete (con riferimento al flusso dell'informazione), ma non tra di loro (sono cioè assenti le connessioni *laterali*); infine, il percettrone singolo è una rete *totalmente connessa*, in quanto tutti i neuroni di uno strato (in questo caso quello di input) sono connessi con tutti i neuroni dello strato successivo (che qui ha un unico neurone). Se più percettroni

[32] Esempi di questo tipo di architettura sono le reti di Hopfield e le Macchine di Bolzman.

singoli vengono combinati insieme, allora si ha una rete di percettroni (Fig. 6b) con architettura *n*x*m*, in cui *n* è il numero dei nodi di input ed *m* è il numero dei nodi di output. In entrambi i casi, le connessioni hanno tutte un peso sinaptico. Per conoscere le caratteristiche di una rete neurale, oltre all'architettura deve essere specificato il tipo di elaboratore elementare, cioè di unità neurale, di cui si è fatto uso. Come si è già visto (Fig. 5), un'unità neurale è caratterizzata da una prima funzione che interviene nel calcolo del potenziale di attivazione e da una seconda funzione che determina lo stato di attivazione. Nel percettrone di Rosenblatt il potenziale di attivazione è calcolato come combinazione lineare dei segnali ingresso (valori continui tra 0 e 1, con i coefficienti dati dai pesi sinaptici), mentre la funzione di attivazione è ancora una funzione a soglia, come nel combinatore lineare a soglia di McCulloch e Pitts. Dunque l'output è binario[33].

Il modello di Rosenblatt ebbe il merito di fissare alcuni parametri fondamentali dell'architettura e delle unità neurali sulla base del lavoro teorico di Kolmogorov, ma ancora più importante fu l'elaborazione di un algoritmo di addestramento, cioè di un processo automatico che sfruttando le conclusioni di Hebb regolava i pesi sinaptici e faceva sì che la rete approssimasse *realmente* le funzioni proposte tramite gli esempi. Era un passo avanti che apriva le porte a sviluppi teorici e applicativi di grande portata e alimentò dunque la curiosità e l'interesse generale della comunità scientifica negli anni '60. Purtroppo il modello di Rosenblatt aveva ancora un limite importante: una rete con due soli strati di neuroni, uno di input e l'altro di output può approssimare solo un certo tipo di funzioni, mentre altre restano al di fuori della sua portata[34]. Questo Rosenblatt lo sapeva (perché Kolmogorov lo aveva dimostrato) e cercò di creare anche dei modelli più sofisticati, con unità non lineari (cioè con funzione di attivazione non lineare) e con tre strati (*Multilayer Perceptron*), in cui lo strato intermedio (detto anche strato nascosto) connetteva quello di input e di output (Fig. 7).

[33] Pertanto il percettrone singolo di Rosenblatt aveva la caratteristica di separare in due parti lo spazio n-dimensionale definito in $\mathbb{R}^n$, riconducendo cioè la posizione di un punto in quello spazio (ogni punto rappresenta l'insieme n-dimensionale di misure necessarie a caratterizzare un dato o esempio) all'appartenenza ad una di due diverse classi.

[34] Se il problema modellistico va oltre l'individuazione di porzioni *linearmente separabili* dello spazio n-dimensionale dei descrittori, il percettrone di Rosenblatt è inadeguato. Si pensi al caso semplificato in cui vi sono solo due nodi di input, per cui i punti che descrivono lo spazio dei descrittori si dispongono su un piano; se la classificazione richiesta al neurone di output implica l'assegnazione del valore 1 e del valore 0 a due diverse parti dello spazio dei descrittori che possono essere identificate tracciando una linea, allora il percettrone di Rosenblatt è efficace; ma se già la linea di separazione è curva, l'efficacia diminuisce drasticamente. Purtroppo anche funzioni relativamente semplici, come quella data dal connettivo logico XOR (eXclusive OR) presentano caratteristiche di non separabilità lineare.

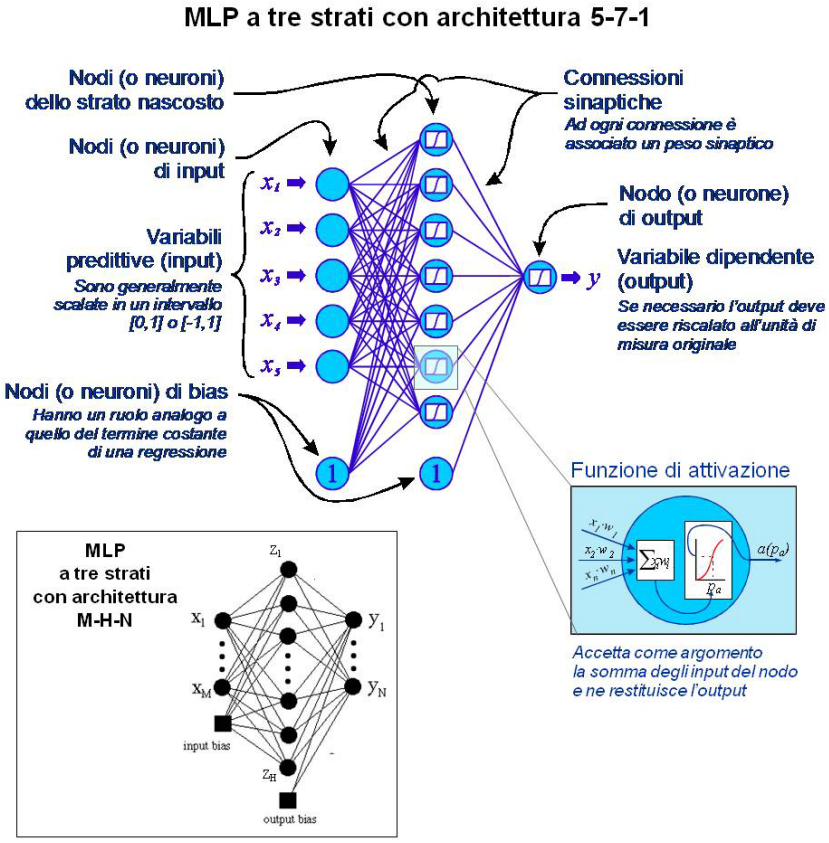


Fig. 7

Ma il problema vero era l'algoritmo di addestramento: con tre strati non si riusciva a creare un algoritmo che convergesse stabilmente fino ad approssimare la funzione desiderata. Neanche Bernard Widrow e Marcian E. Hoff, che in quegli anni perfezionarono ulteriormente l'algoritmo di apprendimento giungendo alla formulazione della *regola delta* (*delta rule*) (1960), riuscirono a trovare una soluzione efficace ai limiti del percettrone. Tuttavia, la regola delta avrebbe avuto un ruolo notevole negli sviluppi di poco più di un decennio più tardi.

In estrema sintesi, la regola delta si basa sulla misura di quanto l'uscita OUT_j del j-esimo nodo di output in un percettrone si discosta dall'uscita attesa OBS_j,

determinata dai dati osservati. Da questo scostamento si ricava una misura dell'errore compiuto dalla j-esima unità di output:

$$E_j = \frac{1}{2}\left(OBS_j - OUT_j\right)^2 = \frac{1}{2}\left(OBS_j - a(\sum_{i=0}^{n_1} w_{ij}\, x_i)\right)^2$$

Lo scostamento viene elevato al quadrato sia per considerare allo stesso modo scostamenti positivi e negativi, che per dare maggiore importanza agli errori grandi; il prodotto per $1/2$ ha l'unico scopo di facilitare i passaggi di calcolo successivi: questa misura dell'errore è l'*MSE* (Mean Square Error), cioè l'errore quadratico medio. Ovviamente, il termine OUT_j dipende dalla funzione di attivazione *a*, che in base al contributo pesato degli n_1 segnali in ingresso (più quello di bias) determina l'uscita del nodo *j*. Qui w_{ij} indica il peso sinaptico della connessione tra l'i-esimo neurone di input e il j-esimo neurone di output: è l'elemento più importante della formula, poiché ciò che vogliamo fare tramite la regola delta è proprio modificarne il valore per ridurre E_j. Dunque, in base all'uscita OUT_j del j-esimo neurone di output, la regola delta ci permette di regolare i pesi sinaptici di tutte le connessioni che quel neurone riceve dallo strato precedente; infatti, considerando costanti gli n_1 stati di attivazione (x_i) dei neuroni in ingresso (di fatto, ogni volta che viene fornito alla rete un vettore di dati, i suoi elementi sono costanti), possiamo dire che E_j è una funzione degli n_1 pesi sinaptici delle relative connessioni e cercare quindi di modificare questi ultimi per rendere l'output del neurone *j* il più possibile conforme ai dati. Inoltre, il peso di ogni connessione può essere regolato separatamente dagli altri, per cui volta per volta è possibile considerare costanti gli $(n_1\text{-}1)$ pesi sinaptici rimanenti. Per ogni peso sinaptico considerato, dunque, l'elemento $a(\sum_{i=0}^{n_1} w_{ij}\, x_i)$ nella formula sopra riportata può essere scomposto in modo da contenere una parte costante *c*, determinata dalla funzione di attivazione applicata alla somma dei prodotti degli altri pesi sinaptici con i relativi stati di attivazione e da una parte variabile $a(wx)$ costituita dalla funzione di attivazione applicata al prodotto tra il peso sinaptico considerato w e lo stato di attivazione x (preso come costante) del neurone a monte:

$$E_j(w) = \tfrac{1}{2}\left(OBS_j - OUT_j\right)^2 = \tfrac{1}{2}\left(OBS_j - c - a(wx)\right)^2.$$

Così trasformata, la curva dell'errore diviene una semplice parabola e per trovare il valore di w in corrispondenza del quale l'errore è minimo, basta trovare gli zeri della sua derivata prima. Ma nella realtà (cioè, al di là delle semplificazioni che hanno condotto alla definizione di questa curva) le cose sono più complicate: i valori di x e di OBS_j cambiano ogni volta che dati diversi vengono presentati alla rete e la costante c varia a sua volta sia in funzione dei diversi dati presentati alla rete che in funzione delle variazioni che l'algoritmo di addestramento induce di epoca in epoca sui pesi sinaptici. Dunque si può immaginare che la funzione d'errore relativa ad un peso sinaptico sia in realtà più complessa di una semplice parabola e che una sua formulazione analitica univoca (cioè un'unica equazione che la definisca) sia impossibile da determinare perché ad ogni ciclo essa cambia al variare dei pesi sinaptici. Tuttavia è comunque utile far riferimento ad un modello generico, ancorché indeterminato, di funzione d'errore al variare del singolo peso sinaptico w (Fig. 8).

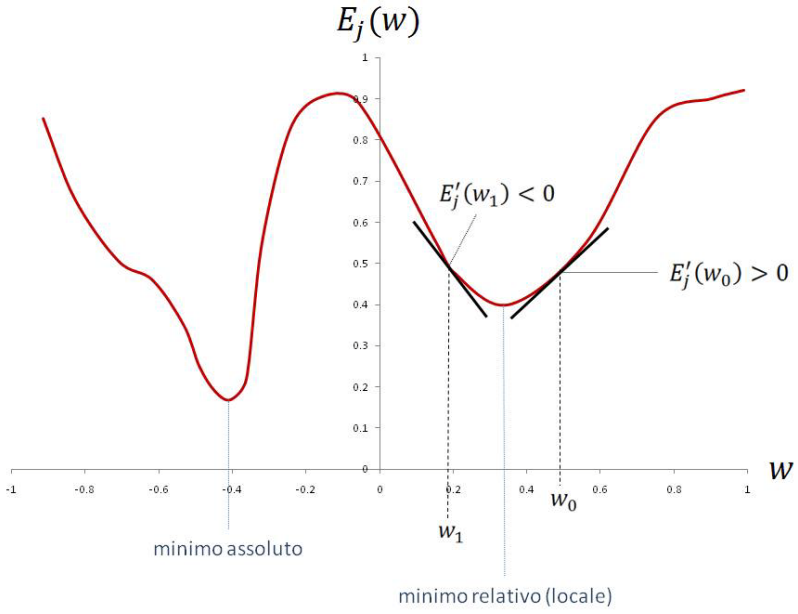


Fig. 8

Possiamo infatti assumere che, almeno per piccole variazioni di w e nell'ambito di un numero limitato di cicli di addestramento, la funzione $E_j(w)$ sopra proposta approssimi bene tale modello generico (Pessa, 2004) e che la sua derivata

prima possa contribuire a seguirne il profilo fino ad avvicinarsi ad un punto di minimo: se si parte da un punto a caso w_0, la derivata misura l'inclinazione della retta tangente alla curva in quel punto, per cui, sottraendo a w_0 un valore Δw proporzionale alla derivata, ci si avvicina certamente al punto di minimo più vicino. Dunque la regola (*regola delta* o *delta rule*)

$$\Delta w = -\eta E_j^{'}(w_0)$$

in cui $E_j^{'}(w_0)$ è la derivata della funzione d'errore nel punto w_0 ed η è un parametro che stabilisce l'intensità dello spostamento, fornisce un algoritmo per decidere di volta in volta di quanto spostarsi per avvicinarsi sempre di più al valore di w che minimizza l'errore; se in prossimità del minimo, uno spostamento un po' troppo pronunciato facesse superare il valore ottimale di w, la derivata cambierebbe segno e determinerebbe immediatamente un'inversione della rotta, puntando nuovamente nella direzione corretta. Questo metodo (cioè quello di usare la derivata di una funzione per seguire il profilo di un'altra con la quale essa non coincide, se non in un range spazio-temporale molto limitato) corrisponde un po' a cercare a tastoni la maniglia di una porta per uscire da un ambiente ampio, sconosciuto, buio e che si modifica nel tempo: non si evita il rischio di cadere in minimi locali (maniglie di porte che non sono quelle dell'uscita), cioè in punti di minimo relativo che non corrispondono alla soluzione ottimale. Un accorgimento è quello di scegliere valori del parametro η (detto tasso di apprendimento o *learning rate*) abbastanza alti da consentire salti più pronunciati ad ogni spostamento, ma non tanto da impedire la convergenza desiderata[35]. Un'altra possibilità è quella di sfruttare il fatto che i pesi sinaptici vengono inizialmente fissati in modo casuale: poiché la loro configurazione iniziale condiziona sia i singoli passi che il risultato finale ottenuto dall'algoritmo di apprendimento, la conduzione di varie sessioni di training aumenta la probabilità di sfuggire ai minimi locali e di raggiungere una configurazione finale dei pesi

[35] Vi è anche un altro parametro che può intervenire nell'applicazione della regola delta. Come il tasso di apprendimento, esso è moltiplicato per un valore che può essere aggiunto alla formula: $\Delta^t w = -\eta E_j^{'}(w_0) + \alpha \Delta^{t-1} w$. In pratica, nel calcolo dello spostamento $\Delta^t w$ da compiere al tempo *t*, può essere considerata anche una frazione dello spostamento effettuato al tempo precedente, $\Delta^{t-1} w$. Quanta parte di questo spostamento debba influenzare lo spostamento attuale è determinato tramite il parametro α, detto *momentum*: più questo è elevato, maggiore è la componente di "inerzia" che tende a mantenere la direzione precedente dello spostamento lungo il gradiente. Lo scopo è evitare spostamenti troppo lenti o troppo veloci, oscillazioni, situazioni di arresto dovute ad una superficie della funzione di errore localmente piatta.

sinaptici che dia un errore minimo in corrispondenza di tutte le connessioni presenti nella rete.

Applicare il principio generale di questa regola (*discesa lungo un gradiente* della funzione d'errore) ad un modello completo di rete neurale non è banale, sia a causa della presenza di più di un descrittore (e quindi di più pesi sinaptici da regolare contemporaneamente), sia perché non si ha a che fare con un unico esempio, ma con molti esempi a cui il modello deve adattarsi, sia, infine, perché l'approssimazione di alcune funzioni richiede l'aggiunta di uno o più strati interni di neuroni (tra quello di input e quello di output) che complica ulteriormente la struttura dei pesi sinaptici da regolare.

Alla fine degli anni '60 non c'era ancora un algoritmo di addestramento efficace per una rete con più di due strati. Così nel 1969 uscì un volume in cui Marvin Minsky e Seymour Papert dimostravano matematicamente i limiti del percettrone e mettevano in evidenza le difficoltà nel trovare un algoritmo convergente per i percettroni multistrato. Anche grazie al prestigio degli autori, lo scritto ebbe una risonanza enorme e determinò praticamente l'abbandono delle ricerche sul percettrone, nonché la cessazione di gran parte dei finanziamenti. L'interesse per le reti neurali si sposta in quegli anni su modelli con architetture diverse, come quelli di Kohonen e di Hopfield. Si tratta spesso di modelli ad apprendimento *non supervisionato*, che è sostanzialmente diverso da quello *supervisionato* tipico dei percettroni. In questi ultimi, gli esempi presentati contengono sia l'input che l'output desiderato, per cui il modello viene guidato e corretto dai dati nel compito di fornire un determinato output in funzione di un certo input; questi modelli sono particolarmente adatti ad approssimare funzioni. L'apprendimento non supervisionato, invece, non prevede l'addestramento del modello a riprodurre un output noto in funzione dell'input, quanto piuttosto una riorganizzazione interna automatica del sistema che gli consenta di trovare un modo originale per riclassificare le informazioni in input; questi modelli sono utili per funzioni di *clustering*.

Alcuni studiosi continuarono a lavorare sugli algoritmi di addestramento dei percettroni e già a metà degli anni '70 fu elaborato un algoritmo in grado di portare a compimento l'addestramento di un percettrone multistrato (MLP, Multilayer Perceptron) con un numero qualsiasi di strati e di neuroni all'interno di ogni strato (era il 1974 quando Paul Werbos illustrava per la prima volta l'algoritmo nella sua

tesi di dottorato[36]). Il medesimo algoritmo venne riscoperto più volte negli anni che seguirono (Floreano, 1996), finché nel 1986 Rumelhart et al. non ebbero successo nel pubblicare e diffondere l'algoritmo EBP (*Error Back Propagation*, propagazione all'indietro dell'errore), che rappresentava la chiave per superare le difficoltà poste da Minsky e Papert nel 1969. Esso si basa essenzialmente sulla *regola delta*, che a sua volta è fondata sul principio di *discesa del gradiente*; nei MLP con un solo strato intermedio, la regolazione dei pesi sinaptici avviene a partire dalle connessioni tra lo strato nascosto e quello di output, per poi spostarsi all'indietro (rispetto al flusso dei segnali nella rete) verso le connessioni tra lo strato di input e quello interno. L'algoritmo lavora in modo iterativo, fino ad ottenere la convergenza.

L'addestramento di una rete tramite algoritmo EBP richiede una serie di scelte che non si limita alla fissazione del *learning rate* o del *momentum*.

Innanzitutto è necessario decidere come usare i dati a disposizione. Da questa scelta può dipendere infatti la *generalità* del modello addestrato, cioè la sua capacità di fornire previsioni accurate anche con dati diversi da quelli utilizzati per il training (Fielding & Bell, 1997; Guisan & Zimmermann, 2000; Özesmi et al., 2006), nonché la possibilità di ottenere misure di performance del modello che ne esprimano anche le capacità di generalizzazione.

La scelta più semplice è quella di usare per il training tutti i dati presenti e di valutare poi la performance del modello sui medesimi dati. Questa scelta, necessaria quando i dati a disposizione sono pochi, è tuttavia sconsigliabile in generale, perché nel valutare la performance del modello non è possibile distinguere i casi in cui la rete ha "appreso" un modello generale delle relazioni tra le variabili in gioco dai casi in cui, invece, non ha fatto altro che "imparare a memoria" i dati di partenza. Vi sono varie tecniche per aggirare parzialmente questo problema (*jack-knife*, *cross-validation*, *bootstrap*, ecc.; si veda, ad esempio, Guisan e Zimmermann, 2000), ma la soluzione decisiva è quella di usare due insiemi indipendenti di dati, uno per il training e uno per la valutazione della performance (Maier & Dandy, 2000). Sarebbe auspicabile che i due insiemi fossero realmente indipendenti, cioè derivanti da campagne separate di raccolta dati, ma una buona alternativa, praticabile quando la quantità di dati lo permette, è quella dell'approccio cosiddetto *split-sample* (Van Houwelingen & Le Cessie, 1990), in cui un unico data set viene opportunamente suddiviso in due insiemi di dati il più possibile omogenei. Non esiste una regola assoluta per stabilire la proporzione di casi da assegnare all'insieme di training e a

[36] In realtà, una prima formulazione dell'algoritmo risale addirittura al 1969 (Bryson et al.).

quello di test, ma esistono varie regole empiriche, tra cui quella di Schaafsma & van Vark (1979) per cui, in presenza di più di 10 predittori, al training set vanno il 75% circa delle osservazioni. Questa percentuale è giustificata dall'esigenza di assegnare più informazione possibile all'insieme su cui si basa il processo di calibrazione

L'uso di un *test set* indipendente garantisce che la performance del modello venga misurata tenendo conto delle sue capacità di generalizzazione, ma non contribuisce in alcun modo a far sì che in fase di addestramento il modello sviluppi tale capacità. Una strategia che può rivelarsi molto efficace in questo senso è l'*early stopping* (Fig. 9), che implica un'ulteriore suddivisione del *training set* in un *training set* vero e proprio e in un *validation set*.

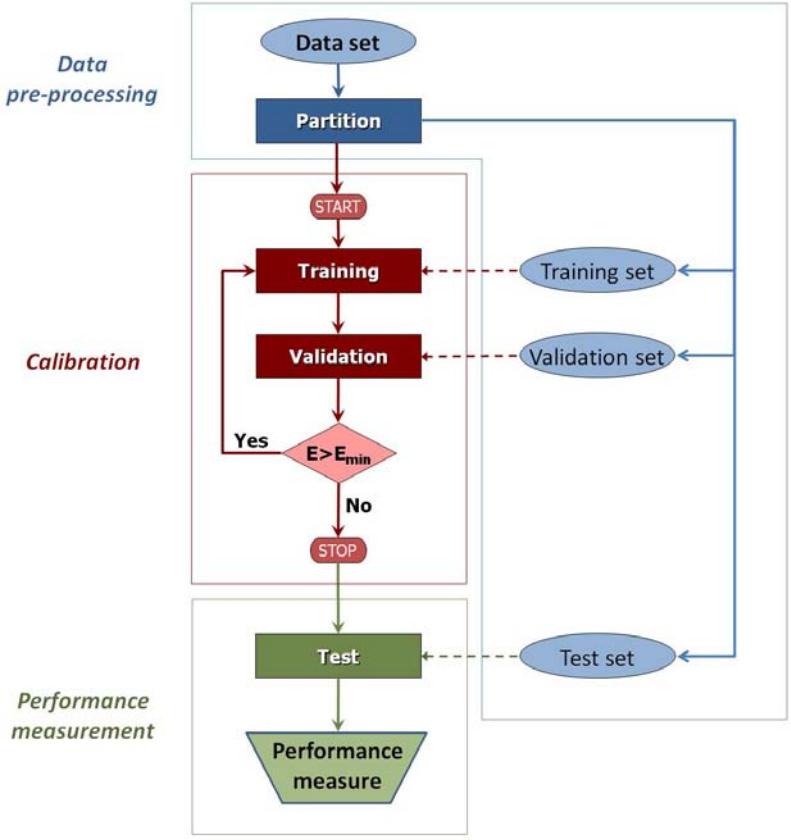


Fig. 9 - Addestramento con *split-sample* ed *early stopping*.

All'insieme di training (*training set*) vengono assegnati i dati in base ai quali si effettua la calibrazione del modello, cioè la regolazione automatica dei pesi sinaptici tramite algoritmo EBP. La presentazione degli esempi tratti dal *training set* può avvenire sia in modalità *on-line*, che in modalità *batch*. Nel primo caso, ogni volta che un singolo esempio viene proposto alla rete, i pesi vengono immediatamente regolati con l'algoritmo EBP; nel secondo caso, alla rete viene proposto un insieme di esempi e l'algoritmo EBP è applicato sull'errore complessivo misurato.

Ad ogni sessione, la nuova configurazione di pesi sinaptici ottenuta viene provata sull'insieme di validazione (*validation set*); finché l'errore misurato in fase di validazione decresce, la procedura viene reiterata, quando invece esso non decresce più e tende a risalire, il processo di calibrazione viene interrotto e si considera concluso. Questa verifica di performance basata su dati separati da quelli di training ha lo scopo di evitare un iperadattamento (*overfitting*) del modello ad un insieme particolare di dati e di garantirne così una capacità di *generalizzazione* che lo renda efficace anche in situazioni leggermente diverse da quelle contemplate dall'insieme di training. Se il training fosse basato su un unico insieme di dati, l'errore tenderebbe sempre a scendere e dopo alcuni cicli si stabilizzerebbe assicurandoci che il modello ha raggiunto la sua massima capacità di previsione; se però i dati usati nel training non contemplano in modo esaustivo tutte le possibili combinazioni tra le variabili in gioco (circostanza di fatto impossibile!), ma presentano dei "buchi" nello spazio degli eventi osservabili, il rischio è che per puntare a minimizzare l'errore nei casi considerati, l'algoritmo EBP determini una regolazione dei pesi sinaptici inefficace con casi che ricadono nei "buchi". Il confronto con l'insieme di validazione permette invece di monitorare costantemente l'efficacia del modello con dati leggermente diversi da quelli usati in calibrazione e di fermare l'addestramento prima che il modello inizi a migliorare le proprie performance con i casi di training a detrimento di quelle con i casi di validazione. In tal modo la capacità di generalizzare può essere considerata una caratteristica costruttiva del modello.

Una seconda strategia (oltre all'*early stopping*) che viene spesso adoperata nell'addestramento dei modelli basati sui percettroni è il *jittering*: durante il training viene sistematicamente aggiunta ai valori dei descrittori una piccola quota di rumore bianco che propone all'algoritmo di addestramento una serie di variazioni marginali attorno alle condizioni ambientali effettivamente osservate; assumendo che a variazioni minime di tali condizioni corrispondano identiche strutture di comunità, questo metodo permette al modello in fase di calibrazione di non irrigidirsi attorno a

poche condizioni che eventualmente si ripetono, garantendo flessibilità e capacità di generalizzazione.

Sempre per prevenire l'*overtraining* è possibile anche non usare ad ogni epoca tutti i casi presenti nel training set, ma selezionarne ogni volta in modo random solo una parte, per far sì che ogni nuovo passo nell'addestramento del modello sia basato su un insieme sempre diverso di osservazioni.

Infine, la scelta di come misurare la performance dipende dalle variabili di output (che ad esempio possono essere continue, discrete o binarie) e dagli obiettivi del modello. Se le variabili di output sono continue, ad esempio, per ognuna di esse si può calcolare il coefficiente di correlazione di Bravais-Pearson tra i valori attesi e quelli predetti dal modello, ottenendo valori vicini ad 1 quando la rete fa un buon lavoro, vicini a 0 quando si comporta come un decisore casuale e vicini a -1 quando le previsioni sono sistematicamente invertite rispetto alle attese[37].

In questo lavoro le variabili di output esprimono la presenza o l'assenza delle varie specie in un certo sito di campionamento e pertanto sono di tipo binario (assumono cioè solo i due valori 1 e 0). Come si vedrà nel secondo paragrafo di questo capitolo, una misura opportuna in casi come questo è la statistica K di Cohen (1960), calcolata tramite una matrice di confusione; essa si comporta in modo non dissimile dal coefficiente di correlazione di Bravais-Pearson, variando tra 1, 0 e -1 a seconda che il modello preveda correttamente tutti gli 1 e gli 0, emetta 0 e 1 a caso o preveda sempre 1 al posto dello 0 e viceversa.

Tornando alla storia, nel 1989 Hornik et al. e Cybenco produssero ulteriori specificazioni delle dimostrazioni matematiche già compiute da Kolmogorov trent'anni prima, con cui divenivano sempre più chiare le potenzialità dei percettroni MLP nell'approssimare funzioni continue e discrete con qualunque numero di variabili indipendenti e dipendenti. Dagli anni '80, dunque, la ricerca sulle reti neurali e in particolare sui percettroni non si è più fermata, sviluppando modelli e applicazioni di vario genere, dal lettore automatico nelle casse dei supermercati, all'ABS nelle automobili, dal controllo di efficienza degli impianti, al monitoraggio nell'industria di precisione, dal riconoscimento di configurazioni complesse (es. volti, dermatoglifi, parlato, suoni, livree animali, caratteri scritti a mano, targhe di automobili in autostrada), alla modellizzazione di funzioni complesse in cui diverse

[37] Se ciò accadesse, ovviamente, basterebbe calcolare una funzione inversa dell'output del modello per avere previsioni corrette.

variabili indipendenti determinano il comportamento di molte variabili dipendenti (modelli di sistemi ecologici, economici, climatici, ecc).

La storia e la descrizione di questo tipo di modelli, e in particolare del percettrone multistrato, dovrebbe sfatare una serie di "miti" che in alcuni casi hanno ostacolato la loro diffusione come strumenti modellistici in ecologia:

1) sono modelli matematici con basi teoriche non meno fondate di quelle degli strumenti statistici tradizionali, di cui si fa largo uso. E' vero che gli algoritmi di calibrazione (come l'algoritmo EBP), pur ben definiti nella loro logica, funzionano a volte in un modo non del tutto chiaro, ma la ricerca in questo campo è aperta e affonda le radici in decenni di dimostrazioni matematiche circa le proprietà di questi modelli. Questo è molto interessante in ecologia: nei sistemi in cui si possa ipotizzare l'esistenza di relazioni funzionali di varia complessità tra variabili anche molto numerose, i modelli MLP costituiscono strumenti potenzialmente adeguati e particolarmente indicati a scopi previsionali;
2) non si vuole sostenere che le reti neurali, o simili tecniche di intelligenza artificiale, costituiscano una panacea per ogni problema modellistico: quando i dati rispettano le assunzioni richieste dai modelli tradizionali (es. regressione logistica), è senz'altro più indicato l'uso di questi ultimi. Inoltre, l'addestramento di una rete neurale necessità di una grande quantità di informazioni significative, il che rende questi strumenti inutili laddove queste condizioni non si verificano. Dunque, la scelta di modelli di questo tipo sottostà ai medesimi principi che valgono per altri: bisogna tenere conto di tutta una serie di questioni metodologiche e bisogna avere un'adeguata conoscenza dei sistemi ecologici da modellizzare; solo in questo modo è possibile sapere quanto ci si può fidare dei risultati ottenuti tramite un particolare modello;
3) i modelli MLP non sono macchine mangia-dati: l'uso di questi modelli richiede una grande competenza teorica. Come si diceva prima, le informazioni fornite alla rete devono essere numerose e significative; quindi solo la conoscenza dei sistemi in esame può permettere di decidere quanto il set di dati che si prevede di utilizzare possa contribuire alla costruzione di un modello affidabile. Dover suddividere il data set in sottoinsiemi il più possibile simili tra loro, richiede inoltre considerazioni teoriche su quali siano le variabili e le condizioni che meglio rappresentano

i fenomeni in esame; compiere in modo errato le operazioni di partizione[38] rischia di produrre addestramenti errati, perché condotti su casistiche non appropriate, o valutazioni errate della performance. Avendo a disposizione quantità molto grandi di osservazioni, una partizione casuale può dare buoni risultati, ma quando la quantità di dati è piccola rispetto alla dimensionalità del problema (cioè al numero di variabili e di relazioni in gioco), una partizione appropriata è decisiva. Si può aggiungere anche che non esiste un'unica partizione adeguata, ma che ve ne sono molte possibili e che ognuna può far sì che il modello colga con maggior risalto alcune caratteristiche piuttosto che altre. Ogni partizione dà una particolare visione del sistema in esame; è come presentare ad una persona la medesima foto, ma con regolazioni di contrasto, luminosità, colore, ecc. diverse: l'informazione è la stessa, ma il modo in cui viene presentata può far risaltare o scomparire intere parti della scena, nonché cambiare il senso generale della percezione. Dunque, anche qui la conoscenza teorica dell'ecologo è decisiva nel proporre all'algoritmo di addestramento la giusta "immagine" rispetto all'obiettivo modellistico che si desidera raggiungere. Vi è poi una serie di parametri, dal numero di unità nascoste, al tasso di apprendimento η, al *momentum* α, alla misura dell'errore, al numero di cicli di training da compiere, la cui regolazione ha molto a che fare con la natura di ciò che si studia o con il tipo di dati di cui si è in possesso;

4) infine, un percettrone multistrato non è *tout court* una black box: la famosa "opacità" di questi modelli dipende dalla forma *distribuita* della conoscenza al loro interno, che si esprime in una complessa matrice di pesi sinaptici, una forma che il cervello non riesce a leggere direttamente. Le funzioni approssimate dal percettrone restano difficili da scomporre e analizzare, per cui è problematico svelare il modo in cui operano e ricavarvi informazioni sul sistema in esame. Esistono però dei metodi (come l'analisi di sensibilità) con cui possiamo analizzare i rapporti tra singole variabili o gruppi di esse, per cui i modelli neurali, pur restando opachi alla codifica

[38] In questo lavoro il termine *partizione* (si veda anche Fig. 9) è usato per indicare la procedura di suddivisione del *data set* in sottoinsiemi funzionali all'addestramento e alla valutazione dei modelli. In genere con "partizione" si intende il risultato della procedura, cioè una particolare suddivisione ottenuta, ma data la frequenza con cui si dovrà fare qui riferimento alle operazioni di suddivisione del *data set*, si è scelto per semplicità di indicare con un termine unico sia la procedura che il suo prodotto.

analitica, offrono comunque la possibilità di "osservare" e misurare il modo in cui si relazionano le variabili al loro interno, gettando luce su ciò che accade nei sistemi che studiamo.

1.2 Obiettivi e ipotesi

Obiettivo di questo studio è l'esame sperimentale di alcune potenziali strategie di ottimizzazione di modelli predittivi del popolamento ittico dei fiumi basati su reti neurali artificiali. Ogni strategia si fonda su particolari criteri di addestramento o di interpretazione dei risultati. Per raggiungere tale obiettivo è stato necessario il passaggio per un obiettivo intermedio: lo sviluppo di un disegno sperimentale e del relativo software per generare le prove ed analizzare i risultati.

Il *modello base* utilizzato è un percettrone multistrato a tre strati: uno di input, uno nascosto e uno di output. Lo strato di input riceve, tramite n_d neuroni di input, altrettanti valori di descrittori ambientali su scale continue o discrete ordinali; ogni neurone dello strato di output restituisce poi le previsioni circa la presenza o l'assenza di una delle n_s specie del popolamento, su una scala continua che varia da 0 a 1. In Fig. 10a si vede come il singolo neurone di output, in funzione del potenziale d'azione calcolato in base agli input ricevuti (ascisse), restituisca il proprio valore di output (ordinate), determinato tramite una funzione di attivazione sigmoidale. Tale valore continuo può essere letto come una "probabilità di presenza", ma può anche essere reso binario tramite una funzione a soglia; poiché ci si aspetta che il modello dia output "vicini allo 0" quando la probabilità di presenza è bassa e "vicini all'1" quando è alta, la soglia tradizionale è quella che appare più naturale, cioè $\vartheta = 0.5$ (Fig. 10b).

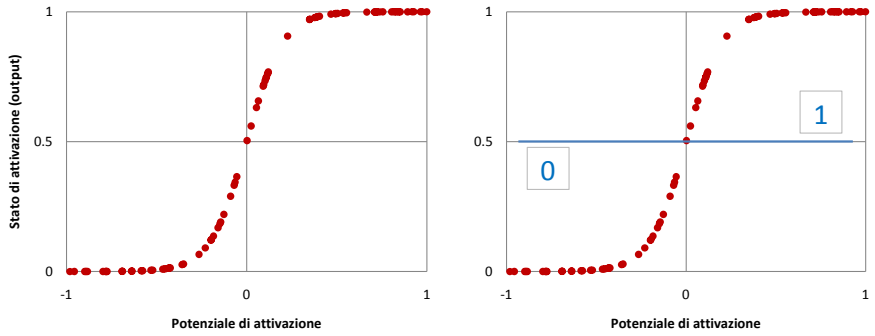


Fig. 10

In tal modo la struttura prevista del popolamento in un certo sito, esprimibile con un vettore n_s-dimensionale di valori binari $\bar{s}_p = (0,1,1,1,0,\ldots,1)$, può essere confrontata tramite una matrice di confusione[39], nella fase di test del modello, con la struttura $\bar{s}_o = (0,1,0,1,1,\ldots,0)$ che in quel sito si è osservata; da qui è poi possibile calcolare una misura di performance del modello stesso.

Una serie di riflessioni, basate sulla letteratura, ha permesso la formulazione di quattro ipotesi (più una da considerare ancillare) circa altrettanti potenziali metodi di ottimizzazione.

Ipotesi 1

Un'opportuna regolazione della soglia che discretizza l'output migliora la performance di un modello di previsione.

Ipotesi 2 (da considerare ancillare rispetto alla prima)

Il valore della soglia ottimale è legato alla frequenza della specie nel data set tramite una funzione non decrescente.

Un fatto noto e spesso riscontrato (Scardi et al., 2005) è la difficoltà di ottenere buone previsioni per specie relativamente rare. La matrice dei dati usati per l'addestramento (calibrazione) del modello ha un numero *r* di righe pari al numero

[39] Si veda in "Materiali e metodi".

delle osservazioni condotte e un numero *c* di colonne pari al numero totale delle variabili indipendenti e dipendenti considerate ($n_d + n_s$). Se prendiamo una colonna relativa ad una specie rara, gran parte delle sue *r* righe sarà occupata da zeri (cioè, da valori di assenza), che testimoniano il fatto che nella maggior parte dei siti campionati la specie non è stata osservata. Durante l'addestramento del modello, dunque, può accadere che i pochi valori 1 non siano sufficienti ad informare la rete circa le condizioni ambientali in cui la specie viene trovata; tanto più che in molti casi le specie rare non sono tali semplicemente in relazione alla rarità delle condizioni ecologiche che preferiscono. Se non esistono condizioni ambientali univoche per determinare la presenza di una specie rara, il modello avrà difficoltà ad approssimare una funzione che preveda la presenza della specie in base ai descrittori. Inoltre, al di sotto di una certa frequenza (cioè, proporzione di 1 su tutte le osservazioni condotte), anche se le condizioni ambientali fossero stringenti rispetto alla presenza della specie, la rete verrebbe comunque addestrata con un *bias* per cui l'output si avvicina sempre allo zero. Ciò è dovuto al fatto che la regola delta si troverà nella stragrande maggioranza dei casi ad orientare i pesi sinaptici in modo da ottenere uno 0 in output e dunque la rete verrà addestrata in base ad una sorta di "pregiudizio" per cui la specie rara è considerata genericamente assente; così la rete prevederà sempre in modo corretto l'assenza della specie, ma non sarà in grado di predirne la presenza.

Ricapitolando, sono due i fattori che ostacolano la previsione della presenza delle specie rare: frequenza troppo bassa dei casi di presenza e legame non sufficientemente forte con i descrittori ambientali. Se il legame con le condizioni ambientali è abbastanza forte, si può supporre che la "frequenza critica" al di sotto della quale l'addestramento non è più in grado di mettere la rete nelle condizioni di predire la presenza si abbassi; e si può supporre anche che ci sia un intorno del valore di frequenza critica all'interno del quale, spostandosi da valori più elevati a valori più bassi di frequenza, la rete passi da una netta capacità di orientare l'output verso l'1 o verso lo 0 nei casi rispettivamente di presenza o di assenza, ad una totale incapacità di farlo, che si traduce in un output costantemente orientato verso lo 0 (Fig. 11).

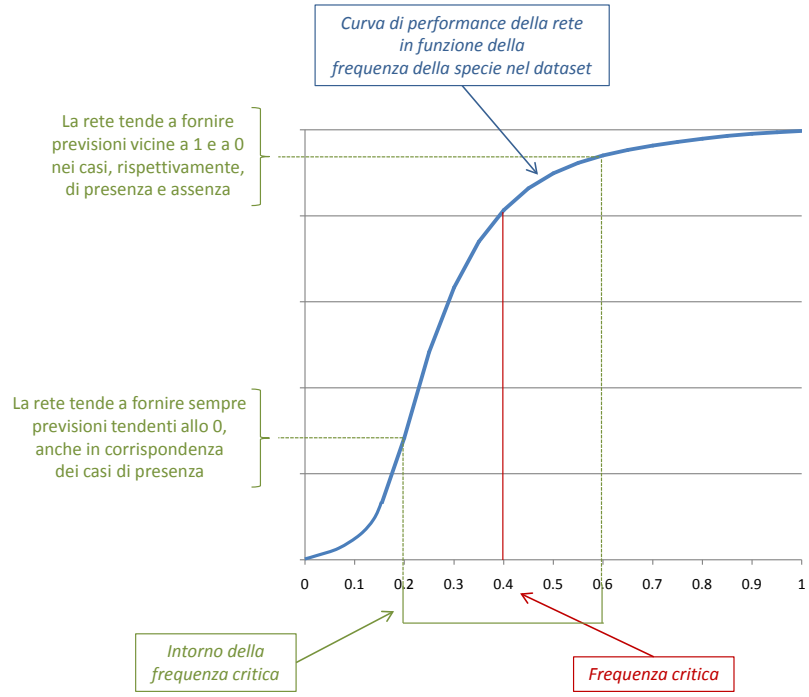


Fig. 11

Spazzando idealmente dal limite superiore a quello inferiore questo intorno, si vedrà in pratica "migrare" molte previsioni vicine all'1 verso lo 0.

Per visualizzare meglio ciò che si intende, si guardi innanzitutto Fig. 12, in cui sono rappresentati alcuni esempi di output nel caso di specie con frequenza intorno al 50%: l'asse orizzontale rappresenta l'output continuo della rete e ogni punto corrisponde ad una previsione[40]; in verde sono indicate previsioni a cui corrispondono osservazioni di assenza, mentre in rosso sono indicate previsioni a cui corrispondono osservazioni di presenza.

[40] Lo sfasamento verticale dei punti ha unicamente lo scopo di facilitare la visualizzazione.

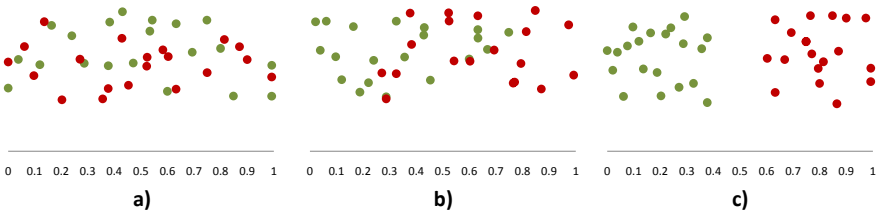


Fig. 12

Il caso rappresentato in 12a è quello di un decisore casuale, cioè di una rete addestrata con dati incoerenti o di qualità scadente, che emette previsioni su tutto il range da 0 a 1 sia per i casi in cui è stata osservata l'assenza (punti verdi) che per casi in cui è stata osservata la presenza (punti rossi). In 12b si ha la previsione di un modello abbastanza buono, che tendenzialmente predice con valori vicini allo 0 i casi di assenza e con valori vicini all'1 i casi di presenza; non vi sono infatti punti rossi troppo vicini allo 0, né punti verdi troppo vicini all'1, e i casi in cui le assenze hanno previsioni al di sopra di 0.5 o le presenze hanno previsioni al di sotto di 0.5 sono abbastanza pochi; un modello del genere è stato certamente addestrato su dati con buona qualità informativa. La Fig. 12c, mostra invece un modello eccellente, che concentra le previsioni di tutti e solo i casi di presenza verso l'1 e di tutti e solo i casi di assenza verso lo 0; la qualità dell'informazione utilizzata è qui molto elevata.

La Fig. 13 rappresenta ciò che accade, secondo la visione proposta dalle ipotesi 1 e 2, quando si considera una situazione analoga a quella vista in Fig. 12, ma con frequenze dei casi di presenza molto più basse di quelle dei casi di assenza (specie rare)[41].

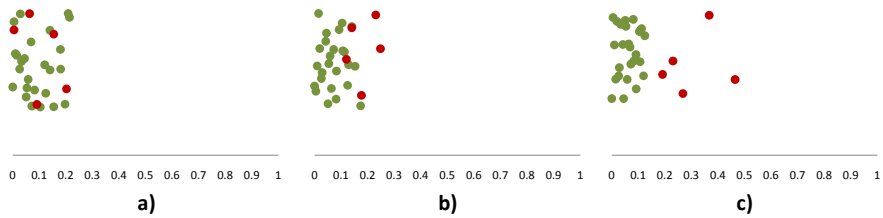


Fig. 13

[41] Per facilitare la visualizzazione dell'esempio, in figura si hanno cinque casi di presenza contro 25 di assenza, ma questo tipo di effetto si verifica con squilibri molto più pronunciati.

Nel passaggio da Fig. 12 a Fig. 13 si notano in pratica due cose: 1) nella seconda un *bias* tende a spostare tutte le previsioni nella direzione del caso più frequente (l'assenza); 2) la contrazione delle previsioni verso il basso dovuta al *bias* fa sì che il modello abbia difficoltà nell'avvicinare le previsioni dei casi di presenza al valore 1, anche quando la qualità delle relazioni di una specie con i descrittori ambientali è buona (Fig. 12c).

Per tornare a ciò che si illustrava con Fig. 11, dunque, finché ci si trova al di sopra di un valore critico della frequenza (che dipende dalla qualità dell'informazione disponibile), benché il modello non riesca più a tenere le previsioni dei casi di presenza nella metà destra del grafico (si confronti Fig. 12b e 13b o Fig. 12c e 13c) esso riesce tuttavia a tenere separati i valori di previsione dell'assenza e della presenza. Quando si va troppo al di sotto della frequenza critica, tutti i valori precipitano comunque verso lo 0.

La medesima idea è rappresentata in Fig. 14a, che mostra ciò che accade alle previsioni (rispetto ad una situazione come in Fig. 10) quando la frequenza si abbassa: molte scivolano al di sotto della soglia stabilita per interpretare l'output come una previsione di presenza.

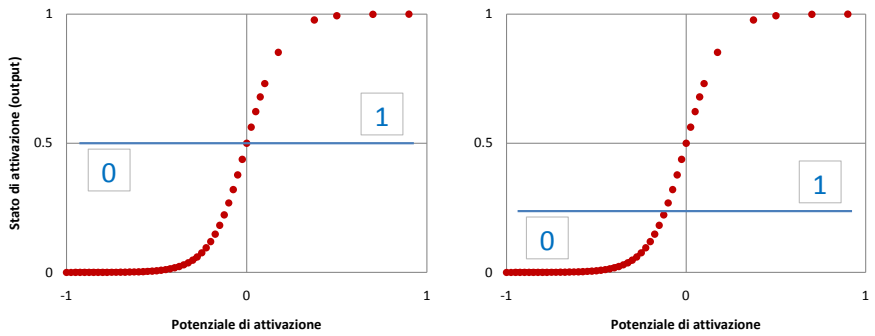


Fig. 14

Per questo si è pensato di studiare la possibilità di "inseguire" questa migrazione spostando verso il basso la soglia ϑ (Fig. 14b), cioè di adoperarsi in fase interpretativa per recuperare ad un giudizio di presenza casi che vi sarebbero sfuggiti con una soglia classica fissata a 0.5. E' chiaro che questa strategia può essere utile solo nei casi in cui è preservato un minimo di equilibrio tra frequenza della specie e

legame con i descrittori; il "recupero" al giudizio di presenza si basa infatti sull'assunzione che un valore di previsione tendente ad 1 in condizioni normali, si sposti verso lo 0 quando la frequenza della specie si avvicina a quella critica, ma resti comunque separato dalle previsioni di assenza (Fig. 13c, in cui si potrebbe usare efficacemente una soglia intorno 0.15). Se una specie è troppo rara e slegata dalle condizioni ambientali, il valore di previsione perde il suo legame con il sistema modellizzato e precipita semplicemente verso lo 0. Poiché tuttavia non possiamo conoscere a priori per nessuna specie l'entità di questo equilibrio o della contrazione che ci si deve attendere (anche perché essi sono legati non solo all'ecologia della specie, ma anche allo specifico set di dati utilizzato), allora può essere interessante tentare uno spostamento della soglia con tutte le specie[42] e confrontare poi i risultati così ottenuti con quelli relativi ad un soglia classica a 0.5. Per far questo è necessario trovare per ogni specie una soglia che ottimizzi le prestazioni del modello[43].

Il rapporto atteso tra la frequenza della specie e la soglia ottimale (ipotesi 2) può essere descritto in generale da una curva come quella riportata in Fig. 15.

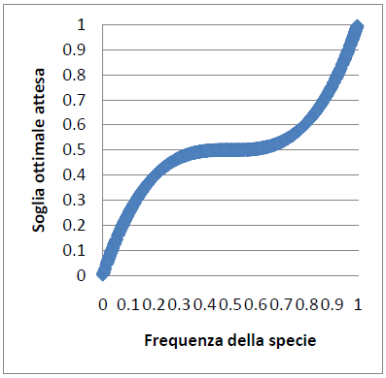


Fig. 15

[42] Nel contesto specifico del data set utilizzato, infatti, anche specie relativamente abbondanti potrebbero fornire alla rete informazioni insufficienti sulle proprie condizioni di presenza; pertanto non si può fissare a priori una frequenza critica al di là della quale non sottoporre le specie a questo tipo di indagine. Inoltre va rilevato che quanto detto finora per le specie rare, vale anche per le specie ubiquitarie, con le quali la rete presenterebbe il *bias* opposto (prevedrebbe sempre la presenza) e il problema della soglia si porrebbe in termini di aumento di quest'ultima rispetto a 0.5. In teoria solo le specie con una frequenza intorno al 50% non dovrebbero indurre un *bias* di questo tipo, ma a parte la difficoltà di fissare a priori un range attorno a 0.5 all'interno del quale escludere le specie dall'indagine, non è detto che la ricerca di una soglia θ ottimale non intercetti *bias* dovuti ad altri fattori; per questo si ritiene più corretto sottoporre tutte le specie al medesimo tipo di indagine.

[43] Nel prossimo capitolo sarà descritto l'algoritmo sviluppato per ottenere questo risultato.

Sempre ad un livello descrittivo, si può immaginare che questa curva tenda ad una retta laddove uno scostamento positivo o negativo della frequenza dal valore di 0.5 rappresenti una buona misura del decadimento della qualità informativa del data set rispetto al rapporto tra le caratteristiche ambientali e la presenza/assenza di una specie; allo stesso modo si può immaginare che il plateau della curva intorno a 0.5 si espanda quando la qualità dell'informazione contenuta nel data set resta elevata nonostante uno scostamento significativo della frequenza da 0.5.

Ovviamente ci si aspetta che le ipotesi 1 e 2 siano confermate solo in generale, perché per alcune specie il rapporto tra frequenza e qualità informativa del data set può essere in grado di determinare modelli che funzionano benissimo con una soglia a 0.5, anche quando la frequenza è ben più alta o più bassa di 0.5.

Ipotesi 3

Con la maggior parte delle specie la performance di modelli multispecie, con architettura $n_d - h - n_s$*, è migliore di quella di modelli monospecie, con architettura* $n_d - h - 1$*, ma con alcune specie è vero il contrario*[44].

Come si diceva nell'introduzione, la possibilità di costruire strumenti che predicano in modo integrato intere comunità è un traguardo importante in ecologia e lo sviluppo di modelli basati su reti neurali artificiali ha reso questo obiettivo sempre più vicino (Olden, 2001; Scardi et al. 2005; Olden et al., 2006). Tuttavia è indispensabile non perdere, nell'ambito della visione integrata, l'attenzione per le singole specie di cui la comunità si compone.

Dal punto di vista dei modelli è possibile addestrare percettroni che in base ai descrittori predicano il popolamento nel suo insieme (modelli con architettura $n_d - h - n_s$; Fig. 16a) o una specie singolarmente (modelli con architettura $n_d - h - 1$; Fig. 16b).

[44] Il valore *h* indica il numero di unità presenti nello strato nascosto.

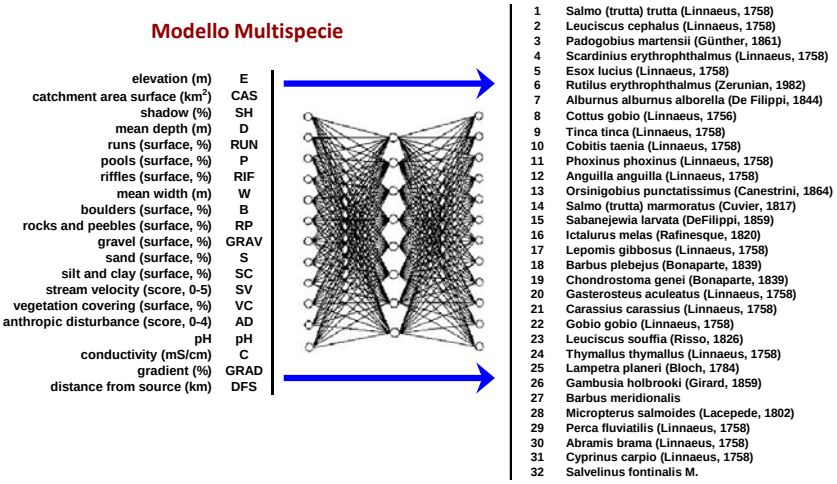


Fig. 16a

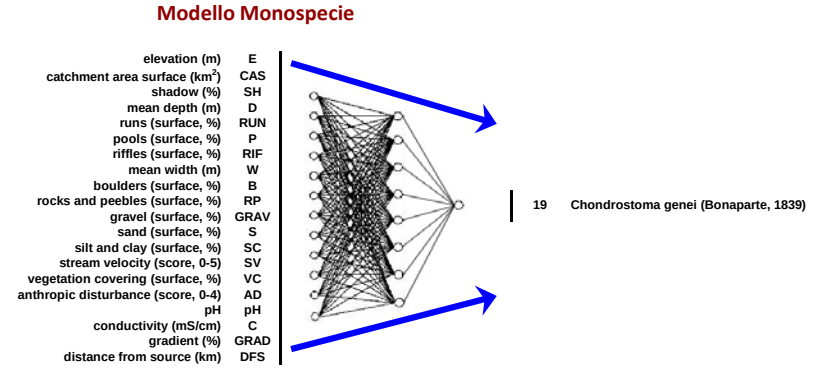


Fig. 16b

Nel primo caso la struttura del popolamento sarà data da un unico modello, mentre nel secondo dovremo costruire n_s modelli indipendenti.

Un percettrone è in grado di tenere conto delle interazioni tra le variabili di output, oltre che di quelle tra le variabili di input e può dunque prevedere un'intera comunità integrata, mostrandosi molto aderente alla realtà ecologica; per questa ragione ci si può attendere che la bontà di previsione per la maggior parte delle specie benefici di questa complessità. Un modello di questo tipo è in grado inoltre di

ridurre la dimensionalità dei problemi, cogliendo implicitamente l'andamento congiunto di diverse variabili e determinando così una riduzione del numero minimo di osservazioni necessarie per l'addestramento. In generale, dunque, ci aspettiamo che un modello multispecie funzioni meglio di n_s modelli monospecie. Tuttavia potrebbero verificarsi circostanze, legate all'ecologia delle specie o alle particolari condizioni rappresentate nel data set, per cui i modelli di previsione monospecie danno performance migliori con alcune specie: se una specie ha una forte autoecologia (è legata cioè strettamente a particolari condizioni ambientali) o è associata in modo ambiguo con altre specie oppure presenta frequenze nel data set vicine a quelle critiche, la previsione multivariata può disturbare la performance del modello; se invece una specie ha relazioni con il resto del popolamento che sono più informative delle relazioni con i descrittori ambientali, un modello multispecie avrebbe più *chances* di cogliere l'informazione significativa nel data set. In casi come questi, la scelta di un modello piuttosto che di un altro potrebbe segnare la differenza tra la possibilità o l'impossibilità di prevedere con accuratezza una certa specie. E' probabile, infine, che ci siano anche specie per le quali l'uno e l'atro modello diano performance non dissimili.

Ipotesi 4

La performance di modelli addestrati con partizione che distribuisce equamente i casi di presenza delle specie tra i subset è migliore di quelli in cui la distribuzione è casuale.

Questa ipotesi e la successiva hanno a che fare con i criteri di pre-elaborazione dei dati e in particolare con il modo di operare la partizione del data set nei subset di training, validazione e test. Come si diceva prima, infatti, è solo quando si ha a disposizione una grandissima quantità di dati molto rappresentativi dello spazio degli eventi osservabile che una distribuzione casuale dei dati ai tre subset dà luogo a tre insiemi omogenei di dati. Quando i dati sono relativamente pochi e frammentari, invece, può essere opportuno usare uno o più criteri per cercare di garantire un'equa distribuzione delle condizioni.

Una partizione casuale rischia, soprattutto nel caso di specie rare, di non distribuire in modo equo i casi di presenza (e quindi anche quelli di assenza). Può succedere, ad esempio, che il subset di training contenga solo casi di assenza, per cui la rete non ha modo di apprendere il legame tra le condizioni ambientali e la

presenza della specie; oppure potrebbero non esserci casi di presenza nel subset di test, rendendo impossibile una misura completa della performance della rete. Allora si può cercare di costruire i tre subset in modo che la frequenza dei casi di presenza in ognuno di essi corrisponda alla frequenza della specie nell'intero data set. L'attesa, secondo la quarta ipotesi, è che modelli addestrati su partizioni di questo tipo funzionino meglio di modelli addestrati su partizioni random.

Ipotesi 5

La performance di modelli addestrati con partizione che distribuisce omogeneamente la casistica relativa all'altitudine tra i subset è in genere migliore di quelli in cui la distribuzione è casuale.

In un caso semplificato all'estremo, in cui un modello compie delle predizioni in base ad un unico descrittore ambientale (lo si immagini continuo), distribuire omogeneamente la casistica presente nei dati fra i tre subset vuol dire essenzialmente far sì che in ogni subset siano presenti tutte le determinazioni (valori bassi, medi, alti) che il descrittore presenta nel data set. Ad esempio, se tale descrittore è la temperatura e il suo range di variazione nel data set è compreso tra 5 e 20 °C, bisogna evitare che nel training set siano totalmente assenti valori tra 5 e 12 °C, perché così il modello non verrebbe addestrato su una porzione importante della casistica osservata; se la stessa cosa avvenisse nell'insieme di validazione, non avremmo garanzie sulla capacità del modello di generalizzare in quel range di temperature, mentre se avvenisse nel test set non potremmo testare in modo completo la performance della rete.

Quando i descrittori sono molti le cose si complicano, perché non è possibile garantire un'equipartizione delle condizioni per tutte le variabili e le loro combinazioni. Se però accade che molte variabili significative covarino e che una di esse tenda ad integrare questo andamento comune, quest'ultima può essere scelta come criterio, procedendo come nel caso esemplificativo del descrittore singolo. Come già si diceva, negli ecosistemi fluviali si può contare su questa covariazione e un descrittore adatto ad integrare l'andamento degli altri può essere l'altitudine. Non è detto che tutte le specie beneficino di questa scelta, perché alcune potrebbero essere legate più fortemente a descrittori che non covariano con l'altitudine, ma l'attesa è che "in generale" si osservi un miglioramento nelle predizioni.

2. MATERIALI E METODI

Mettendo da parte per un istante l'ipotesi 2, che può essere considerata ancillare rispetto alla 1, ogni ipotesi propone un confronto tra due determinazioni alternative di un'ideale variabile binaria: soglia tradizionale (0.5) / soglia ottimizzata (ipotesi 1), previsione monospecie / previsione multispecie (ipotesi 3), partizione random / partizione con equa distribuzione dei casi di presenza e assenza tra i subset (ipotesi 4), partizione random / partizione con assegnazione di una casistica omogenea rispetto all'altitudine tra i subset (ipotesi 5).

Per ogni specie è dunque possibile predisporre un disegno sperimentale in grado di confrontare le 2^4=16 condizioni, che corrispondono alle combinazioni possibili tra le varie determinazioni delle variabili di ipotesi; ad ogni condizione corrisponde una variante di modello di previsione da valutare.

Ognuna delle 16 condizioni sperimentali viene esplorata tramite una serie di addestramenti di una rete neurale[45] (prove sperimentali), ottenendo una distribuzione di valori di performance che può essere poi confrontata con le altre.

2.1 Il *data set*

Poiché il target principale che ci si proponeva era lo sviluppo di un metodo e del relativo software per realizzare gli esperimenti previsti dal disegno sperimentale, si è scelto di non includere nel progetto di ricerca un'attività originale di raccolta dati, ma di usare piuttosto un data set già disponibile e noto (Scardi et al., 2005).

I siti di campionamento da cui provengono i dati sono relativi ai tratti montani e pedemontani dei corsi d'acqua presenti nelle province di Vicenza e Belluno (Fig. 17).

[45] Le caratteristiche generali del tipo di rete neurale utilizzato sono state descritte all'inizio del paragrafo 2 del capitolo introduttivo; le varianti specifiche di ogni addestramento sono invece fissate dalla particolare combinazione di determinazioni delle variabili di ipotesi che definisce la singola condizione sperimentale.

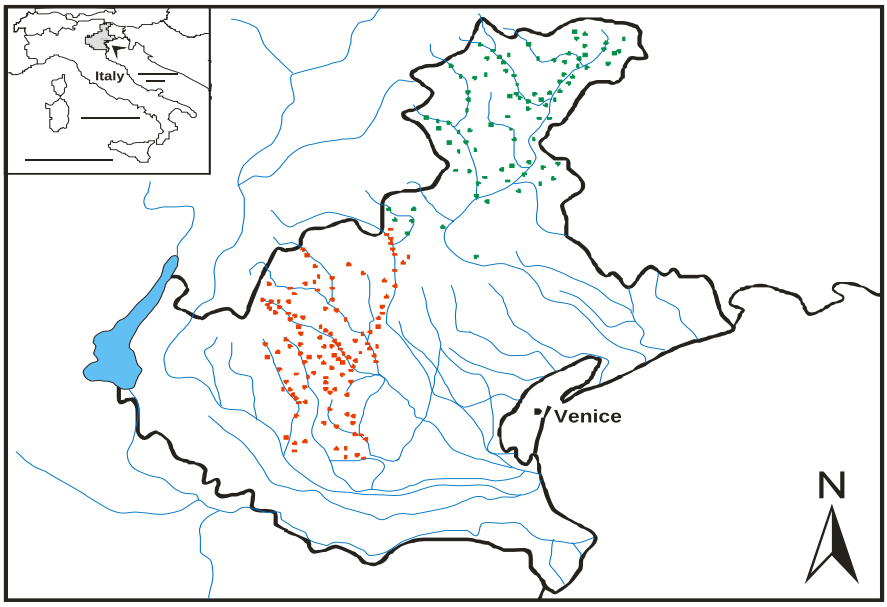


Fig. 17

Il data set acquisito consiste di $r = 264$ osservazioni (records) relative a due gruppi di variabili. Il primo gruppo è costituito da 20 (n_d) descrittori ambientali, utilizzati come variabili predittive (Tab. 1), che assumono in alcuni casi valori continui (es. altitudine), mentre in altri sono codificate secondo una scala ordinale (es. disturbo antropico).

ID	Descrittori	Unità di misura
1	Altitudine	*m*
2	Profondità media	*m*
3	Correnti	*% superficie*
4	Pozze	*% superficie*
5	Raschi	*% superficie*
6	Larghezza media	*m*
7	Massi	*% superficie*
8	Sassi e ciottoli	*% superficie*
9	Ghiaia	*% superficie*
10	Sabbia	*% superficie*
11	Peliti	*% superficie*
12	Velocità flusso	*punteggio 0-5*
13	Copertura vegetale	*% superficie*
14	Ombreggiatura	*%*
15	Disturbo antropico	*punteggio 0-4*
16	Acidità dell'acqua	*pH*
17	Conducibilità	*mS/cm*
18	Gradiente	*%*
19	Superficie del bacino versante	km^2
20	Distanza dalla sorgente	*km*

Tab. 1

Il secondo gruppo di variabili è quello delle 32 (n_s) specie (vairabili predette), che sono tutte di tipo binario, assumendo valore 1 quando la specie è presente in un certo sito e 0 quando non lo è (Tab. 2).

ID	Specie	Nome italiano	Codice sintetico
1	*Salmo (trutta) trutta* (Linnaeus, 1758)	Trota fario	Sat
2	*Leuciscus cephalus* (Linnaeus, 1758)	Cavedano	Lec
3	*Padogobius martensii* (Günther, 1861)	Ghiozzo padano	Pam
4	*Scardinius erythrophthalmus* (Linnaeus, 1758)	Scardola	Sce
5	*Esox lucius* (Linnaeus, 1758)	Luccio	Esl
6	*Rutilus erythrophthalmus* (Zerunian, 1982)	Triotto	Rue
7	*Alburnus alburnus alborella* (De Filippi, 1844)	Alborella	Ala
8	*Cottus gobio* (Linnaeus, 1756)	Scazzone	Cog
9	*Tinca tinca (* Linnaeus, 1758)	Tinca	Tit
10	*Cobitis taenia* (Linnaeus, 1758)	Cobite	Cot
11	*Phoxinus phoxinus* (Linnaeus, 1758)	Sanguinerola	Php
12	*Anguilla anguilla* (Linnaeus, 1758)	Anguilla	Ana
13	*Orsinigobius punctatissimus* (Canestrini, 1864)	Panzarolo	Orp
14	*Salmo (trutta) marmoratus* (Cuvier, 1817)	Trota marmorata	Sam
15	*Sabanejewia larvata* (DeFilippi, 1859)	Cobite mascherato	Sal
16	*Ictalurus melas* (Rafinesque, 1820)	Pescegatto	Icm
17	*Lepomis gibbosus* (Linnaeus, 1758)	Persico sole	Leg
18	*Barbus plebejus* (Bonaparte, 1839)	Barbo	Bap
19	*Chondrostoma genei* (Bonaparte, 1839)	Lasca	Chg
20	*Gasterosteus aculeatus* (Linnaeus, 1758)	Spinarello	Gaa
21	*Carassius carassius* (Linnaeus, 1758)	Carassio	Cac
22	*Gobio gobio* (Linnaeus, 1758)	Gobione	Gog
23	*Leuciscus souffia* (Risso, 1826)	Vairone	Les
24	*Thymallus thymallus* (Linnaeus, 1758)	Temolo	Tht
25	*Lampetra planeri* (Bloch, 1784)	Lampreda di ruscello	Lap
26	*Gambusia holbrooki* (Girard, 1859)	Gambusia	Gah
27	*Barbus meridionalis (* Risso, 1826)	Barbo canino	Bam
28	*Micropterus salmoides* (Lacepede, 1802)	Persico trota	Mis
29	*Perca fluviatilis* (Linnaeus, 1758)	Persico reale	Pef
30	*Abramis brama* (Linnaeus, 1758)	Abramide	Abb
31	*Cyprinus carpio* (Linnaeus, 1758)	Carpa	Cyc
32	*Salvelinus fontinalis* (Mitchell, 1814)	Salmerino	Saf

Tab. 2

La matrice dei dati che ne risulta è dunque composta da 264 righe (records), ognuna corrispondente ad una osservazione, e 52 colonne (fields), ognuna corrispondente ad una misura effettuata.

Nell'elenco presentato le specie sono state ordinate in modo non crescente secondo la loro frequenza nel *data set*; si tenga dunque presente che la numerazione progressiva loro assegnata corrisponde a tale ordine (Fig. 18)

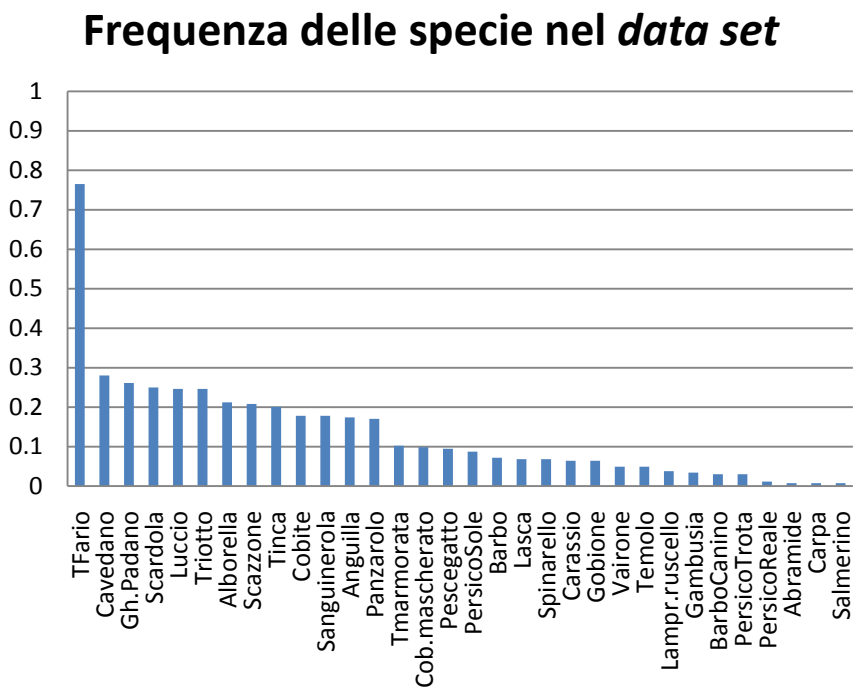


Fig. 18

I dati non sono stati raccolti originariamente per scopi scientifici, ma per la realizzazione delle *carte ittiche provinciali*[46]; essi tuttavia rappresentano bene la varietà delle condizioni ambientali e dei popolamenti ittici nelle aree campionate[47].

L'osservazione delle specie presenti nei vari siti è avvenuta tramite tecniche di *electrofishing*, eventualmente supportate da reti nel caso di siti caratterizzati da ampia larghezza dell'alveo. Si tratta di metodi che garantiscono con buona efficacia che tutte le specie presenti vengano rilevate; il fatto, poi, che il dato raccolto sia di tipo binario (presenza/assenza) e non, ad esempio, legato all'abbondanza, riduce ulteriormente l'incertezza. Infine, per tenere conto delle variazioni stagionali, alcune

[46] Secondo le Linee Guida dell'Associazione Italiana Ittiologi d'Acque Dolci, citate in Gelosi & Colombari (2004), "l'obiettivo della CARTA ITTICA è di pianificare la gestione razionale dell'ittiofauna e delle attività alieutiche", [...] in base a "valutazioni tecnico-scientifiche quantitative, verificabili e migliorabili nel tempo. Le carte ittiche studiano lo stato dell'ittiofauna in relazione alle caratteristiche ambientali delle zone umide ad acque correnti permanenti".

[47] La raccolta dei dati è stata compiuta da Aquaprogram S.r.l. e Bioprogram S.c.r.l.

delle 264 osservazioni sono state condotte nel medesimo sito in momenti diversi dell'anno.

Per visualizzare le associazioni tra le specie, le ultime 32 colonne della matrice dei dati (valori di presenza/assenza delle specie) sono state sottoposte ad ordinamento non metrico (NMDS) in base all'indice di Fager & McGowan (1963), elaborato specificamente per lo studio delle associazioni tra specie. L'indice è calcolato così[48]:

$$FM = \frac{a}{\sqrt{(a+b)(a+c)}} - \frac{1}{2\sqrt{max[(a+b),(a+c)]}}$$

e ha la caratteristica di dare particolare importanza alle associazioni positive tra i casi di presenza (*a*) piuttosto che alle associazioni positive tra i casi di assenza (*d*), il che riduce il rischio di ottenere valori troppo elevati nel caso di specie rare (che nella maggior parte dei siti sono contemporaneamente assenti). Include inoltre un fattore di correzione che è tanto più grande quanto meno frequente è la specie meno rara tra le due confrontate: anche questo accorgimento riduce il rischio di ottenere valori troppo elevati nel caso di specie rare.

Il grafico NMDS (Fig. 19) è stato ottenuto eliminando dall'analisi le 4 specie più rare del data set, poiché l'informazione disponibile era insufficiente a collocarle correttamente nel quadro del popolamento.

[48] Si veda Fig. 24 per comprendere la notazione basata sulla matrice di confusione.

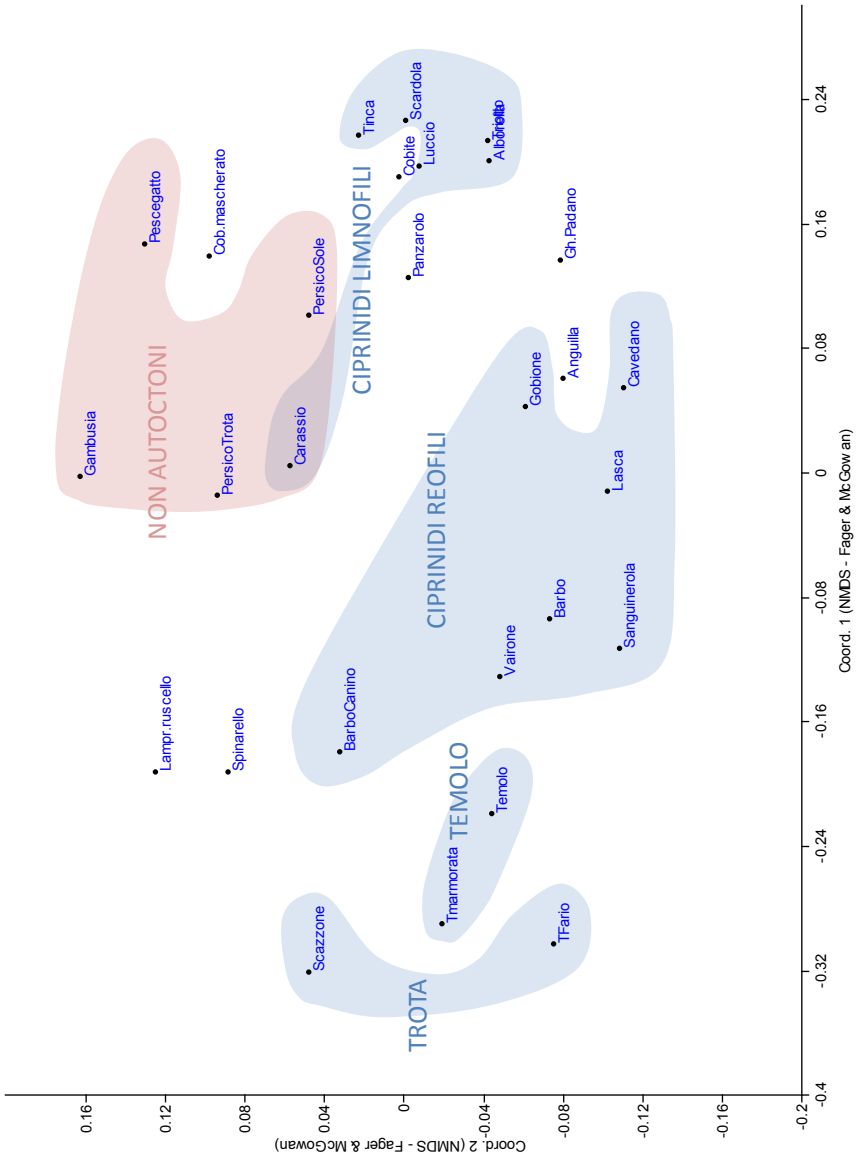


Fig. 19

Il grafico conferma le associazioni che a grandi linee ci si attendeva. In azzurro sono state evidenziate le specie più rappresentative dei gruppi legati alla zonizzazione classica (lungo la coordinata 1 si sviluppa chiaramente un gradiente

monte-valle). Alcune di queste specie (tra cui lo spinarello, la lampreda, il ghiozzo padano, il panzarolo, il luccio) possono essere legate anch'esse in modo tipico alla zonizzazione tradizionale o ritrovarsi in ambienti particolari come quelli di risorgiva insieme ad altre specie. L'anguilla fa un po' caso a sé: è una specie ampiamente euriecia e mobile che è possibile trovare quasi dappertutto lungo l'asta fluviale. In rosso sono stati evidenziati quei pesci che, pur non autoctoni, fanno oramai parte dei popolamenti ittici di molti fiumi italiani.

2.2 Disegno sperimentale

L'insieme delle prove previsto dal disegno sperimentale è stato generato incrociando le alternative proposte dalle ipotesi. Per prime sono state prese in considerazione le ipotesi 4 (H4) e 5 (H5), che definiscono specifiche modalità di partizione dei dati e implicano dunque una serie di operazioni da compiere prima dell'addestramento vero e proprio. Una tabella a doppia entrata rappresenta l'incrocio delle alternative legate alle ipotesi 4 e 5 per la generica specie X (Fig. 20).

SPECIE X

Ipotesi 5

Ipotesi 4	1_{H5}	0_{H5}	
1_{H4}	*n*	*n*	*2n*
0_{H4}	*n*	*n*	*2n*
	2n	*2n*	*4n*

a)

Ipotesi 5

Ipotesi 4	1_{H5}	0_{H5}	
1_{H4}	*72*	*72*	*144*
0_{H4}	*72*	*72*	*144*
	144	*144*	*288*

b)

Fig. 20

Sono quattro le possibili combinazioni:

- partizione per valori di presenza e di altitudine [1_{H4}-1_{H5}];
- partizione per valori di presenza, ma random per l'altitudine [1_{H4}-0_{H5}];

- partizione random per i valori di presenza, ma non per l'altitudine [0_{H4}-1_{H5}];
- partizione random sia per i valori di presenza che per l'altitudine [0_{H4}-0_{H5}].

Poiché per ognuna di queste quattro combinazioni esistono molte possibili partizioni dei dati, ognuna delle quali propone al modello in addestramento una "visione" dell'informazione disponibile leggermente diversa dalle altre, è utile generare una popolazione di *n* partizioni per ogni combinazione, a cui corrisponderà l'addestramento di altrettanti modelli (prove sperimentali). Il risultato ottenuto su una singola partizione, infatti, può dar luogo a un valore molto basso o molto elevato di performance solo per effetto del caso, in dipendenza di un particolare disporsi dei dati nei tre subset che ne disturba o ne facilita la previsione[49]. Con una distribuzione di misure di performance per ogni condizione è invece possibile valutare i risultati su un istogramma (ricavandone eventualmente indici sintetici, come la media o la mediana) e confrontare poi tramite test statistici le distribuzioni relative alle diverse condizioni.

Nel capitolo successivo si vedrà che per ogni condizione sono state ricavate, tramite due diversi algoritmi, 72 partizioni diverse (Fig. 20b), per un totale di 288 partizioni generate in base ad ogni singola specie. E' necessario costruire una tabella di questo tipo per ogni specie, poiché l'ordinamento (partizione) per i valori di presenza/assenza ha senso solo quando si considera una specie singola. Un'ulteriore tabella di questo tipo è stata inoltre generata utilizzando, al posto dell'ordinamento per una specie, quello per la "ricchezza di specie" complessiva. Dunque si hanno $n_s + 1 = 33$ tabelle, con 288 partizioni ciascuna, per un totale di $288 \times (n_s + 1) = 9504$ partizioni su cui addestrare i modelli, per sottoporre a verifica le ipotesi 4 e 5.

Le 72 partizioni contenute in ogni cella sono state costruite sulla base di algoritmi che ne garantiscono complessivamente l'omogeneità (tra di loro e con l'intero data set) e la capacità di esplorare lo spazio delle diverse varianti di partizione ottenibili sotto un certo criterio. Questo fa sì che la distribuzione delle misure di performance che si ottengono per ogni condizione sperimentale dia una visione realistica delle prestazioni che ci si può aspettare in quelle condizioni.

In realtà, da ogni gruppo di partizioni relative ad una condizione non si ottiene una distribuzione di valori, ma quattro; ciò perché si deve tenere conto anche delle ipotesi 3 e 1.

[49] Ad esempio, per effetto del caso il *training set* e il *validation set* potrebbero risultare leggermente più ricchi di informazione rispetto al *test set*, per cui la misura di performance verrebbe condotta su una sottocasistica di quella utilizzata durante il training, tendendo a dare valori relativamente alti.

Innanzitutto su ogni partizione va saggiato sia il modello monospecie (0_{H3}) che quello multispecie (1_{H3}), da cui discende che per ogni condizione si avranno due distribuzioni di 72 risultati; ma poiché dobbiamo confrontare anche le diverse performance ottenute tramite una soglia tradizionale a 0.5 (0_{H1}) e una ottimizzata (1_{H1}), per ogni modello addestrato avremo due diversi risultati, giungendo così, per ognuna delle 9504 partizioni, ad avere quattro misure di performance. Questo insieme di misure, va poi ottenuto per ogni singola specie.

Il modello multispecie, come già si diceva, presenta un'architettura di tipo $n_d - h - n_s$, che nel caso del nostro data set diventa $20 - h - 32$. Ciò implica un modello che in base a 20 descrittori ambientali predice i valori di presenza/assenza di 32 specie[50].

Addestrato su ogni partizione, il modello multispecie dà 32 valori di performance, uno per ogni specie. Per ottenere i medesimi 32 valori con i modelli monospecie, è invece necessario addestrare 32 diversi modelli sulla stessa partizione. Quindi, il modello multispecie richiede 9504 training in tutto, mentre con i modelli monospecie (incluso quello che predice la ricchezza di specie) è necessario condurre 9504 training per ognuno.

In realtà, questo insieme corposo di training è stato pensato anche in previsione di successivi approfondimenti, per cui non tutti i risultati ottenuti sono stati poi utilizzati in questo lavoro. La parte relativa alla ricchezza di specie, ad esempio, non è stata considerata. E non sono state considerate nemmeno le previsioni di una specie quando la partizione era stata compiuta in base ai valori di presenza/assenza di un'altra specie.

In sintesi, per ogni partizione utilizzata si sono ottenuti due modelli per specie: uno multispecie e uno monospecie. Da ognuno di questi modelli si sono poi ottenute due misure di performance: una con soglia tradizionale e una con soglia ottimizzata. Dunque, quattro valori di performance (corrispondenti alle alternative valutate dalle ipotesi 3 e 1) per ogni partizione. Poiché tutte le partizioni sono suddivise in 4 in base alle condizioni definite secondo le ipotesi 4 e 5, si chiude così il cerchio delle 16 combinazioni previste dal disegno sperimentale.

I risultati sperimentali effettivamente utilizzati, alla fine, sono stati 323136.

[50] *h* è il numero di unità nascoste, di cui si dirà tra breve.

2.3 Modello base e procedure associate

Il modello base di rete neurale utilizzato per lo sviluppo dei singoli modelli sperimentali è un percettrone multistrato a tre strati, cioè una rete *feed-forward* a tre strati totalmente connessi, con architettura 20-17-32 e 20-17-1 (rispettivamente, per i modelli multispecie e monospecie) e con addestramento basato su un algoritmo EBP.

Il numero di unità nascoste è fondamentale nel determinare le capacità di approssimazione di un modello (Bishop, 1995; Özesmi et al. 2006). Benché esistano dei metodi per determinare in modo analitico il numero di unità nascoste adeguato per una certa architettura[51], nella maggior parte dei casi, sviluppando un modello di previsione che si desidera ottimizzare, si compiono delle prove empiriche variando tale numero fino a determinare quello che fornisce i migliori valori di performance. Un numero troppo basso di *hidden units* dà un *bias* elevato nell'approssimazione della funzione desiderata (cioè, non permette al modello di rappresentare adeguatamente la complessità delle relazioni modellizzate), mentre un numero troppo elevato tende a far aumentare la variabilità delle predizioni a causa dell'*overfitting* con i dati di training (Geman et al., 1992; Olden et al., 2008). Poiché il numero di modelli da addestrare e confrontare in questo lavoro era considerevole, si è ritenuto opportuno scegliere per semplicità un unico numero di unità nascoste, $h = 17$, calcolato in modo empirico da Scardi et al. (2005) lavorando su modelli multispecie e sul medesimo data set. Alcune prove preliminari hanno permesso di rilevare anche sui modelli monospecie variazioni complessivamente non significative delle performance in seguito a regolazioni alternative del numero di unità nascoste.

L'addestramento è stato condotto tramite un algoritmo EBP con *early stopping*, *jittering* (con rumore bianco a valori pescati nell'intervallo [-0.01,0.01]) e selezione casuale dei dati su cui di volta in volta condurre il training. All'inizio del training i pesi sinaptici venivano inizializzati casualmente con valori entro un range compreso tra $-\frac{1}{\sqrt{n_i+1}}$ e $+\frac{1}{\sqrt{n_i+1}}$ (cioè tra -0.22 e 0.22 circa). Il numero massimo di epoche era fissato a 10000, per cui ogni volta che il training veniva interrotto dall'*early*

[51] Uno di questi metodi si basa, ad esempio, sulla regola empirica che il numero dei dati necessari all'addestramento di un percettrone multistrato dovrebbe corrispondere a circa 10 volte il numero di parametri (cioè, pesi sinaptici) presenti nel modello. Con un ragionamento inverso, dato un certo numero di dati a disposizione, il numero delle unità nascoste può essere scelto per far sì che il numero di connessioni presenti nella rete corrisponda a circa un decimo dei dati.

stopping, il processo ricominciava daccapo, a partire dall'inizializzazione casuale dei pesi sinaptici, e questo ciclo veniva ripetuto fino al raggiungimento delle 10000 epoche complessive, salvando alla fine la configurazione di pesi sinaptici che aveva dato l'errore minore.

L'applicativo per la conduzione del training, batchnnc.exe, è stato elaborato dal Prof. M. Scardi in Fortran 90 e utilizzato in modo automatico e reiterato tramite un sistema di controllo che verrà descritto nel capitolo successivo. Il programma richiede un file dati (*.tra) formattato in un certo modo (con i casi predisposti per l'*early stopping* e le indicazioni circa il numero di variabili predittive e predette) e un file di configurazione (*.cfg) che istruisca l'algoritmo di training su come leggere il file dati e su che parametri utilizzare per l'addestramento (Fig. 21).

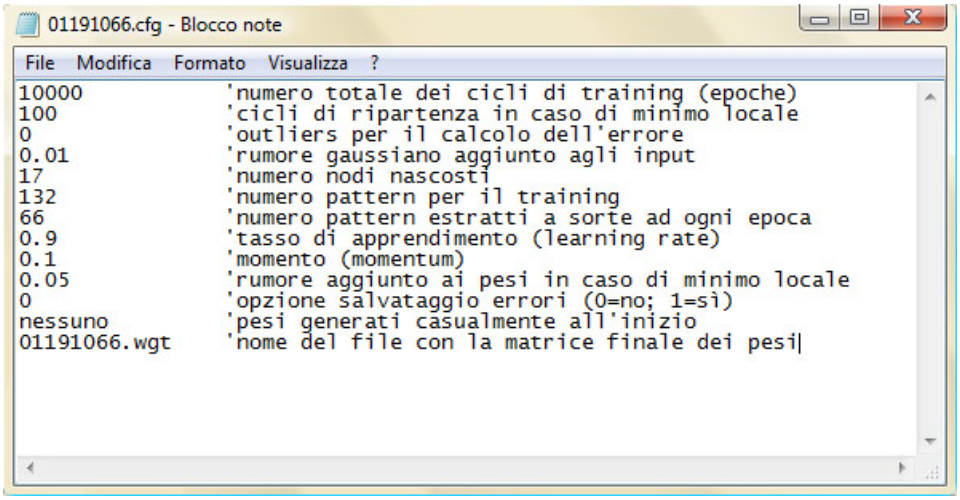


Fig. 21

L'output di batchnnc.exe è un file (*.wgt) con la matrice dei pesi sinaptici calibrati.

Un secondo applicativo elaborato dal Prof. Scardi, nnout4.exe, veniva usato per eseguire il modello predittivo addestrato[52]. Esso riceve in input il file dei pesi (*.wgt) prodotto da batchnnc.exe e un file dati (*.tes) con il *test set* (ottenuto con approccio *split-sample*) e restituisce i valori predittivi da confrontare con quelli osservati tramite un nuovo file (*.out). La Fig. 22 mostra un esempio con un vettore di osservazioni e il corrispondente vettore delle previsioni, che deve essere reso a valori binari per poter essere confrontato con le osservazioni.

Osservazioni contenute nel file di test	Previsioni fornite da nnout4.exe	Previsioni rese binarie tramite una soglia fissata a 0.5
0	0.076	0
0	0.1901	0
1	0.9133	1
0	0.5865	1
0	0.2795	0
0	0.4969	0
1	0.6715	1
1	0.8876	1
0	0.0922	0
1	0.8857	1
0	0.7168	1
1	0.726	1
1	0.8001	1
1	0.7421	1
1	0.7351	1
0	0.3192	0
0	0.4421	0
0	0.3238	0
1	0.45	0

Fig. 22

[52] "Eseguire il modello" vuol dire calcolare, per ogni osservazione (riga) nel *test set*, il valore di

$$y_k = \varphi_o\left[\beta_k + \sum_j w_{jk}\,\varphi_h\left(\beta_j + \sum_i w_{ij}\,x_i\right)\right]$$

in cui y_k è lo stato di attivazione del k-esimo neurone di output, corrispondente alla previsione relativa alla k-esima specie, φ_o e φ_h sono, rispettivamente, le funzioni di attivazione dei neuroni dello strato di output e di quello nascosto, β_k e β_j sono le costanti di bias associate allo strato di output e a quello nascosto, w_{jk} è il peso della connessione tra il k-esimo neurone di output e il j-esimo neurone dello strato nascosto, w_{ij} è il peso della connessione tra il j-esimo neurone dello strato nascosto e l'i-esimo neurone di input e x_i è lo stato di attivazione dell'i-esimo neurone di input, dipendente direttamente dall'i-esimo predittore considerato.

Anche nnout4.exe funzionava all'interno di una procedura automatica.

Le procedure automatiche elaborate in questo lavoro sono state scritte in FreeBASIC[53] in ambiente di programmazione FBIde[54] e hanno permesso di generare i file necessari a condurre le prove previste tramite batchnnc.exe e nnout4.exe, nonché di analizzare poi i risultati sia dal punto di vista descrittivo che con l'uso di test statistici. Nel prossimo capitolo si entrerà maggiormente nei particolari relativi agli algoritmi sviluppati.

I file prodotti come output sono stati poi ulteriormente analizzati tramite PAST 2.0[55] (Hammer et al., 2001) e Microsoft Excel.

2.4 Misura della *performance*

Come accennato nel capitolo introduttivo, la scelta del metodo per misurare la performance di un modello predittivo dipende sia dal tipo di variabile di output (continua, discreta, binaria) che dagli obiettivi della previsione.

L'output dei modelli utilizzati in questo lavoro deve essere valutato come binario, nonostante la rete neurale restituisca in realtà un valore continuo tra 0 e 1. Indicando semplicemente le due condizioni possibili di presenza (1) e assenza (0) di una specie in un certo sito, i dati di partenza, su cui la rete viene addestrata, sono binari e l'output predittivo deve dunque essere "interpretato" in modo binario tramite una soglia al di sotto della quale leggere 0 e al di sopra della quale 1.

Dato un *test set* con r_t osservazioni, la lettura di una delle colonne relative alle specie darà un vettore di r_t valori binari osservati che possono essere confrontati con quelli predetti. Per ogni singola previsione le possibilità sono quattro: a) si ottiene 1 quando anche il dato osservato è 1; b) si ottiene 1 quando il dato osservato è 0; c) si ottiene 0 quando il dato osservato è 1; d) si ottiene 0 quando anche il dato osservato è 0 (in Fig. 23[56] è riportato un esempio in cui r_t=20).

[53] Erede del vecchio QBasic, FreeBASIC è un linguaggio di programmazione utilizzabile tramite un compilatore dotato di licenza free/open source (GPL), che viene gratuitamente e continuamente aggiornato e che è utilizzato da una vasta comunità di programmatori (http://www.freebasic.net/).

[54] http://fbide.freebasic.net/.

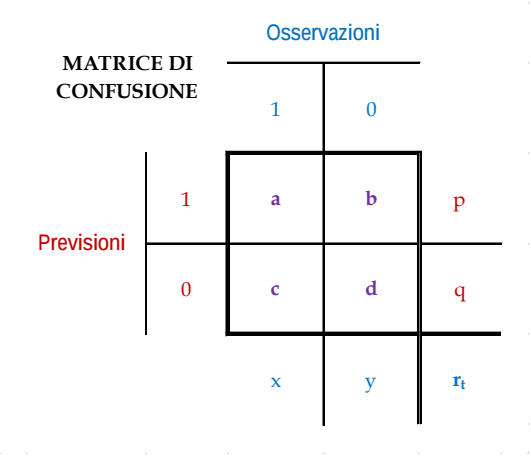
[55] http://folk.uio.no/ohammer/past/.

[56] In tutte le immagini legate alla matrice di confusione si è scelto di adottare un codice visuale che rappresenta in blu i valori e le grandezze dipendenti esclusivamente dalle osservazioni, in rosso i valori e le grandezze dipendenti esclusivamente dalle previsioni fornite dai modelli e in viola i valori e le grandezze determinati dal confronto tra valori osservati e predetti.

Valori osservati	Valori predetti	a	b	c	d
0	1	0	1	0	0
0	0	0	0	0	1
1	1	1	0	0	0
0	0	0	0	0	1
0	0	0	0	0	1
0	1	0	1	0	0
0	1	0	1	0	0
0	1	0	1	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	0	0	0	1	0
0	1	0	1	0	0
0	0	0	0	0	1
1	1	1	0	0	0
0	1	0	1	0	0
1	1	1	0	0	0
0	0	0	0	0	1
1	1	1	0	0	0
1	1	1	0	0	0
1	0	0	0	1	0
		7	6	2	5

Fig. 23

Contando i casi relativi ad ognuna di queste quattro condizioni è possibile riempire le quattro celle di una tabella di contingenza 2x2, detta anche *matrice di confusione* (Fig. 24), uno strumento particolarmente adatto a rappresentare la relazione tra due variabili binarie.



a
numero casi **TP**
[True Positive - correttamente predetti come positivi]

b
numero casi **FP**
[False Positive - erroneamente predetti come positivi]

c
numero casi **FN**
[False Negative - erroneamente predetti come negativi]

d
numero casi **TN**
[True Negative - correttamente predetti come negativi]

x: **a+c** [totale dei casi di presenza osservati]
y: **b+d** [totale dei casi di assenza osservati]

p: **a+b** [totale dei casi di presenza predetti]
q: **c+d** [totale dei casi di assenza predetti]

r_t: **a+b+c+d** [numero totale casi osservati]

Prevalenza
(Prevalence)
x/r_t
frequenza dei casi positivi osservati

Indici derivati dalla matrice di confusione

Tasso di classificazione corretta (CCR , Correct Classification Rate)
$(a+d)/r_t$ → proporzione assegnazioni corrette

Sensibilità (Sensitivity)
a/x → proporzione di casi positivi correttamente identificati

Specificità (Specificity)
d/y → proporzione di casi negativi correttamente identificati

Valore Predittivo Positivo (PPP, Positive Predictive Power)
a/p → probabilità che una previsione positiva sia corretta

Valore Predittivo Negativo (NPP, Negative Predictive Power)
d/q → probabilità che una previsione negativa sia corretta

Odds-ratio
ad/cb → rapporto tra probabilità di assegnazioni corrette e scorrette

Statistica K (Cohen K-statistics)
$[\ r_t(a+d) - (px+qy)\ ] / [\ r_t^2 - (px+qy)\ ]$
rapporto tra la porzione di accordo non dovuto al caso
e la massima porzione possibile di accordo non dovuto al caso $(A_{obs}-A_{exp}) / (1-A_{exp})$

NMI (normalized mutual information)
$1-[-a\ln(a)-b\ln(b)-c\ln(c)-d\ln(d)+p\ln(p)+q\ln(q)]/[r_t\ln(r_t)-(x\ln(x)+y\ln(y))]$
in base all'entropia di Shannon, misura la trasmissione dell'informazione tra osservazioni e previsione

Fig. 24

Vari indici di performance possono essere estratti dalla matrice di confusione. Il più semplice e intuitivo di essi è il tasso di classificazione corretta (CCR), che calcola il rapporto tra le previsioni esatte (riportate nelle celle **a** e **d**, che compongono la diagonale principale della matrice) e tutte le previsioni compiute. E' un indice facilmente comprensibile e calcolabile, a cui si può utilmente far riferimento in molte circostanze diverse; per queste ragioni esso è forse l'indice più utilizzato in generale. Dà valori tendenti all'1 quando le previsioni sono quasi tutte corrette (cioè tutti i casi si concentrano nella diagonale principale della matrice di confusione), valori tendenti allo 0 quando quasi tutte le previsioni sono errate (cioè, il modello predice '1' quando il dato osservato è 0 e '0' quando il dato osservato è 1) e valori tendenti a 0.5 quando il modello predittivo si comporta come un decisore casuale (per cui emette '1' e '0' in modo praticamente random, determinando una distribuzione tendenzialmente uniforme dei casi tra le due diagonali della matrice). Finché la frequenza dei casi positivi osservati (prevalenza) si trova intorno a 0.5, il tasso di classificazione corretta riesce abbastanza bene a discriminare tra efficacia predittiva (Fig. 25a), decisione casuale (Fig. 25b) e associazione negativa (Fig. 25c) tra predizioni e osservazioni.

OBS	PRED	a	b	c	d
0	0	0	0	0	1
1	1	1	0	0	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	1	0
1	1	1	0	0	0
0	0	0	0	0	1
0	0	0	0	0	1
1	1	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
		7	1	2	10

ACCORDO POSITIVO CON PREVALENZA ≈ 0.5

PREVALENZA: 0.45

		OBS 1	OBS 0	
PRED	1	7	1	8
	0	2	10	12
		9	11	20

Indici di performance

CCR:	0.85
Sensibilità:	0.78
Specificità:	0.91
VPP:	0.88
VPN:	0.83
Odds-ratio:	35.00
Statistica K:	0.69
NMI:	0.39

Fig. 25a

OBS	PRED	a	b	c	d
1	1	1	0	0	0
0	1	0	1	0	0
0	0	0	0	0	1
0	1	0	1	0	0
0	1	0	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	0	0	0	1
1	1	1	0	0	0
1	0	0	0	1	0
0	1	0	1	0	0
0	1	0	1	0	0
1	1	1	0	0	0
1	0	0	0	1	0
1	1	1	0	0	0
0	1	0	1	0	0
1	1	1	0	0	0
0	0	0	0	0	1
1	1	1	0	0	0
0	0	0	0	0	1
		6	7	3	4

ACCORDO CASUALE
CON PREVALENZA ≈ 0.5

PREVALENZA: 0.45

		OBS 1	OBS 0	
PRED	1	6	7	13
	0	3	4	7
		9	11	20

Indici di performance

CCR:	0.50
Sensibilità:	0.67
Specificità:	0.36
VPP:	0.46
VPN:	0.57
Odds-ratio:	1.14
Statistica K:	0.03
NMI:	0.00

Fig. 25b

OBS	PRED	a	b	c	d
0	1	0	1	0	0
1	0	0	0	1	0
0	1	0	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	0	0	0
1	0	0	0	1	0
1	0	0	0	1	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	0	0	0
1	0	0	0	1	0
0	1	0	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	0	0	0	1
0	1	0	1	0	0
0	1	0	1	0	0
0	1	0	1	0	0
0	1	0	1	0	0
		2	10	7	1

ACCORDO NEGATIVO
CON PREVALENZA ≈ 0.5

PREVALENZA: 0.45

		OBS 1	OBS 0	
PRED	1	2	10	12
	0	7	1	8
		9	11	20

Indici di performance

CCR:	0.15
Sensibilità:	0.22
Specificità:	0.09
VPP:	0.17
VPN:	0.13
Odds-ratio:	0.03
Statistica K:	-0.67
NMI:	0.39

Fig. 25c

Quando però la prevalenza si avvicina a 0 (specie rare; Fig. 26a) o a 1 (specie ubiquitarie) l'indice CCR tende comunque a dare valori alti, rendendo impossibile una corretta distinzione tra predizione efficace e predizione casuale o costante[57].

OBS	PRED	a	b	c	d
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	1	0	1	0	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
		0	**1**	**1**	**18**

ACCORDO CASUALE CON PREVALENZA ≈ 0.1

PREVALENZA: **0.05**

		OBS 1	OBS 0	
PRED	1	0	1	1
	0	1	18	19
		1	19	20

Indici di performance

CCR:	**0.90**
Sensibilità:	**0.00**
Specificità:	**0.95**
VPP:	**0.00**
VPN:	**0.95**
Odds-ratio:	**0.00**
Statistica K:	**-0.05**
NMI:	non calcolabile

Fig. 26a

C'è anche un'altra condizione possibile: quella in cui il modello predittivo fornisca un'unica previsione sempre uguale, cioè solo '1' o solo '0', indipendentemente dai valori delle variabili in ingresso; come già spiegato nell'illustrare l'ipotesi 2, con le reti neurali questa circostanza non è rara da osservare quando la prevalenza si avvicina, rispettivamente, all'1 o allo 0. In Fig. 26b è presentato un esempio con una specie rara.

[57] Avendo a che fare con modelli sensibili alla prevalenza, che tendono a fornire valori predittivi con frequenze di 1 e 0 vicine a quelle dei valori osservati, quando qui si parla di "previsione causale" è sottointeso che il processo di generazione casuale delle previsioni sia comunque vincolato dalle frequenze osservate e non dia luogo, dunque, ad un numero orientativamente pari di 1 e 0, come con il lancio di una moneta. In queste condizioni, una previsione casuale non fornisce valori molto diversi da quelli di una previsione costante.

OBS	PRED	a	b	c	d
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
		0	0	2	18

PREVISIONE COSTANTE CON PREVALENZA ≈ 0.1

PREVALENZA: 0.10

		OBS 1	OBS 0	
PRED	1	0	0	0
	0	2	18	20
		2	18	20

Indici di performance

CCR:	0.90
Sensibilità:	0.00
Specificità:	1.00
VPP:	non calcolabile
VPN:	0.90
Odds-ratio:	non calcolabile
Statistica K:	0.00
NMI:	non calcolabile

Fig. 26b

Poiché lo studio della struttura di una comunità implica anche l'analisi di specie rare, che tra l'altro costituiscono spesso un elemento speciale di interesse in ecologia, il tasso di classificazione corretta non può essere considerato un buon indice di performance in questi casi.

Altri indici, come la sensibilità e la specificità o come il valore predittivo positivo o negativo, sono principalmente utilizzati in casi in cui ai due tipi di errore possibili (FP ed FN) sono associati diversi livelli di rischio o di costo. Questo accade, ad esempio, nella ricerca medica, in biologia della conservazione o nelle analisi di mercato. Se tali differenze di rischio o di costo non sussistono, difficilmente questi indici vengono usati singolarmente, perché forniscono una visione sbilanciata e parziale dell'informazione contenuta nella matrice di confusione. Casomai essi vengono pesati in base a matrici di rischio o di costo e combinati insieme a determinare misure specifiche.

Gli ultimi tre indici riportati in Fig. 24 hanno invece la notevole proprietà di tenere conto in modo equilibrato di tutta l'informazione contenuta nella matrice di confusione (Fielding & Bell, 1997). Ad essi si può fare riferimento quando si è

interessati ad una valutazione globale e completa delle capacità predittive dei modelli generati. Ognuno presenta potenzialità e limiti che lo rendono particolarmente adatto o inadatto in particolari circostanze.

Per una serie di ragioni (Fielding & Bell, 1997; Scardi et al., 2005) che è possibile condensare in tre considerazioni fondamentali, in questo studio si è preferito l'uso della statistica K di Cohen (1960): 1) essa varia tra -1, 0 e 1 rappresentando in modo immediato la transizione tra accordo negativo, casuale e positivo[58] tra previsioni e osservazioni, a differenza, ad esempio, dell'*odds-ratio* che non ha limiti prefissati di immediata lettura (Fig. 25a); 2) finché il vettore delle osservazioni non è banale (cioè non contiene tutti 1 o tutti 0), l'indice è sempre calcolabile, a differenza dell'*odds-ratio* e del NMI di Forbes (1995), che non ammettono valori nulli in alcune celle della matrice (Figg. 26a e 26b); 3) la statistica K fornisce una misura della performance che differenzia i modelli di previsione in base alla loro capacità di scostarsi dal comportamento di un decisore casuale, permettendo così di valutare specificamente quanto l'addestramento di ogni singolo modello abbia condotto quest'ultimo ad una reale modellizzazione dei rapporti tra le variabili[59].

Poiché la matrice di confusione si modifica in funzione della soglia applicata per la discretizzazione dell'output continuo della rete, tutti gli indici calcolabili in base ad essa, inclusa la statistica K, dipendono da tale soglia. Fissare una soglia è indispensabile per ottenere una vera e propria previsione di presenza/assenza riguardo alle varie specie; per questo motivo ha senso il confronto proposto in

[58] In realtà, nel caso della statistica K sarebbe corretto parlare di "associazione" piuttosto che di "accordo" (Forbes, 1995). Previsioni casuali o errate manifestano infatti entrambe una mancanza di accordo, mentre un insieme di previsioni tutte errate dà luogo ad un'associazione perfetta, per quanto negativa, tra le due variabili binarie confrontate. Dunque, nonostante qui si userà a volte il termine "accordo" anche nel senso allargato di associazione, è importante sottolineare che vi sono indici, come la statistica K, che misurano l'associazione, mentre altri, come l'NMI, misurano l'accordo.

[59] La formula in Fig. 24, che può essere riscritta anche in base alle frequenze relative,

$$K = \frac{A_{OBS} - A_{EXP}}{1 - A_{EXP}} = \frac{(p_a + p_d) - (p_{ab}\,p_{ac} + p_{cd}\,p_{bd})}{1 - (p_{ab}\,p_{ac} + p_{cd}\,p_{bd})}$$

calcola la differenza tra l'accordo osservato (rappresentato dalla frequenza cumulata dei casi presenti nella diagonale principale, p_a+p_d) e quello atteso per effetto del caso (calcolato in base alle frequenze marginali, $p_{ab}p_{ac}+p_{cd}p_{bd}$) e fa il rapporto tra questa differenza e quella che si sarebbe avuta se l'accordo osservato fosse stato il massimo possibile (con tutti i casi concentrati nella diagonale principale e dunque $p_a+p_d=1$).

questo lavoro dall'ipotesi 1. Tuttavia non si tratta di un confronto tra modelli diversi, ma tra diverse interpretazioni del risultato del medesimo modello.

Molti studi sottolineano l'importanza di ottenere anche misure della performance che siano indipendenti dalla soglia (Fielding & Bell, 1997; Guisan & Zimmermann, 2000), così da valutare la performance di un modello senza il "disturbo" introdotto da una precisa scelta interpretativa. Negli studi di ecologia (e non solo) l'indice più usato per ottenere una misura di questo tipo è l'area sotto la curva (AUC, Area Under Curve) che si determina in un grafico ROC (Receiver Operating Characteristics; Fig. 27).

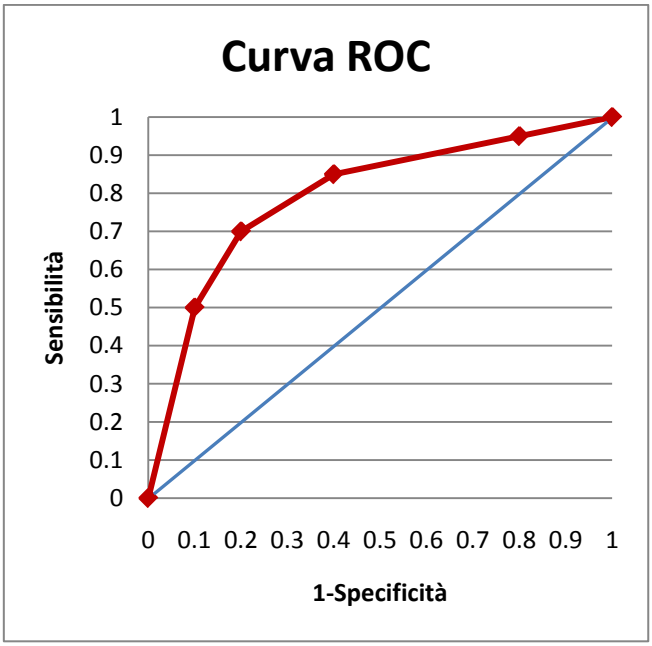


Fig. 27

I punti che danno luogo alla curva sono dati dalle coppie di valori della proporzione di casi negativi erroneamente predetti come positivi (1-Specificità) e della proporzione di casi positivi correttamente identificati (Sensibilità) che si determinano dalla matrice di confusione al variare della soglia[60]. Quindi, ogni punto riflette la performance di un modello con una determinata soglia. Se si fa variare la soglia da 0 a 1 incrementandola ogni volta di un intervallo *c*, e per ogni matrice di

[60] Si veda, ad esempio, Fielding & Bell (1997), per una descrizione più completa.

confusione ottenuta si determina un punto nel grafico ROC, alla fine la spezzata ottenuta rifletterà la performance del modello indipendentemente dalla singola scelta possibile della soglia. L'area sotto tale spezzata[61] può essere poi usata come misura sintetica della performance. La linea blu nel grafico rappresenta in modo ideale la curva che si otterrebbe da un decisore casuale; dunque un valore di AUC=0.5 indica un modello le cui previsioni non si discostano da quelle di un generatore casuale di 1 e 0. Tra 0.5 e 1 si hanno invece i valori di AUC che indicano una capacità via via crescente del modello di prevedere correttamente i casi di presenza/assenza delle specie (una lettura inversa si ha invece quando si scende da 0.5 a 0; AUC=0 indica un modello che sbaglia sistematicamente tutte le previsioni).

Si è descritta questa misura di performance indipendente dalla soglia in quanto molto diffusa nell'analisi di modelli predittivi con output binario. Tuttavia in questo lavoro alcune prove preliminari hanno mostrato una sostanziale inadeguatezza di AUC nel confrontare tra loro i modelli generati. E' probabile che questo dipenda essenzialmente dalla bassa prevalenza di molte delle specie considerate. In tali condizioni, gli indici tratti dalla matrice di confusione tendono ad essere molto sensibili anche a piccole variazioni nella distribuzione dei valori nella tabella, fornendo a volte singoli valori estremamente bassi o elevati che influiscono poi però notevolmente sul calcolo di AUC. Si pensi, ad esempio, ad un caso limite in cui una di queste "strane coppie" di valori viene ottenuta una sola volta al variare della soglia, a fronte di 99 risultati banali (coppie [0,0] o [1,1]) ottenuti con le altre soglie[62]: la curva avrà tre soli punti, di cui due coincidenti con gli estremi della linea blu e uno molto vicino all'angolo alto a sinistra o basso a destra; il valore di AUC sarà qui molto alto o molto basso, ma rifletterà, paradossalmente, una proprietà del modello legata ad un'unica soglia ben precisa, al di là e al di qua della quale il modello non funziona.

Volendo comunque far uso di una misura di performance indipendente dalla soglia, si è dunque scelto di elaborarne una *ad hoc*. Nel prossimo capitolo tale indice, chiamato AUK (Area Under K-Curve), sarà descritto nel dettaglio. Essendo

[61] In realtà il calcolo dell'area implica la costruzione di una curva interpolata, ma in questa sede non è importante scendere in particolari.

[62] Questo si verifica, ad esempio, quando all'unico caso di presenza osservato il modello di previsione associa un valore predittivo che è più elevato di quello associato ai casi di assenza di una misura inferiore al passo *c* con cui viene elevata di volta in volta la soglia. Allora può succedere che una sola volta la soglia si trovi tra i due valori, dando una previsione perfetta. Prima di giungere in quel punto, però, il modello darebbe solo previsioni di presenza e subito dopo darebbe solo previsioni di assenza, cadendo nei casi banali.

indipendente dalla soglia, esso fornirà un criterio per l'analisi delle ipotesi 3, 4 e 5, ma ovviamente lascerà da parte l'ipotesi 1.

2.5 Test statistici

All'inizio del capitolo si diceva che per ogni specie si otterranno 16 serie di modelli, ognuna corrispondente ad una condizione sperimentale.

Per le 8 condizioni sperimentali che implicano una partizione secondo i dati di presenza/assenza delle specie, le serie sono composte dai 72 modelli generati (Fig. 20). Per le restanti 8 condizioni sperimentali, ogni specie è invece stata testata su tutte le partizioni (72x33) generate per le varie specie[63].

Guardando nel loro complesso le 16 distribuzioni di risultati (statistica K) che si ottengono per ogni specie, si noterà che alcune di esse possono essere confrontate tramite test per dati appaiati, in quanto i modelli sono stati addestrati sulle medesime partizioni. Come si è detto, infatti, le performance relative alle condizioni legate alle ipotesi 1 e 3 vengono tratte da modelli addestrati sulle medesime 72 partizioni relative ad un cella della tabella che incrocia le condizioni delle ipotesi 4 e 5. Anche nei casi di partizione random rispetto ai casi di presenza assenza, i modelli sono stati addestrati tutti sulle medesime 72x33 partizioni[64]; dunque in tutti i casi c'è la possibilità di confrontare a quattro a quattro le alternative di ipotesi, ad esempio, tramite un test di Wilcoxon. I confronti tra condizioni legate a celle diverse della tabella di Fig. 20 vanno invece condotti con test per dati indipendenti.

Una prima scelta di base è stata quella di orientarsi su statistiche di tipo non parametrico. Non vi è ragione, infatti, di ritenere che le distribuzioni dei risultati ottenuti nelle varie condizioni si conformino in generale alle assunzioni richieste dalla statistica parametrica (normalità, omoscedasticità, indipendenza, ecc).

Per quanto riguarda invece la scelta specifica dei test da utilizzare, si è preferito semplificare le cose e scegliere un unico test in grado di effettuare tutti i confronti

[63] Mentre, infatti, le 72x2 partizioni per valori di presenza/assenza di una singola tabella (Fig. 20) sono legate specificamente ad una specie, le altre 72x2 possono essere usate con qualunque specie. In totale queste partizioni sono 72x33 e non 72x32 perché sono state usate anche le partizioni random relative alla tabella della ricchezza di specie, anche se poi, come già sottolineato, la ricchezza di specie in quanto tale non è stata presa in considerazione in questo lavoro.

[64] In realtà, soprattutto nel caso di specie rare, quando non viene effettuata una partizione per i casi di presenza/assenza, può capitare che nessun caso di presenza venga assegnato al test set, rendendo di fatto la partizione inutile per qualunque valutazione di performance. Ciò significa che per alcune specie le distribuzioni di performance ottenute non contengono 72x33 valori, ma qualcuno di meno.

tra le 16 distribuzioni di risultati ottenute per ogni specie. Il motivo principale è disporre di un unico strumento per valutare le differenze tra tutte le possibili coppie di distribuzioni da confrontare. Il test scelto è il Wilcoxon-Mann-Whitney, o test della somma dei ranghi, per una serie di ragioni: 1) è da ritenersi equivalente al test di Wilcoxon per campioni dipendenti, che si sarebbe scelto per confrontare le distribuzioni con dati appaiati; 2) rispetto al primo, al prezzo di una potenza leggermente minore, impone condizioni meno stringenti riguardo alle distribuzioni da confrontare (in particolare, forme analoghe contro forme simmetriche); 3) la minore potenza è da considerarsi, al limite, un vantaggio in termini di cautela nel rigetto dell'ipotesi nulla; 4) il Wilcoxon richiede l'eliminazione degli zeri dalla distribuzione delle differenze, mentre il WMW tiene conto di tutta l'informazione presente, con il vantaggio di non introdurre un potenziale elemento di eterogeneità tra i confronti, rispetto alla quantità di informazione utilizzata[65].

Per ogni specie si hanno dunque 16 modelli, da ognuno dei quali si ricava una distribuzione di risultati; i confronti possibili tra modelli diversi sono in tutto 120, ma solo alcuni di essi sono utili per il confronto diretto delle due alternative legate ad ogni ipotesi.

Ogni modello può essere identificato con un codice di quattro cifre binarie (ad esempio, 0010, 1011, 1100, ecc.): la prima cifra, relativa alle condizioni prese in causa dall'ipotesi 1, indica se la performance è stata misurata in base ad una soglia tradizionale a 0.5 (nel qual caso la cifra è '0') o in base ad una soglia *best* (nel qual caso la cifra è '1'); la seconda cifra, relativa all'ipotesi 3, indica se il modello è monospecie ('0') o multispecie ('1'); la terza cifra, relativa all'ipotesi 4, indica se la partizione ha tenuto conto dei valori presenza/assenza ('1') o se è random da questo punto di vista ('0'); la quarta cifra, relativa all'ipotesi 5, indica se la partizione è stata condotta in base all'altitudine ('1') oppure no ('0'). Dunque '0010' indica un modello ottimizzato solo per presenza/assenza; '1011' indica invece un modello con soglia ottimizzata, previsione monospecie e partizione sia per presenza/assenza che per altitudine; '1100' si riferisce invece ad un modello con soglia ottimizzata e

[65] Prove preliminari hanno mostrato una certa frequenza di valori identici di K tra dati appaiati, dovuta essenzialmente ad una riduzione delle caratteristiche di continuità della statistica K quando la matrice di confusione contiene un numero non elevato di casi. In particolare, quando si confrontano distribuzioni di 72 misure di performance nel caso di specie rare, può accadere che i valori relativi a diversi modelli siano uguali e, in particolare, pari a 0. Poiché questo fenomeno può dar luogo ad un certo squilibrio tra le quantità di informazione usate in vari confronti, la scelta di un unico test che tenga sempre conto di tutti i valori presenti appare più adatta.

previsione multispecie, ma con partizione random rispetto alla presenza/assenza e all'altitudine.

Per ogni possibile combinazione delle altre tre ipotesi, ogni ipotesi propone due alternative; poiché le possibili combinazioni di tre ipotesi sono 8, ogni ipotesi va testata effettuando 8 diversi confronti. Ad esempio, per testare l'ipotesi 3 i confronti da effettuare sono:

0000 *vs.* 0100 (**0#00**)
0001 *vs.* 0101 (**0#01**)
0010 *vs.* 0110 (**0#10**)
0011 *vs.* 0111 (**0#11**)
1000 *vs.* 1100 (**1#00**)
1001 *vs.* 1101 (**1#01**)
1010 *vs.* 1110 (**1#10**)
1011 *vs.* 1111 (**1#11**).

Si può ragionare nello stesso modo per le altre ipotesi, ognuna delle quali implica dunque 8 confronti, nel caso venga presa in considerazione una sola specie, o 256 (8x32), se si tiene conto di tutte le specie insieme. Che si desideri trarre conclusioni a livello di specie oppure ad un livello più generale, quindi, la verifica di ipotesi va impostata comunque su confronti multipli, richiedendo di conseguenza una correzione delle soglie di significatività. Tale correzione è indispensabile per tenere conto del rischio di rilevare differenze significative per effetto del caso, rischio che aumenta quando aumenta il numero dei confronti.

Ad esempio, restando a livello di specie, se anche uno solo degli 8 confronti relativi ad un'ipotesi desse risultati significativi, si può sostenere che almeno in certe circostanze si riesce a falsificare l'ipotesi nulla con la particolare specie in esame; quindi si trae una conclusione generale (sull'ipotesi) sulla base di un risultato particolare (ottenuto solo in una delle condizioni in cui essa può essere testata). Quando il confronto è unico, la probabilità di rilevare correttamente l'assenza di effetti sistematici (cioè di non rifiutare l'ipotesi nulla quando questa è vera) è pari a $1-\alpha$; se si conducono n confronti, la probabilità di non ottenere risultati significativi scende a $(1-\alpha)^n$, dunque aumenta la probabilità $[1-(1-\alpha)^n]$ di ottenere almeno un risultato significativo per effetto del caso. Per compensare l'aumento di tale rischio, il livello di significatività scelto come soglia, ad esempio $\alpha=0.01$, può essere ridotto in funzione del numero di confronti effettuati, in modo da rendere più stringenti le condizioni di rigetto dell'ipotesi nulla.

Un tipo di correzione largamente in uso per questo scopo è la *correzione di Bonferroni*, che richiede semplicemente che il livello di significatività corretto α_B da usare come soglia risulti dal rapporto tra il livello di significatività desiderato e il numero di confronti effettuati:

$$\alpha_B = \frac{\alpha}{n}$$

Ad esempio, se si volessero trarre conclusioni in base agli 8 confronti relativi ad una singola ipotesi con una singola specie, dato un livello di significatività desiderato pari a 0.01, si avrebbe

$$\alpha_B = \frac{0.01}{8} = 0.00125$$

per cui bisognerebbe raggiungere una significatività almeno pari a 0.00125 nel singolo test per poter sostenere di aver rilevato un effetto con una significatività pari a 0.01 nel contesto dei confronti multipli effettuati.

In questo lavoro, le condizioni adottate sono ancora più stringenti, perché l'interesse è quello di rilevare effetti della scelta di diverse alternative di ottimizzazione non tanto su una specie in particolare, ma nel contesto dell'intero popolamento. Dunque il numero di confronti da considerare è 8x32=256. I livelli di significatività corretti sono indicati in Tab. 3.

Livello di significatività desiderato	Significatività con correzione di Bonferroni per 256 confronti
0.05	0.00019531
0.01	0.00003906
0.001	0.00000391
0.0001	0.00000039

Tab. 3

Poiché almeno nel caso di due delle ipotesi considerate (3 e 5) non ci si può attendere sistematicamente la prevalenza di una alternativa sull'altra, i test condotti saranno bilaterali[66].

Nel paragrafo precedente (2.4) si è accennato ad una misura della performance indipendente dalla soglia (AUK, che verrà descritta nel prossimo capitolo), in base alla quale saranno condotti, su una base diversa da quella fornita dalla statistica K basata sulla soglia, i confronti relativi alle sole ipotesi 3, 4 e 5. In tal caso, per ogni specie e ogni ipotesi, non si hanno più 8 confronti, ma solo 4. Il caso dell'ipotesi 3, usato poco sopra come esempio, va dunque così riadattato:

X000 *vs.* X100 (**X#00**)
X001 *vs.* X101 (**X#01**)
X010 *vs.* X110 (**X#10**)
X011 *vs.* X111 (**X#11**)

Qui la 'X' non ha alcun significato, se non quello di ricordare che c'è un'altra ipotesi in gioco, di cui tuttavia in questi confronti non si tiene conto. Il numero di confronti da usare per il calcolo della correzione di Bonferroni è qui 4x32=128 (Tab. 4).

Livello di significatività desiderato	Significatività con correzione di Bonferroni per 128 confronti
0.05	0.00039063
0.01	0.00007813
0.001	0.00000781
0.0001	0.00000078

Tab. 4

[66] L'estensione del test bilaterale anche alle ipotesi 1 e 4 è giustificata da un'esigenza di omogeneità nella probabilità di rilevare differenze sistematiche, che con l'ipotesi bilaterale e più bassa di quanto sarebbe con quella unilaterale. Inoltre, le varianti legate alle ipotesi 1 e 4 sono costruite in modo tale da lasciare aperta la possibilità di effetti opposti rispetto alle attese.

3. RISULTATI

I risultati di questo lavoro si dividono essenzialmente in due gruppi: quello dei risultati veri e propri, legati ai valori di performance dei modelli e al confronto tra le condizioni sperimentali indagate, e quello dei metodi e degli algoritmi originali elaborati per condurre in modo appropriato l'indagine progettata.

3.1. Metodi e algoritmi

Come già illustrato nel capitolo precedente, l'intero disegno sperimentale è stato generato, eseguito e valutato tramite alcuni applicativi sviluppati all'interno del progetto.

Il primo applicativo è delphish1.exe, il cui codice sorgente è visionabile integralmente in Allegato A.

Il programma legge innanzitutto la matrice dei dati da un file data.csv[67] appositamente predisposto e genera le 9504 partizioni relative all'incrocio tra le condizioni indagate dalle ipotesi 4 e 5. Al suo interno due algoritmi eseguiti da due diverse subroutine provvedono a determinare le 72 partizioni (36 per ognuno) omogenee per ogni condizione di incrocio.

Entrambi gli algoritmi si basano su un principio generale molto semplice: ordinate le righe della matrice dei dati secondo un gradiente la cui casistica si vuole distribuire equamente tra i subset, è possibile assegnare i casi ad ognuno pescandoli in modo omogeneo da ogni porzione di tale gradiente. In questo caso il gradiente è legato ad uno specifico descrittore, cioè l'altitudine, per cui la matrice dei dati viene innanzitutto ordinata in base alla colonna relativa. Poiché però il numero totale dei pattern (osservazioni) è discreto e relativamente limitato; poiché la distribuzione del descrittore non è del tutto continua e regolare dal suo minimo al suo massimo; e poiché la distribuzione del singolo descrittore, ancorché legata a quella degli altri, non ne spiega tutta la variabilità; allora, nonostante ognuna delle 72 assegnazioni miri al medesimo scopo, cioè un'equa distribuzione delle condizioni osservate ai

[67] Il formato *csv* (Comma Separated Values) si presta molto bene a scambiare dati con programmi del tipo sviluppato in questo lavoro. Può essere infatti letto e creato in modo molto semplice, oltre che dai programmi in BASIC, anche da comuni editor di testo o da Microsoft Excel.

subset, ogni partizione finisce in realtà con l'individuare una distribuzione dei pattern leggermente diversa dalle altre e si crea dunque una sorta di "rumore" attorno ad un'ideale assegnazione ottimale che non è possibile definire a priori.

L'output dei due algoritmi consiste semplicemente in vettori colonna i cui elementi possono essere solo i numeri 1, 2 e 3 a seconda che una particolare riga risulti assegnata al subset di training, di validazione o di test, rispettivamente. In generale, l'algoritmo **RFA_Partitioner** (Regular Filling Algorithm) mira a distribuire ad intervalli il più possibile regolari gli '1', i '2' e i '3' lungo il vettore delle assegnazioni. L'algoritmo **MCA_Partitioner** (Modular Chain Algorithm) riempie invece il vettore delle assegnazioni usando le 12 possibili 4-ple composte da due '1', un '2' e un '3'[68].

Determinate le partizioni, il programma crea poi 33 directory e salva in ognuna di esse i file necessari agli applicativi batchnnc.exe e nnout4.exe per condurre i training previsti dal disegno sperimentale. In una di esse vanno i file per l'addestramento del modello multispecie, mentre in ognuna delle restanti 32 sono salvati i file per l'addestramento dei modelli monospecie relativi ad una singola specie. In ogni cartella il programma salva anche una copia di batchnnc.exe e di nnout4.exe e due file batch creati dal programma stesso. Il primo file batch contiene le istruzioni per far eseguire tutti i training al programma batchnnc.exe, mentre il secondo esegue tutti i modelli addestrati tramite nnout4.exe.

In pratica, quando il programma delphish1.exe termina il suo lavoro (che richiede circa un paio d'ore di elaborazione), nella cartella da cui lo si è lanciato si trovano le 33 cartelle con i file già predisposti. Ogni cartella è indipendente e può essere copiata anche su altri computer. Da ognuna di esse le operazioni di training possono essere avviate con un semplice doppio click sul primo file batch; al termine delle operazioni si potrà fare lo stesso con il secondo file batch. Inoltre, i file batch cancellano via via i file non più necessari, lasciando nelle cartelle solo quelli indispensabili. Nel primo caso, i file con i dati di training e quelli di configurazione, una volta usati possono essere cancellati; nel secondo caso, i file con i pesi sinaptici, una volta usati, possono essere cancellati. In ogni cartella restano così solo 9504 file *.out con le previsioni da testare e 9504 file *.tes con i dati di test usati per generare le previsioni. L'unica operazione manuale che l'utente deve eseguire per ogni

[68] In allegato A, il codice è corredato da commenti e spiegazioni che approfondiscono i particolari tecnici del funzionamento dei due algoritmi.

cartella è la copia di un ulteriore programma, delphish11.exe, che una volta avviato compatta tutti i file *.out in un unico file *.res con i risultati. Al termine di queste operazioni, dunque, dentro ognuna delle 33 cartelle iniziali resta solo un unico file *.res.

La ragione di questa organizzazione del lavoro basata su cartelle indipendenti è legata ai tempi di elaborazione: fare tutto in un'unica soluzione avrebbe richiesto che un'unica macchina lavorasse intensamente e ininterrottamente per circa un mese, esposta ad incidenti di percorso che potevano pregiudicare un lavoro di molti giorni. Articolando invece in moduli l'insieme dei training previsti è stato possibile dividere il lavoro in più *tranches* e su più macchine, con una notevole riduzione dei tempi e dei rishi. Lo sviluppo *ex novo* di un sistema di questo tipo, infatti, può richiedere molte prove in fase sperimentale, per cui il fattore tempo assume un'importanza notevole.

Il programma delphish12.exe[69] prende poi i 33 file *.res e li compatta in 4 soli file: uno con i dati osservati (OBS.res), uno con le previsioni del modello multispecie (PRES_A.res), uno con le previsioni dei modelli monospecie (PRED_S.res) e uno con le etichette relative ad ogni sessione di training condotta (label.res).

A questo punto interviene il programma delphish2.exe, che sulla base dei quattro file con le previsioni ottenute e con le osservazioni corrispondenti, ha il compito di calcolare i valori di performance e di fornire una serie di file che a livello sia descrittivo che inferenziale diano un quadro completo della verifica di ipotesi che ci si proponeva[70].

Il ciclo principale del programma passa in rassegna le previsioni (continue) relative a tutti i modelli addestrati e le confronta con le corrispondenti osservazioni (binarie). Volendo studiare gli effetti che opportune variazioni della soglia possono avere sulle performance (ipotesi 1 e 2), il programma determina le matrici di confusione, e i relativi valori della statistica K, per molti valori possibili della soglia. In particolare, si è scelto di partire dalla soglia minima, pari a 0, e di incrementarla ogni volta di un valore ε pari a 0.01, giungendo così al valore massimo di 1 in 100 passi. Escluse le due soglie minima e massima, che danno valori di performance banali, si hanno dunque per ogni modello 99 valori di statistica K. La Fig. 28 mostra

[69] I due programmi delphish11.exe e delphish12.exe non vengono riportati in allegato con il codice sorgente in quanto legati più ad accorgimenti tecnici che a questioni di contenuto.

[70] Il codice sorgente di delphish2.exe è disponibile integralmente in Allegato B.

un esempio, tratto dai risultati ottenuti, di come la statistica K possa variare in funzione della soglia di discretizzazione.

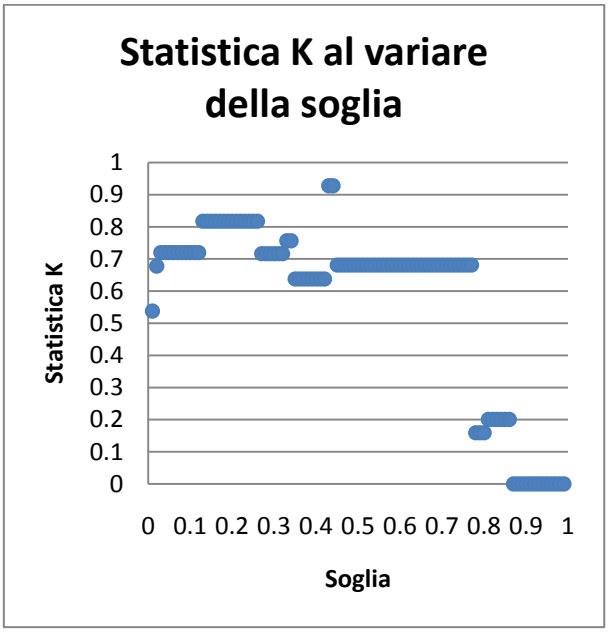


Fig. 28

Il grafico rappresenta il caso di una specie predetta in modo non eccellente, ma comunque molto buono. Come si vede, scegliere una soglia piuttosto che un'altra può essere determinante: anche se in questo caso una soglia tradizionale a 0.5 darebbe comunque predizioni buone, una soglia intorno a 0.2 darebbe performance ancora migliori. Nell'esempio in Fig. 29, invece, si ha un caso in cui una scelta tradizionale della soglia condurrebbe ad un modello inefficace, con prestazioni pari a quelle di un decisore casuale, mentre la scelta di una soglia pari a circa 0.2 darebbe risultati molto buoni.

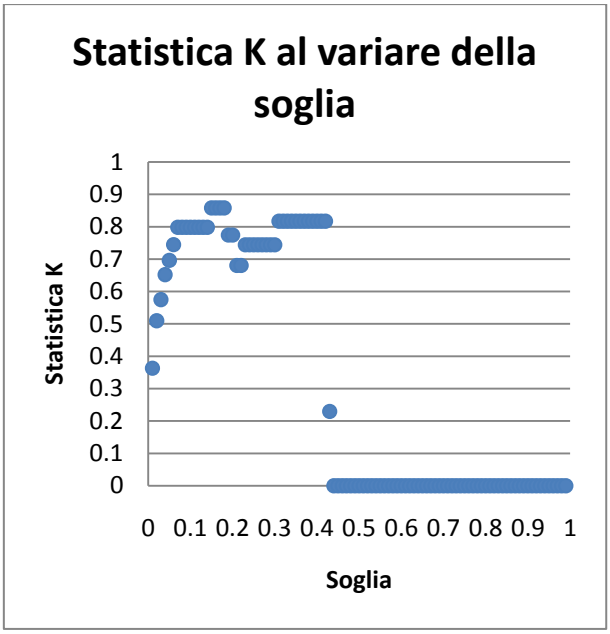


Fig. 29

Per trovare la soglia *best* per un modello, il programma utilizza un algoritmo per il quale è stato necessario sviluppare una particolare forma di media mobile pesata (μ_R) da calcolare sui valori della statistica K ottenuti al variare della soglia. La funzione che definisce i pesi è calcolata fissando un raggio (R) attorno ad un punto centrale. Per ognuna delle soglie, la media viene calcolata considerando centrale il valore di K relativo a quella soglia; il peso assegnato a quel valore è 1, mentre quello assegnato agli altri valori è via via decrescente man mano che questi si allontanano dal valore centrale; esso si riduce al valore *a* in corrispondenza degli estremi del raggio R, per poi avvicinarsi asintoticamente a 0 allontanandosi da essi. La funzione per il calcolo dei pesi è la seguente:

$$w(c,R,a,t) = e^{\frac{\ln a}{R^2}(t-c)^2}$$

in cui *c* è la soglia su cui la funzione è centrata e *t* è il valore soglia cui via via (in base ad incrementi progressivi pari a ε) la funzione deve attribuire un peso. I parametri della funzione qui utilizzata sono stati fissati arbitrariamente in modo che i

valori di K corrispondenti ad una soglia distante R=0.1 dalla soglia centrale *c* ricevessero un peso pari a 0.1:

$$w(c, 0.1, 0.1, t) = e^{\frac{\ln 0.1}{0.1^2}(t-c)^2}$$

La Fig. 30 mostra un esempio della funzione peso utilizzata, quando essa è centrata sul valore soglia di 0.3.

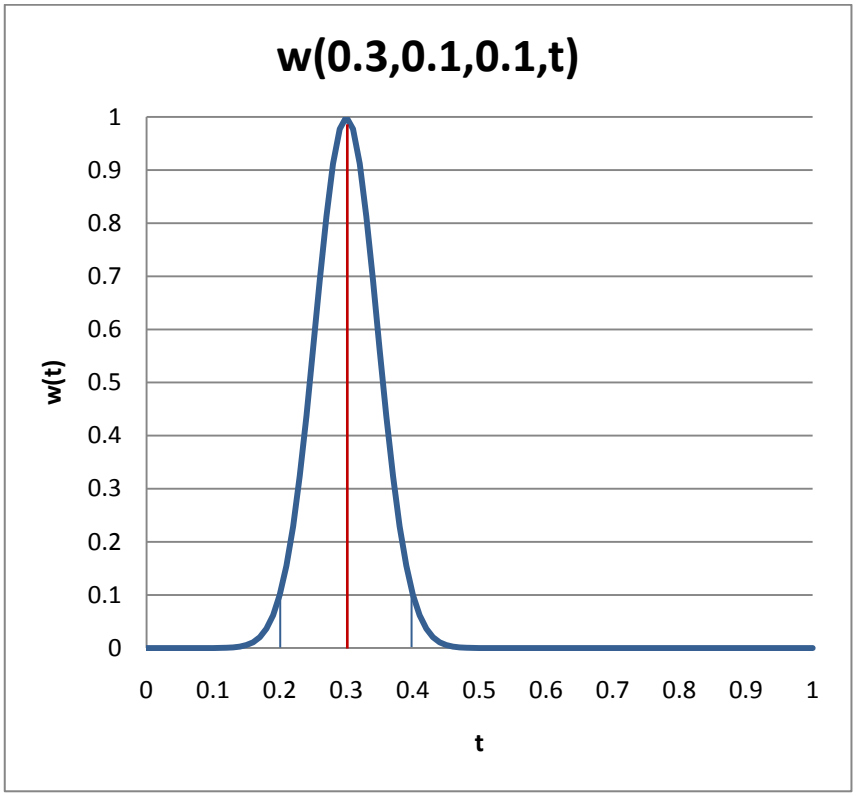


Fig. 30

Si tratta di una media pesata che funziona in pratica come un filtro passa-banda in elettrotecnica, assegnando un peso elevato ai valori presenti in un intorno di un valore considerato centrale e smorzando fortemente quelli che invece si trovano

all'esterno di esso. Dato un particolare valore centrale *c* della soglia, la media μ_R è definita dalla funzione

$$\mu_R(c,K) = \frac{\sum_{i=0}^{m} k_i \cdot e^{\frac{\ln 0.1}{0.1^2}\left(\frac{i}{m}-c\right)^2}}{\sum_{i=0}^{m} e^{\frac{\ln 0.1}{0.1^2}\left(\frac{i}{m}-c\right)^2}}$$

Centrata su ognuna delle soglie utilizzate, questa media permette una visione dei singoli risultati legata anche al loro intorno e riduce così il rischio, nella ricerca della soglia *best*, di selezionare picchi isolati, puntando piuttosto verso porzioni di grafico con valori stabilmente elevati (punti in rosso in Fig. 31).

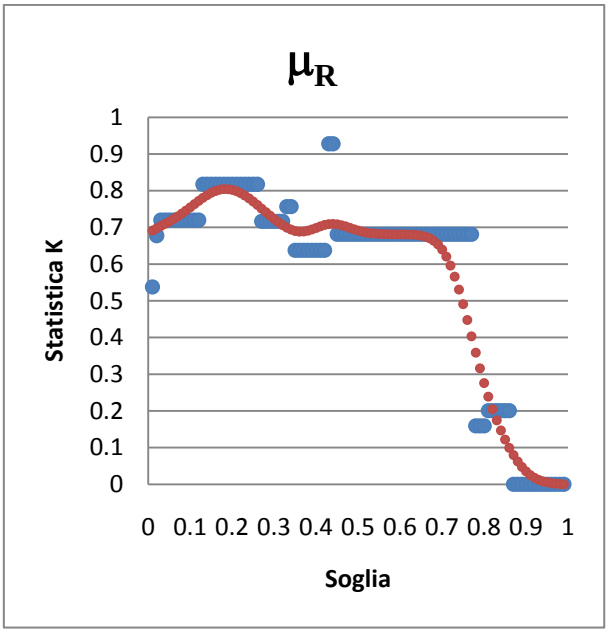


Fig. 31

Facendo riferimento a μ_R, la soglia *best* in quest'esempio risulta intorno a 0.2, mentre usando solo i valori di K si sarebbe selezionato un valore tra 0.4 e 0.5 che, pur migliore in assoluto, si trova però in un contesto di valori mediamente più bassi. Questi picchi manifestano la presenza di valori di previsione molto vicini sulla scala continua, ancorché correttamente orientati verso l'1 nei casi di presenza osservata e verso lo 0 in quelli di assenza. La loro vicinanza fa sì che solo pochi valori di soglia

che cadono al loro interno permettano una previsione più corretta in termini discreti, mentre con soglie esterne le prestazioni possono decadere anche notevolmente. Valori elevati più stabili al variare della soglia, come quelli ottenuti con soglie intorno a 0.2, manifestano invece previsioni ben orientate, ma anche ben distanziate, che esprimono una reale capacità del modello di concentrare correttamente le previsioni di assenza e presenza verso gli estremi del continuum 0-1.

Dunque, per ogni modello, sia monospecie che multispecie, un algoritmo elaborato *ad hoc* all'interno di delphish2.exe (**BTFA**, Best Threshold Finder Algorithm) cerca per ogni specie la soglia *best* in base a μ_R e determina il valore di K-*best*, relativo a quella soglia, da confrontare con il valore di K ottenuto con una soglia tradizionale a 0.5.

Una volta addestrati i modelli monospecie e multispecie (ipotesi 3) su tutte le partizioni studiate per incrociare le condizioni relative alle ipotesi 4 e 5, l'operazione di determinazione dei K tradizionali e dei K-*best* (ipotesi 1) su tutti i modelli ottiene i risultati necessari per compiere i confronti previsti dal disegno sperimentale.

Poiché, come già illustrato nel capitolo precedente, in letteratura si sostiene da più parti l'importanza di ottenere anche misure della performance indipendenti dalla scelta di una soglia e poiché prove preliminari hanno scoraggiato l'uso dell'area sotto la curva del grafico ROC[71], si è cercato di sviluppare un semplice indice alternativo adatto allo scopo. Il punto di partenza è ancora il grafico con le variazioni della statistica K in funzione della soglia (Fig. 28): l'area sotto la spezzata individuata dai punti del grafico può essere infatti assunta come misura sintetica della capacità generale (indipendente dalla soglia) di un modello di fornire previsioni vicine allo 0 nei casi di assenza e vicine all'1 nei casi di presenza (Fig. 32).

[71] Si veda in proposito quanto riportato nel capitolo precedente.

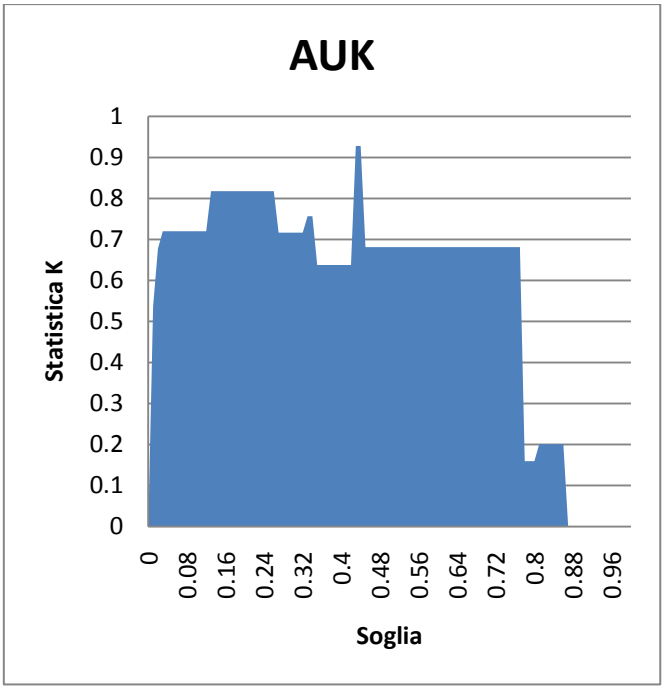


Fig. 32

L'indice (**AUK**, Area Under K-curve) si basa sul rapporto tra l'area individuata nel grafico e il suo massimo teorico:

$$AUK = \frac{area\ K}{\max(area\ K)} = \frac{\frac{\varepsilon}{2}\sum_{i=1}^{m}(k_i + k_{i-1})}{\frac{\varepsilon}{2} \cdot 2(m-1)} = \frac{1}{2(m-1)}\sum_{i=1}^{m}(k_i + k_{i-1})$$

con

$$-1 \leq AUK \leq 1.$$

La formula fa sì che in presenza di valori di K negativi, l'area sotto l'asse delle ascisse venga semplicemente sottratta a quella sopra. Si ottiene AUK=1 quando tutti i valori di K al variare della soglia sono uguali a 1, cioè quando il modello fornisce valori di previsione per le presenze sempre maggiori o uguali a 1- ε e valori di previsione per le assenze sempre minori di ε (viceversa, quando AUK=-1).

Poiché questa misura di performance rende priva di senso la scelta di una specifica soglia, essa non può essere usata per confrontare le condizioni legate all'ipotesi 1, ma contribuirà solo alla verifica delle ipotesi 3, 4 e 5.

E' importante sottolineare la peculiarità dell'informazione veicolata da questo indice: nonostante le previsioni di ogni modello *debbano* essere lette tramite una soglia, una misura dell'efficacia legata a tutte le soglie possibili valuta la performance in un modo robusto rispetto al variare della soglia stessa ed esprime la capacità del modello di essere affidabile anche con scelte non ottimali della soglia. Basandosi su tutto il range di variazione della soglia, AUK si rivela inoltre una misura molto severa dell'efficacia dei modelli (che tende cioè raramente ad assumere valori vicini ad 1).

Come già chiarito nel capitolo precedente, la parte inferenziale del confronto tra le varie condizioni sperimentali è affidata al test di Wilcoxon-Mann-Whitney applicato alle distribuzioni di valori di K legate alle condizioni sperimentali stesse. Ritenendo utile, a livello puramente descrittivo, sintetizzare ognuna di queste distribuzioni anche con un indice di tendenza centrale in grado di rappresentare in modo generale la performance del modello connesso e di confrontarla con la performance degli altri, la scelta è caduta inizialmente sulla mediana, particolarmente adatta data l'eterogeneità delle distribuzioni che ci si poteva attendere. Purtroppo, in alcune condizioni relativamente frequenti, la mediana non riusciva ad esprimere in modo adeguato le differenze tra distribuzioni. Esempi fittizi, ma aderenti a quanto osservato (Fig. 33), riguardano distribuzioni con poco più del 50% dei valori pari a 0 e distribuzioni con più del 90% dei valori pari a 0 (esempio A): la mediana qui dà 0 per entrambe; oppure distribuzioni con poco più del 50% dei valori pari a 0 e una media dei restanti valori molto diversa (esempio B): anche qui la mediana dà comunque 0; oppure, ancora, distribuzioni in cui la mediana è uguale ma ciò che vi sta intorno decisamente no (esempio C).

	Esempio A		Esempio B		Esempio C		Esempio D	
	Distrib. 1	Distrib. 2	Distrib. 1	Distrib. 2	Distrib. 1	Distrib. 2	Distrib. 1	Distrib. 2
1	-0.07	-0.07	-0.12	-0.12	-0.35	0.11	0	0
2	-0.03	-0.03	-0.05	-0.05	-0.27	0.11	0.01	0.01
3	0	0	-0.02	-0.02	-0.27	0.11	0.02	0.02
4	0	0	0	0	-0.27	0.11	0.03	0.03
5	0	0	0	0	-0.27	0.11	0.04	0.04
6	0	0	0	0	-0.19	0.11	0.05	0.05
7	0	0	0	0	-0.19	0.11	0.06	0.06
8	0	0	0	0	-0.19	0.14	0.07	0.07
9	0	0	0	0	-0.19	0.14	0.08	0.08
10	0	0	0	0	-0.19	0.14	0.09	0.09
11	0	0	0	0	-0.19	0.15	0.1	0.1
12	0	0	0	0	-0.19	0.15	0.11	0.11
13	0	0	0	0	-0.19	0.15	0.12	0.12
14	0	0	0	0	0	0.15	0.13	0.13
15	0	0	0	0	0	0.15	0.14	0.14
16	**0**	**0**	**0**	**0**	**0.15**	**0.15**	**0.15**	**0.15**
17	0.9	0	0	0	0.15	0.23	0.16	0.16
18	0.9	0	0	0	0.15	0.23	0.17	0.17
19	0.9	0	0.1	0.9	0.15	0.23	0.18	0.18
20	0.9	0	0.1	0.9	0.15	0.46	0.19	0.19
21	0.9	0	0.1	0.9	0.15	0.46	0.2	0.2
22	0.9	0	0.1	0.9	0.15	0.46	0.21	0.21
23	0.9	0	0.1	0.9	0.15	0.46	0.22	0.22
24	0.9	0	0.1	0.9	0.15	0.46	0.23	0.23
25	0.9	0	0.1	0.9	0.15	0.65	0.24	0.24
26	0.9	0	0.1	0.9	0.15	0.65	0.25	0.25
27	0.9	0	0.1	0.9	0.15	0.87	0.26	0.26
28	0.9	0	0.1	0.9	0.15	0.87	0.27	1
29	0.9	0.9	0.1	0.9	0.15	0.87	0.28	1
30	0.9	0.9	0.1	0.9	0.17	0.87	0.29	1
31	0.9	0.9	0.1	0.9	0.17	0.87	0.3	1
Mediana	**0**	**0**	**0**	**0**	**0.15**	**0.15**	**0.15**	**0.15**
Media	**0.43**	**0.08**	**0.04**	**0.37**	**-0.02**	**0.35**	**0.15**	**0.24**
WMAB	**0.39**	**0**	**0.02**	**0.2**	**0.04**	**0.22**	**0.15**	**0.15**

Fig. 33

Nei tre esempi citati la media si comporta meglio della mediana, riuscendo a registrare le differenze esistenti tra le distribuzioni. Si nota però, soprattutto nell'esempio D, ma in parte anche negli altri, come la media tenda ad essere

sensibile alle code delle distribuzioni, amplificando a volte eccessivamente le differenze in base a valori che rischiano di essere particolarmente bassi o elevati per effetto del caso. Si è dunque cercato di individuare un indice sintetico che mettesse insieme le caratteristiche desiderabili della mediana e della media, ma che non risentisse troppo dei loro limiti[72]. Dopo alcune prove, tale indice è stato individuato in una forma di media pesata centrata sulla mediana (μ_M), caratterizzata da una funzione dei pesi che assegna peso 1 al valore legato alla posizione mediana nella distribuzione ordinata e pesi via via decrescenti man mano che ci si allontana, simmetricamente, da quella posizione (Fig. 34).

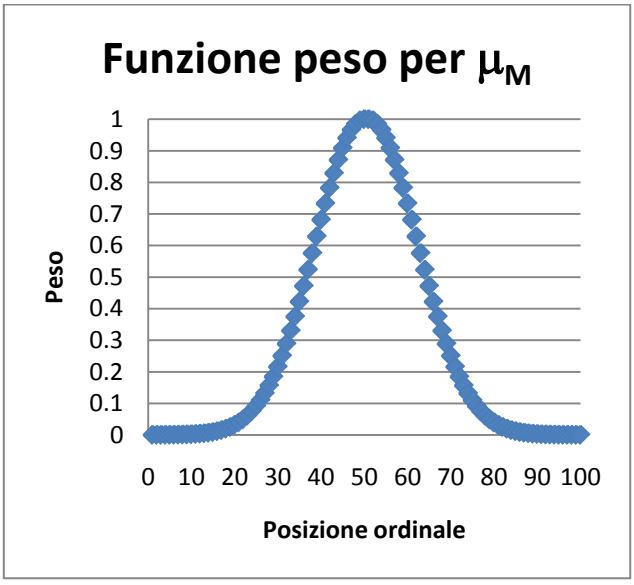


Fig. 34

La generica funzione peso per l'indice μ_M è

$$w(i,n) = \begin{cases} n\ dispari, & e^{\ln a \left(\frac{2i-n-1}{1-n}\right)^2} \\ n\ pari, & \begin{cases} i \le \frac{n}{2}, & e^{\ln a \left(\frac{2i-n}{2-n}\right)^2} \\ i > \frac{n}{2}, & e^{\ln a \left(\frac{2i-n-2}{2-n}\right)^2} \end{cases} \end{cases}$$

[72] Tale operazione ha l'unico scopo di evidenziare a livello descrittivo alcune caratteristiche dei risultati qui ottenuti e non quello di proporre un nuovo indice di tendenza centrale.

in cui il parametro *a* definisce il peso che si desidera venga assegnato alle posizioni estreme (1 e *n*). Ponendo *a*=0.0001 si ottiene che le posizioni esterne al 50% centrale della distribuzione ordinata ricevano pesi sempre inferiori a 0.1.

Dunque, posto che *n* sia il numero di valori presenti nella distribuzione di K considerata, l'indice μ_M può essere così calcolato:

$$\mu_M(K) = \frac{\sum_{i=1}^{n} k_i w(i)}{\sum_{i=1}^{n} w(i)}$$

L'indice ha il pregio (rispetto alla mediana) di utilizzare tutta l'informazione presente nella distribuzione e (rispetto alla media) di essere molto sensibile a ciò che accade in un intorno della posizione mediana piuttosto che in corrispondenza delle posizioni ordinali più estreme (Fig. 34).

3.2. Risultati delle prove

Il modo più immediato per visualizzare i risultati è affiancare i boxplot relativi alle distribuzioni delle performance ottenute per ogni specie nelle varie condizioni (Fig. 35). Ad ogni specie è dedicata una coppia di grafici: il primo, a sinistra, riporta le distribuzioni dei valori di K nelle 16 condizioni sperimentali; il secondo, a destra, riporta le distribuzioni dei valori di AUK nelle 8 condizioni sperimentali che restano se si esclude l'ipotesi 1, come spiegato nel paragrafo precedente.

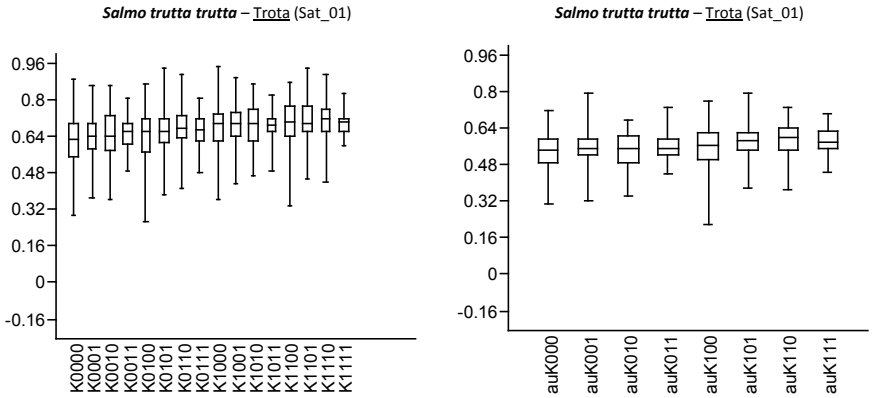


Fig. 35 (pagg. 97-108)

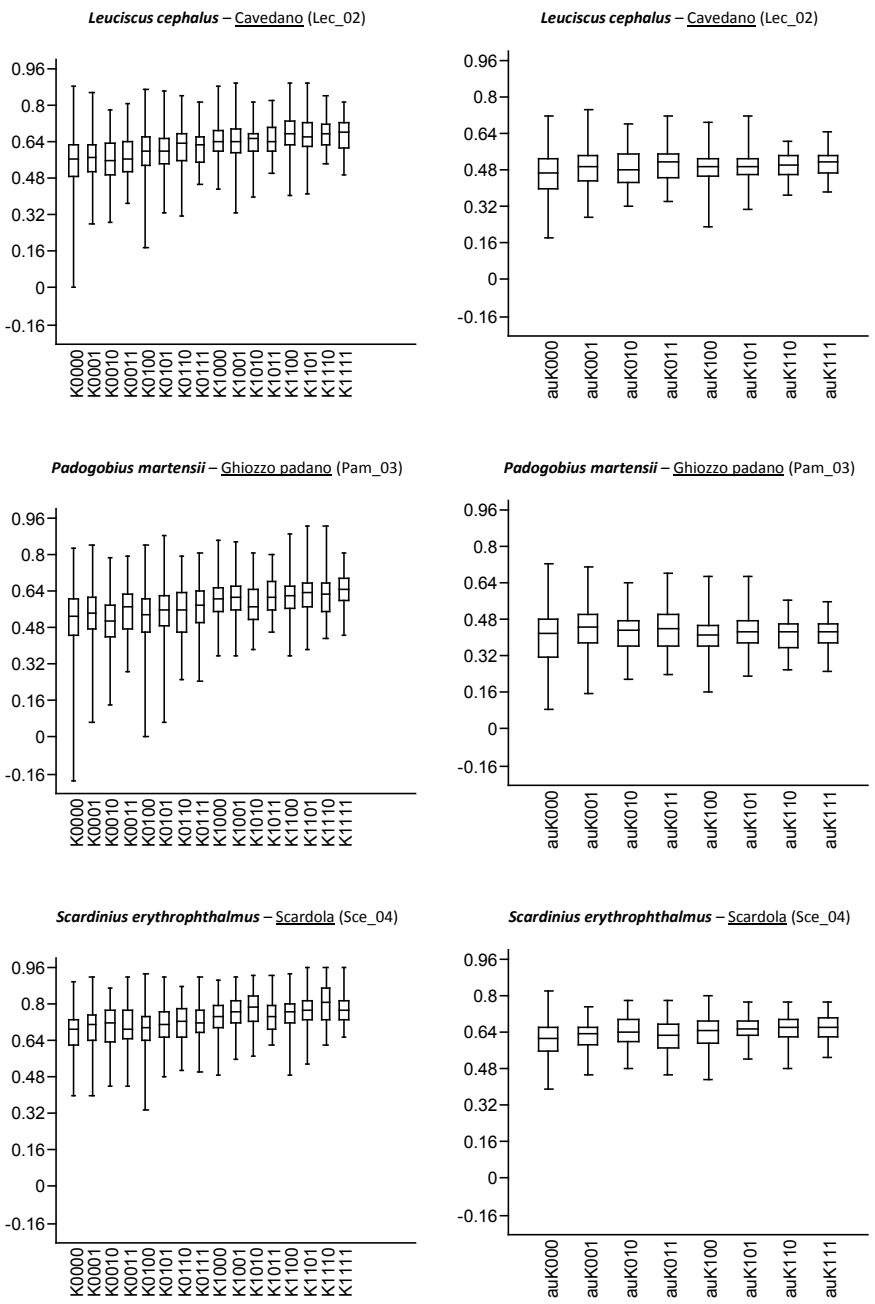


Fig. 35 (pagg. 97-108)

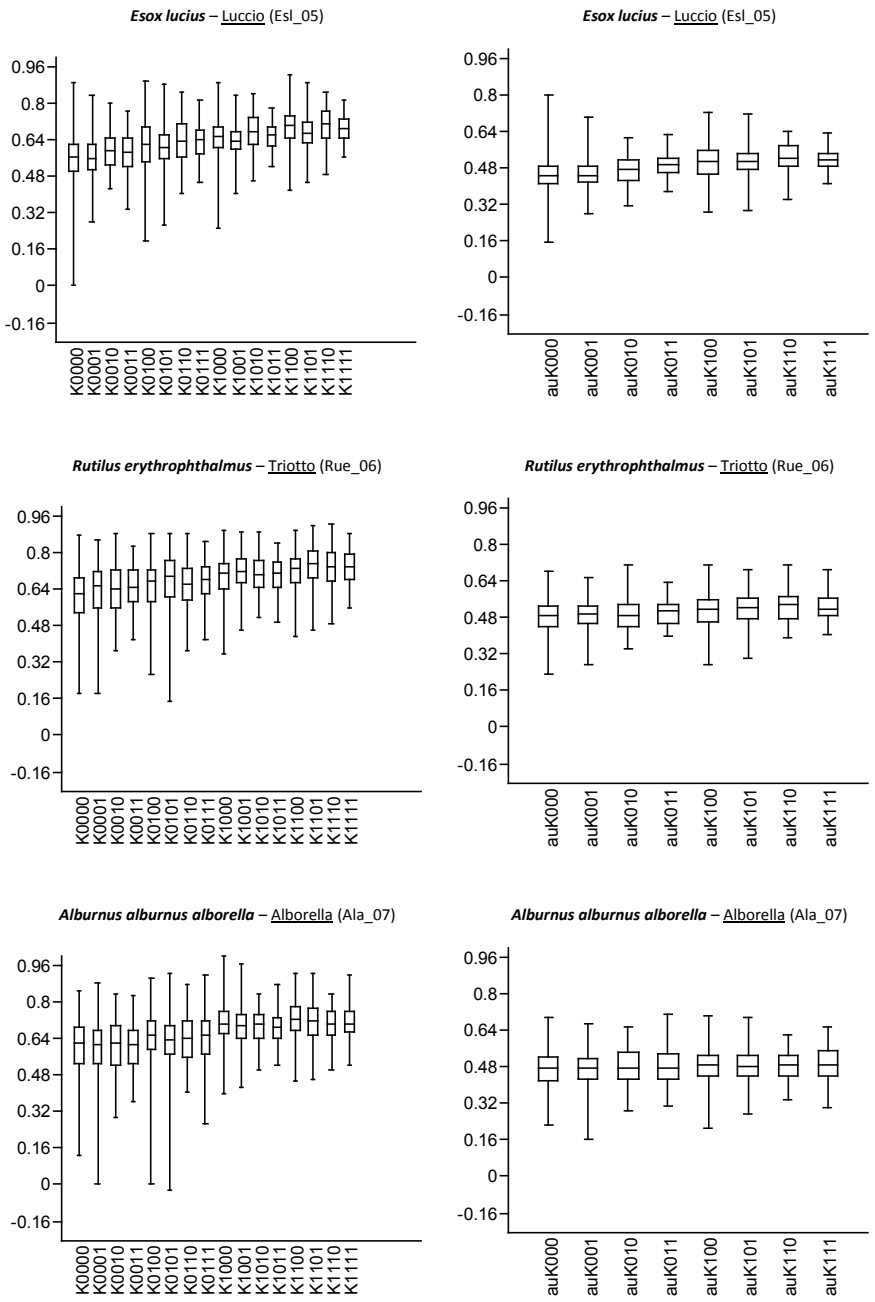


Fig. 35 (pagg. 97-108)

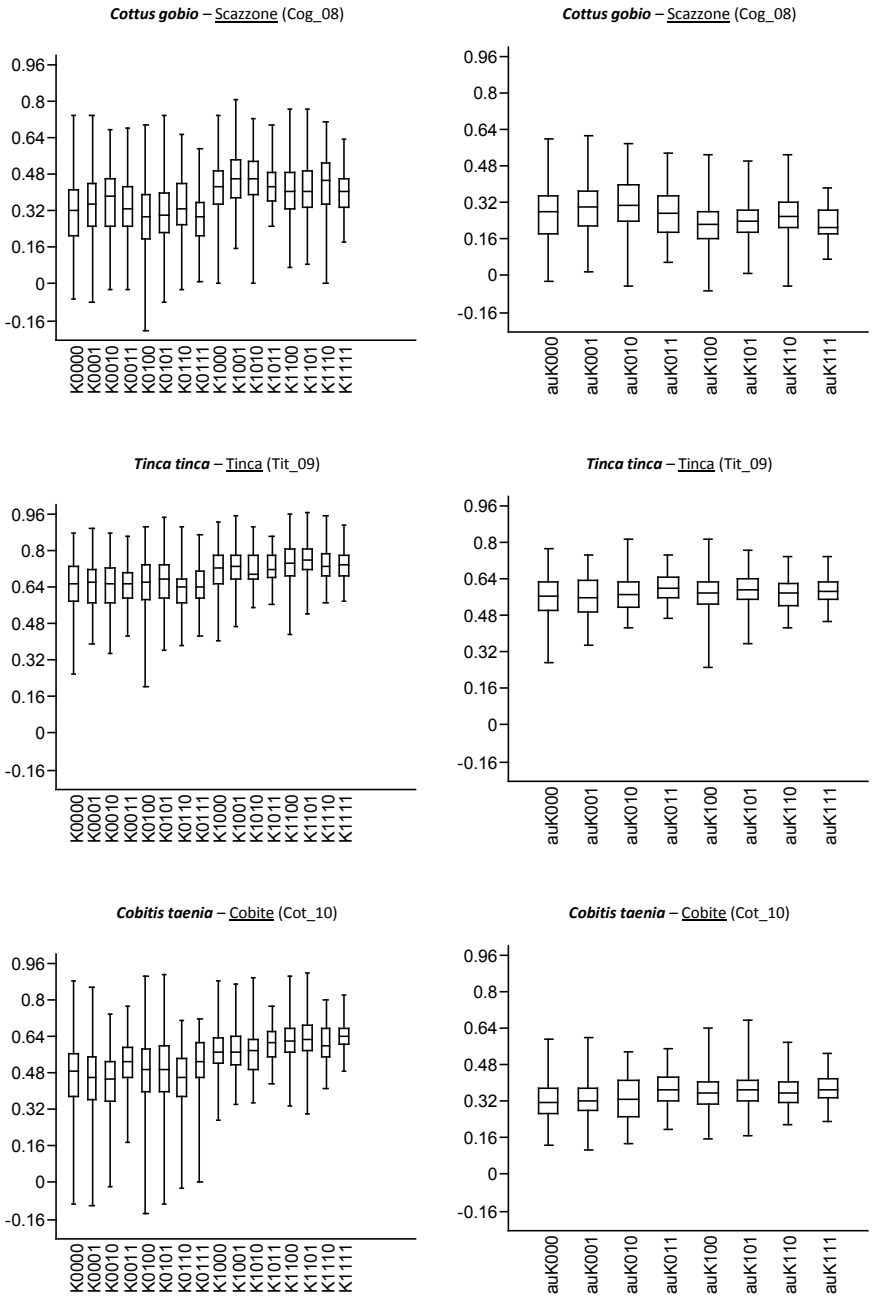


Fig. 35 (pagg. 97-108)

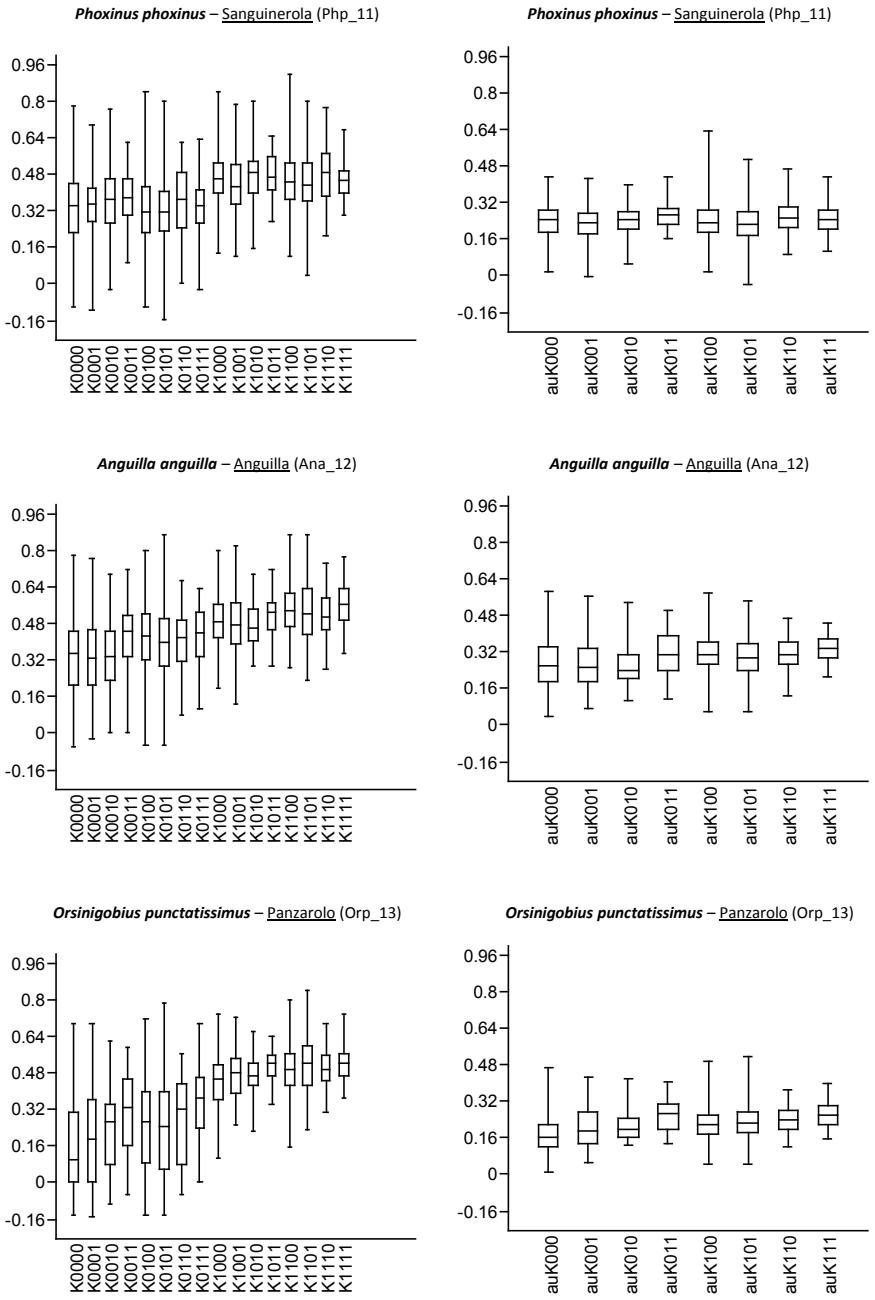


Fig. 35 (pagg. 97-108)

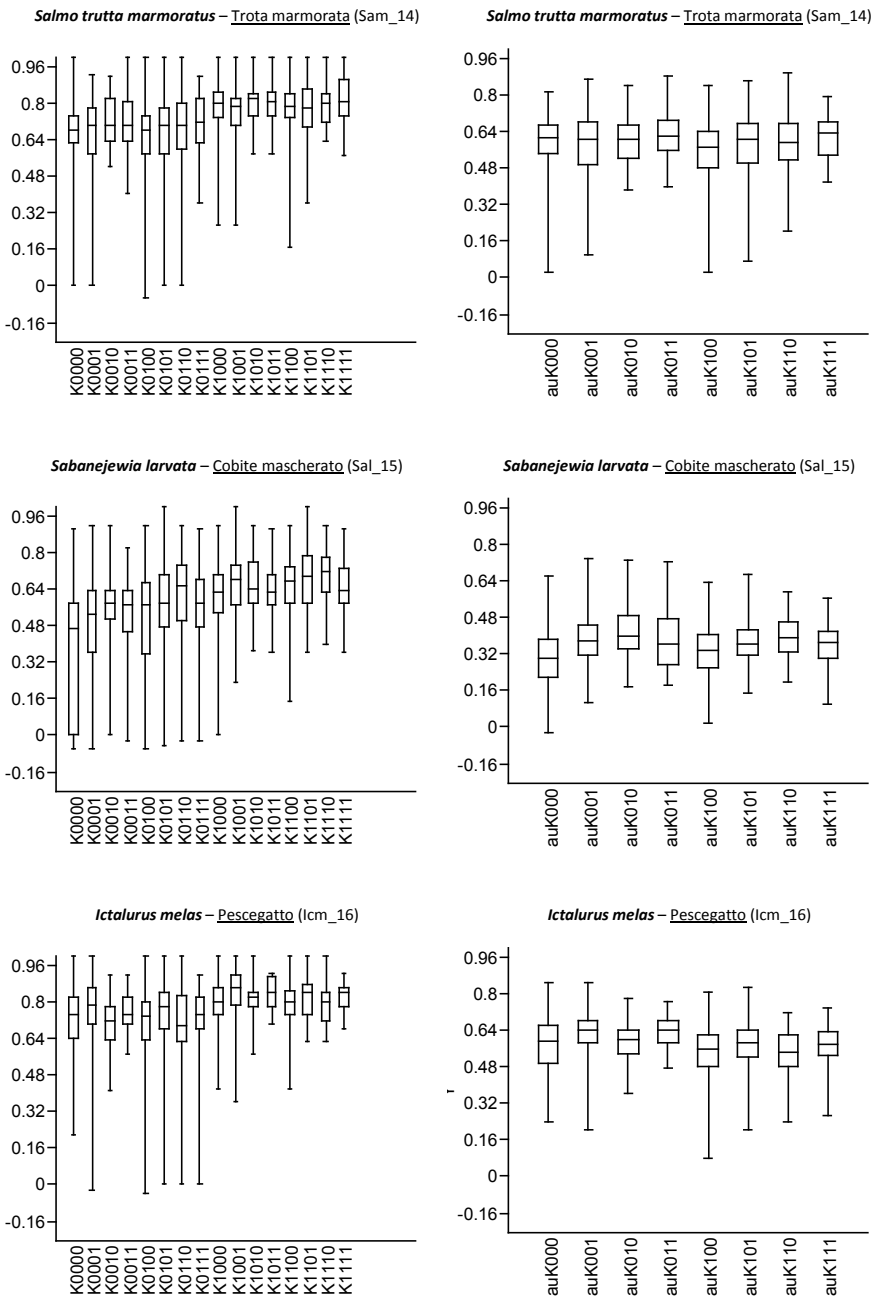


Fig. 35 (pagg. 97-108)

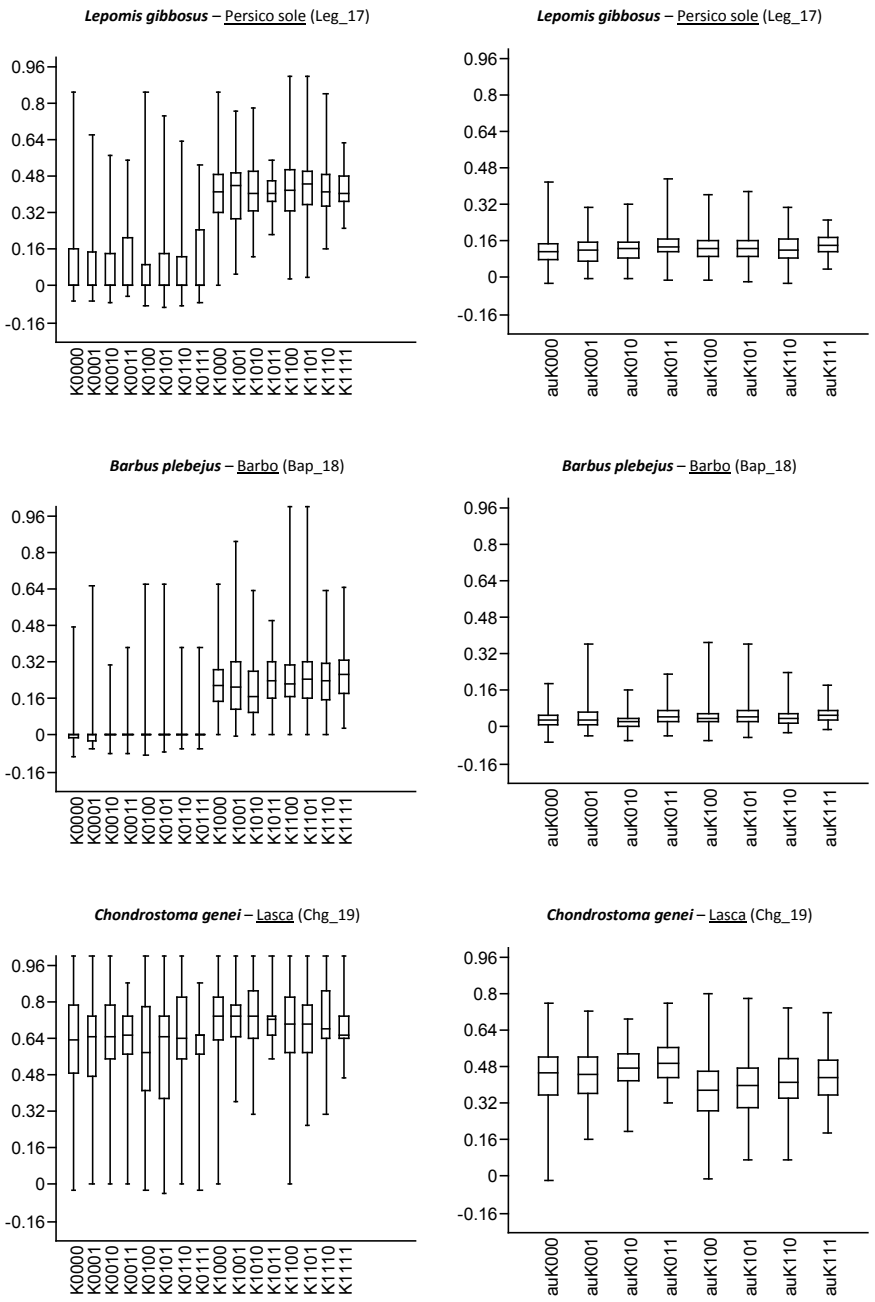


Fig. 35 (pagg. 97-108)

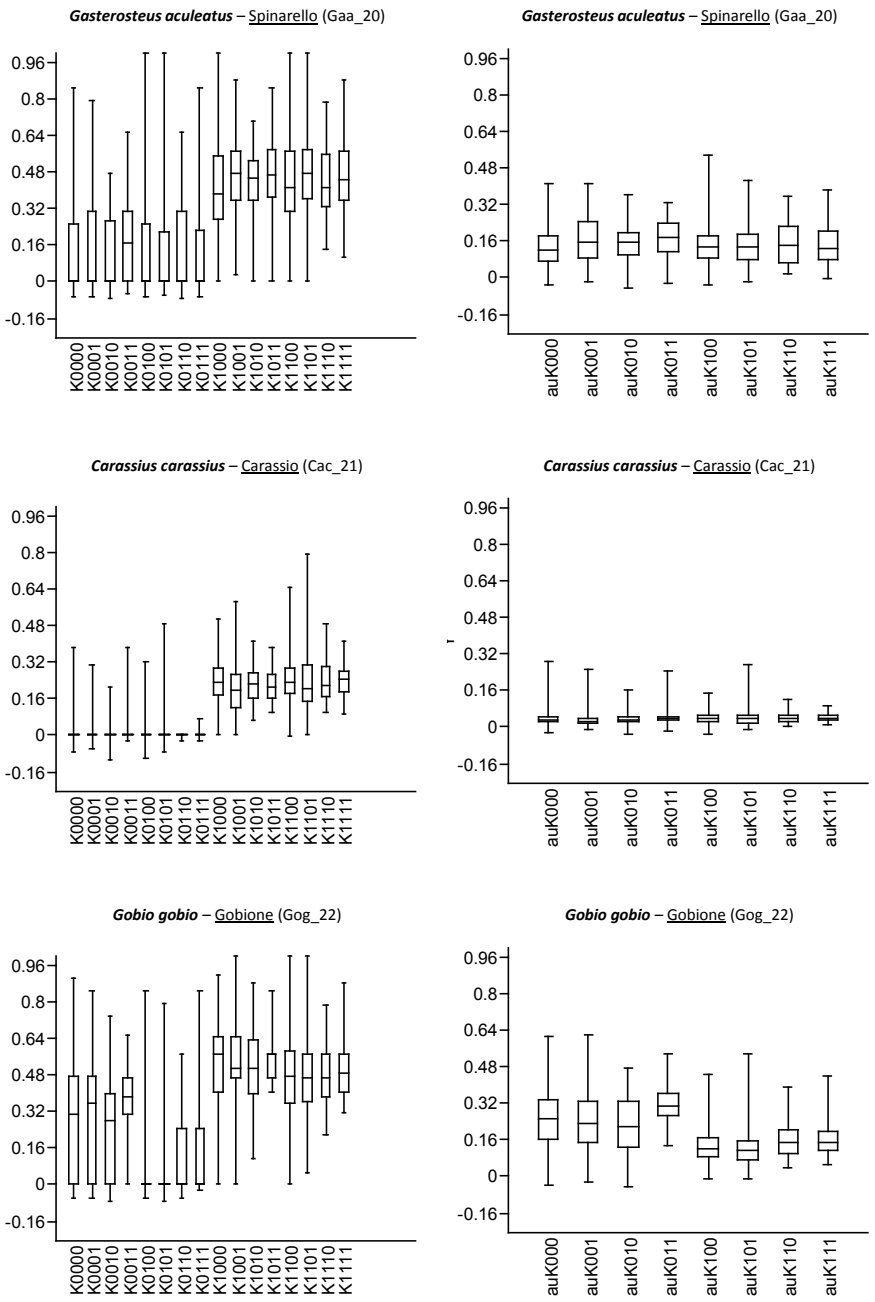


Fig. 35 (pagg. 97-108)

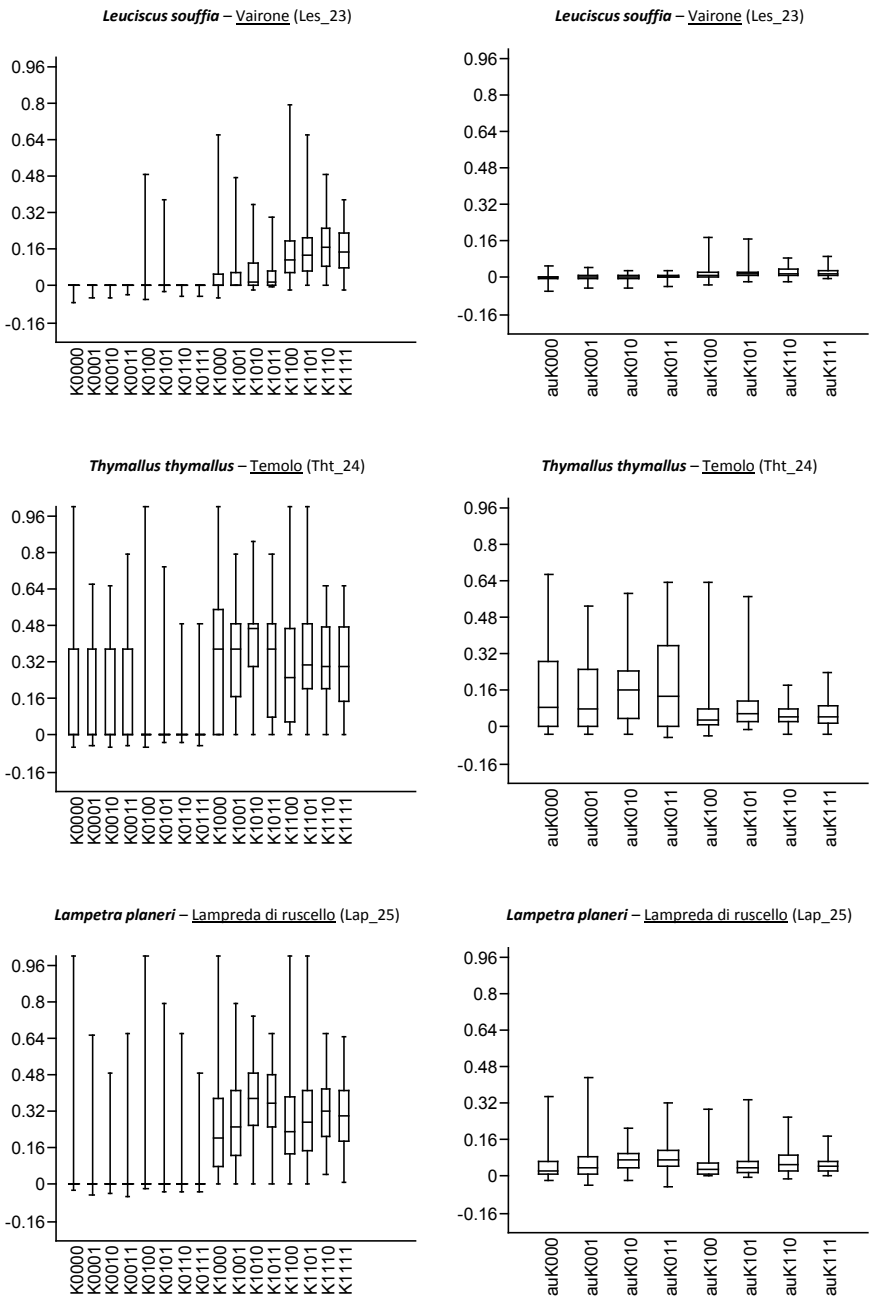


Fig. 35 (pagg. 97-108)

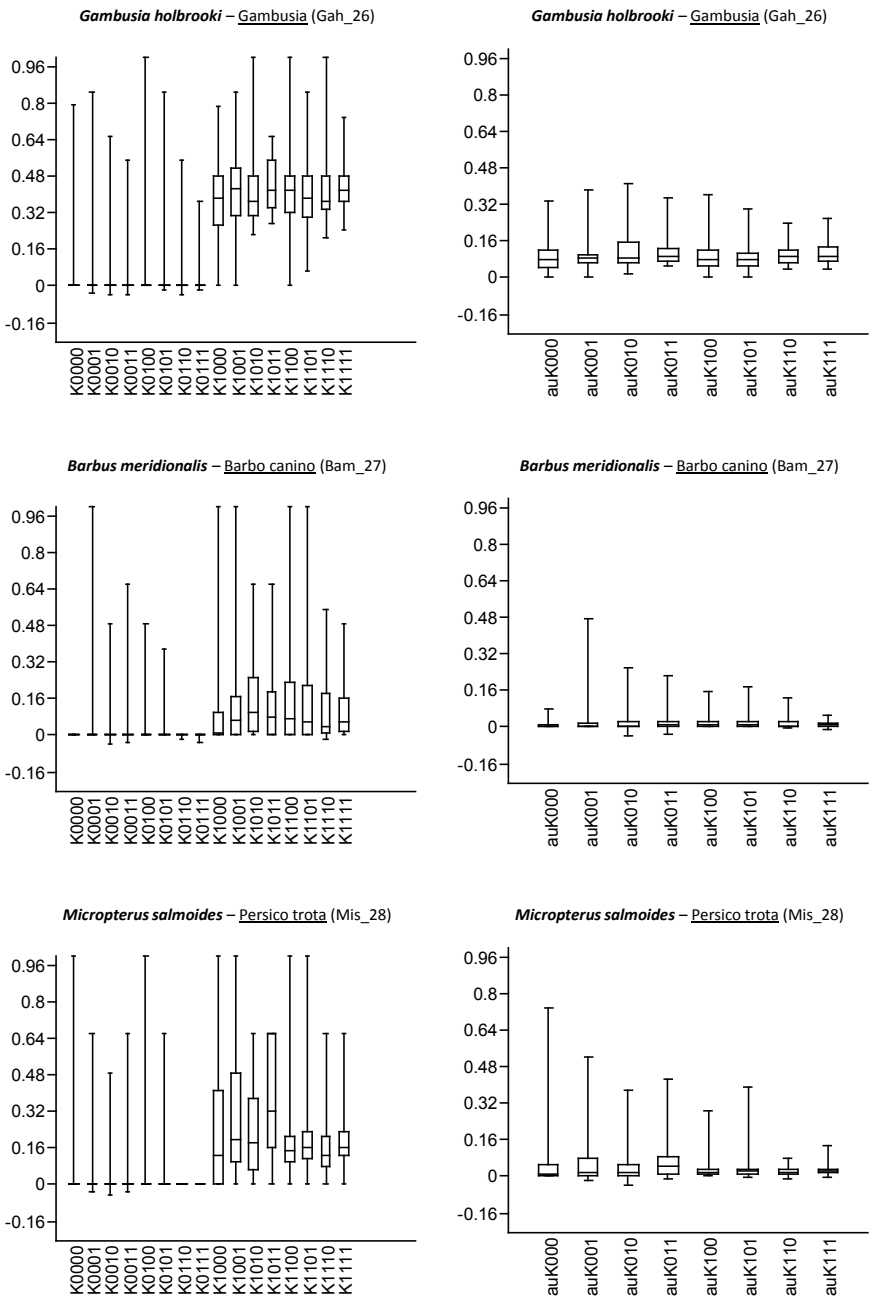


Fig. 35 (pagg. 97-108)

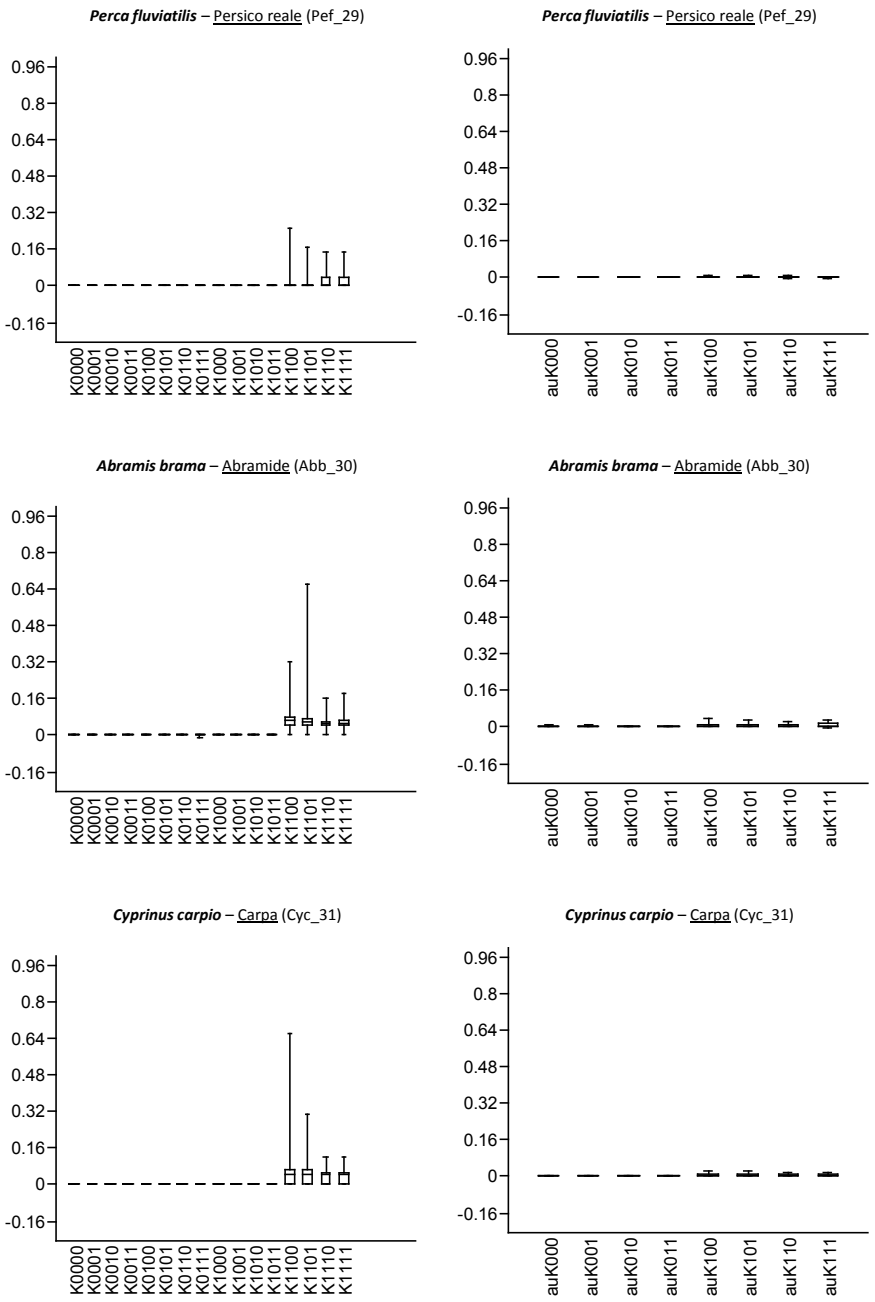


Fig. 35 (pagg. 97-108)

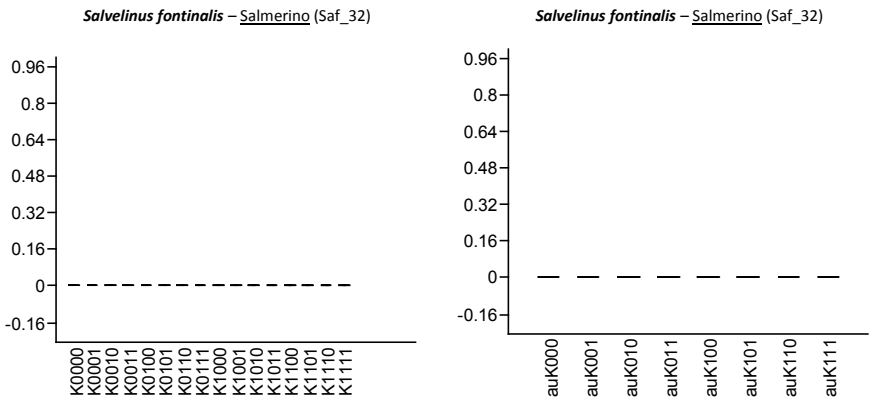


Fig. 35 (pagg. 97-108)

Già un rapido sguardo ai risultati rappresentati tramite boxplot permette di notare alcuni fatti macroscopici.

Innanzitutto alcune specie, indipendentemente dal modello scelto, vengono predette meglio di altre; ad esempio, hanno mediane complessivamente più elevate e/o variabilità ridotte. Facendo riferimento ai grafici di sinistra e sintetizzando ogni distribuzione con l'indice μ_M[73] è possibile ad esempio mostrare per ogni specie il valore di performance più basso ottenuto tra i 16 confrontati (barre in blu in Fig. 36), evidenziando il fatto che con alcune specie anche una scelta casualmente infelice del modello può dare buoni risultati, mentre con altre questo non accade.

[73] Si veda il paragrafo precedente, 3.1.

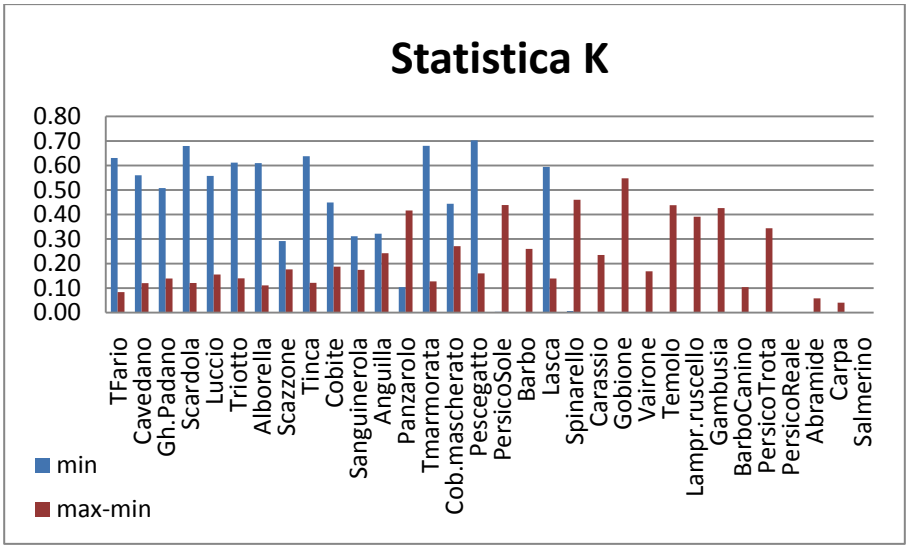


Fig. 36

Ricordando che le specie sono disposte in ordine di frequenza decrescente da sinistra a destra, le barre blu mostrano anche una tendenza delle specie più frequenti ad essere predette meglio di quelle meno frequenti, fenomeno già introdotto nel primo capitolo e legato alla quantità di informazione che l'algoritmo di training ha a disposizione per addestrare i modelli. Secondo quanto si diceva, si osservano anche le eccezioni di alcune specie relativamente frequenti in cui la performance è ridotta (scazzone, sanguinerola, anguilla, panzarolo) e di specie relativamente rare con prestazioni buone (lasca): ciò è dovuto al fatto che l'informazione utile non dipende solo dalla frequenza. E' interessante notare che con quasi la metà delle specie la performance predittiva minima (barre blu) è prossima allo zero.

Nella stessa figura (Fig. 36) le barre rosse indicano per ogni specie la differenza tra il valore massimo di performance ottenuto tra i 16 modelli e quello minimo, mostrando il massimo incremento possibile di prestazione che si può ottenere scegliendo opportunamente un modello piuttosto che un altro. Il fenomeno più evidente qui è che proprio le specie che mostravano maggiori problemi di previsione presentano il maggiore incremento di prestazione, evidenziando l'importanza di lavorare sulle caratteristiche dei modelli per ottenere buoni risultati con tutte le specie del popolamento. Con alcune specie l'incremento è notevole in assoluto (anguilla, panzarolo, cobite mascherato, persico sole, barbo, spinarello, carassio, gobione, temolo, lampreda di ruscello, gambusia e persico trota), mentre

con altre è da notare il passaggio tra una totale assenza di previsione e una possibilità di previsione almeno minima (persico sole, barbo, spinarello, carassio, gobione, vairone, temolo, lampreda di ruscello, gambusia, barbo canino, persico trota, abramide e carpa).

Ci sono comunque due casi, il persico reale e il salmerino, davvero troppo rari perché un qualsiasi modello ne potesse ricostruire le relazioni con i descrittori ambientali e con le altre specie. Si tratta tuttavia di due casi particolari (ai quali possono essere associati anche l'abramide e la carpa) poiché il limite principale alla loro frequenza nel data set è dato dal campionamento stesso, condotto in massima parte su tratti montani e pedemontani che si trovano ai margini del naturale areale di queste specie. Pertanto non si può concludere nulla di generale circa l'insuccesso nella previsione di queste specie.

In Fig. 36 si notano anche i risultati ottenuti con specie relativamente facili da predire (trota fario, cavedano, ghiozzo padano, scardola, luccio, triotto, alborella, tinca, cobite, trota marmorata, pesce gatto e lasca). Nonostante sia più importante, per certi versi, rendere predicibili specie difficili da predire, anche il miglioramento nella previsione di specie "facili" costituisce un risultato importante nel contesto integrato della previsione di un popolamento.

Il medesimo grafico, ottenuto però con AUK (Fig. 37) facendo riferimento alle distribuzioni rappresentate nei grafici di destra di Fig. 35, mette in evidenza soprattutto un aspetto dei risultati: la scelta della soglia di discretizzazione dell'output è decisiva nel determinare la performance reale dei modelli. Infatti nel diagramma a barre di AUK, che non contempla le differenze dovute alla scelta della soglia, non si osservano i netti miglioramenti rilevati precedentemente (barre rosse); non si osserva però neanche quel 50% di risultati minimi (barre blu) prossimi allo zero che era evidente in Fig. 36.

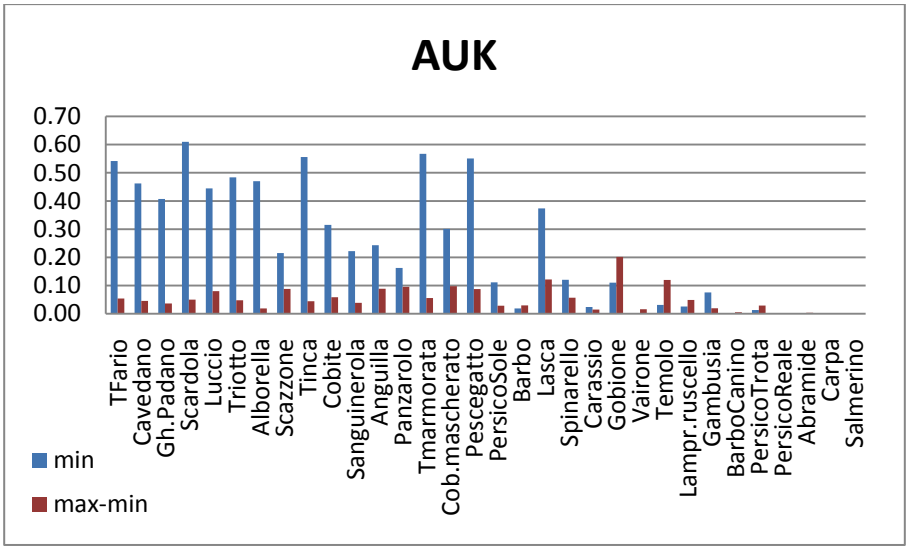


Fig. 37

Il fatto che gran parte degli incrementi di prestazione siano dovuti alla scelta di un'opportuna soglia è testimoniato, nei grafici di sinistra in Fig. 35, anche dalla differenza tra le mediane delle prime 8 distribuzioni (senza soglia ottimizzata) e quelle delle altre 8 (con soglia ottimizzata). In alcuni dei casi in cui tale differenza non è netta, si nota comunque una riduzione della variabilità nei modelli con soglie ottimizzate, a testimonianza di una maggiore stabilità dei modelli al variare dei dati usati durante il training.

Tuttavia la Fig. 37 mostra anche la presenza di miglioramenti relativamente consistenti (scazzone, anguilla, panzarolo, cobite mascherato, gobione, temolo, lampreda di ruscello) o comunque di miglioramenti notevoli rispetto al minimo di partenza (barbo, spinarello, carassio, vairone, lampreda di ruscello, persico trota), non dovuti alla scelta della soglia.

Un'osservazione attenta dei grafici (soprattutto di destra, ma anche di sinistra) in Fig. 35 lascia intravedere che nella maggior parte dei casi tali miglioramenti sono legati all'ipotesi 3 e dunque alla scelta di modelli monospecie (scazzone, pesce gatto, lasca, gobione, temolo, persico trota) o multispecie (luccio, anguilla, panzarolo, vairone, persico reale, abramide, carpa). Ad esempio, se si guarda il grafico di destra relativo alla lasca (specie 19, *Chondrostoma genei*) e si confrontano i modelli contrapposti per la terza ipotesi ma omologhi per le altre (000 *vs.* 100, 001 *vs.* 101, 010 *vs.* 110, 011 *vs.* 111), si vedrà che in ogni coppia il primo modello, cioè

quello monospecie, ha sempre una mediana più elevata e una minore dispersione. Per il vairone (specie 23, *Leuciscus souffia*), invece, si verifica l'opposto (preferenza per il modello multispecie), ma in questo caso la maggiore evidenza si ha guardando il grafico di sinistra ed effettuando i confronti 1000 *vs.* 1100, 1001 *vs.* 1101, 1010 *vs.* 1110, 1011 *vs.* 1111. Questo accade perché il vairone in questo data set è una specie difficile da predire e senza una soglia ottimizzata quasi non si hanno risultati validi (primi 8 boxplot del grafico a sinistra), per cui l'indice AUK (grafico a destra) tende soprattutto a registrare la tendenza generale dei modelli a non dare previsioni efficaci; quando però la soglia viene ottimizzata (secondo gruppo di 8 boxplot nel grafico di sinistra) allora l'effetto della scelta tra modello monospecie e multispecie diviene evidente, benché il grafico a destra la segnali solo debolmente.

Questo del vairone è uno dei pochi casi di *interazione* rilevati in questo studio. Un altro è quello del gobione (specie 22, *Gobio gobio*) per il quale, in assenza di ottimizzazione della soglia, l'effetto di preferenza per il modello monospecie risulta amplificato. Interazioni simili sono rilevabili anche per altre specie, ma la loro intensità è in genere trascurabile, forse anche a causa della differenza tra gli ordini di grandezza degli effetti determinati dalle alternative delle diverse ipotesi. Questo risultato non era prevedibile; anzi, la ragione per cui è stato progettato un disegno sperimentale che incrocia le diverse combinazioni di alternative era proprio legata alla possibile presenza di interazioni tra le ipotesi.

Per quanto riguarda le altre due ipotesi, gli effetti macroscopici rilevabili con un'occhiata ai boxplot sono molti meno: il panzarolo (specie 13, *Orsinigobius punctatissimus*), la lampreda di ruscello (specie 25, *Lampetra planeri*) e il persico reale (specie 29, *Perca fluviatilis*) mostrano una preferenza per l'ordinamento in base a presenza/assenza, mentre il panzarolo, il pesce gatto (specie 16, *Ictalurus melas*) e il persico trota (specie 28, *Micropterus salmoides*) beneficiano dell'ordinamento in base all'altitudine.

L'insieme dei valori di sintesi (μ_M) relativi a tutti i modelli per le varie specie può essere raggruppato e rappresentato in forma tabellare, sia per quanto riguarda la statistica K con i 16 modelli relativi a tutte le ipotesi (Tab. 5), che l'indice AUK con gli 8 modelli relativi alle ipotesi 3, 4 e 5 (Tab. 6).

Specie	0-0-0-0	0-0-0-1	0-0-1-0	0-0-1-1	0-1-0-0	0-1-0-1	0-1-1-0	0-1-1-1	1-0-0-0	1-0-0-1	1-0-1-0	1-0-1-1	1-1-0-0	1-1-0-1	1-1-1-0	1-1-1-1
1 Sat	0.63	0.64	0.65	0.65	0.66	0.67	0.68	0.67	0.69	0.69	0.69	0.69	0.71	0.71	0.71	0.70
2 Lec	0.56	0.57	0.56	0.56	0.60	0.60	0.63	0.61	0.64	0.64	0.65	0.64	0.68	0.66	0.67	0.68
3 Pam	0.53	0.55	0.51	0.56	0.54	0.56	0.55	0.58	0.60	0.61	0.58	0.62	0.62	0.63	0.63	0.65
4 Sce	0.68	0.71	0.71	0.69	0.70	0.70	0.71	0.72	0.74	0.76	0.78	0.74	0.76	0.77	0.80	0.77
5 Esl	0.56	0.56	0.59	0.59	0.62	0.61	0.64	0.63	0.65	0.63	0.67	0.66	0.70	0.67	0.71	0.69
6 Rue	0.61	0.65	0.64	0.65	0.67	0.69	0.66	0.68	0.70	0.72	0.70	0.70	0.72	0.75	0.74	0.73
7 Ala	0.62	0.61	0.61	0.61	0.65	0.63	0.64	0.64	0.70	0.69	0.69	0.68	0.72	0.71	0.71	0.70
8 Cog	0.32	0.35	0.38	0.34	0.29	0.30	0.33	0.29	0.42	0.46	0.47	0.42	0.41	0.41	0.44	0.40
9 Tit	0.65	0.66	0.64	0.65	0.66	0.67	0.64	0.64	0.72	0.73	0.71	0.72	0.75	0.76	0.73	0.74
10 Cot	0.47	0.46	0.45	0.52	0.50	0.50	0.46	0.54	0.57	0.58	0.57	0.61	0.62	0.63	0.60	0.64
11 Php	0.34	0.35	0.37	0.37	0.32	0.31	0.37	0.35	0.46	0.43	0.47	0.47	0.44	0.44	0.49	0.45
12 Ana	0.34	0.32	0.33	0.44	0.42	0.39	0.41	0.44	0.48	0.47	0.47	0.52	0.54	0.52	0.52	0.56
13 Orp	0.10	0.19	0.24	0.32	0.26	0.24	0.30	0.36	0.45	0.48	0.47	0.51	0.50	0.52	0.50	0.51
14 Sam	0.68	0.70	0.71	0.71	0.68	0.69	0.70	0.72	0.80	0.78	0.81	0.79	0.79	0.78	0.80	0.81
15 Sal	0.44	0.54	0.58	0.55	0.55	0.59	0.63	0.57	0.62	0.67	0.65	0.63	0.67	0.69	0.71	0.64
16 Icm	0.74	0.80	0.72	0.76	0.72	0.76	0.70	0.75	0.81	0.86	0.81	0.84	0.80	0.84	0.80	0.83
17 Leg	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.02	0.40	0.43	0.41	0.41	0.42	0.44	0.42	0.41
18 Bap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.21	0.21	0.17	0.23	0.23	0.24	0.23	0.26
19 Chg	0.62	0.61	0.65	0.66	0.59	0.62	0.63	0.65	0.72	0.72	0.73	0.71	0.70	0.69	0.70	0.66
20 Gaa	0.03	0.07	0.06	0.15	0.03	0.01	0.07	0.01	0.39	0.46	0.45	0.47	0.42	0.47	0.43	0.45
21 Cac	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.19	0.22	0.21	0.23	0.21	0.22	0.23
22 Gog	0.29	0.30	0.25	0.39	0.00	0.00	0.02	0.01	0.55	0.53	0.52	0.54	0.48	0.46	0.49	0.49
23 Les	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.01	0.11	0.13	0.17	0.15
24 Tht	0.06	0.05	0.09	0.08	0.00	0.00	0.00	0.00	0.36	0.36	0.44	0.39	0.26	0.32	0.31	0.31
25 Lap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.21	0.26	0.39	0.35	0.24	0.27	0.33	0.30
26 Gah	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.41	0.37	0.43	0.40	0.40	0.38	0.41
27 Bam	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.06	0.10	0.08	0.08	0.06	0.05	0.07
28 Mis	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.22	0.19	0.34	0.15	0.16	0.13	0.16
29 Pef	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30 Abb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.05	0.05	0.05
31 Cyc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.04	0.03
32 Saf	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Tab. 5

Specie	X-0-0-0	X-0-0-1	X-0-1-0	X-0-1-1	X-1-0-0	X-1-0-1	X-1-1-0	X-1-1-1
1 Sat	0.54	0.55	0.55	0.55	0.56	0.58	0.60	0.58
2 Lec	0.46	0.49	0.48	0.50	0.49	0.49	0.50	0.51
3 Pam	0.41	0.44	0.42	0.43	0.41	0.43	0.42	0.42
4 Sce	0.61	0.63	0.64	0.63	0.64	0.66	0.66	0.66
5 Esl	0.44	0.45	0.47	0.49	0.51	0.51	0.53	0.51
6 Rue	0.48	0.49	0.49	0.50	0.51	0.52	0.53	0.52
7 Ala	0.47	0.47	0.48	0.47	0.48	0.48	0.48	0.49
8 Cog	0.28	0.30	0.30	0.28	0.22	0.24	0.25	0.22
9 Tit	0.56	0.56	0.57	0.60	0.58	0.59	0.57	0.59
10 Cot	0.32	0.32	0.33	0.37	0.35	0.36	0.35	0.37
11 Php	0.24	0.23	0.25	0.26	0.23	0.22	0.25	0.24
12 Ana	0.26	0.25	0.24	0.31	0.31	0.29	0.30	0.33
13 Orp	0.16	0.19	0.19	0.26	0.22	0.22	0.24	0.25
14 Sam	0.61	0.60	0.60	0.62	0.57	0.60	0.59	0.62
15 Sal	0.30	0.37	0.40	0.36	0.33	0.36	0.39	0.36
16 Icm	0.59	0.64	0.59	0.64	0.55	0.59	0.55	0.58
17 Leg	0.11	0.11	0.12	0.13	0.12	0.12	0.12	0.14
18 Bap	0.03	0.03	0.02	0.04	0.03	0.04	0.03	0.05
19 Chg	0.44	0.44	0.47	0.49	0.37	0.39	0.42	0.43
20 Gaa	0.12	0.15	0.15	0.18	0.13	0.13	0.14	0.13
21 Cac	0.03	0.02	0.03	0.03	0.03	0.03	0.04	0.04
22 Gog	0.25	0.23	0.22	0.31	0.12	0.11	0.14	0.15
23 Les	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.01
24 Tht	0.10	0.10	0.15	0.15	0.03	0.06	0.04	0.05
25 Lap	0.03	0.04	0.07	0.07	0.03	0.04	0.05	0.04
26 Gah	0.08	0.08	0.09	0.09	0.08	0.08	0.09	0.09
27 Bam	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.01
28 Mis	0.01	0.02	0.02	0.04	0.02	0.02	0.02	0.02
29 Pef	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30 Abb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
31 Cyc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32 Saf	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Tab. 6

Queste tabelle sono le medesime da cui si sono tratti i due diagrammi a barre precedentemente presentati e possono essere utili, ad un livello descrittivo, anche per ricavare un'idea indicativa di quanto le specie possano preferire una delle due alternative proposte dalle 4 ipotesi. Si parla di "idea indicativa" in quanto non è possibile trarre in questo modo conclusioni generali; lo scopo è semplicemente visualizzare una misura quantitativa della preferenza per una delle due alternative di un'ipotesi (Figg. 38 e 39)[74].

[74] Ogni barra è calcolata come differenza tra la media delle performance dei modelli in cui l'alternativa dell'ipotesi in esame è quella ottimizzante e la media delle performance dei modelli costruiti in base

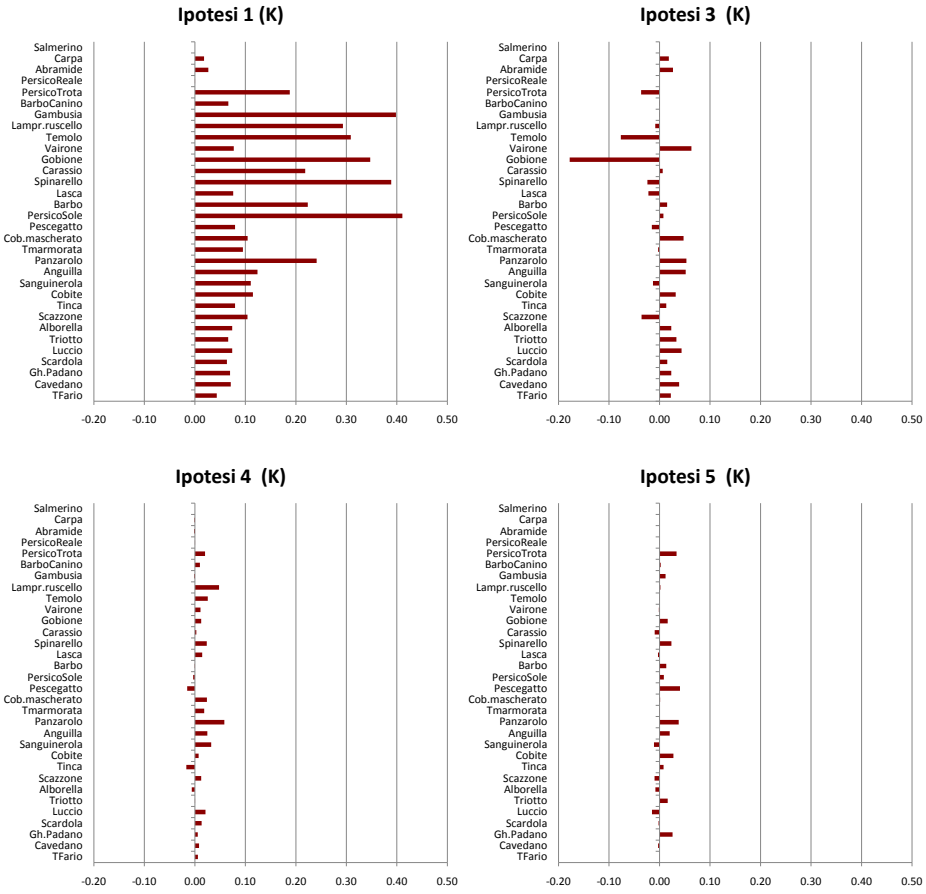


Fig. 38

all'altra ipotesi. Pertanto, un valore positivo (barra a destra) indica la preferenza per l'alternativa considerata ottimizzante, mentre un valore negativo (barra a sinistra) indica una preferenza per l'altra ipotesi. Che si sviluppi verso destra o verso sinistra, la barra è tanto più lunga quanto più grande è la differenza di prestazione tra un'alternativa e l'altra. Quando la barra è molto piccola o assente, le prestazioni legate alle due alternative si equivalgono e dunque non vi è preferenza.

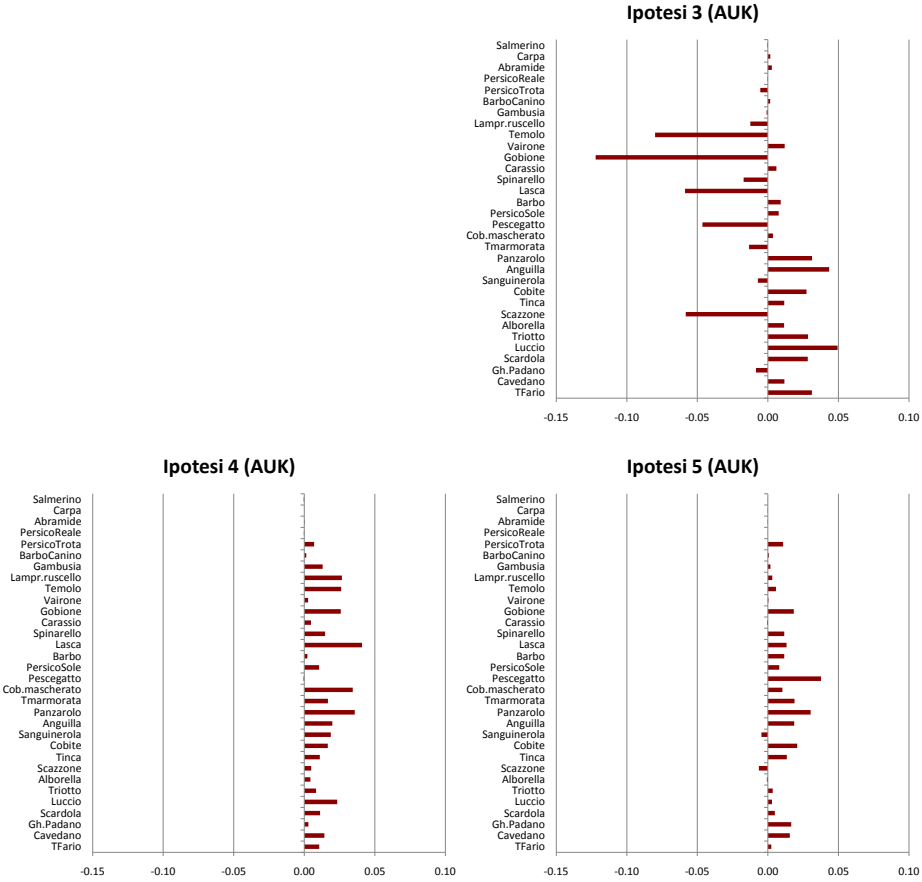


Fig. 39

I grafici ottenuti confermano innanzitutto la grande importanza dell'ottimizzazione della soglia di discretizzazione (ipotesi 1), sia perché a quest'ultima soluzione non viene mai preferito l'uso di una soglia tradizionale a 0.5 (le barre sono tutte orientate a destra), che a causa dell'entità del miglioramento che in molti casi essa produce (fino a più di 0.4), che confrontato con quello indotto dalle scelte relative alle altre ipotesi risulta davvero notevole.

Anche le alternative legate all'ipotesi 3 producono miglioramenti apprezzabili con alcune specie, ma in questo caso non in modo univoco come per la soglia ottimizzata. Le barre sono infatti orientate sia da una parte che dall'altra, ad indicare che alcune specie preferiscono una previsione monospecie (gobione, temolo,

scazzone, lasca, pesce gatto, spinarello, persico trota, ecc.) ed altre una previsione multispecie (luccio, anguilla, vairone, panzarolo, triotto, scardola, cobite, trota, cavedano, alborella, ecc.). Il grafico dà la sensazione che vi siano più specie favorite dalla previsione multispecie (come ci si aspettava), ma che gli effetti positivi di quest'ultima sulle singole specie siano meno pronunciati di quelli prodotti dall'alternativa monospecie.

Le altre due ipotesi propongono alternative la cui scelta dà luogo in generale ad effetti relativamente piccoli in termini di miglioramento della performance media. Tuttavia l'effetto dell'alternativa ottimizzante sembra in entrambi i casi trasversalmente positivo, con poche eccezioni di modesta entità.

I risultati possono essere analizzati a livello descrittivo anche tramite un'analisi delle componenti principali (PCA, Principal Component Analysis) effettuata sulla Tab. 5. Lungo la prima componente (Fig. 40) varia essenzialmente la bontà generale con cui le specie sono predette (dal gruppo persico reale, salmerino, carpa ed abramide, che ottengono le prestazioni peggiori, al pesce gatto e alla trota marmorata che mostrano le performance migliori). Come suggeriscono le barre verdi delle variabili, la seconda componente contribuisce invece a rappresentare, lungo un asse che si può immaginare perpendicolare alla linea rossa, l'effetto che ha la scelta di una soglia ottimizzante (ipotesi 1) sulle performance con le varie specie; l'effetto sulla previsione della carpa o dell'abramide è molto piccolo, mentre via via che ci si sposta verso il basso l'effetto cresce, fino a diventare importante sul gruppo del persico sole, della gambusia, dello spinarello e del gobione. La componente 3 (Fig. 41) sintetizza invece la preferenza per un modello monospecie o multispecie: le specie che si trovano in prossimità della linea arancione sono abbastanza indifferenti alle due alternative, mentre quelle che se ne discostano verso l'alto hanno un proporzionale beneficio dalla scelta del modello multispecie e quelle che se ne discostano verso il basso hanno un proporzionale beneficio con l'alternativa monospecie. Si noti, in particolare, l'importanza di tale effetto sul gobione. Le barre delle variabili si strutturano qui in quattro fasci contrapposti, mostrando come la gran parte della variabilità dei risultati ottenuti sia spiegata, oltre che dalla tendenza intrinseca di ogni specie ad essere predetta bene o male, dalle scelte legate alle ipotesi 1 e 3; si ha dunque un'ulteriore conferma del limitato effetto che le alternative delle ipotesi 4 e 5 determinano sulle performance dei modelli.

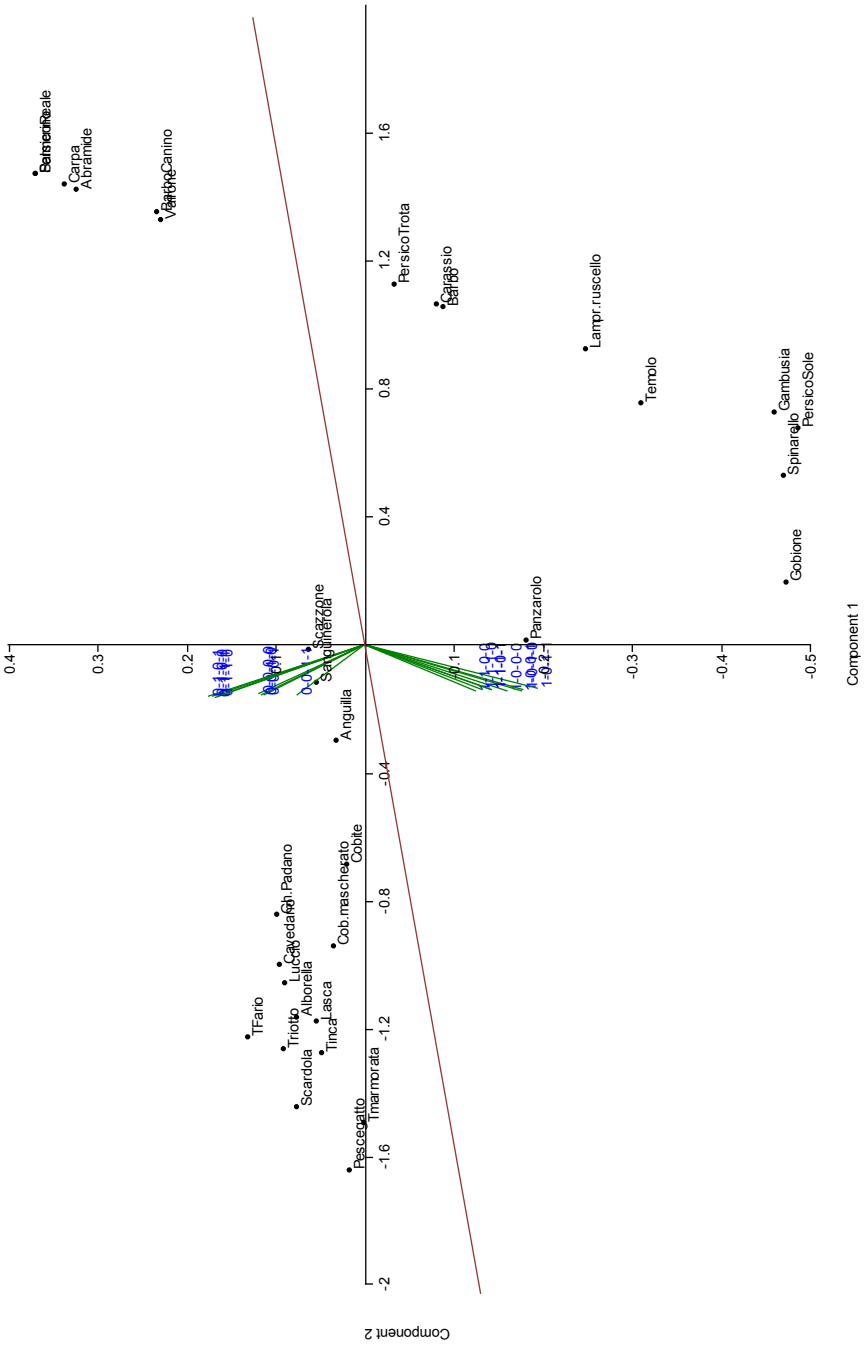


Fig. 40

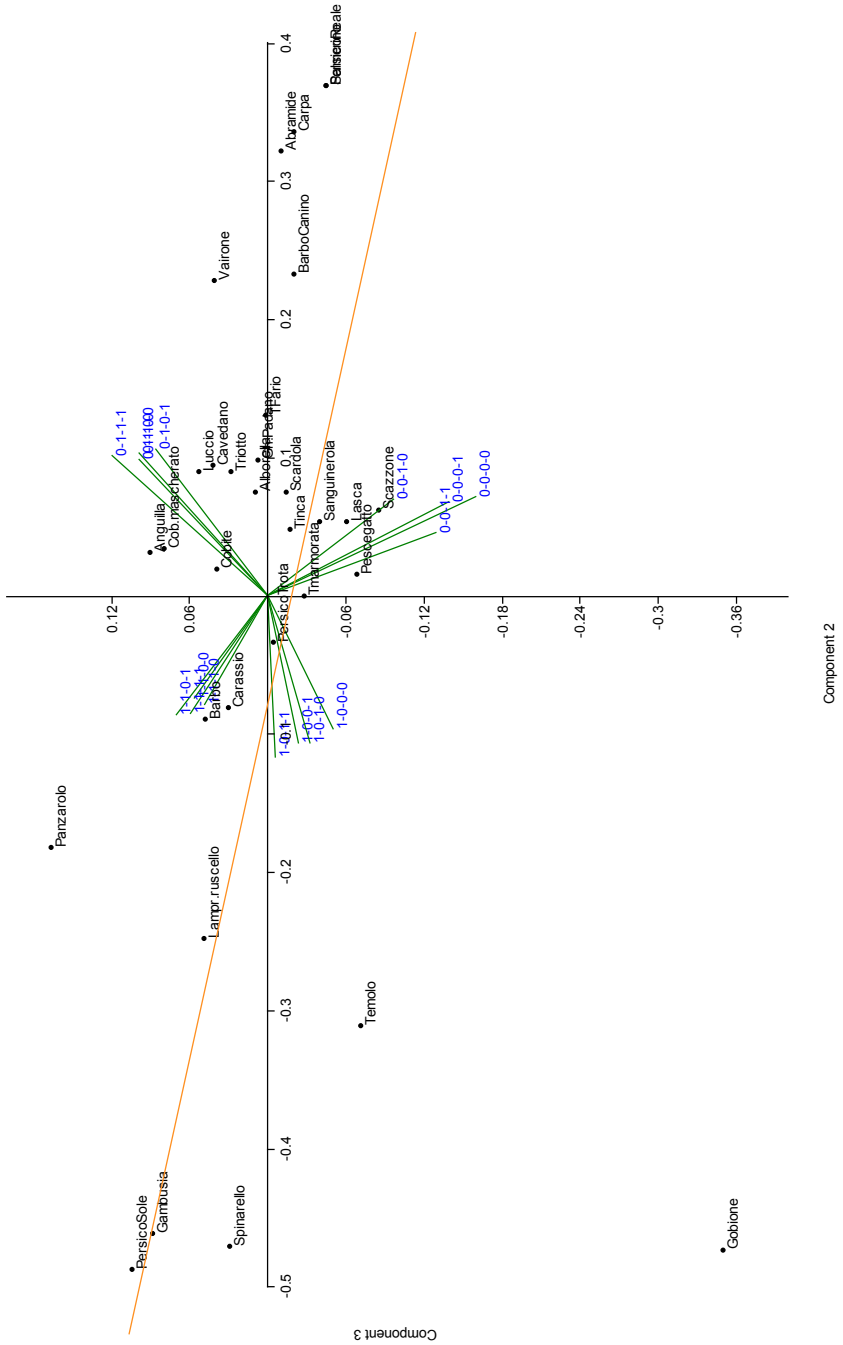


Fig. 41

La sintesi descrittiva ha il pregio di evidenziare alcune caratteristiche macroscopiche nei risultati e renderle visualizzabili in termini grafici, ma nessuna analisi può ritenersi completa senza una fase inferenziale che permetta di discriminare in termini probabilistici tra gli effetti osservati per caso e quelli legati ai fattori sistematici indagati. I risultati ottenuti con il test di Wilcoxon-Mann-Whitney confrontando per ogni specie i modelli contrapposti secondo le varie ipotesi sono presentati nelle tabelle che seguono. L'ipotesi 1 (Tab. 7) è l'unica ad avere una sola tabella, mentre l'ipotesi 3 (Tabb. 8 e 9), l'ipotesi 4 (Tabb. 10 e 11) e l'ipotesi 5 (Tabb. 12 e 13), studiate tramite l'indice AUK oltre che con la statistica K, necessitano di due tabelle per ciascuna. In ogni tabella, lungo le righe si hanno le specie e lungo le colonne si hanno i confronti effettuati, legati a tutte le possibili combinazioni determinate altre ipotesi; ad esempio, il confronto #010 relativo al triotto (Rue) in Tab. 7 dice che il modello 1010 ha mostrato prestazioni superiori al modello 0010 con una significatività di 0.05, cioè che con un modello monospecie addestrato su dati ordinati per presenza/assenza ma non per altitudine, l'uso di una soglia ottimizzata ha dato risultati migliori dell'uso di una soglia tradizionale a 0.5. Dunque i risultati presenti in una riga mostrano, per una specie, gli esiti dei confronti tra le alternative dell'ipotesi contemplata dalla tabella (indicata agli angoli) in tutte le condizioni possibili secondo le altre ipotesi. I risultati favorevoli all'alternativa ottimizzante sono stati evidenziati in verde, mentre quelli favorevoli all'altra alternativa sono stati evidenziati in blu[75]; la scritta "ns" è stata invece utilizzata per segnalare i confronti che non hanno mostrato differenze significative.

[75] Come già accennato, mentre per alcune ipotesi è naturale considerare una delle due alternative come quella in grado di dare modelli potenzialmente ottimizzati, per altre la scelta è più arbitraria. In particolare, sia dalla ricerca di una soglia ottimale (ipotesi 1) che da un'equilibrata ripartizione dei casi di presenza/assenza tra i subset (ipotesi 4) si può pensare di ottenere con tutte le specie un miglioramento delle prestazioni; solo a causa di "incidenti" dovuti ad errori o a caratteristiche particolari dei dati è ragionevole aspettarsi prestazioni migliori dalle altre alternative. Nel caso invece delle ipotesi 3 e 5, come già illustrato precedentemente, alcune specie potrebbero "preferire" una delle due alternative in ragione di caratteristiche peculiari del loro rapporto con i descrittori ambientali e con le altre specie; dunque la scelta di ritenere ottimizzanti l'alternativa multispecie (ipotesi 3) e la partizione per l'altitudine (ipotesi 5) è puramente di comodo e giustificata solo dalla ragionevole attesa che esse siano tali per la maggior parte delle specie.

H1	#000	#001	#010	#011	#100	#101	#110	#111	H1
Sat	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Sat
Lec	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.05	0.01	Lec
Pam	0.0001	0.0001	0.05	0.05	0.0001	0.0001	0.01	0.01	Pam
Sce	0.0001	0.0001	0.001	ns	0.0001	0.0001	0.0001	0.01	Sce
Esl	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.01	0.001	Esl
Rue	0.0001	0.0001	0.05	ns	0.0001	0.0001	0.01	0.05	Rue
Ala	0.0001	0.0001	0.01	0.0001	0.0001	0.0001	0.01	0.01	Ala
Cog	0.0001	0.0001	0.01	0.001	0.0001	0.0001	0.01	0.0001	Cog
Tit	0.0001	0.0001	0.01	0.0001	0.0001	0.0001	0.0001	0.0001	Tit
Cot	0.0001	0.0001	0.0001	0.001	0.0001	0.0001	0.0001	0.0001	Cot
Php	0.0001	0.0001	0.01	0.0001	0.0001	0.0001	0.01	0.0001	Php
Ana	0.0001	0.0001	0.0001	0.01	0.0001	0.0001	0.0001	0.0001	Ana
Orp	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Orp
Sam	0.0001	0.0001	0.001	0.01	0.0001	0.0001	0.001	0.001	Sam
Sal	0.0001	0.0001	0.05	0.05	0.0001	0.0001	ns	ns	Sal
Icm	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.01	0.0001	Icm
Leg	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Leg
Bap	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Bap
Chg	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Chg
Gaa	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Gaa
Cac	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Cac
Gog	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Gog
Les	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Les
Tht	0.0001	0.0001	0.0001	0.001	0.0001	0.0001	0.0001	0.0001	Tht
Lap	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Lap
Gah	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Gah
Bam	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Bam
Mis	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	Mis
Pef	ns	ns	ns	ns	0.0001	0.0001	0.001	0.001	Pef
Abb	ns	ns	ns	ns	0.0001	0.0001	0.0001	0.0001	Abb
Cyc	ns	ns	ns	ns	0.0001	0.0001	0.0001	0.0001	Cyc
Saf	ns	ns	ns	ns	ns	ns	ns	ns	Saf
H1	#000	#001	#010	#011	#100	#101	#110	#111	H1

Tab. 7

H3	0#00	0#01	0#10	0#11	1#00	1#01	1#10	1#11	H3
Sat	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Sat
Lec	0.0001	0.0001	0.05	ns	0.0001	0.0001	ns	ns	Lec
Pam	ns	ns	ns	ns	0.0001	0.0001	ns	ns	Pam
Sce	0.0001	ns	ns	ns	0.0001	0.0001	ns	ns	Sce
Esl	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Esl
Rue	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Rue
Ala	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Ala
Cog	-0.0001	-0.0001	ns	ns	-0.001	-0.0001	ns	ns	Cog
Tit	0.05	0.0001	ns	ns	0.0001	0.0001	ns	ns	Tit
Cot	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Cot
Php	-0.05	-0.0001	ns	ns	-0.0001	ns	ns	ns	Php
Ana	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Ana
Orp	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Orp
Sam	ns	ns	ns	ns	ns	ns	ns	ns	Sam
Sal	0.0001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Sal
Icm	-0.001	-0.0001	ns	ns	ns	-0.0001	ns	ns	Icm
Leg	ns	ns	ns	ns	ns	0.0001	ns	ns	Leg
Bap	0.001	0.0001	ns	ns	0.0001	0.0001	ns	ns	Bap
Chg	-0.001	ns	ns	ns	ns	-0.0001	ns	ns	Chg
Gaa	ns	-0.0001	ns	ns	0.0001	ns	ns	ns	Gaa
Cac	ns	ns	ns	ns	ns	0.0001	ns	ns	Cac
Gog	-0.0001	-0.0001	ns	-0.0001	-0.0001	-0.0001	ns	ns	Gog
Les	ns	ns	ns	ns	0.0001	0.0001	0.0001	0.0001	Les
Tht	-0.0001	-0.0001	ns	ns	-0.0001	ns	ns	ns	Tht
Lap	0.0001	0.0001	ns	ns	0.0001	ns	ns	ns	Lap
Gah	0.001	0.0001	ns	ns	ns	ns	ns	ns	Gah
Bam	0.0001	ns	ns	ns	0.0001	ns	ns	ns	Bam
Mis	0.0001	ns	ns	ns	ns	-0.0001	ns	-0.01	Mis
Pef	ns	ns	ns	ns	0.0001	0.0001	0.001	0.001	Pef
Abb	ns	ns	ns	ns	0.0001	0.0001	0.0001	0.0001	Abb
Cyc	ns	ns	ns	ns	0.0001	0.0001	0.0001	0.0001	Cyc
Saf	ns	ns	ns	ns	ns	ns	ns	ns	Saf
H3	0#00	0#01	0#10	0#11	1#00	1#01	1#10	1#11	H3

Tab. 8

H3	X#00	X#01	X#10	X#11	H3
Sat	0.0001	0.0001	ns	ns	Sat
Lec	0.0001	ns	ns	ns	Lec
Pam	ns	-0.0001	ns	ns	Pam
Sce	0.0001	0.0001	ns	ns	Sce
Esl	0.0001	0.0001	0.001	ns	Esl
Rue	0.0001	0.0001	ns	ns	Rue
Ala	0.0001	0.0001	ns	ns	Ala
Cog	-0.0001	-0.0001	ns	ns	Cog
Tit	0.0001	0.0001	ns	ns	Tit
Cot	0.0001	0.0001	ns	ns	Cot
Php	ns	ns	ns	ns	Php
Ana	0.0001	0.0001	ns	ns	Ana
Orp	0.0001	0.0001	ns	ns	Orp
Sam	-0.0001	ns	ns	ns	Sam
Sal	0.0001	-0.01	ns	ns	Sal
Icm	-0.0001	-0.0001	ns	-0.001	Icm
Leg	0.0001	0.0001	ns	ns	Leg
Bap	0.0001	0.0001	ns	ns	Bap
Chg	-0.0001	-0.0001	ns	-0.01	Chg
Gaa	0.01	-0.0001	ns	ns	Gaa
Cac	0.0001	0.0001	ns	ns	Cac
Gog	-0.0001	-0.0001	-0.05	-0.0001	Gog
Les	0.0001	0.0001	0.0001	0.0001	Les
Tht	-0.0001	-0.0001	-0.001	ns	Tht
Lap	0.01	ns	ns	-0.05	Lap
Gah	ns	ns	ns	ns	Gah
Bam	0.0001	ns	ns	ns	Bam
Mis	ns	ns	ns	ns	Mis
Pef	-0.0001	-0.0001	-0.0001	-0.001	Pef
Abb	0.0001	0.0001	0.0001	0.0001	Abb
Cyc	0.0001	0.0001	0.05	ns	Cyc
Saf	ns	ns	ns	ns	Saf
H3	X#00	X#01	X#10	X#11	H3

Tab. 9

H4	00#0	00#1	01#0	01#1	10#0	10#1	11#0	11#1	H4
Sat	ns	ns	ns	ns	ns	ns	ns	ns	Sat
Lec	ns	ns	ns	ns	ns	ns	ns	ns	Lec
Pam	ns	ns	ns	ns	ns	ns	ns	ns	Pam
Sce	ns	ns	ns	ns	0.01	ns	0.01	ns	Sce
Esl	ns	ns	ns	ns	ns	ns	ns	ns	Esl
Rue	ns	ns	ns	ns	ns	ns	ns	ns	Rue
Ala	ns	ns	ns	ns	ns	ns	ns	ns	Ala
Cog	ns	ns	ns	ns	ns	ns	ns	ns	Cog
Tit	ns	ns	ns	ns	ns	ns	ns	ns	Tit
Cot	ns	0.001	ns	ns	ns	ns	ns	ns	Cot
Php	ns	ns	ns	ns	ns	ns	ns	ns	Php
Ana	ns	0.001	ns	ns	ns	ns	ns	ns	Ana
Orp	ns	ns	ns	0.05	ns	ns	ns	ns	Orp
Sam	ns	ns	ns	ns	ns	ns	ns	ns	Sam
Sal	0.0001	ns	0.05	ns	ns	ns	ns	ns	Sal
Icm	ns	ns	ns	ns	ns	ns	ns	ns	Icm
Leg	ns	ns	ns	ns	ns	ns	ns	ns	Leg
Bap	ns	ns	ns	ns	ns	ns	ns	ns	Bap
Chg	ns	ns	ns	ns	ns	ns	ns	ns	Chg
Gaa	ns	ns	ns	ns	ns	ns	ns	ns	Gaa
Cac	ns	ns	ns	ns	ns	ns	ns	ns	Cac
Gog	ns	0.05	ns	0.05	ns	ns	ns	ns	Gog
Les	ns	ns	ns	ns	ns	ns	ns	ns	Les
Tht	ns	ns	ns	ns	ns	ns	ns	ns	Tht
Lap	ns	ns	ns	ns	0.0001	0.05	ns	ns	Lap
Gah	ns	ns	ns	ns	ns	ns	ns	ns	Gah
Bam	ns	ns	ns	ns	0.0001	ns	ns	ns	Bam
Mis	ns	ns	ns	ns	ns	ns	ns	ns	Mis
Pef	ns	ns	ns	ns	ns	ns	ns	ns	Pef
Abb	ns	ns	ns	ns	ns	ns	ns	ns	Abb
Cyc	ns	ns	ns	ns	ns	ns	ns	ns	Cyc
Saf	ns	ns	ns	ns	ns	ns	ns	ns	Saf
H4	00#0	00#1	01#0	01#1	10#0	10#1	11#0	11#1	H4

Tab. 10

H4	X0#0	X0#1	X1#0	X1#1	H4
Sat	ns	ns	ns	ns	Sat
Lec	ns	ns	ns	ns	Lec
Pam	ns	ns	ns	ns	Pam
Sce	0.05	ns	ns	ns	Sce
Esl	ns	0.0001	ns	ns	Esl
Rue	ns	ns	ns	ns	Rue
Ala	ns	ns	ns	ns	Ala
Cog	ns	ns	0.05	ns	Cog
Tit	ns	0.01	ns	ns	Tit
Cot	ns	0.0001	ns	ns	Cot
Php	ns	0.001	ns	ns	Php
Ana	ns	0.001	ns	0.01	Ana
Orp	0.001	0.0001	ns	0.01	Orp
Sam	ns	ns	ns	ns	Sam
Sal	0.0001	ns	0.001	ns	Sal
Icm	ns	ns	ns	ns	Icm
Leg	ns	0.01	ns	ns	Leg
Bap	ns	ns	ns	ns	Bap
Chg	ns	0.001	ns	ns	Chg
Gaa	ns	ns	ns	ns	Gaa
Cac	ns	0.001	ns	ns	Cac
Gog	ns	0.0001	ns	0.0001	Gog
Les	ns	ns	0.05	ns	Les
Tht	ns	ns	ns	ns	Tht
Lap	0.0001	0.0001	0.01	ns	Lap
Gah	ns	0.01	ns	0.01	Gah
Bam	ns	ns	ns	ns	Bam
Mis	ns	ns	ns	ns	Mis
Pef	ns	ns	0.05	0.01	Pef
Abb	ns	ns	ns	ns	Abb
Cyc	ns	ns	ns	ns	Cyc
Saf	ns	ns	ns	ns	Saf
H4	X0#0	X0#1	X1#0	X1#1	H4

Tab. 11

H5	000#	001#	010#	011#	100#	101#	110#	111#	H5
Sat	0.001	ns	0.0001	ns	ns	ns	ns	ns	Sat
Lec	0.01	ns	ns	ns	ns	ns	-0.01	ns	Lec
Pam	0.0001	ns	0.0001	ns	0.01	ns	ns	ns	Pam
Sce	0.0001	ns	0.001	ns	0.0001	ns	0.0001	ns	Sce
Esl	ns	ns	ns	ns	-0.0001	ns	-0.0001	ns	Esl
Rue	0.0001	ns	0.0001	ns	0.0001	ns	0.0001	ns	Rue
Ala	ns	ns	-0.0001	ns	ns	ns	-0.01	ns	Ala
Cog	0.0001	ns	0.01	ns	0.0001	ns	ns	ns	Cog
Tit	ns	ns	ns	ns	ns	ns	0.0001	ns	Tit
Cot	ns	0.01	ns	ns	ns	ns	0.01	ns	Cot
Php	ns	ns	ns	ns	-0.0001	ns	ns	ns	Php
Ana	ns	ns	-0.01	ns	ns	ns	-0.001	ns	Ana
Orp	0.0001	ns	ns	ns	0.0001	ns	0.0001	ns	Orp
Sam	0.05	ns	ns	ns	-0.0001	ns	-0.0001	ns	Sam
Sal	0.0001	ns	0.0001	ns	0.0001	ns	0.0001	ns	Sal
Icm	0.0001	ns	0.0001	ns	0.0001	ns	0.0001	ns	Icm
Leg	ns	ns	ns	ns	ns	ns	ns	ns	Leg
Bap	ns	ns	ns	ns	ns	ns	ns	ns	Bap
Chg	ns	ns	ns	ns	ns	ns	-0.0001	ns	Chg
Gaa	0.0001	ns	ns	ns	0.0001	ns	0.0001	ns	Gaa
Cac	ns	ns	ns	ns	-0.0001	ns	-0.0001	ns	Cac
Gog	ns	0.05	-0.0001	ns	ns	ns	-0.05	ns	Gog
Les	ns	ns	ns	ns	ns	ns	0.01	ns	Les
Tht	ns	ns	0.0001	ns	ns	ns	0.0001	ns	Tht
Lap	ns	ns	ns	ns	0.0001	ns	0.01	ns	Lap
Gah	ns	ns	ns	ns	ns	ns	ns	ns	Gah
Bam	0.05	ns	ns	ns	0.0001	ns	ns	ns	Bam
Mis	0.0001	ns	ns	ns	0.0001	ns	0.0001	ns	Mis
Pef	ns	ns	ns	ns	ns	ns	ns	ns	Pef
Abb	ns	ns	ns	ns	ns	ns	-0.01	ns	Abb
Cyc	ns	ns	ns	ns	ns	ns	ns	ns	Cyc
Saf	ns	ns	ns	ns	ns	ns	ns	ns	Saf
H5	000#	001#	010#	011#	100#	101#	110#	111#	H5

Tab. 12

H5	X00#	X01#	X10#	X11#	H5
Sat	0.0001	ns	0.0001	ns	Sat
Lec	0.01	ns	ns	ns	Lec
Pam	0.0001	ns	0.0001	ns	Pam
Sce	0.0001	ns	0.001	ns	Sce
Esl	ns	ns	ns	ns	Esl
Rue	0.0001	ns	0.0001	ns	Rue
Ala	ns	ns	-0.0001	ns	Ala
Cog	0.0001	ns	0.01	ns	Cog
Tit	ns	ns	ns	ns	Tit
Cot	ns	0.01	ns	0.05	Cot
Php	ns	ns	ns	ns	Php
Ana	ns	0.05	-0.01	ns	Ana
Orp	0.0001	ns	ns	ns	Orp
Sam	0.05	ns	ns	ns	Sam
Sal	0.0001	ns	0.0001	ns	Sal
Icm	0.0001	ns	0.0001	ns	Icm
Leg	ns	ns	ns	ns	Leg
Bap	ns	ns	ns	ns	Bap
Chg	ns	ns	ns	ns	Chg
Gaa	0.0001	ns	ns	ns	Gaa
Cac	ns	ns	ns	ns	Cac
Gog	ns	0.05	-0.0001	ns	Gog
Les	ns	ns	ns	ns	Les
Tht	ns	ns	0.0001	ns	Tht
Lap	ns	ns	ns	ns	Lap
Gah	ns	ns	ns	ns	Gah
Bam	0.01	ns	ns	ns	Bam
Mis	0.0001	ns	ns	ns	Mis
Pef	ns	ns	ns	ns	Pef
Abb	ns	ns	ns	ns	Abb
Cyc	ns	ns	ns	ns	Cyc
Saf	ns	ns	ns	ns	Saf
H5	X00#	X01#	X10#	X11#	H5

Tab. 13

Nell'interpretare i risultati si ricordi che la statistica non parametrica utilizzata, basandosi sui ranghi, tende sia a ridimensionare le differenze grandi che a dare importanza alle differenze piccole all'interno delle distribuzioni confrontate, puntando non sull'intensità, ma sulla sistematicità con cui i valori di una distribuzione sono più grandi o più piccoli di quelli di un'altra.

I test relativi all'ipotesi 1 (Tab. 7) confermano appieno le attese e ribadiscono quanto già osservato a livello descrittivo: la ricerca di una soglia ottimizzata per l'interpretazione binaria dell'output permette di sfruttare al meglio le capacità dei modelli ottenuti. I casi di non significatività sono limitati qui alle sole situazioni in cui una specie è predetta molto bene (gli output continui sono ben vicini allo zero per l'assenza e all'uno per la presenza) o troppo male (gli output sono casuali o banali).

Sull'insieme di soglie *best* individuate per ogni specie e ogni condizione è stata calcolata la mediana, così da fornire un'informazione sintetica sull'orientamento preso dalle soglie nelle varie situazioni esaminate (Tab. 14). Valori pari a zero indicano una degenerazione del processo di ricerca della soglia, che si verifica quando il modello non esprime in pratica alcuna capacità predittiva.

	#000	#001	#010	#011	#100	#101	#110	#111
Sat	0.5	0.43	0.48	0.485	0.52	0.5	0.44	0.51
Lec	0.38	0.37	0.385	0.35	0.37	0.41	0.405	0.42
Pam	0.38	0.39	0.335	0.395	0.39	0.4	0.42	0.395
Sce	0.43	0.48	0.475	0.495	0.4	0.45	0.435	0.41
Esl	0.35	0.35	0.415	0.36	0.4	0.405	0.43	0.405
Rue	0.385	0.39	0.35	0.405	0.43	0.43	0.395	0.45
Ala	0.33	0.33	0.28	0.29	0.38	0.36	0.325	0.325
Cog	0.37	0.34	0.365	0.355	0.34	0.34	0.3	0.375
Tit	0.42	0.41	0.39	0.4	0.38	0.4	0.36	0.27
Cot	0.31	0.3	0.28	0.36	0.32	0.3	0.245	0.33
Php	0.35	0.42	0.395	0.34	0.32	0.31	0.375	0.33
Ana	0.31	0.31	0.37	0.34	0.33	0.33	0.325	0.335
Orp	0.22	0.22	0.21	0.25	0.3	0.26	0.295	0.33
Sam	0.39	0.39	0.42	0.475	0.35	0.33	0.325	0.41
Sal	0.36	0.39	0.405	0.37	0.41	0.4	0.4	0.435
Icm	0.41	0.38	0.34	0.38	0.37	0.49	0.385	0.35
Leg	0.18	0.2	0.2	0.195	0.2	0.17	0.195	0.195
Bap	0.11	0.12	0.09	0.14	0.11	0.12	0.12	0.165
Chg	0.49	0.49	0.52	0.495	0.465	0.52	0.51	0.525
Gaa	0.22	0.21	0.225	0.2	0.22	0.17	0.205	0.185
Cac	0.09	0.09	0.1	0.11	0.09	0.11	0.1	0.11
Gog	0.29	0.32	0.28	0.315	0.21	0.2	0.255	0.235
Les	0.01	0.01	0.03	0.035	0.05	0.06	0.07	0.085
Tht	0.16	0.17	0.25	0.21	0.1	0.14	0.13	0.14
Lap	0.08	0.08	0.14	0.16	0.09	0.1	0.11	0.105
Gah	0.14	0.18	0.155	0.17	0.14	0.15	0.16	0.155
Bam	0.01	0.01	0.025	0.02	0.03	0.01	0.01	0.03
Mis	0.04	0.08	0.06	0.135	0.05	0.05	0.045	0.07
Pef	0	0	0	0	0	0	0	0
Abb	0	0	0	0	0.01	0.01	0.01	0.01
Cyc	0	0	0	0	0.01	0.01	0.01	0.01
Saf	0	0	0	0	0	0	0	0

Tab. 14

I valori presentati in questa tabella saranno utilizzati per l'esame dell'ipotesi 2.

Anche i risultati legati all'ipotesi 3 (Tab. 8 e 9) sono conformi alle attese: la maggior parte delle specie preferisce quasi sempre il modello multispecie (risultati in verde), ma ve n'è comunque un certo numero che mostra una netta preferenza per il modello monospecie (risultati in blu).

Nelle due tabelle si notano alcune colonne decisamente caratterizzate da risultati non significativi: questo purtroppo è un effetto dovuto alla diversa numerosità dei casi considerati nei vari confronti secondo il disegno sperimentale. Si

tratta in particolare dei casi di partizione per presenza/assenza, basati su distribuzioni di 72 risultati, mentre le distribuzioni usate per gli altri confronti ne hanno anche più di 2000. L'effetto di questa discrepanza si nota, con forme e intensità diverse, in tutte le tabelle con le significatività.

Poiché lo studio dell'ipotesi 3 ha dato luogo a risultati articolati, nonché potenzialmente interessanti dal punto di vista ecologico, si è ritenuto opportuno approfondire l'indagine. Innanzitutto, su entrambe le tabelle è stata condotta una PCA che le Figg. 42 e 43 propongono in forma grafica[76].

[76] La PCA è stata condotta sulla versione originale (non semplificata in base alle significatività di riferimento) delle tabelle, in seguito ad una trasformazione logaritmica, necessaria per riportare i valori ad un andamento lineare, e ad una successiva normalizzazione basata sulla radice quadrata dell'arcoseno.

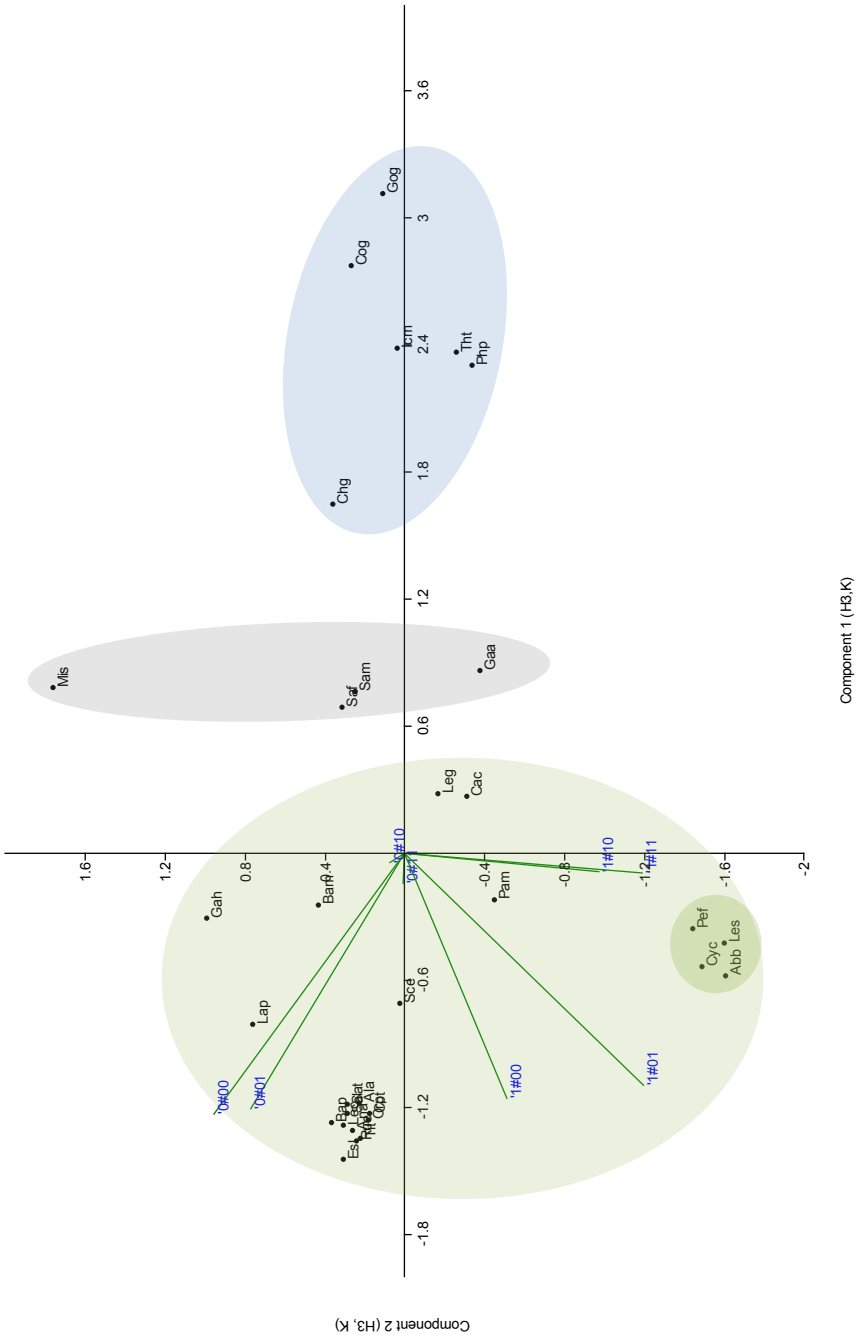


Fig. 42

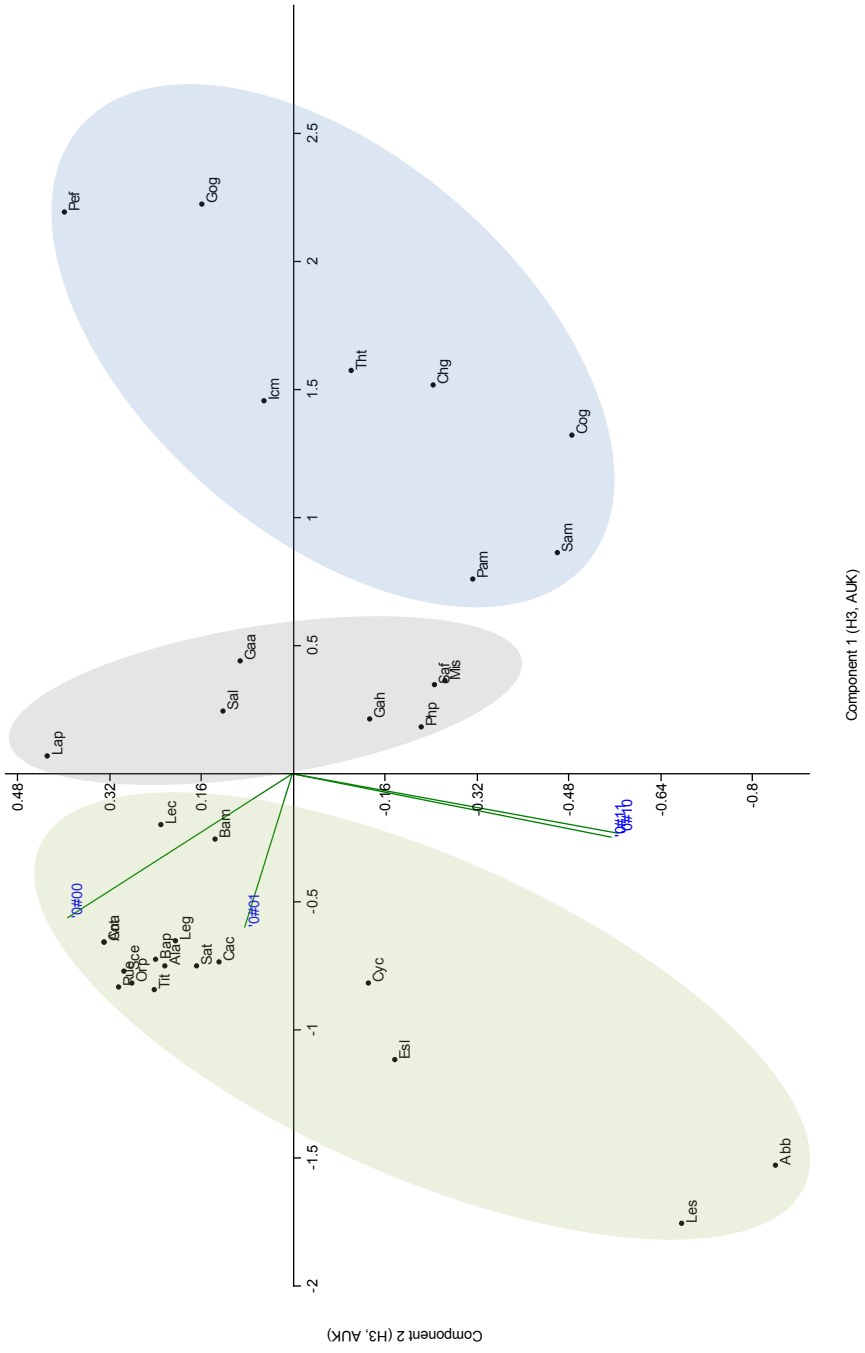


Fig. 43

In entrambi i grafici si riscontra una tendenza delle specie a raggrupparsi lungo la prima componente in base alla preferenza per un modello monospecie (ellisse in azzurro), ad una condizione di incertezza (ellisse in grigio) o ad una preferenza per il modello multispecie (ellisse in verde). I due grafici sono parzialmente diversi, perché nel primo la considerazione delle alternative legate all'ipotesi 1 impone un'organizzazione particolare che tende a spostare verso il basso i casi in cui gli effetti maggiori si riscontrano quando la soglia è ottimizzata. Questo, ad esempio, permette di riconoscere il gruppo (ellisse in verde scuro) delle specie "salvate" dall'ottimizzazione della soglia. Il secondo grafico, però, non affetto dal "disturbo" dell'ipotesi 1 e basato su una misura severa come AUK, è più efficace nel separare le specie in base all'effetto che sulla loro previsione hanno le scelte legate all'ipotesi 3.

Sarebbe interessante anche fare delle riflessioni sul significato ecologico di queste differenze, visto che potenzialmente esso ha a che fare con i rapporti che ogni specie intrattiene con l'ambiente e con il resto del popolamento. Poiché questo richiederebbe una conoscenza profonda del contesto ecologico da cui è stato tratto il *data set* utilizzato, conoscenza di cui non si dispone in modo adeguato, ci si limiterà qui ad esplorare la semplice idea che la tendenza di una specie a preferire un modello monospecie, rinunciando ai vantaggi del modello multispecie, dipenda positivamente dalla forza della relazione con i descrittori ambientali e negativamente dalla forza della relazione con le altre specie presenti nella comunità di cui fa parte. In pratica, se una specie ha relazioni più definite con l'ambiente che con la comunità potrebbe preferire una previsione monospecie, perché il principale vantaggio del modello multispecie (cioè la capacità di tenere conto delle relazioni tra le specie) aggiungerebbe in questo caso solo un elemento di confusione. Per poter visualizzare graficamente, ad un livello puramente descrittivo, le relazioni tra le specie e queste due variabili si è costruito un diagramma a due dimensioni (Fig. 44): sulle ascisse è riportata una misura del grado di associazione con i parametri ambientali e sulle ordinate una misura del grado di associazione con le altre specie[77].

[77] Visto lo scopo puramente descrittivo del grafico, le due misure sono state ottenute in modo totalmente arbitrario utilizzando, nel primo caso, una media delle prestazioni ottenute con i modelli monospecie, e nel secondo una media dei valori di K ottenuti da ogni specie confrontata con tutte le altre. Nel primo caso si assume che la prestazione del percettrone a singolo output misuri nel modo migliore possibile il legame della specie con i predittori ambientali; nel secondo si assume che l'uso dei valori di presenza/assenza di una specie come predittori dei medesimi valori per un'altra sia un buon modo di testare il grado di associazione tra due specie. La media dei valori K ottenuti rappresenterebbe così il grado di associazione medio di una specie con le altre. Pur considerandola nei calcoli, si è scelto

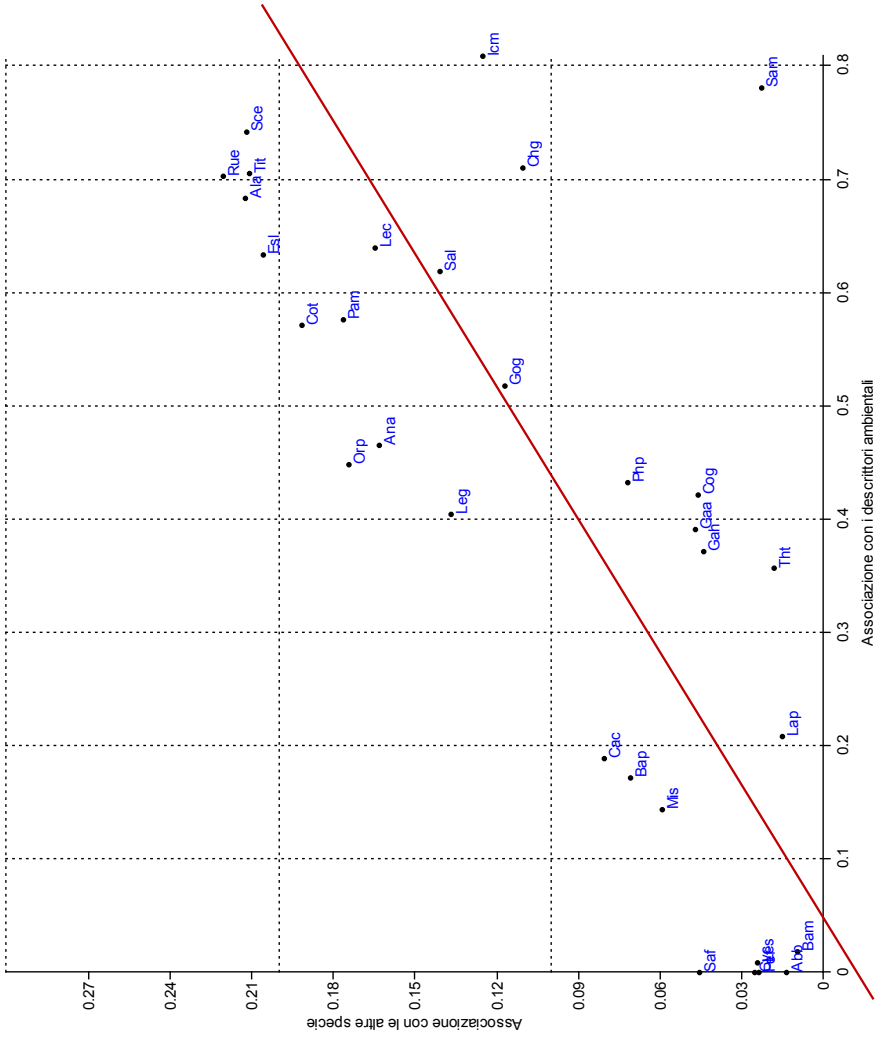


Fig. 44

Sul grafico è possibile tracciare una linea al di sopra della quale cade la maggior parte delle specie che mostra in generale una preferenza per il modello multispecie e al di sotto della quale si trovano invece quasi tutte le specie che

di eliminare la trota dal grafico, poiché, in quanto specie oggetto di ripopolamento, i suoi rapporti con i parametri ambientali e con le altre specie possono essere influenzati da fattori non legati strettamente all'ecologia naturale della specie.

preferiscono il modello monospecie o che non mostrano una preferenza decisa. E' un risultato puramente indicativo e come tale va preso; tuttavia esso incoraggia un approfondimento dell'insieme di fattori ecologici e metodologici che possono influenzare in un senso o nell'altro le preferenze delle varie specie.

I risultati relativi all'ipotesi 4 (Tabb. 10 e 11), come già accennato, risentono probabilmente del ridotto numero di misure ottenute per ogni modello. Tuttavia i confronti sono risultati significativi almeno in un certo numero di casi, evidenziando le specie e le condizioni per cui una partizione basata sui valori di presenza/assenza migliora la performance dei modelli. La non significatività del miglioramento dipende essenzialmente dall'abbondanza e dalla varietà di casi rappresentativi nel data set, che possono essere sufficienti da garantire subset omogenei anche con un'assegnazione causale, o dalla mancanza/incoerenza dell'informazione, tale da dar luogo comunque a modelli simili a decisori casuali o banali, non migliorabili in alcun modo.

E' da segnalare un risultato che, per quanto non significativo a livello inferenziale, è interessante comprendere: due specie (il pesce gatto e la tinca) hanno mostrato in alcuni casi una preferenza per il modello non ottimizzato. Trattandosi di due tra le specie che meglio si riesce a predire, esse rientrano certamente in quei casi in cui l'abbondanza e la qualità dell'informazione disponibile sono tali da non richiedere ottimizzazioni in fase di partizione; è anzi probabile che l'artificiosità di tale operazione introduca elementi indesiderabili nel processo di addestramento. Per quanto dunque la riduzione di performance non sia stata significativa, questo risultato suggerisce di non estendere in modo automatico tale strategia di ottimizzazione a tutte le specie e ad ogni condizione, ma di verificarne il beneficio.

Anche per l'ipotesi 5 (Tabb. 12 e 13) si è riusciti a mostrare come la scelta di un'alternativa piuttosto che dell'altra possa determinare valori di performance significativamente diversi. La variabilità tra le specie nel preferire l'una o l'altra alternativa lungo le condizioni stabilite dalle altre ipotesi non permette qui di ottenere un quadro complessivo della risposta del popolamento tramite metodi di ordinamento. Ci si è limitati dunque a cercare una rappresentazione sintetica dell'informazione contenuta nelle due tabelle (Fig. 45), mostrando la tendenza media delle specie a preferire l'alternativa ottimizzante (valori positivi) o l'altra (valori negativi).

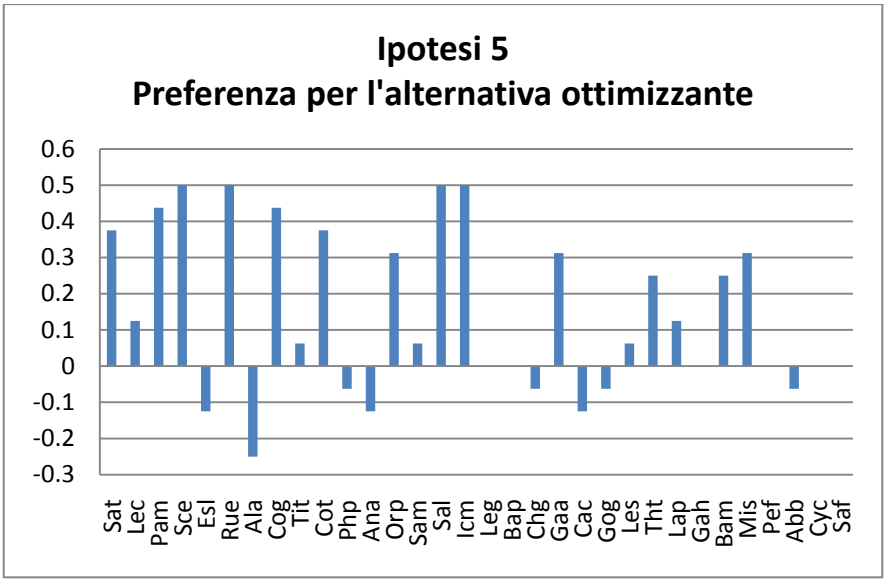


Fig. 45

Come ci si attendeva, la maggior parte delle specie trae beneficio da un ordinamento dei dati che favorisce una distribuzione omogenea ai subset di tutti livelli delle diverse variabili ambientali che covariano con l'altitudine. Qualche altra specie, invece, risulta disturbata da questo ordinamento, il che si verifica potenzialmente quando i descrittori ambientali con cui essa ha le relazioni più forti non covariano con l'altitudine. Per le stesse ragioni addotte in merito all'ipotesi 3, non si entra qui in modo approfondito nell'interpretazione ecologica delle differenze riscontrate, ma si può ipotizzare, ad esempio, che l'anguilla (Ana), essendo una specie euriecia rispetto alle condizioni che covariano con l'altitudine (infatti la si può trovare quasi dappertutto tra monte e valle, almeno nelle zone a ciprinidi), è probabilmente condizionata da variabili di tipo diverso, che possono risultare meglio distribuite con un ordinamento casuale. Tuttavia, qualunque approfondimento di questo tipo non può prescindere da una conoscenza adeguata del contesto ambientale da cui i dati sono stati tratti.

Infine, si riportano i risultati relativi all'ipotesi 2, secondo cui la relazione tra la frequenza dei casi di presenza di una specie e la soglia che determina la migliore performance del modello (ipotesi 1) può essere descritta quantitativamente da una funzione non decrescente del tipo proposto in Fig. 15 (per semplificare l'analisi si è

però preferito far riferimento al modello lineare). Poiché il test va effettuato sulle 8 distribuzioni di soglie *best* ottenute (Tab. 14), si hanno alla fine 8 grafici che mettono in relazione la frequenza e la soglia e altrettanti valori di significatività relativi alla linearità della relazione. Si presenta qui un solo grafico (Fig. 46) in quanto i risultati ottenuti nelle 8 diverse condizioni sono molto simili tra loro; la Tab. 15 riporta invece i valori di significatività ottenuti.

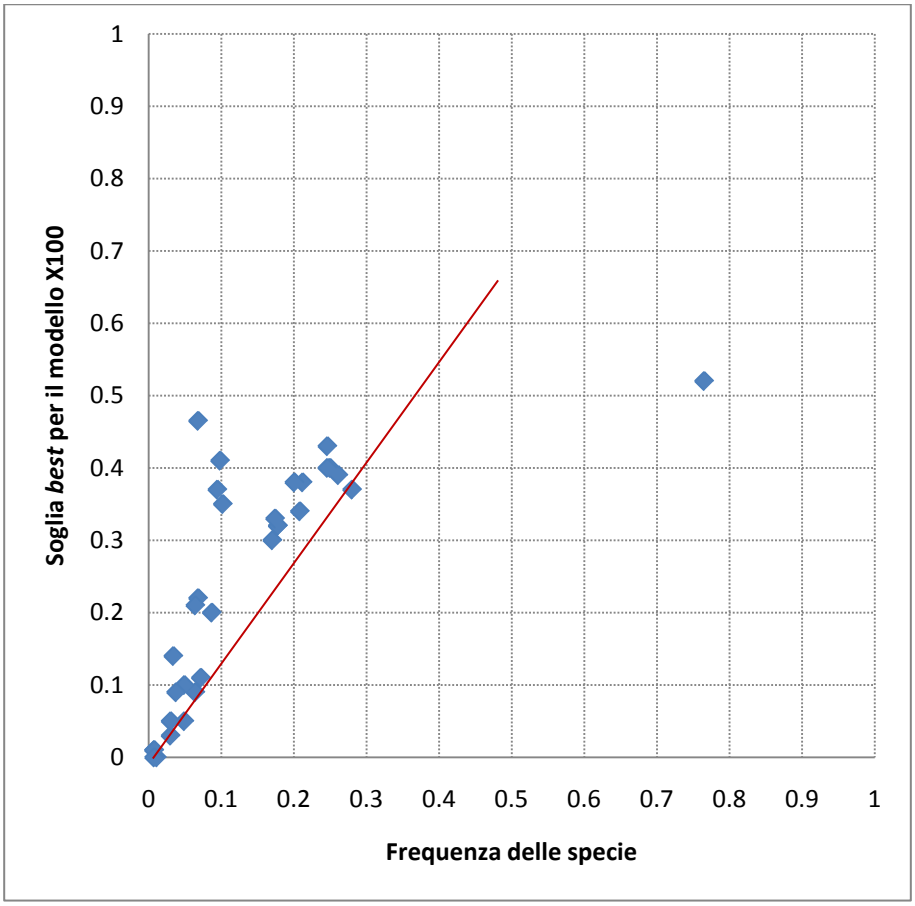


Fig. 46

	X000	X001	X010	X011	X100	X101	X110	X111
p	0.00021	0.00163	0.00090	0.00076	0.00003	0.00021	0.00055	0.00015

Tab. 15

Il punto isolato nel grafico rappresenta la situazione della trota, che per le ragioni già illustrate, ha una frequenza molto più alta delle altre specie. Non si tratta però di un vero e proprio outlier, perché se il modello di riferimento usato non fosse stato quello lineare, ma quello originale (Fig. 15), la sua posizione sarebbe stata ben in linea con il modello stesso; per questa ragione si è scelto di includere comunque la trota nell'analisi, anche in considerazione del fatto che la relazione ipotizzata è così forte in tutte le 8 condizioni studiate, che la presenza di un caso *sui generis* non ne pregiudica la significatività.

Il risultato ottenuto conferma il ragionamento alla base dell'ipotesi 1 e dunque, almeno in parte, spiega perché un'opportuna regolazione della soglia permetta di ottenere performance migliori con alcune specie.

4. CONCLUSIONI

I risultati ottenuti dimostrano innanzitutto l'interesse pratico e teorico di lavorare sui modelli di previsione. Da una parte, infatti, si evidenzia la possibilità di ottenere modelli più efficaci, mentre dall'altra si nota come sia possibile innescare un circolo virtuoso in cui le riflessioni sulla realtà ecologica determinano soluzioni modellistiche, le quali a loro volta propongono nuove domande e idee circa la realtà ecologica stessa.

Il risultato più rilevante per intensità e sistematicità è quello relativo alla regolazione delle soglie di discretizzazione per l'interpretazione binaria dell'output (ipotesi 1), con cui per tutte le specie si è ottenuto un significativo miglioramento delle performance e che in molti casi ha determinato il superamento di una condizione di totale non predicibilità.

Da un punto di vista pratico, questo risultato incoraggia l'uso di un metodo efficace ed economico per migliorare le prestazioni di modelli come quelli usati in questo studio: non richiede infatti alcuna modifica delle procedure di addestramento eventualmente già fissate e, in più, può essere applicato anche a modelli già addestrati.

Da un punto di vista metodologico esso cambia un po' la prospettiva da cui osservare il problema della previsione delle specie più o meno rare, mettendo in luce l'esistenza di una porzione di potere predittivo che i dati conferiscono ai modelli tramite le procedure di addestramento, ma che rischia di restare in ombra a causa di un'interpretazione rigida dell'output.

Grazie ai riscontri significativi riguardo alla relazione tra la frequenza di una specie e la soglia che ne discretizza in modo ottimale le previsioni (ipotesi 2), si è potuto avvalorare l'ipotesi che la soglia *best* insegua un bias dovuto essenzialmente alla frequenza. La dispersione attorno al modello usato come riferimento (Fig. 46) suggerisce tuttavia che altri fattori (e più degli altri, probabilmente, la qualità dell'informazione) generino fluttuazioni rispetto all'andamento generale dovuto alla

frequenza, per cui la funzione frequenza-soglia[78] è da considerarsi solo un modello predittivo di massima per la soglia *best*.

Il bias dovuto alla frequenza potrebbe quindi non agire in modo lineare e univoco sull'output, determinandone semplicemente uno spostamento verso l'alto o verso il basso in funzione della frequenza, ma è possibile che esso interagisca con la qualità dell'informazione; aver trovato il modo di determinare una soglia *best* non assicura pertanto l'annullamento degli effetti deleteri del bias. Per esser certi di aver eliminato del tutto questi effetti, bisognerebbe utilizzare data set in cui, per ogni specie, i casi di presenza siano tanti quanti quelli di assenza, condizione praticamente impossibile con le specie rare.

Una possibile soluzione è quella di integrare il data set con repliche dei casi di presenza che compensino lo squilibrio[79]. Nonostante si tratti di una procedura difficile da attuare con i modelli multispecie (per i quali si dovrebbe garantire un 50% di casi di presenza per tutte le specie!), condurre delle prove può essere utile (Tourenq et al., 1999), sia per verificare se si riesce ad ottenere risultati ancora migliori che con l'ottimizzazione della soglia (visto che il bias verrebbe eliminato del tutto e non solo compensato in fase di interpretazione), sia per saggiare la robustezza dei risultati qui ottenuti con l'ipotesi 2, in base ai quali si può prevedere che una ricerca della soglia ottimale in queste condizioni darebbe valori di soglia complessivamente molto più vicini alla soglia tradizionale (e variabili solo secondo una funzione non banale della qualità dell'informazione disponibile per ogni specie nel data set).

Tornando all'algoritmo per l'identificazione della soglia *best* (BTFA[80]), potrebbe essere opportuno svilupparne una variante, sempre basata su μ_R, che non cerca la soglia *best* sul singolo modello, ma sull'insieme dei modelli addestrati in ogni condizione e per ogni specie (Fig. 47).

[78] La funzione ipotizzata era del tipo mostrato in Fig. 15, ma per semplicità si è preferito testare l'ipotesi su un modello lineare. Lavorando sulla funzione originaria, se ne sarebbero potuti determinare i parametri in modo da avere un modello predittivo di massima per la soglia *best*.

[79] Aggiungendo però ai valori dei descrittori una porzione di rumore tipo *jittering*, per evitare un overfitting del modello sui casi ripetuti artificialmente.

[80] Si veda par. 3.1.

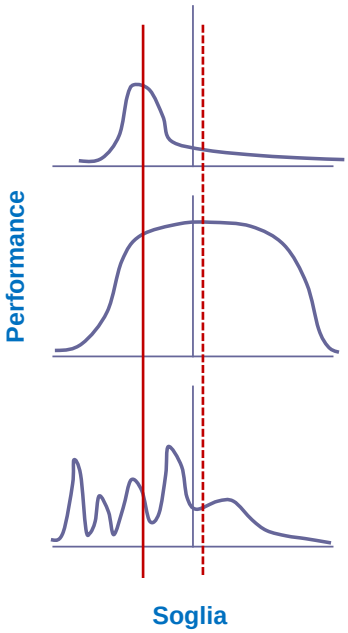


Fig. 47

Infatti, quando si addestra un modello per scopi pratici non è tanto importante ottenere la migliore prestazione possibile dal modello addestrato, quanto determinare una soglia che per diverse varianti di partizione mantenga le performance stabilmente al loro massimo (visto che il modello andrà poi utilizzato con dati diversi da quelli adoperati per l'addestramento). La strategia usata in questo lavoro, che consiste nel trovare la soglia *best* per ogni singolo modello, determinando poi eventualmente una soglia più generale tramite il calcolo della mediana sulla distribuzione di soglie ottenute, ha il pregio di mettere in maggiore risalto le potenzialità dell'algoritmo (che può ottenere il massimo da ogni singolo modello), ma rischia di non garantire in modo adeguato la generalità della soglia *best* identificata. Un algoritmo che invece consideri per ogni soglia ciò che accade alle prestazioni di una popolazione di modelli (Fig. 47) potrebbe invece cogliere in modo più realistico l'effetto generale della variazione della soglia sulle performance. Bisogna però essere certi di aver generato una popolazione di casi rappresentativa del problema modellistico affrontato, pena l'intrusione di una serie di fattori di disturbo legati a modelli devianti (si veda, ad esempio, il grafico in basso di Fig. 47):

le variazioni tra le performance dei diversi modelli devono infatti rientrare nel dominio di una dispersione casuale attorno ad una regola.

I risultati relativi alle preferenze per il modello monospecie o multispecie (ipotesi 3) sono stati parzialmente commentati nel capitolo precedente e messi in connessione con le relazioni delle specie con l'ambiente fisico e tra di loro. Un esame approfondito dell'influenza che le diverse variabili ambientali hanno sulle specie (condotto, ad esempio, tramite un'analisi della sensibilità) e delle relazioni delle specie tra di loro (già in parte effettuato tramite diverse misure di associazione; si vedano in proposito Fig. 19 e 44) potrebbe contribuire a chiarire meglio le ragioni ecologiche di tali preferenze; tale esame, come già si accennava, non può tuttavia sostituire una conoscenza approfondita del contesto ecologico e del data set analizzato. Pertanto un passo avanti in questo senso potrà essere compiuto solo lavorando con dati riguardo ai quali si ha una conoscenza diretta.

Potrebbe essere interessante anche provare una soluzione intermedia tra il modello monospecie e quello multispecie che prevede l'intero popolamento, cioè un modello mulispecie che preveda solo un sottoinsieme del popolamento. In tal modo si potrebbe evitare di prevedere una specie insieme con altre che ne disturbano la previsione e associarla invece con specie che la facilitano. Il problema è scegliere per ogni specie il sottoinsieme del popolamento che meglio di tutti permette di utilizzare a pieno sia l'informazione legata ai descrittori che quella legata alle altre specie. Strumenti utili in questo senso potrebbero essere gli algoritmi genetici. Poiché infatti un qualsiasi sottoinsieme del popolamento può essere rappresentato tramite un vettore con n_s elementi binari, dove n_s è il numero totale delle specie del popolamento e ogni elemento assume valore 1 se la specie è inclusa e 0 se è esclusa, tali vettori potrebbero essere usati come "geni" da un algoritmo genetico che in base ad essi conduce e valuta una serie di addestramenti, fino a determinare quello o quelli che danno risultati migliori. Le popolazioni di partenza per l'algoritmo potrebbero essere determinate a caso o sulla base di opportune misure di associazione. Sarebbe interessante, alla fine, valutare anche il significato ecologico dei sotto popolamenti selezionati.

Riguardo alla valutazione della preferenza per modelli addestrati con partizione bilanciata per presenza/assenza (ipotesi 4), si rileva innanzitutto l'effetto negativo di una caratteristica già segnalata del disegno sperimentale, per cui i casi valutati sono in numero notevolmente inferiore rispetto a quelli usati complessivamente per le

altre ipotesi. Insieme con l'uso della correzione di Bonferroni, questo ha fatto sì che il numero di confronti significativi sia risultato qui piuttosto basso (Tabb. 10 e 11). Tuttavia un certo numero di confronti ha mostrato una decisa tendenza in favore dell'ordinamento ottimizzante e l'approccio descrittivo (Figg. 38 e 39) conferma effetti sulle performance non inferiori (e in alcuni casi anche superiori) a quelli ottenuti con l'ordinamento per altitudine (ipotesi 5); inoltre, l'effetto sembra maggiore, come ci si attendeva, su specie con frequenza medio-bassa, che possono risentire in modo particolare di assegnazioni random che lasciano almeno parzialmente scoperti di casi di presenza uno o più subset. Il caso del pesce gatto e della tinca, che hanno mostrato in alcuni casi una preferenza (comunque non significativa) per l'ordinamento random, suggerisce che l'ordinamento per presenza/assenza potrebbe risultare, nei casi in cui l'informazione nel data set è di buona qualità, addirittura di disturbo.

Nell'insieme, dunque, i risultati ottenuti confermano l'opportunità di chiedersi, durante l'addestramento di un modello, se usare o no questa strategia, ma richiedono l'attenzione di uno studio dedicato, in cui il disegno sperimentale non debba tener conto di troppe cose insieme. Almeno il problema del numero di prove squilibrato per le diverse ipotesi, comunque, può essere risolto in modo relativamente semplice, aumentando il numero di addestramenti all'incrocio tra le condizioni fissate dalle ipotesi 4 e 5 (che in questo lavoro era pari a 72) e prevedendo per le condizioni più generali, cioè quelle in cui non veniva effettuata una partizione per presenza/assenza, il medesimo numero di prove.

La partizione dei dati per altitudine (ipotesi 5) ha dato risultati interessanti, mostrando come le diverse specie possano preferire o meno tale strategia (Tabb. 12 e 13), anche in base al fatto che i modelli siano monospecie o multispecie oppure ottimizzati per la soglia o no. Questo evidenzia per le varie specie un'interazione complessa tra aspetti relativi alla frequenza, all'autoecologia e alle associazioni con le altre specie del popolamento; per questa ragione, anche qui sarebbe interessante compiere uno studio dedicato che prenda esplicitamente in considerazione tutti questi aspetti.

La scelta dell'altitudine come variabile di partizione è stata dettata da ragioni teoriche che sono già state illustrate e in base alle quali la si può ritenere un fattore sintetico che spiega la covariazione di molte variabili fisiche e biologiche. Nel compiere la partizione dei dati è infatti indispensabile far riferimento ad un criterio semplice, ma significativo. Sia le attese che i risultati, però, indicano questa scelta

come una soluzione non ottimale in generale, cioè non efficace con tutte le specie. In casi particolari, inoltre, si potrebbe avere a che fare con sistemi ecologici in cui altri fattori intervengono nel disturbare in modo sistematico gli effetti di questa tipica covariazione sulle comunità degli ecosistemi fluviali; o si potrebbe avere a che fare con tratti fluviali che oltre la loro porzione montana e pedemontana si sviluppano per molti chilometri lungo tratti pianeggianti, senza grandi variazioni di altitudine. Così la scelta dell'altitudine come riferimento, benché ragionevole in molti casi, rischia di diventare forzata in contesti specifici o quando gli interessi applicativi sono legati a porzioni di comunità influenzate poco o negativamente da questa scelta.

Determinare di volta in volta un criterio di partizione adeguato, però, non è un problema da poco, perché gli strumenti a disposizione per compiere tale scelta (misure di correlazione o associazione, metodi di ordinamento, cluster analysis, ecc.) sono in genere i medesimi che vengono giudicati inadeguati a modellizzare un sistema quando si sceglie una rete neurale come modello di riferimento. La soluzione migliore è probabilmente quella di lasciare che un esperto scelga caso per caso le variabili da lui ritenute più rilevanti e che una serie di prove empiriche permetta poi di scartare le scelte meno efficaci e di conservare quella o quelle migliori: questa soluzione darà quasi sempre buoni risultati. Un'alternativa più sofisticata può essere ricavata riarrangiando un metodo proposto in Pessa (2004) per ottenere una sorta di analisi delle componenti principali tramite una rete neurale. In breve, si addestra un percettrone multistrato con uno o due strati nascosti, assegnando allo strato nascosto immediatamente precedente a quello di output poche unità (da una a tre). Lo strato di input riceve i descrittori ambientali e quello di output la specie o le specie da prevedere. Gli stati di attivazione dei neuroni immediatamente precedenti quelli di output vanno così a determinare variabili artificiali che integrano le variazioni complessive dei descrittori e da cui dipendono, a meno di un passaggio non lineare, i valori di presenza e assenza delle specie; pertanto tali variabili possono essere considerate analoghe agli assi di una PCA (con la differenza che non è possibile attribuire loro la quantità di varianza spiegata) ed usate come fattori di partizione.

Un lavoro di ricerca non produce solo risultati, ma anche idee. Idee su come migliorare il lavoro svolto o su come approfondirne alcuni aspetti o, ancora, su nuove ipotesi da studiare. Alcune di queste idee sono già state accennate, in quanto legate direttamente alle ipotesi studiate. Qui di seguito se ne presentano alcune altre.

Innanzitutto si ricorderà (par. 2.3) che il numero delle unità nascoste è stato mantenuto pari a 17 per tutti i modelli, in base a prove empiriche compiute in un precedente lavoro (Scardi et al., 2005) su modelli multispecie addestrati sui medesimi dati. Tale numero sarebbe da verificare almeno riguardo ai modelli monospecie, per i quali probabilmente è più adeguato un numero più basso. Tuttavia, anche in base a prove preliminari effettuate in questo studio, non ci si aspetta da questa pur doverosa integrazione sostanziali cambiamenti nei risultati.

Dal punto di vista tecnico, l'intera procedura di sviluppo e training illustrata nel terzo capitolo, basata su una serie di applicativi e di file che lavorano in tandem, può essere semplificata e velocizzata riconducendo alcuni applicativi a subroutine che non richiedono la scrittura e la lettura di grandi quantità di file, riducendo così notevolmente i tempi. Si tratta però di un'operazione che va progettata con attenzione perché i tempi di sviluppo del software restino ragionevolmente contenuti.

Un riadattamento potenzialmente utile riguarda gli algoritmi di partizione (si veda par. 3.1). Essi erano stati studiati per esplorare nel modo più completo possibile, con un numero limitato di partizioni, lo spazio delle possibili assegnazioni dei casi ai subset secondo un particolare criterio; se però si vorrà, come indicato precedentemente, reimpostare il disegno sperimentale per riequilibrare il numero di prove nelle diverse condizioni, è più opportuno l'uso di un unico algoritmo che permetta di generare molte partizioni diverse sulla base di un criterio che corrisponda alla condizione sperimentale da testare. Tale algoritmo è già stato progettato (anche se il software deve ancora essere realizzato) e consiste in un'evoluzione dell'algoritmo RFA già utilizzato.

Come si è visto, la statistica K di Cohen (1960) fornisce un indice che ha più di un pregio nel misurare la performance di predittori con output binario. In particolare, essa tiene conto di tutta l'informazione contenuta nella matrice di confusione e confronta il comportamento predittivo complessivo dei modelli testati con il comportamento di un teorico predittore casuale che si basa solo sulla proporzione dei casi positivi e negativi osservati. Si tratta dunque di una misura adatta a discriminare tra modelli che posseggono una reale regola di previsione e modelli che predicono a caso. Poiché l'addestramento di un percettrone multistrato consiste proprio nel condurre un modello da uno stato di predittore casuale ad uno di

predittore efficace, la statistica K potrebbe essere usata direttamente come misura dell'errore nel processo di training (Fig. 9). In tal modo le sue proprietà sarebbero usate non solo per la valutazione dell'efficacia di un modello, ma anche per il raggiungimento di tale efficacia. L'errore quadratico medio (MSE) utilizzato tradizionalmente, infatti, espone facilmente l'addestramento, soprattutto nel caso di specie relativamente rare o ubiquitarie, al bias dovuto alla frequenza che in questo lavoro si è "inseguito" tramite l'aggiustamento della soglia di discretizzazione dell'output. L'uso della statistica K con soglia[81] a 0.5 potrebbe invece costringere la rete ad usare tutta l'informazione disponibile nei dati per costruire un modello il più possibile efficace. Una strategia simile è stata già utilizzata con successo da Scardi et al. (2005), che al MSE hanno sostituito l'indice di Rogers e Tanimoto per l'addestramento di modelli multispecie.

L'utilità di una misura di performance indipendente dalla soglia, nei capitoli precedenti più volte ribadita, ha condotto in questo lavoro alla formulazione dell'indice AUK (par. 3.1). Per valori vicini al proprio massimo (1) o al proprio minimo (-1) l'indice riflette in modo chiaro e sensibile l'efficacia e la stabilità predittiva dei modelli, indicando una capacità più o meno sviluppata di assegnare valori vicini allo 0 ai casi di assenza e vicini all'1 ai casi di presenza. Quando però si scende verso valori intermedi, cioè prossimi allo 0, il significato del valore assunto dall'indice diviene ambiguo. In Fig. 48 sono riportati 4 casi estremi in cui AUK assumerebbe sempre valore 0.

[81] Poiché il calcolo della statistica K si basa sulla costruzione di una matrice di confusione è indispensabile fissare una soglia per discretizzare l'output anche durante l'addestramento. La soglia più naturale sembra proprio 0.5, anche perché così si indurrebbe la rete a determinare un modello privo di bias dovuto alla frequenza. Si potrebbe però pensare di provare anche soglie diverse, per includere la possibilità che la rete non possa evitare una sorta di bias fisiologico in certe condizioni.

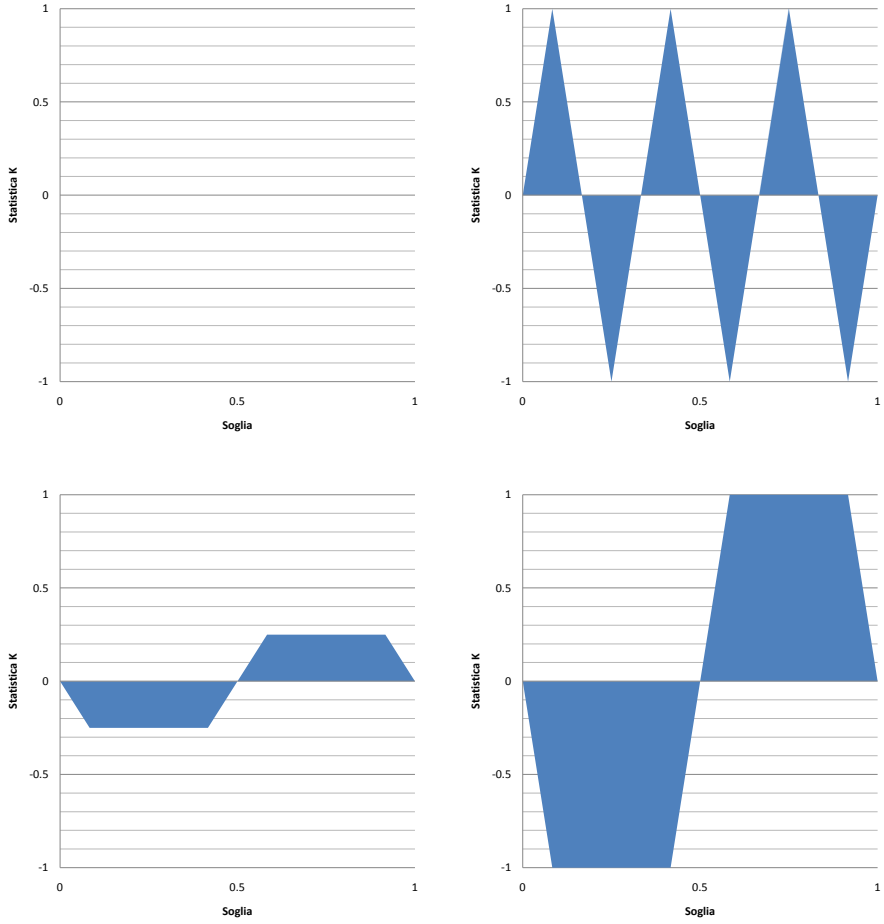


Fig. 48

Nel primo caso (in alto a sinistra) abbiamo il grafico di un predittore casuale perfetto, cioè di un modello totalmente inutile in quanto non fa che emettere 1 e 0 con una probabilità pari alla frequenza dei due casi nel test set. In alto a destra, invece, abbiamo il grafico (improbabile da ottenere nella realtà) di un modello che si comporta in modo molto diverso a seconda della soglia, raggiungendo in alcuni casi performance perfette, cioè pari a 1; il problema di un modello con questo grafico è l'instabilità della previsione al variare della soglia, per cui con mutamenti minimi di quest'ultima si rischia di passare da performance perfette a prestazioni casuali o, addirittura, sistematicamente errate. Il grafico in basso a sinistra riflette invece un modello che in modo abbastanza stabile mostra prestazioni non particolarmente

elevate, ma neanche gravemente insufficienti; se infatti si sceglie una soglia tra 0.5 e 1 si può essere ragionevolmente certi di ottenere un qualche risultato. L'ultimo grafico, quello in basso a destra (anche questo improbabile da ottenere nella realtà, ma proposto a scopo esclusivamente esemplificativo), riflette invece una performance eccellente in un range abbastanza ampio di soglie (sempre tra 0.5 e 1).

Il fatto che l'indice AUK dia 0 in tutti questi casi mette in evidenza un suo limite nel discriminare, in particolari circostanze, modelli potenzialmente molto diversi. Il problema di come ottenere una misura indipendente dalla soglia che sia efficace in molte condizioni diverse, resta dunque aperto. Una soluzione potrebbe essere quella di calcolare AUK su un intervallo limitato del range di variazione della soglia, centrato sulla soglia utilizzata. Si consideri ad esempio il valore di AUK che si otterrebbe se lo si calcolasse solo all'interno del range definito dalle due linee rosse in Fig. 49.

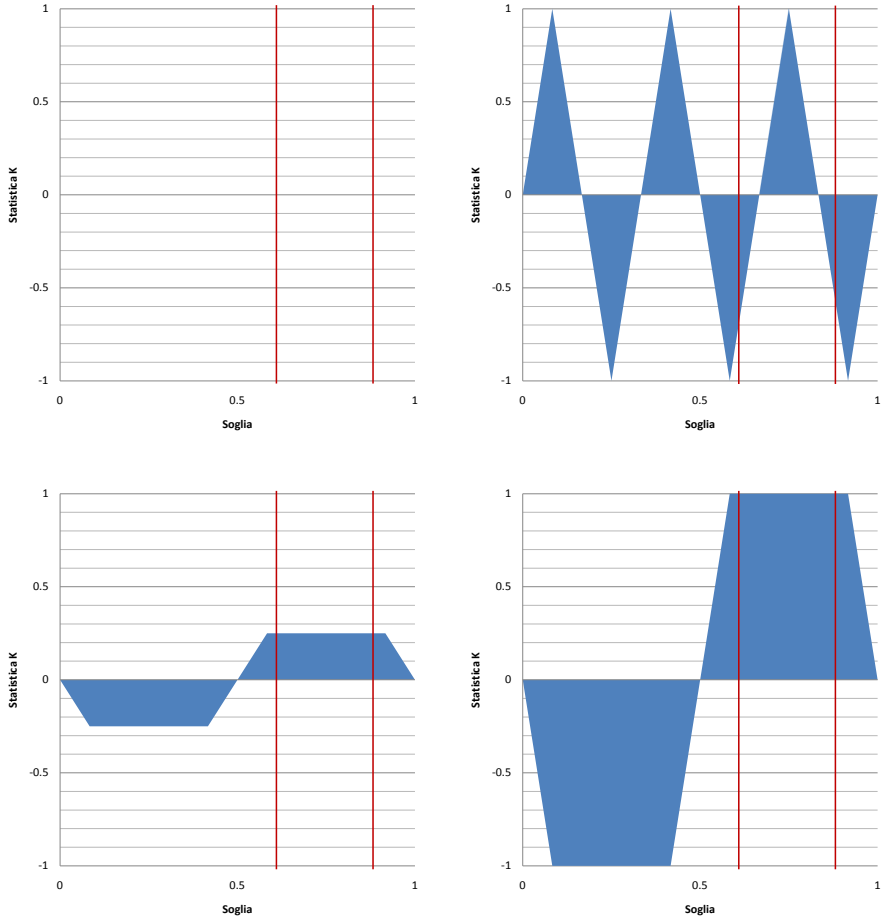


Fig. 49

Nel primo caso si otterrebbe 0, nel secondo e nel terzo si avrebbe un valore superiore allo 0 e nell'ultimo caso si otterrebbe 1, evitando così l'appiattimento a 0 mostrato dall'indice AUK semplice e ricavando una misura più valida delle capacità predittive del modello. Per fissare la posizione precisa del range di soglie da considerare, ci si potrebbe basare sul centro definito dalla soglia utilizzata (tradizionale o ottimizzata), oppure si potrebbe scegliere l'intervallo che dà il valore più elevato di AUK (considerando il suo centro come la soglia ottimizzata da utilizzare)[82]; la scelta dell'ampiezza del range resta invece parzialmente arbitraria,

[82] Chiaramente, questo fatto renderebbe la nuova misura solo parzialmente indipendente dalla soglia.

poiché legata all'uso specifico che si intende fare delle previsioni e dunque, ad esempio, alla maggiore importanza che possono rivestire nei vari casi l'efficacia o la stabilità (al variare della soglia) della previsione. Basarsi su range ampi circa un terzo dell'intero intervallo di variazione della soglia, come in figura, potrebbe costituire un criterio generale abbastanza buono per equilibrare l'esigenza di riflettere sia l'efficacia che la stabilità dei modelli.

Un'idea generale, per la quale non si è ancora pensato alle modalità di realizzazione, è quella di includere nei modelli l'aspetto spaziale, cioè fornire al modello una sorta di mappa che permetta di tenere conto, oltre che delle condizioni ambientali puntuali, anche della posizione del sito da prevedere rispetto al contesto determinato dall'intero sistema fluviale. Rappresentando l'intero bacino idrografico (o una porzione significativa di esso), i modelli potrebbero così includere in modo integrato tutto l'ecosistema, tenendo conto anche delle relazioni spaziali tra i diversi siti. Si potrebbe ottenere questo risultato, ad esempio, combinando modelli basati su reti neurali artificiali con una tecnologia GIS (Geographic Information System) oppure fornendo in modo adeguato alla rete neurale informazioni sul grafo determinato dalle aste fluviali considerate.

Nell'immediato futuro è in previsione lo sviluppo di alcune di queste idee con un data set più ampio (circa 400 osservazioni), relativo a corsi d'acqua del Centro Italia. Non si tratta in questo caso di dati acquisiti dall'esterno, ma prodotti *ad hoc* nell'ambito delle attività del Dipartimento di Biologia dell'Università di Roma "Tor Vergata"; sarà dunque possibile superare i problemi, qui più volte segnalati, dovuti al lavorare con dati relativi a contesti ambientali di cui non si ha una conoscenza adeguata. Questi dati contengono misure di tipo quantitativo riguardo alle variabili biologiche (non solo presenza/assenza, dunque, ma anche abbondanze e biomassa) e consentiranno pertanto di aggiungere nuovi ed interessanti elementi alla ricerca presentata in questo lavoro.

Cercando di trarre un bilancio complessivo, è innanzitutto doveroso riconoscere un limite fondamentale di questo lavoro, legato alla complessità del disegno sperimentale utilizzato. Essa, infatti, ha determinato tempi lunghi per la scrittura del software, ha richiesto compromessi indispensabili perché le procedure e gli strumenti di analisi si adattassero contemporaneamente a tutta un'eterogeneità di

esigenze e ha in parte reso difficile la comunicazione, sia scritta che orale, degli aspetti metodologici dello studio compiuto. Il disegno è stato così progettato per permettere di rilevare eventuali interazioni tra gli effetti legati alle diverse ipotesi, ma alla luce dell'esperienza compiuta, potrebbe rivelarsi più pratico un approccio iniziale dedicato ad ipotesi singole e un successivo approfondimento delle interazioni tra le condizioni ad esse legate.

Il pregio fondamentale del lavoro qui descritto, che ne rappresenta anche un elemento di originalità, è invece l'aver puntato non tanto alla costruzione di un modello di previsione ottimizzato o al confronto tra singoli modelli ottimizzati che differiscono per qualche caratteristica, quanto a dimostrare, attraverso la generazione di cospiscue popolazioni di modelli costruiti secondo diverse alternative di ottimizzazione, l'esistenza di differenze tra le performance di modelli rispondenti a strategie di ottimizzazione diverse (approccio sperimentale). Il livello di generalità delle conclusioni che è possibile trarre da uno studio di questo tipo è elevato e permette, a fronte di una profonda conoscenza dei sistemi naturali modellizzati, di connettere gli aspetti modellistici con la realtà ecologica.

Oltre ai risultati in sè, questo lavoro ha prodotto un metodo e uno strumento informatico congegnati per testare strategie di ottimizzazione anche diverse da quelle qui considerate e predisposti per funzionare con qualsiasi data set. Nella prospettiva di usare questi strumenti in ricerche future, l'investimento di tempo ed energie che il loro approntamento ha richiesto (legato soprattutto alla scrittura del software) assume dunque un significato che va al di là dell'interesse dei risultati ottenuti in questa sede.

RIFERIMENTI BIBLIOGRAFICI

Bishop, C.M. (1995). *Neural Networks and Pattern Recognition*. Oxford: Clarendon Press.

Bryson, A.E. Jr., & Ho Y.C. (1969). *Applied optimal control: optimization, estimation, and control*. Waltham, MA: Blaisdell [Stanford Univ., Stanford, CA and Harvard Univ., Cambridge, MA].

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 37-46.

Cybenco, *G.* (1989). Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signal and Systems*, 2, 303–314.

Fager, E.W., & McGowan, J.A., (1963). Zooplankton species groups in the North Pacific. *Science*, 140, 453–460.

Fielding, A.H. (Ed.). (1999). *Machine Learning Methods for Ecological Applications*. Boston (MA): Kluwer Academic Publishers.

Fielding, A.H., & Bell, J.F. (1997). A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environmental Conservation*, 24 (1), 38–49.

Floreano, D. (1996). Manuale sulle Reti Neurali. Bologna: Il Mulino.

Forbes, A.D. (1995) Classification algorithm evaluation: five performance measures based on confusion matrices. *Journal of Clinical Monitoring*, 11, 189–206.

Gelosi, E., & Colombari, P.T. (2004). Manuale della pesca. Ambiente fauna, pesca, attrezzi, leggi delle acque del Lazio. Roma: Romana Editrice S.r.l. (distribuzione gratuita sotto il patrocinio dello Stabilimento Ittiogenico di Roma, dell'Assessorato all'Agricoltura della Regione Lazio e dell'ARSIAL, Agenzia regionale per lo sviluppo e l'innovazione dell'agricoltura nel Lazio).

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4 (1), 1–58.

Guisan, A., & Zimmermann, N.E. (2000). Predictive habitat distribution models in ecology. *Ecological Modelling*, 135 (2–3), 147–186.

Hammer, Ø., Harper, D.A.T., & Ryan, P.D. (2001). PAST: Paleontological Statistics Software Package for Education and Data Analysis. *Palaeontologia Electronica*, 4 (1), 1-9.

Hebb, D.O. (1949). *The Organization of Behavior*. New York: Wiley.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2 (5), 359 –366.

Huet, M. (1949). Apercu des relations enetre la pente et les populations piscicoles des eaux courents. Schweiz. Zeitschr. Hydrol., 11, 332-351.

Illies, J., & Botosaneanu, L. (1963). Problèmés et méthodes de la classification et de la zonation écologique des eaux courantes, considérées surtout du point de vue faunistique. *Mitt. Int. Verein theor. Agew Limnol.*, 12, 1-57.

Kolmogorov, A.N. (1957). On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition. *Doklady Akademii Nauk SSSR*, 114, 953-956.

Lek, S., Scardi, M., Verdonshot, P., Descy, J.P., & Park Y.S. (Eds.). (2005). *Modelling Community structure in Freshwater Ecosystems*. Berlin: Springer-Verlag.

Levins, R. (1966). The Strategy of Model Building in Population Biology. *American Scientist*, 54, 421-431.

Maier, H.R., & Dandy, G.C. (2000). Neural networks for the prediction and forecasting of water resource variables: a review of modelling issues and applications. *Environmental Modelling and Software*, 15 (1), 101–124.

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.

Minsky, M.L., & Papert, S. A. (1969). *Perceptrons. An Introduction to Computational Geometry*. Cambridge, MA: MIT Press.

Olden, J. D., & Jackson, D.A. (2001). Fish-habitat relationships in lakes: gaining predictive and explanatory insight by using artificial neural networks. *Transactions of the American Fisheries Society*, 130 (5), 878–897.

Olden, J. D., Joy, M. K., & Death, R. G. (2006). Rediscovering the species in community-wide modeling. *Ecological Applications*, 16 (4), 1449 –1460.

Olden, J.D., Lawler, J.J., & Poff., N.L. (2008). Machine learning methods without tears: a primer for ecologists. *Quarterly Review of Biology*, 83, 171-193.

Özesmi, S. L., Tan, C.O., Özesmi, U. (2006). Methodological issues in building, training, and testing artificial neural networks in ecological applications. *Ecological Modelling*, 195 (1–2), 83–93.

Pessa, E. (2004*). Statistica con le reti neurali. Un'introduzione.* Roma: Di Renzo Editore.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65 (6), 386-408.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. New York: Spartan.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E, Rumelhart, & J.L., McClelland (Eds.), *Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition* (vol. 1, pp. 318-362). Cambridge, MA: MIT Press.

Scardi, M., Cataudella, S., Ciccotti, E., Di Dato, P., Maio, G., Marconato, E., Salviati, S., Tancioni, L., Turin, P., & Zanetti, M. (2005). Optimization of artificial neural networks for predicting fish assemblages in rivers. In S. Lek, M. Scardi, P.

Verdonshot, J.P. Descy, & Y.S. Park (Eds.), *Modelling Community structure in Freshwater Ecosystems* (pp. 89-122). Berlin: Springer-Verlag.

Schaafsma, W., & van Vark, G.N. (1979). Classification and discrimination problems with applications. Part IIa. *Statistica Neerlandica*, 33, 91–126.

Stoch, F. (2002). *Torrenti montani. La vita nelle acque correnti.* Collana Quaderni habitat (n. 5) del Ministero dell'Ambiente e della Tutela del Territorio e del Museo Friulano di Storia Naturale. Udine: Graphic linea print factory (distribuito gratuitamente).

Tancioni, L., Scardi, M., & Cataudella, S. (2005). I pesci nella valutazione dello stato ecologico dei sistemi acquatici. *Annali dell'Istituto Superiore di Sanità*, 41 (03), 399-402.

Tourenq, C., Aulagnier, S., Mesleard, F., Durieux, L., Johnson, A., Gonzalez, G., & Lek, S. (1999). Use of artificial neural networks for predicting rice crop damage by greater flamingos in the Camargue, France. *Ecological Modelling*, 120, 349–358.

Van Houwelingen, J.C., & Le Cessie, S. (1990). Predictive value of statistical models. *Statistics in Medicine*, 9, 1303–1325.

Vannote, R.L., Minshall, G.W., Cummins, K.W., Sedell, J.R., & Cushing, C.E. (1980). The river continuum concept. *Can. J. Fish. Aquat. Sci.*, 37, 130-137.

Werbos, P. (1974). *Beyond regression: new tools for prediction and analysis in the behavourial sciences*. Cambridge, MA: Harvard University (Ph.D. Thesis Dissertation).

Widrow, B. & Hoff, M.E., Jr. (1960). Adaptive switching circuits. In *IRE WESCON Convention Record*, 4, 96-104.

ALLEGATO A

```
'                              delphish1.bas
'                            (per brevità dph1)
'                       (by Quma, versione finale 2009)

'Questo programma predispone i file necessari alla conduzione dei test previsti
'dal disegno sperimentale del progetto di dottorato di Marco Quartararo
'(Università di Roma "Tor Vergata", XXI ciclo, tutor Prof. Michele Scardi)


'DICHIARAZIONI

Declare Sub DataInput(fname As String,ematrix() As Double,enames() As
String,vnum As Integer)
Declare Sub SenseInsertionSort(array() As Double,key As Integer,sense As
Double)
Declare Sub RawRandomSort(array() As Double)
Declare Function Minimum(matrix() As Double,column As Integer) As Double
Declare Function Maximum(matrix() As Double,column As Integer) As Double
Declare Sub RFA_Partitioner(assign() As Ubyte,SetNum() As
Uinteger,permutations() As Ubyte,iperm As Ubyte,lim1 As Ubyte,lim2 As Ubyte)
Declare Sub MCA_Partitioner(assign() As Ubyte,SetNum() As
Uinteger,permutations() As Ubyte,iperm As Ubyte)

Dim As Integer i,j,h,k,m,n,cnt 'variabili contatore da usare nei cicli; in
                               'genere si usano i e j per indicare,
                               'rispettivamente le righe e le colonne
Dim As String Input_File 'file dati
Dim As String path 'directory (con path) da cui il programma viene lanciato
Dim As String newpath
Dim As Integer cI,cD 'cI=numero variabili indipendenti (descrittori ambientali)
                     'cD=numero variabili dipendenti (specie)
Redim As Double Dmat() 'matrice dei dati
Dim As Integer r,c 'r=numero righe di Dmat(); c=numero colonne di Dmat()
Redim As String Dmat_Labels() 'etichette di colonna (nomi delle variabili)
Redim As Double Nmat() 'matrice con i dati normalizzati
Redim As Double min_I(),diff_I() 'vettori con il minimo e con la differenza tra
                                 'minimo e massimo di ogni variabile
                                 'indipendente (I); usati per normalizzare le
                                 'variabili indipendenti e ottenere dalla
                                 'matrice Dmat() la matrice Nmat()
Redim As Ubyte Pmat() 'matrice con le partizioni elaborate dal programma
Redim As Ubyte Pmat_Param() 'matrice con i parametri che identificano le varie
                            'partizioni contenute in Pmat()
Dim As Integer cP 'numero partizioni
Dim As Ubyte s,so,eo 'variabili contatore da usare nei cicli di partizione per
                     'rappresentare, rispettivamente, specie (1-33),
ordinamento
                     'specie (0/1) e ordinamento elevazione (0/1)

Dim As Ubyte iperm,lim1,lim2 'variabili contatore da usare nei cicli di
                             'partizione con la Sub RFA_Partitioner;
                             'rappresentano, rispettivamente, le permutazioni
di
                             '1-2-3 (1-6), il limite di riempimento relativo al
                             'primo (0-2) e al secondo (0-1) subset
Redim As Double RESmat() 'matrice con tre colonne: id pattern, altitudine,
                         'specie X (viene usata come versione alleggerita della
                         'matrice dati, da ordinare e randomizzare per riga nel
                         'processo di partizione, senza perdere
l'identificativo
                         'dei pattern)
```

```
Dim As Integer one,zero 'numero casi di presenza e assenza (volevo dichiararli
                        'come Uinteger, ma c'è un problema con il calcolo dei
                        'valori di SetNum())
Dim As Single meanrichness

Redim As Ubyte assign(),assign1(),assign0() 'vettori attraverso cui le Sub
                                            'RFA_Partitioner e MCA_Partitioner
                                            'restituiscono le partizioni
                                            'elaborate; quando i dati vengono
                                            'suddivisi tenendo conto dei valori
                                            'di presenza e assenza, le Sub
                                            'agiscono separatamente sui gruppi
                                            'di pattern con 1 e con 0: in quei
                                            'casi si usano assign1() e
assign0()

Dim As Ubyte cntRFA,cntMCA
Dim As Uinteger Pmat_column

Dim As String program1,program2 'sono i due programmi che fanno già parte della
                                '"cassetta degli attrezzi" di questa ricerca;
                                'scritti dal Prof. Michele Scardi,
                                'rispettivamente, conducono il training
                                '(program1="batchnnc.exe") e determinano
                                'l'output della rete addestrata
                                '(program2="NNOUT4.EXE)) per ognuno dei casi
                                'considerati; in questa versione del presente
                                'programma essi sono salvati in cartelle da
                                'cui vengono lanciati successivamente tramite
                                'file batch; in una versione successiva
                                'potrebbero essere lanciati da questo stesso
                                'programma (dph1); poiché i due programmi
                                'sono stati realizzati per eseguire e testare
                                'singoli training usando file dati e di
                                'configurazione in particolari formati, dph1
                                'genera tali file e predispone delle procedure
                                'per poter condurre attraverso di essi molte
                                'prove consecutive in modo automatico

'inizio funzione cronometro
Dim As Double total_time
total_time=timer

'in questo file viene salvato un report delle operazioni compiute dal programma
Dim As String Report_File
Report_File="REPORT.DPH"


'BLOCCO PRINCIPALE

'FASE DI INPUT

'nella prospettiva di usare questo programma per condurre in futuro la medesima
'indagine su dataset diversi, le seguenti informazioni andrebbero fornite
'manualmente; qui si è preferito tuttavia indicarle già nel codice;
'sia Input_File che i due programmi utilizzati da dph1 devono essere
'presenti, insieme a dph1 stesso, nella cartella indicata nella stringa path
Input_File="dati.csv"
cI=20
cD=32
path="c:\Delphish"
program1="batchnnc.exe"
program2="NNOUT4.EXE"
```

```
'copia dei dati in Dmat
c=cI+cD
DataInput Input_File,Dmat(),Dmat_Labels(),c
r=Ubound(Dmat,1)-Lbound(Dmat,1)+1

Open Report_File For Output As #8
Print "File dei dati: '";Input_File;"'"
Print "Path: ";path
Print
Print "   Righe: ";r
Print "   Colonne: ";c
Print "   Descrittori ambientali: ";cI
Print "   Specie:";cD
Print
Print
Print "Sto generando le partizioni. Attendere..."
Print
Print #8, "Data: ";Date;"   Orario: ";Time
Print #8,
Print #8, "File dei dati: '";Input_File;"'"
Print #8, "Path: ";path
Print #8,
Print #8, "   Righe: ";r
Print #8, "   Colonne: ";c
Print #8, "   Descrittori ambientali: ";cI
Print #8, "   Specie:";cD
Print #8,



'FASE DI ELABORAZIONE PER LA GENERAZIONE DELLE PARTIZIONI

'la matrice Pmat() conterrà le partizioni elaborate dal programma; le sue righe
'corrispondono, nel numero e nell'ordine, alle righe della matrice dei dati,
'mentre le colonne contengono ognuna una partizione (in ogni riga viene
'riportato '1', '2' o '3' a seconda che il pattern corrispondente a quella riga
'nella matrice dei dati risulti assegnato, nella specifica partizione
'rappresentata dalla singola colonna, rispettivamente, al subset di training,
'validazione o test); il numero cP delle colonne di questa matrice dipende dai
'criteri che si sono scelti per operare le partizioni e sarà determinato a
breve

'la matrice delle etichette delle varie partizioni Pmat_Param() è molto
'importante: la lettura sequenziale dei numeri in essa riportati dalla prima
'alla quarta riga in corrispondenza di ogni colonna indica, rispettivamente,
'1) la specie che è stata presa come riferimento nell'operare la partizione (i
'numeri da 1 a 32 indicano le singole specie, il numero 33 indica la variabile
'"ricchezza di specie"), 2) se si è curato di distribuire equamente i pattern
in
'cui la specie è presente o assente nei tre subset ('1' se sì, '0' se no),
'3) se si è curato di distribuire equamente ai tre subset i vari livelli del
'continuum del descrittore 'altitudine' ('1' se sì, '0' se no) e 4) il metodo
'con cui è stata effettuata la suddivisione in subset (i metodi usati sono due,
'ma ognuno può dar luogo a 36 diverse assegnazioni, per cui il numero riportato
'in Pmat_Param() andrà da 1 a 36 in riferimento al primo metodo, detto "Regular
'Filling Assignation" ed eseguito dalla Sub RFA_Partitioner(), e da 37 a 72 in
'riferimento al secondo, detto "Modular Chain Assignation" ed eseguito dalla
Sub
'MCA_Partitioner())

'la caratteristica principale delle 72 possibili modalità di assegnazione dei
'pattern ai tre subset è che, dato un ordinamento secondo una specie e/o un
'descrittore, l'assegnazione distribuisce in modo equo i casi di
'presenza/assenza e/o il continuum del descrittore tra i subset.
```

```
'Volendo provare ad addestrare tutti i modelli su partizioni che tengano conto
o
'no dei dati di presenza/assenza delle specie e che tengano conto o no della
'distribuzione del descrittore 'altitudine', si generano quattro condizioni
'(indicate dalle coppie 00, 01, 10, 11 presenti alle righe 2 e 3 di
'Pmat_Param()... una tabella a doppia entrata rappresenta bene questo incrocio
'tra condizioni). Se per ogni condizione generiamo 72 assegnazioni,
'avremo 72x4=288 diverse partizioni generate sulla base di ogni singola specie.
'Poiché le specie sono 32 e ad esse si aggiunge la variabile "ricchezza di
'specie", complessivamente si avranno dunque 288x33=9504 diverse partizioni dei
'dati (è come se le 33 tabelle a doppia entrata formassero adesso un lungo
'parallelepipedo con la sezione quadrata). La metà di queste partizioni (metà
'superiore del parallelepipedo) non tiene conto delle specie (lascia al caso la
'distribuzione dei casi di presenza/assenza). Un'altra metà (ovviamente non
'complementare alla prima; metà sinistra del parallelepipedo) non tiene conto
'della distribuzione dell'altitudine. Un quarto di tutte le partizioni non
tiene
'conto di nulla; un altro quarto contiene i 33 angoli a destra in alto delle
'tabelle a doppia entrata, con i casi in cui si tiene conto di una specie e
'dell'altitudine.
Dim As Double thistime
thistime=Timer

cP=72*4*(cD+1) '=9504

Redim As Ubyte Pmat(1 To r,1 To cP)
Redim As Ubyte Pmat_Param(1 To 8,1 To cP) 'i parametri relativi alle otto righe
                                          'sono:
                                            '1) specie
                                            '2) ordinamento specie (0/1)
                                            '3) ordinamento per altitudine
(0/1)
                                            '4) algoritmo di partizione (1-72)
                                            '5) numero casi di presenza
                                            '6-7-8) numerosità subset
                                            'di training-validazione-test

Redim As Double RESmat(1 To r,1 To 3)
Redim As Ubyte assign(1 To r)

Dim As Ubyte specol 'colonna della specie s
Redim As Uinteger SetNum0(1 To 3),SetNum1(1 To 3) 'numerosità dei casi di
                                                  'assenza (SetNum0()) e di
                                                  'presenza (SetNum1()) nei tre
                                                  'subset quando si tenga conto
                                                  'dell'ordinamento per specie
Redim As Uinteger SetNum(1 To 3) 'numerosità dei subset quando non si tiene
                                 'conto dei casi di presenza e assenza

'equivalenti dei precedenti, usati per il passaggio alla Sub MCA_Partitioner,
'che li determina di volta in volta
Redim As Uinteger SetNum0_MCA(1 To 3),SetNum1_MCA(1 To 3),SetNum_MCA(1 To 3)

'riporta le permutazioni di 1, 2 e 3, necessarie alla Sub RFA_Partitioner
Redim As Ubyte perm123(1 To 6,1 To 3)

'(rivedere: questa parte di codice può essere migliorata)
For j=1 To 3
    perm123(1,j)=j
    perm123(4,j)=-j+4
Next
For i=2 To 3
    For j=1 To 3
        perm123(i,1)=perm123(i-1,3)
```

```
        perm123(i,2)=perm123(i-1,1)
        perm123(i,3)=perm123(i-1,2)
        perm123(i+3,1)=perm123(i+2,2)
        perm123(i+3,2)=perm123(i+2,3)
        perm123(i+3,3)=perm123(i+2,1)
    Next
Next

'riporta le permutazioni dei numeri 1, 1, 2 e 3 (numericamente corrispondenti
'alla metà delle permutazioni di 4 numeri diversi), necessarie alla Sub
'MCA_Partitioner
Redim As Ubyte perm1123(1 To 12,1 To 4)
Redim As Ubyte temp(1 To 12,1 To 8)

'(rivedere: questa parte di codice può essere migliorata)
For i=1 To 11 step 2
    temp(i,3)=2
    temp(i,6)=3
    temp(i+1,3)=3
    temp(i+1,6)=2
Next
For i=1 To 5 step 2
    temp(i,(3/2)*i-(1/2))=1
    temp(i,(3/2)*i+(1/2))=1
    temp(i+1,(3/2)*i-(1/2))=1
    temp(i+1,(3/2)*i+(1/2))=1
Next
i=7
For j=1 To 8 step 7
    temp(i,j)=1
    temp(i+1,j)=1
Next
For i=9 To 11 step 2
    temp(i,4)=1
    temp(i,3*i-26)=1
    temp(i+1,4)=1
    temp(i+1,3*i-26)=1
Next

For i=1 To 12
    cnt=0
    For j=1 To 8
        if temp(i,j)<>0 then
            cnt+=1
            perm1123(i,cnt)=temp(i,j)
        end if
    Next
Next

'determinazione del numero di casi da assegnare ai tre subset; il criterio è
che
'al subset di training viene attribuita la metà (eventualmente approssimata per
'eccesso) dei casi, al subset di validazione la metà (eventualmente
approssimata
'per eccesso) dei casi rimanenti e al subset di test i casi rimanenti; le tre
'numerosità sono salvate in SetNum()
SetNum(1)=-Int(-r/2)
SetNum(2)=-Int(-((r-SetNum(1))/2))
SetNum(3)=r-SetNum(1)-SetNum(2)

'questo ciclo compie per ogni specie le operazioni necessarie ad ottenere le
'partizioni in tutte le condizioni desiderate (9504 in tutto)
For s=1 To cD
```

```
    'fissazione delle condizioni valide sempre, una volta fissata la specie di
    'riferimento

    'calcolo del numero dei pattern da attribuire ai tre subset nel caso di
    'assenza o presenza; tali valori vengono salvati, rispettivamente, nei
    'vettori a tre elementi SetNum0() e SetNum1(); definite le quantità 'zero'
    'e 'one' dei casi di assenza e presenza, si procede come per SetNum()

    'vengono contati i casi di assenza (zero) e presenza (one)
    specol=cI+s 'poiché le colonne dei cI descrittori sono le prime nella
                'matrice dei dati, 'specol', a partire da s, identifica in tale
                'la colonna corrispondente alla specie considerata

    'riempimento di RESmat
    one=0 'numero dei casi di presenza inizializzato a 0
    For i=1 To r
        RESmat(i,1)=i 'la prima colonna contiene l'ID di riga, corrispondente
                      'alla posizione di ogni pattern nella matrice dei dati
        RESmat(i,2)=Dmat(i,1) 'nella seconda colonna vengono copiati i valori
                              'del descrittore "elevation"
        RESmat(i,3)=Dmat(i,specol) 'nella terza colonna vengono copiati i
valori
                                   'di presenza/assenza (0/1) della specie s
        one+=RESmat(i,3)
    Next
    zero=r-one

    'quando si tiene conto dei valori di presenza/assenza nella partizione, il
    'numero di casi di assenza e presenza da assegnare ai subset viene
calcolato
    'separatamente
    SetNum0(1)=-Int(-zero/2)
    SetNum0(2)=-Int(-((zero-SetNum0(1))/2))
    SetNum0(3)=zero-SetNum0(1)-SetNum0(2)
    SetNum1(1)=-Int(-one/2)
    SetNum1(2)=-Int(-((one-SetNum1(1))/2))
    SetNum1(3)=one-SetNum1(1)-SetNum1(2)

    'dimensionamento dei vettori usati dalle Sub RFA_Partitioner e
    'MCA_Partitioner per restituire le partizioni elaborate
    Redim As Ubyte assign0(1 To zero)
    Redim As Ubyte assign1(1 To one)

    For so=0 To 1
        For eo=0 To 1
            RawRandomSort RESmat() 'che si desideri o meno un ordinamento per
                                   'l'altitudine o per la specie, l'ordine dei
                                   'pattern viene innanzitutto randomizzato,
sia
                                   'per eliminare eventuali ordinamenti
                                   'precedenti, che per trovare quel minimo di
                                   'variazione ancora possibile nonostante i
                                   'vincoli di ordinamento (la matrice completa
                                   'dei dati non si riduce solo alla specie in
                                   'esame e all'altitudine, ma a molte altre
                                   'specie e variabili, per cui, se vi sono due
                                   'o più righe identiche per i valori
                                   'dell'altitudine e della specie considerata,
                                   'esse potrebbero comunque differire in altri
                                   'campi; una randomizzazione pre-ordinamento
                                   'assicura una distribuzione random tra le
                                   'partizione di queste variazioni)
            SenseInsertionSort RESmat(),2,eo
            SenseInsertionSort RESmat(),3,so
```

```
            cntRFA=0
            cntMCA=0

            'partizioni con RFA_Partitioner
            For iperm=1 To 6
                For lim1=0 To 2
                    For lim2=0 To 1
                        cntRFA+=1
                        Pmat_column+=1
                        Pmat_Param(1,Pmat_column)=s
                        Pmat_Param(2,Pmat_column)=so
                        Pmat_Param(3,Pmat_column)=eo
                        Pmat_Param(4,Pmat_column)=cntRFA
                        Pmat_Param(5,Pmat_column)=one

                        If so=1 Then
                            'poiché i valori di SetNum() possono essere
                            'modificati dalla Sub (con RFA succede solo quando
                            'one/zero<=3, mentre con MCA succede sempre) la
loro
                            'copia in Pmat_Param() avviene dopo la chiamata
                            RFA_Partitioner
assign0(),SetNum0(),perm123(),iperm,lim1,lim2
                            RFA_Partitioner
assign1(),SetNum1(),perm123(),iperm,lim1,lim2
                            For i=1 To zero
                                Pmat(RESmat(i,1),Pmat_column)=assign0(i)
                            Next
                            For i=1 To one
                                Pmat(RESmat(zero+i,1),Pmat_column)=assign1(i)
                            Next
                            For i=1 To 3

Pmat_Param(i+5,Pmat_column)=SetNum0(i)+SetNum1(i)
                            Next
                        Else
                            RFA_Partitioner
assign(),SetNum(),perm123(),iperm,lim1,lim2
                            For i=1 To r
                                Pmat(RESmat(i,1),Pmat_column)=assign(i)
                            Next
                            For i=1 To 3
                                Pmat_Param(i+5,Pmat_column)=SetNum(i)
                            Next
                        End If

                    Next
                Next
            Next

            'partizioni con MCA_Partitioner
            For iperm=1 To 36
                cntMCA+=1
                Pmat_column+=1
                Pmat_Param(1,Pmat_column)=s
                Pmat_Param(2,Pmat_column)=so
                Pmat_Param(3,Pmat_column)=eo
                Pmat_Param(4,Pmat_column)=36+cntMCA '36=cntRFA
                Pmat_Param(5,Pmat_column)=one

                If so=1 Then
                    MCA_Partitioner assign0(),SetNum0_MCA(),perm1123(),iperm
                    MCA_Partitioner assign1(),SetNum1_MCA(),perm1123(),iperm
```

```
                    For i=1 To zero
                        Pmat(RESmat(i,1),Pmat_column)=assign0(i)
                    Next
                    For i=1 To one
                        Pmat(RESmat(zero+i,1),Pmat_column)=assign1(i)
                    Next
                    For i=1 To 3

Pmat_Param(i+5,Pmat_column)=SetNum0_MCA(i)+SetNum1_MCA(i)
                    Next
                Else
                    MCA_Partitioner assign(),SetNum_MCA(),perm1123(),iperm
                    For i=1 To r
                        Pmat(RESmat(i,1),Pmat_column)=assign(i)
                    Next
                    For i=1 To 3
                        Pmat_Param(i+5,Pmat_column)=SetNum_MCA(i)
                    Next
                End If

            Next
        Next
    Next
Next

'le operazioni effetuate sopra per le varie specie, vanno ora condotte, mutatis
'mutandis, per la variabile "ricchezza di specie"
For i=1 To r
    RESmat(i,1)=i
    RESmat(i,2)=Dmat(i,1)
    RESmat(i,3)=0
    For j=cI+1 To c
        RESmat(i,3)+=Dmat(i,j)
    Next
Next

s=cD+1
For i=1 To r
    meanrichness+=RESmat(i,3)
Next
meanrichness/=r

For so=0 To 1
    For eo=0 To 1
        RawRandomSort RESmat()
        SenseInsertionSort RESmat(),2,eo
        SenseInsertionSort RESmat(),3,so

        cntRFA=0
        cntMCA=0

        'partizioni con RFA_Partitioner
        For iperm=1 To 6
            For lim1=0 To 2
                For lim2=0 To 1
                    cntRFA+=1
                    Pmat_column+=1
                    Pmat_Param(1,Pmat_column)=s
                    Pmat_Param(2,Pmat_column)=so
                    Pmat_Param(3,Pmat_column)=eo
                    Pmat_Param(4,Pmat_column)=cntRFA
                    Pmat_Param(5,Pmat_column)=meanrichness
                    RFA_Partitioner assign(),SetNum(),perm123(),iperm,lim1,lim2
                    For i=1 To r
```

```
                    Pmat(RESmat(i,1),Pmat_column)=assign(i)
                Next
                For i=1 To 3
                    Pmat_Param(i+5,Pmat_column)=SetNum(i)
                Next
            Next
        Next
    Next

    'partizioni con MCA_Partitioner
    For iperm=1 To 36
        cntMCA+=1
        Pmat_column+=1
        Pmat_Param(1,Pmat_column)=s
        Pmat_Param(2,Pmat_column)=so
        Pmat_Param(3,Pmat_column)=eo
        Pmat_Param(4,Pmat_column)=36+cntMCA '36=cntRFA
        Pmat_Param(5,Pmat_column)=meanrichness
        MCA_Partitioner assign(),SetNum_MCA(),perm1123(),iperm
        For i=1 To r
            Pmat(RESmat(i,1),Pmat_column)=assign(i)
        Next
        For i=1 To 3
            Pmat_Param(i+5,Pmat_column)=SetNum(i)
        Next
    Next
Next
Next


Dim As Ubyte models
models=cD+1+1 '(numero delle specie + ricchezza di specie + modello
complessivo)

Print "   Sono state generate";cP;" partizioni"
Print Using "   in #.## secondi";Timer-thistime
Print
Print "Attenzione: da questo momento in poi il programma lavorera' senza
soluzione"
Print "di continuita' per circa un paio d'ore!"
Print
Print #8, cP;" partizioni sono state generate"
Print #8, "in ";Timer-thistime;" secondi"
Print #8,

'FASE DI CREAZIONE DEI FILE E DELLE DIRECTORY

'   Vengono create tante directory quanti sono i modelli (dunque cD+1+1) e
'ognuna viene riempita con i relativi file di training (*.tra), test (*.tes) e
'configurazione (*.cfg) nonché con gli eseguibili batchncc.exe e NNOUT4.EXE,
che
'vengono copiati dalla directory di lancio di dph1, e con i due file batch per
'lanciare manualmente, al termine del lavoro di dph1, ogni serie di 9504
'esperimenti relativa ad ogni singolo modello

Dim As Uinteger tnum,itnum
tnum=cP*models 'numero totale di esperimenti, corrispondente al prodotto tra
               'le partizioni e i modelli
Redim As Uinteger Train_Param(1 To 10,1 To tnum)
'in modo analogo a Pmat_Param(), questa matrice riporta in ogni colonna i
'parametri relativi ai tnum addestramenti che verranno condotti; i parametri
'sono: 1) modello di addestramento (monospecie/multispecie-->0/1); 2) specie
'(ognuna è identificata tramite un numero da 1 a cD); 3) ordinamento specie
'(0/1); 4) ordinamento altitudine (0/1); 5) algoritmo di partizione;
```

```
'6) partizione di riferimento (corrispondente alla colonna di Pmat); 7) numero
'dei casi di presenza; 8-10) numerosità dei tre subset

For i=0 To models-1
    For j=1 To cP
        itnum+=1
        Train_Param(1,itnum)=i
        For h=1 To 4
            Train_Param(h+1,itnum)=Pmat_Param(h,j)
        Next
        Train_Param(6,itnum)=j
        For h=7 To 10
            Train_Param(h,itnum)=Pmat_Param(h-2,j)
        Next
    Next
Next

Redim As String Model_Label(1 To tnum)
Redim As String config_file(1 To tnum),weight_file(1 To tnum)
Redim As String training_file(1 To tnum),test_file(1 To tnum)
Redim As String out_file(1 To tnum)

For i=1 To tnum
    For j=1 To 5
        If (j<3 or j=5) and Train_Param(j,i)<10 Then
            Model_Label(i)+=Str(0)+Str(Train_Param(j,i))
        Else
            Model_Label(i)+=Str(Train_Param(j,i))
        End If
    Next
    training_file(i)=Model_Label(i)+".tra"
    test_file(i)=Model_Label(i)+".tes"
    config_file(i)=Model_Label(i)+".cfg"
    weight_file(i)=Model_Label(i)+".wgt"
    out_file(i)=Model_Label(i)+".out"
Next

'normalizzazione delle variabili indipendenti in Dmat() e copia dei risultati
'nella matrice Nmat(), da cui saranno presi i pattern da copiare nei file di
'training (.tra) e test (.tes); questa matrice avrà una colonna in più, per
'poter aggiungere la ricchezza di specie
SenseInsertionSort RESmat(),1,1 'riordiniamo secondo la prima colonna RESmat(),
                                'che nella terza colonna contiene ancora i
                                'valori della ricchezza di specie
Redim As Double Nmat(1 To r,1 To c+1)
Redim As Double min_I(1 To cI+1),diff_I(1 To cI+1)
For j=1 To cI
    min_I(j)=Minimum(Dmat(),j)
    diff_I(j)=Maximum(Dmat(),j)-min_I(j)
Next
min_I(cI+1)=Minimum(RESmat(),3)
diff_I(cI+1)=Maximum(RESmat(),3)-min_I(cI+1)
For j=1 To cI
    For i=1 To r
        Nmat(i,j)=(Dmat(i,j)-min_I(j))/diff_I(j)
    Next
Next
For i=1 To r
    Nmat(i,c+1)=(RESmat(i,3)-min_I(cI+1))/diff_I(cI+1)
Next
For j=cI+1 To c 'i dati binari delle variabili dipendenti vengono solo copiati
    For i=1 To r
        Nmat(i,j)=Dmat(i,j)
    Next
```

```
Next

Redim As String config_rows(1 To 13) 'vettore con le righe da copiare nei file
                                     'di configurazione; la maggior parte delle
                                     'righe è uguale per tutti i modelli e
viene
                                     'fissata subito una volta per tutte;
                                     'le righe variabili vengono determinate
                                     'volta per volta quando i singoli file di
                                     'configurazione vengono scritti
config_rows(1)=Str(10000) 'cicli di training (epoche)
config_rows(2)=Str(100) 'cicli di ripartenza in caso di minimo locale
config_rows(3)=Str(0) 'outliers per il calcolo dell'errore
config_rows(4)=Str(0.01) 'rumore gaussiano aggiunto agli input
config_rows(5)=Str(17) 'numero nodi nascosti
'config_rows(6)=Str(Train_Param(8,k)) - numero pattern per il training
'config_rows(7)=Str(Int(Train_Param(8,k)/2)) - pattern estratti a sorte per
ogni epoca
config_rows(8)=Str(0.9) 'learning rate
config_rows(9)=Str(0.1) 'momentum
config_rows(10)=Str(0.05) 'rumore aggiunto ai pesi al riaggiungimento di un
minimo locale
config_rows(11)=Str(0) 'opzione salvataggio errori (0=no; 1=sì)
config_rows(12)="nessuno" 'pesi generati casualmente all'inizio
'config_rows(13)=weight_file(k) - nome file pesi di output


Print "Sto creando ";models;" directory (una per modello) e ";cP;" file di
configurazione,"
Print "training e test da salvare in ognuna di esse (per un totale di ";cP*3;"
file"
Print "per directory e ";cP*3*models;" file in tutto)."
Print "All'interno di ogni directory sto salvando anche una copia del
programma"
Print "batchnnc.exe e una del programma NNOUT4.EXE. Attendere..."
Print
Print #8, "Sono state create ";models;" directory (una per modello) e ";cP;"
file"
Print #8, "di configurazione, training e test per in ognuna di esse (";cP*3;"
file"
Print #8, "per directory e ";cP*3*models;" file in tutto)."
Print #8, "All'interno di ogni directory è stata salvata una copia del
programma"
Print #8, "batchnnc.exe e una del programma NNOUT4.EXE. Attendere..."
Print #8,
Print "   Sto creando i file per il modello 0/";Str(models-1);". Attendere..."
Print #8, "   Creazione dei file per il modello 0/";Str(models-1);

Redim As String folder(0 To models-1)
Dim As String input_nodes,output_nodes
input_nodes=Str(cI)
output_nodes=Str(cD)
itnum=0
For i=0 To models-1
    thistime=Timer

    If i>0 Then
        output_nodes=Str(1)
    End If

    folder(i)=Mid(Model_Label((i+1)*cP),1,2)
    Mkdir folder(i) 'creo la directory in cui salvare i file di configurazione,
                    'training, test, ecc. del modello i
    Chdir folder(i) 'mi sposto nella nuova directory
```

```
Shell "copy "+path+"\"+program1 'il programma viene copiato in folder(i)
Shell "copy "+path+"\"+program2
For j=1 To cP 'creo i file di configurazione
    itnum+=1
    Open config_file(itnum) For Output As #4
    For h=1 To 5
        Print #4, config_rows(h)
    Next
    Print #4, Str(Train_Param(8,itnum))
    Print #4, Str(Int(Train_Param(8,itnum)/2))
    For h=8 To 12
        Print #4, config_rows(h)
    Next
    Print #4, weight_file(itnum)
    Close(4)

    'creazione dei file di training e test;
    'i due programmi batchnnc.exe e NNOUT4.EXE richiedevano in origine
    'la trasformazione dei file dati grezzi (in formato testo) da parte di
    'un terzo programma, BAS2FOR2.EXE, per assumere il formato richiesto
    'dai due programmi; dph1 implementa le operazioni di BAS2FOR2.EXE e
    'genera direttamente i file di training e test nel formato opportuno
    Open training_file(itnum) For Output As #12
    Open test_file(itnum) For Output As #3
    Print #12, input_nodes
    Print #3, input_nodes
    Print #12, output_nodes
    Print #3, output_nodes
    Print #12, Str(Train_Param(8,itnum)+Train_Param(9,itnum))
    Print #3, Str(Train_Param(10,itnum))
    For h=1 To r
        If Pmat(h,Train_Param(6,itnum))=1 Then 'se la riga h è stata
                                               'assegnata al subset 1
            For k=1 To cI
                Print #12, Nmat(h,k)
            Next
            If Train_Param(1,itnum)=0 Then 'se il modello è quello che
                                           'predice tutte le specie
                For k=cI+1 To c
                    Print #12, Nmat(h,k)
                Next
            Else
                Print #12, Nmat(h,cI+Train_Param(1,itnum))
            End If
        Elseif Pmat(h,Train_Param(6,itnum))=3 Then  'se la riga h è stata
                                                    'assegnata al subset 3
            For k=1 To cI
                Print #3, Nmat(h,k)
            Next
            If Train_Param(1,itnum)=0 Then
                For k=cI+1 To c
                    Print #3, Nmat(h,k)
                Next
            Else
                Print #3, Nmat(h,cI+Train_Param(1,itnum))
            End If
        End If
    Next
    For h=1 To r
        If Pmat(h,Train_Param(6,itnum))=2 Then  'se la riga h è stata
                                               'assegnata al subset 2
            For k=1 To cI
                Print #12, Nmat(h,k)
            Next
```

```
                If Train_Param(1,itnum)=0 Then
                    For k=cI+1 To c
                        Print #12, Nmat(h,k)
                    Next
                Else
                    Print #12, Nmat(h,cI+Train_Param(1,itnum))
                End If
            End If
        Next
        Close(12)
        Close(3)

    Next
    Chdir path 'scendo di un livello e torno alla directory da cui ho lanciato
               'il programma, in modo che al prossimo ciclo Mkdir crei la
               'directory folder(i+1) sullo stesso livello di folder(i)

    Print Using "   ...operazione completata in #.## minuti";(Timer-
thistime)/60
    Print #8, "   completata in #.## minuti";(Timer-thistime)/60
    Print
    Print #8,
    If i<models-1 Then
        Print "   Sto creando i file per il modello";i;"/";Str(models-1);".
Attendere..."
        thistime=Timer
    End If

Next

Print
Print "Sto creando per ogni modello i due file batch che guidano il training e"
Print "il test (rispettivamente 00TR.BAT, che lancia batchnnc.exe e cancella"
Print "contestualmente i file non più necessari, e 01TE.BAT, che lancia"
Print "NNOUT4.EXE)."
Print "Sto creando anche un file di report, 'DPH1.rep', che costituirà il punto
di"
Print "partenza per DPH2.EXE, che analizza i risultati dei vari test."
Print "   Attendere..."
Print
Print #8,
Print #8, "Sono stati creati per ogni modello i due file batch che guidano il
training"
Print #8, "e il test (rispettivamente 0000TR.BAT, che lancia batchnnc.exe e
cancella"
Print #8, "contestualmente i file non più necessari, e 01TE.BAT, che lancia"
Print #8, "NNOUT4.EXE)."
Print #8, "E' stato creato anche un file di report, 'DPH1.rep', che costituirà
il"
Print #8, "punto di partenza per DPH2.EXE, che analizza i risultati dei vari
test."
Print #8,

Redim As String training_bat(0 To models-1),test_bat(0 To models-1),dph_rep(0
To models-1)
thistime=Timer
itnum=0

For i=0 To models-1
    newpath=path+"\"+folder(i)+"\"
    training_bat(i)="00TR"+folder(i)+".BAT"
    test_bat(i)="01TE"+folder(i)+".BAT"
    If i<10 Then
        dph_rep(i)="dph0"+Str(i)+".rep"
```

```
    Else
        dph_rep(i)="dph"+Str(i)+".rep"
    End If
    Open training_bat(i) For Output As #5
    Open test_bat(i) For Output As #6
    Open dph_rep(i) For Output As #7
    Print #7, Str(cI)
    Print #7, Str(cD)
    Print #7, Str(r)
    For j=1 To cP
        itnum+=1
        Print #5, program1;" ";training_file(itnum);" ";config_file(itnum)
        Print #5, "del ";Model_Label(itnum);".cff"
        Print #5, "del ";training_file(itnum)
        Print #5, "del ";config_file(itnum)
        Print #6, program2;" ";test_file(itnum);" ";weight_file(itnum);"
";out_file(itnum)
        Print #6, "del ";weight_file(itnum)
        For h=1 To 9
            Print #7, Train_Param(h,itnum);",";
        Next
        Print #7, Train_Param(10,itnum)
    Next
    Print #5, "del batchnnc.mse"
    Print #5, "del temp.err"
    Close(5)
    Close(6)
    Close(7)
    Close(9)
    name training_bat(i), newpath+training_bat(i)
    name test_bat(i), newpath+test_bat(i)
    name dph_rep(i), newpath+dph_rep(i)
Next
Print
Print
Print Using "FINE PROGRAMMA. TEMPO TOTALE: #.## ore";(Timer-total_time)/3600
Print "   (premere un tasto per uscire)"
Print #8,
Print #8, "TEMPO TOTALE DI ELABORAZIONE: ";(Timer-total_time)/3600;" ore."
Close(8)

sleep
End



'SUBROUTINE E FUNZIONI

  'acquisisce una matrice di dati (inclusa una prima riga con i nomi delle
  'variabili) da un file di testo e la copia in una matrice vuota [ematrix()
  ''empty matrix'] che viene passata insieme al nome del file [fname
  ''file name'], al numero di variabili (colonne) [vnum 'variables number'] e
ad
  'un vettore vuoto in cui copiare i nomi delle variabili [enames()
  ''empty names']
Sub DataInput(fname As String,ematrix() As Double,enames() As String,vnum As
Integer)
    Dim As Integer i,r
    Redim enames(1 To vnum) As String
    Open fname For input As #1
    For i=1 To vnum
        input #1, enames(i)
    Next
    r=0
```

```
    Do While Not Eof (1)
        r+=1
        Redim preserve ematrix(1 To r,1 To vnum)
        For i=1 To vnum
            Input #1, ematrix(r,i)
        Next
    Loop
End Sub


  'versione modificata di InsertionSort: in base ai valori contenuti nella
  'colonna 'key', ordina le righe della matrice array in modo crescente o
  'decrescente a seconda che 'sense' venga passato con valore, rispettivamente,
  'positivo o negativo); se sense=0, la matrice resta invariata;
Sub SenseInsertionSort(array() As Double,key As Integer,sense As Double)
    If sense <> 0 Then
        Dim As Integer i,j,r,c,k
        'determina i limiti della matrice da ordinare
        r = Ubound(array,1) - Lbound(array,1) + 1
        c = Ubound(array,2) - Lbound(array,2) + 1
        'vettore temporaneo (tanti elementi quante colonne nella matrice da
        'ordinare)
        Dim As Double tmp(1 To c)
        For i = 2 To r
            For j = 1 To c
                tmp(j) = array(i,j)
            Next
            For k = i - 1 To 1 Step -1
                If sense > 0 Then
                    If tmp(key) < array(k,key) Then
                        For j=1 To c
                            array(k+1,j) = array(k,j)
                        Next
                    Else
                        Exit For
                    End If
                Else
                    If tmp(key) > array(k,key) Then
                        For j=1 To c
                            array(k+1,j) = array(k,j)
                        Next
                    Else
                        Exit For
                    End If
                End If
            Next
            For j = 1 To c
                array(k+1,j) = tmp(j)
            Next
        Next
    End If
End Sub


'trova il massimo tra i valori di una colonna in una matrice
Function Maximum(matrix() As Double,column As Integer) As Double
     Dim As Integer r,i
     Dim As Double max
     r=Ubound(matrix,1)-Lbound(matrix,1)+1
     max=matrix(1,column)
     For i=1 To r
         If matrix(i,column)>max Then
             max=matrix(i,column)
         End If
```

```
     Next
     Return max
End Function

'trova il minimo tra i valori di una colonna in una matrice
Function Minimum(matrix() As Double,column As Integer) As Double
     Dim As Integer r,i
     Dim As Double min
     r=Ubound(matrix,1)-Lbound(matrix,1)+1
     min=matrix(1,column)
     For i=1 To r
         If matrix(i,column)<min Then
             min=matrix(i,column)
         End If
     Next
     Return min
End Function


 'rimescola le righe di una matrice cambiando a caso almeno una volta la
 'posizione di ognuna
Sub RawRandomSort(array() As Double)
    Dim As Integer r,c,i,j
    Dim As Integer two
    r=Ubound(array,1)-Lbound(array,1)+1
    c=Ubound(array,2)-Lbound(array,2)+1
    Redim As Double tmp(1 To c)
    Randomize Timer
    For i=1 To r
        two=Int(Rnd*r+1)
        For j=1 To c
            tmp(j)=array(i,j)
            array(i,j)=array(two,j)
            array(two,j)=tmp(j)
        Next
    Next
End Sub

 'restituisce un vettore colonna di valori '1', '2' o '3' (assign()) che
 'indicano l'assegnazione dei pattern (righe) di una matrice di dati
 'rispettivamente al subset di training, validazione o test per l'addestramento
 'di reti neurali; assign() è un vettore che viene passato già dimensionato: la
 'Sub ne pone a zero gli elementi (poichè il vettore potrebbe già essere stato
 'utilizzato nel programma principale) e attraverso di esso restituisce il
 'proprio elaborato; SetNum() è un vettore a tre elementi, attraverso cui la
Sub
 'riceve le numerosità che devono avere i tre subset; permutations() è una
 'matrice fornita dall'esterno, con le sei possibili permutazioni dei numeri 1,
 '2 e 3; iperm indica quale permutazione si desidera che la Sub usi per
 'stabilire un'ordine tra i subset, da cui dipende a quale subset verranno
 'assegnati i pattern per primo, secondo o terzo; questo implica due cose: 1)
il
 'primo, il secondo e il terzo pattern verranno assegnati, rispettivamente al
 'primo, al secondo e al terzo subset nell'ordine stabilito da iperm; 2) lim1
 'stabilisce se l'ultimo pattern da assegnare al primo subset è l'ultimo
 '(lim1=0), il penultimo (lim1=1) o il terzultimo (lim1=2), mentre lim2, in
modo
 'analogo, stabilisce quale dei due ultimi pattern rimanenti viene assegnato al
 'secondo subset (in pratica lim1 e lim2 fissano il limite inferiore dello
 'spazio da riempire con le assegnazioni, rispettivamente, al primo e al
secondo
 'subset); l'algoritmo procede al riempimento regolare (cioè ad intervalli
 'regolari, a meno dell'approssimazione necessaria a far rientrare il tutto
 'nella dimensione discreta della numerazione ordinata dei pattern) del vettore
```

```
 'con le assegnazioni al primo subset; poi, nello stesso modo, assegna i posti
 'restati liberi al secondo subset; al terzo subset restano semplicemente gli
 'spazi non riempiti dagli altri due
Sub RFA_Partitioner(assign() As Ubyte,SetNum() As Uinteger,permutations() As
Ubyte,iperm As Ubyte,lim1 As Ubyte,lim2 As Ubyte)

    Dim As Uinteger num,i,cnt,j
    Dim As Double increm
    num=Ubound(assign)-Lbound(assign)+1

    Redim As Uinteger SN(1 To 3)
    For j=1 To 3
        SN(j)=SetNum(permutations(iperm,j))
    Next

    If num>3 Then

        Redim As Uinteger a(1 To num) 'durante l'elaborazione le assegnazioni
ai
                                      'subset sono copiate in questo vettore di
                                      'lavoro, i cui valori non possono essere
                                      'in formato Ubyte perché, con matrici già
                                      'di qualche decina di righe, in alcune
                                      'fasi del processamento si supera
                                      'facilmente il 255

        Redim As Double decimalposition1(1 To SN(1)),decimalposition2(1 To
SN(2))
        Redim As Uinteger pos1(1 To SN(1)),pos2(1 To SN(2))

        decimalposition1(1)=1
        decimalposition2(1)=1
        pos1(1)=1
        pos2(1)=1

        'inizio operazioni per assegnare i pattern al primo subset (indicato in
        'permutations(iperm,1))

        'calcolo dell'incremento necessario a riempire in modo regolare num-
lim1
        'posizioni con SN(1) oggetti (cioè l'etichetta del primo subset
        'assegnato)
        increm=(num-1-lim1)/(SN(1)-1)

        For i=2 To SN(1)
            decimalposition1(i)=decimalposition1(i-1)+increm
            pos1(i)=cint(decimalposition1(i))
        Next

        'assegnazione dei pattern al primo subset
        For i=1 To SN(1)
            a(pos1(i))=permutations(iperm,1)
        Next

        'inizio operazioni per assegnare i pattern al secondo subset (indicato
        'in permutations(iperm,2)
        For i=1 To num
            If a(i)=0 Then
                cnt+=10
                a(i)=cnt
            End If
        Next

        increm=(num-SN(1)-1-lim2)/(SN(2)-1)
```

```
        For i=2 To SN(2)
            decimalposition2(i)=decimalposition2(i-1)+increm
            pos2(i)=Cint(decimalposition2(i))
        Next

        'operazioni finali per assegnare i pattern al secondo subset (indicato
        'in permutations(iperm,2))
        j=2
        For i=1 To SN(2)
            cnt=0
            Do
                If a(j)/10=pos2(i) Then
                    a(j)=permutations(iperm,2)
                    cnt=1
                End If
                j+=1
            Loop Until cnt=1 or j>num
        Next

        'operazioni per assegnare i pattern al terzo subset (indicato in
        'permutations(iperm,3))
        For i=1 To num
            If a(i)>19 Then
                a(i)=permutations(iperm,3)
            End If
        Next

        'i valori in a(), adesso certamente tornati nell'ambito Ubyte, vengono
        'trasferiti al vettore assign() con cui la Sub restituisce il proprio
        'elaborato
        For i=1 To num
            assign(i)=a(i)
        Next

    Else 'caso in cui il numero dei pattern è inferiore o uguale a tre; questo
         'accade soprattutto quando si distribuiscono separatamente tra i
subset
         'i casi di presenza e assenza, per cui, nel caso di specie rare o
         'ubiquitarie può succedere che l'assegnazione, rispettivamente, delle
         'presenze e delle assenze si riduca ad un numero di casi molto
piccolo;
         'l'algoritmo esegue comunque l'assegnazione, ma è probabile che
         'l'addestramento non dia buoni risultati

        'se si ha un unico pattern, lo si assegna al subset di test per avere
        'la possibilità di rilevare un'eventuale fortunosa capacità della rete
        'di discernere tra presenza e assenza
        If num=1 Then
            assign(1)=3

            SetNum(1)=0
            SetNum(2)=0
            SetNum(3)=1

        'se se ne hanno due, uno va al subset di training e l'altro a quello di
        'test, nell'ordine in cui i numeri 1 e 3 compaiono nella riga iperm di
        'permutations
        Elseif num=2 Then
            i=1
            j=0
            Do
                j+=1
                If permutations(iperm,j)<>2 Then
```

```
                    If cnt=0 Then
                        assign(i)=permutations(iperm,j)
                        i+=1
                        If assign(i-1)=1 Then
                            cnt=1
                        End If
                    Else
                        If permutations(iperm,j)=3 Then
                            assign(i)=3
                            i+=1
                        End If
                    End If
                End If
            Loop Until i=3

            SetNum(1)=1
            SetNum(2)=0
            SetNum(3)=1

        'se se ne hanno tre, l'assegnazione ai tre subset va nell'ordine in cui
        'i numeri 1, 2 e 3 compaiono nella riga iperm di permutations
        Else
            For j=1 To 3
                assign(j)=permutations(iperm,j)
            Next

            SetNum(1)=1
            SetNum(2)=1
            SetNum(3)=1

        End If

    End If
End Sub


 'Il principio generale del riempimento di un vettore con le assegnazioni dei
 'vari casi ai subset 1, 2 e 3 è il medesimo della Sub RFA_Partitioner.
 'Qui però l'assegnazione procede su una base diversa. L'intenzione è quella di
 'assegnare un 50% dei casi al subset 1 e un 25% ai due rimanenti; se i casi
 'fossero solo quattro, il vettore delle assegnazioni dovrebbe resentare
 'due 1 un 2 e un 3. Ad esempio, si potrebbe avere (1,2,1,3), (2,3,1,1),
 '(1,3,2,1), ecc. Nel caso di un numero di osservazioni superiore a quattro, un
 'un vettore delle assegnazioni che riportasse una ripetizione concatenata di
 'uno o più di questi moduli, conseguirebbe sia l'obiettivo di assegnare i casi
 'nelle percentuali desiderate, che quello di distribuire in modo omogeneo i
 'diversi livelli del gradiente ordinante tra i subset. Poiché esistono 12
 'possibili 4-ple del tipo (a,a,b,c), l'algoritmo fornisce 12 assegnazioni
 'basate su una ripetizione concatenata di ognuno dei possibili moduli, più
 'altre 24 assegnazioni basate su una concatenazione random dei 12 moduli
 'possibili, ottenendo così un totale di 36 diverse partizioni dei dati. Tale
 'numero è arbitrario e legato esclusivamente all'esigenza di equiparare il
 'numero di partizioni ottenute con l'algoritmo RFA e con MCA, ma l'obiettivo
 'è comunque quello di determinare un numero adeguato di varianti di partizione
 'su un particolare gradiente
Sub MCA_Partitioner(assign() As Ubyte,SetNum() As Uinteger,permutations() As
Ubyte,iperm As Ubyte)

    Dim As Uinteger num,i,j,quot,ilimit,qxd,cnt
    Dim As Ubyte rest,row
    num=Ubound(assign)-Lbound(assign)+1
    For i=1 To num
        assign(i)=0
    Next
```

```
    If num>3 Then

        quot=Int(num/4)
        qxd=4*quot
        ilimit=qxd-4
        SetNum(1)=qxd/2
        SetNum(2)=qxd/4
        SetNum(3)=qxd/4

        If iperm<13 Then

            For i=0 To ilimit step 4
                For j=1 To 4
                    assign(i+j)=permutations(iperm,j)
                Next
            Next
            If quot<num/4 Then
                rest=num-qxd
                For j=1 To rest
                    assign(qxd+j)=permutations(iperm,j)
                    SetNum(assign(qxd+j))+=1
                Next
            End If

        Else

            'PROBLEMA CON "Randomize Timer"
            For i=0 To ilimit step 4
                row=Int(Rnd*12)+1
                For j=1 To 4
                    assign(i+j)=permutations(row,j)
                Next
            Next
            If quot<num/4 Then
                rest=num-qxd
                row=Int(Rnd*12)+1
                For j=1 To rest
                    assign(qxd+j)=permutations(row,j)
                    SetNum(assign(qxd+j))+=1
                Next
            End If

        End If

    Else 'caso in cui il numero dei pattern è inferiore o uguale a tre

        'anche qui, la riga di riferimento viene scelta secondo iperm o in modo
random a
        'seconda, rispettivamente, che iperm sia minore di 13 o no
        If iperm<13 Then
            row=iperm
        Else
            'PROBLEMA CON "Randomize Timer"
            row=Int(Rnd*12)+1
        End If

        'se si ha un unico pattern, lo si assegna al subset di test
        If num=1 Then
            assign(1)=3

            SetNum(1)=0
            SetNum(2)=0
            SetNum(3)=1
```

```
        'se se ne hanno due, uno va al subset di training e l'altro a quello di
        'test, nell'ordine in cui i numeri 1 e 3 compaiono nella riga row di
        'permutations (l'eventuale secondo 1 viene ignorato)
        Elseif num=2 Then
            i=1
            j=0
            Do
                j+=1
                If permutations(row,j)<>2 Then
                    If cnt=0 Then
                        assign(i)=permutations(row,j)
                        If assign(i)=1 Then
                            cnt=1
                        End If
                        i+=1
                    Else
                        If permutations(row,j)=3 Then
                            assign(i)=3
                            i+=1
                        End If
                    End If
                End If
            Loop Until i=3

            SetNum(1)=1
            SetNum(2)=0
            SetNum(3)=1

        'se se ne hanno tre, l'assegnazione ai tre subset va nell'ordine in cui
        'i numeri 1, 2 e 3 compaiono nella riga iperm di permutations
        '(l'eventuale secondo 1 viene ignorato)
        Else
            i=1
            j=0
            Do
                j+=1
                If cnt=0 Then
                    assign(i)=permutations(row,j)
                    If assign(i)=1 Then
                        cnt=1
                    End If
                    i+=1
                Else
                    If permutations(row,j)<>1 Then
                        assign(i)=permutations(row,j)
                        i+=1
                    End If
                End If
            Loop Until i=4

            SetNum(1)=1
            SetNum(2)=1
            SetNum(3)=1

        End If
    End If
End Sub
```

ALLEGATO B

```
print "delphish2.exe (by Quma, 2009)"
print
print "DECLARATIONS"

declare Sub SenseInsertionSort(array() as double,key as integer,sense as
double)
declare Sub ColumnRank(FromMatrix() as double,FColumn as integer,ToMatrix() as
double,TColumn as integer)
declare function Combination(n as uinteger,k as uinteger) as uinteger
declare function pNorm(z as double) as double
declare function QuantileStop(array() as double,c as integer,k as integer,t as
integer,istop as integer) as double
declare Function WMW(A() as double,cA as integer,B() as double,cB as integer)
as double
declare function ProWMW(astop as integer,bstop as integer,A() as double,cA as
integer,B() as double,cB as integer) as double

dim as string filein,fileout
dim as uinteger col,i,j,h,k,t,waste,cnt,rows0,rows1,datarows,specpart
dim as ubyte cnd
dim as double sum,sum1,sum2,meanfreq,temp1,temp2
dim as integer a,b,c,d,r,s,p,q,n 'variabili relative alla matrice di confusione

const cinc=0.01 'soglia minima, nonché valore costante di incremento del
                'criterio, da utilizzare nella valutazione delle prestazioni
                'al variare del criterio; deve avere 0.5 e 1 come multipli

dim as uinteger csteps 'numero di passi necessari per arrivare da zero ad uno
                       'aggiungendo cinc
csteps=1/cinc

'vettore con tutte le soglie da considerare
redim as double thresholds(1 to csteps)

const part=9504 'numero partizioni
rows0=part/4
rows1=72
const tnum=67 'dimensione massima dei test set
const cD=32 'numero specie
col=cD*tnum
datarows=264
specpart=288

'vettore con la frequenza delle specie nel dataset
redim as double freq(1 to cD)

'matrice con le soglie ottimali per specie e per modello
redim as double opth(1 to cD,1 to 8)


'per ogni partizione (una per riga) riporta nove informazioni (una per colonna)
'necessarie per caratterizzarla: 1) specie di riferimento (1-32); 2) equa
ripartizione
'dei casi di presenza (0-1); 3) ordinamento in base all'altitudine (0-1); 4)
variante
'dell'algoritmo di partizione utilizzata (1-72); 5) numero identificativo della
'partizione (1-9504); 6) frequenza assoluta della specie (2-202); 7) numero di
casi
'assegnati al training set (130-134); 8) numero di casi assegnati al validation
set
'(65-67); 9) numero di casi assegnati al test set (65-67)
```

```
redim as uinteger label(1 to part,1 to 9)

'matrice con i valori osservati di presenza (1) o assenza (0) delle varie
specie
'nei part test set utilizzati; vi sono tnum gruppi di cD colonne; il primo
gruppo
'riporta, relativamente al test set corrispondente ad ogni riga, i valori di
presenza
'o assenza delle cD specie nel primo caso assegnato al test set considerato...
'nel loro complesso, i tnum gruppi di cD colonne, riportano i valori di
presenza
'o assenza di ogni specie in relazione ad ogni caso assegnato al test set
relativo
'alla riga considerata; lungo le righe si hanno dunque le stesse informazioni
per
'ognuno dei test set utilizzati
redim as double OBS(1 to part,1 to col)

'matrice con due strati bidimensionali che contengono i valori di previsione
del
'modello popolamento (strato 0) e del modello specie (strato 1); la struttura
di
'ogni strato è identica a quella della matrice bidimensionale OBS(): in tal
modo,
'per ogni strato, ogni valore identificato dalle due prime coordinate,
corrisponde
'al valore in OBS() identificato dalle medesime coordinate, permettendo un
'confronto immediato tra valori osservati e previsti e tra valori previsti da
un
'modello o dall'altro
redim as double PRED(1 to part,1 to col,0 to 1)

'riporta la frequenza assoluta di ogni specie (una per colonna) in ognuno dei
part
'test set utilizzati (lungo le righe); a parte l'utilità in sé di questa
informazione,
'ad esempio per valutare quali specie risultano poco rappresentate nei vari
'test set, la sua funzione principale è evitare il calcolo della statistica k
nei
'casi in cui cp(i,j)=0, per cui la specie j non è mai presente nel test set i
redim as uinteger cp(1 to part,1 to cD) 'cp sta per Check Presence

'matrice delle performance; per ogni partizione (dim. 1), per ogni specie
'(dim. 2), per ogni modello (dim. 3), riporta (dim. 4) sei misure:
'1) statistica K con criterio 0.5 (perform(i,h,j,1)); 2) area sotto la curva di
'K al variare del criterio (perform(i,h,j,2)); 3) statistica K con criterio
'ottimale per tutta la popolazione di modelli omogenei, identificato
'dall'algoritmo OTF (perform(i,h,j,3)); 4) criterio ottimale così determinato
'(perform(i,h,j,4)); 5) statistica K con criterio ottimale per il singolo
'modello (perform(i,h,j,5)); 6) criterio ottimale per il singolo modello
'(perform(i,h,j,6));
redim as double perform(1 to part,1 to cD,0 to 1,1 to 6)

'matrice in cui sono salvati i valori di K ottenuti dalla matrice di confusione
'per ogni soglia (dim. 1), ogni partizione (dim. 2), ogni specie (dim. 3)
'e ogni modello (dim. 4); tutte le misure di performance che vengono calcolate
'e salvate nella matrice perform() attingono da questa matrice, che costituisce
'in pratica il contenitore dei risultati grezzi di ogni esperimento
redim as double everyK(0 to csteps,0 to part,1 to cD,0 to 1)

'nella matrice wmarK() i valori di everyK() vengono ricalcolati e riscritti
come
'medie mobili pesate (weighted moving average) calcolate sulle distribuzioni
'della statistica K al variare della soglia in ogni modello (dunque lungo la
```

```
'prima dimensione di everyK()); la media è mobile nel senso che di volta in
'volta pone il suo centro in corrispondenza di uno dei valori della
'distribuzione; una funzione permette poi di assegnare pesi via via decrescenti
'(il centro ha peso 1) ai diversi valori che, da un lato o dall'altro, si
'allontanano dal valore preso come centrale; la matrice wmar() fornisce i pesi
'che determinano la "sfumatura" della media, cioè il decrescere dell'importanza
'dei valori, man mano che si allontanano dal K centrale
redim as double wmar(0 to csteps+1,0 to csteps)
redim as double wmarK(0 to csteps,0 to part,1 to cD,0 to 1)


'generazione delle 32 matrici (e relativi file, benché con i soli valori max,
min,
'mediana, Q1 e Q2) con i valori di K, M, B e C (4), in ognuna delle condizioni
'studiate (00, 01, 10 e 11) (4) e per entrambi i modelli (2): 4x4x2=32.
redim as double K0100(1 to rows0,0 to cD)
redim as double K0101(1 to rows0,0 to cD)
redim as double K0110(1 to rows1,0 to cD)
redim as double K0111(1 to rows1,0 to cD)
redim as double K0000(1 to rows0,0 to cD)
redim as double K0001(1 to rows0,0 to cD)
redim as double K0010(1 to rows1,0 to cD)
redim as double K0011(1 to rows1,0 to cD)

redim as double auK100(1 to rows0,0 to cD)
redim as double auK101(1 to rows0,0 to cD)
redim as double auK110(1 to rows1,0 to cD)
redim as double auK111(1 to rows1,0 to cD)
redim as double auK000(1 to rows0,0 to cD)
redim as double auK001(1 to rows0,0 to cD)
redim as double auK010(1 to rows1,0 to cD)
redim as double auK011(1 to rows1,0 to cD)

redim as double K1100(1 to rows0,0 to cD)
redim as double K1101(1 to rows0,0 to cD)
redim as double K1110(1 to rows1,0 to cD)
redim as double K1111(1 to rows1,0 to cD)
redim as double K1000(1 to rows0,0 to cD)
redim as double K1001(1 to rows0,0 to cD)
redim as double K1010(1 to rows1,0 to cD)
redim as double K1011(1 to rows1,0 to cD)

redim as double thr100(1 to rows0,0 to cD)
redim as double thr101(1 to rows0,0 to cD)
redim as double thr110(1 to rows1,0 to cD)
redim as double thr111(1 to rows1,0 to cD)
redim as double thr000(1 to rows0,0 to cD)
redim as double thr001(1 to rows0,0 to cD)
redim as double thr010(1 to rows1,0 to cD)
redim as double thr011(1 to rows1,0 to cD)

'vengono generate anche altre quattro matrici che registrano i valori assoluti
'di presenza delle specie nei vari test-set, come da matrice cp()
redim as double CHP00(1 to rows0,0 to cD)
redim as double CHP01(1 to rows0,0 to cD)
redim as double CHP10(1 to rows1,0 to cD)
redim as double CHP11(1 to rows1,0 to cD)

redim as uinteger jcnt00(1 to cD),jcnt01(1 to cD),jcnt10(1 to cD),jcnt11(1 to
cD)
dim as uinteger cnt00,cnt01

print "DECLARATIONS COMPLETED"
print
```

```
'FASE DI LETTURA DEI FILE CON I RISULTATI E DI RIEMPIMENTO DELLE MATRICI
label(),
'OBS(), PRED() E cp()
print "READING OUTCOME FILES..."
filein="label.res"
print " 1/4 ";filein
open filein for input as #1
for i=1 to part
    for j=1 to 9
        input #1, label(i,j)
    next
next
close(1)
filein="OBS.res"
print " 2/4 ";filein
open filein for input as #1
for i=1 to part
    for j=1 to 9
        input #1, waste
    next
    for j=1 to label(i,9)*cD
        input #1, OBS(i,j)
        cp(i,j-cD*int((j-1)/cD))+=OBS(i,j)
    next
next
close(1)
filein="PRED_A.res"
print " 3/4 ";filein
open filein for input as #1
for i=1 to part
    for j=1 to 9
        input #1, waste
    next
    for j=1 to label(i,9)*cD
        input #1, PRED(i,j,0)
    next
next
close(1)
filein="PRED_S.res"
print " 4/4 ";filein
open filein for input as #1
for i=1 to part
    for j=1 to 9
        input #1, waste
    next
    for j=1 to label(i,9)*cD
        input #1, PRED(i,j,1)
    next
next
close(1)
print "READING OUTCOME FILES COMPLETED"
print

'riempimento del vettore con le frequenze delle specie
for h=1 to cD
    freq(h)=label(h*specpart,6)/datarows
next

print "MEASURING PERFORMANCE..."

'le righe estreme di wmarK() non verranno aggiornate durante la procedura in
'quanto sempre pari a 0
```

```
'determinazione della matrice dei pesi per il calcolo di wmar, che pur
'considerando tutti i valori (ordinati) di un insieme, permette in pratica di
'calcolare la media nell'intorno di un punto, sfumando i pesi dei valori
'adiacenti man mano che si allontanano dal valore indicato come centrale
dim as double wrif,radius
wrif=0.1
radius=0.1
for i=0 to csteps
    wmar(i,0)=exp((log(wrif)/(radius)^2)*(i/csteps)^2)
    wmar(csteps+1,0)+=wmar(i,0)
next
for j=1 to csteps
    for i=0 to csteps
        if i>=j then
            wmar(i,j)=wmar(i-1,j-1)
            wmar(csteps+1,j)+=wmar(i,j)
        else
            wmar(i,j)=wmar(j,i)
            wmar(csteps+1,j)+=wmar(i,j)
        end if
    next
next

'area under K-statistics curve as a function of threshold varying from cinc
'to 1-cinc
dim as double auK

for i=0 to csteps
    thresholds(i)=i*cinc
next

redim as string species(1 to cD)
species(1)="Sat"
species(2)="Lec"
species(3)="Pam"
species(4)="Sce"
species(5)="Esl"
species(6)="Rue"
species(7)="Ala"
species(8)="Cog"
species(9)="Tit"
species(10)="Cot"
species(11)="Php"
species(12)="Ana"
species(13)="Orp"
species(14)="Sam"
species(15)="Sal"
species(16)="Icm"
species(17)="Leg"
species(18)="Bap"
species(19)="Chg"
species(20)="Gaa"
species(21)="Cac"
species(22)="Gog"
species(23)="Les"
species(24)="Tht"
species(25)="Lap"
species(26)="Gah"
species(27)="Bam"
species(28)="Mis"
species(29)="Pef"
species(30)="Abb"
species(31)="Cyc"
```

```
species(32)="Saf"

'viene creato o aperto (append) il file che per una particolare
        'specie (h), un particolare modello (j), una particolare
        'modalità di partizione legata alla presenza/assenza della
        'specie h (label(i,2)) e una modalità analoga legata all'
        'altitudine (label(i,3)), salva i valori di K al variare della
        'soglia per ognuna delle 72 partizioni (label(i,4)) su cui viene
        'studiata la particolare combinazione dei casi h, j, ecc.
redim as string thrKfilesname(1 to cD)
for h=1 to cD
    if h<10 then
        thrKfilesname(h)="thrK_0"+str(h)+"_"+species(h)+".csv"
    else
        thrKfilesname(h)="thrK_"+str(h)+"_"+species(h)+".csv"
    end if
    open thrKfilesname(h) for output as #100
    close(100)
next


'calcolo di ctK, otK e auK
for i=1 to part 'per ogni test-set

    print "   Processing partition ";i;"/";str(part)
    for h=1 to cD 'per ogni specie

        'se i casi di presenza non sono distribuiti omogenamente tra i subset
        '(label(i,2)=0), ma la specie in questione ha comunque almeno un caso
di
        'presenza nel test-set (cp(i,h)>0)
        'oppure
        'se i casi di presenza della specie in questione (h=label(i,1)) sono
        'distribuiti omogeneamente tra i subset (label(i,2)=1)
        'allora si procede al calcolo della performance;
        'se né la prima, né la seconda condizione sono soddisfatte, calcolare
        'la performance non ha senso perché, rispettivamente, non è possibile
        'testare la capacità del modello di predire la presenza oppure i casi
di
        'presenza/assenza distribuiti omogeneamente tra i subset sono quelli
        'relativi ad un altra specie
        if (label(i,2)=0 and cp(i,h)>0) or (label(i,2)=1 and h=label(i,1)) then

            for j=0 to 1 'per ogni modello (0=popolamento; 1=singola specie)

                open thrKfilesname(h) for append as #100
                print #100, "threshold,";species(h);"-";str(j);"-
";str(label(i,2));"-";str(label(i,3));"-";str(label(i,4))

                'ciclo di riempimento della matrice wmarK() relativa alla
partizione i,
                'che porterà al calcolo di ctK, otK e kmean per quella
partizione
                for t=1 to csteps-1

                    a=0
                    b=0
                    c=0
                    d=0

                    'calcolo di K per i due modelli 0 e 1 sulla partizione i,
                    'al variare del criterio (lo scopo è calcolare auK e
trovare
                    'il criterio che dà il otK)
```

```
                    for k=h to h+cD*(label(i,9)-1) step cD

                        if OBS(i,k)=0 then
                            if PRED(i,k,j)<thresholds(t) then
                                d+=1
                            else
                                b+=1
                            end if
                        else
                            if PRED(i,k,j)>=thresholds(t) then
                                a+=1
                            else
                                c+=1
                            end if
                        end if

                    next

                    r=a+c
                    s=b+d
                    p=a+b
                    q=c+d
                    n=r+s

                    everyK(t,i,h,j)=(n*(a+d)-(p*r+q*s))/(n^2-(p*r+q*s)) 'K

                    auK+=everyK(t,i,h,j)+everyK(t-1,i,h,j) 'passaggio
intermedio per il calcolo
                                        'di auK

                    print #100, str(thresholds(t));",";str(everyK(t,i,h,j))

                next

                print #100,
                print #100,
                print #100,
                close(100)

                'si salva in perform() la statistica K ottenuta con
criterio=0.5
                perform(i,h,j,1)=everyK(csteps/2,i,h,j)

                'calcolo e salvataggio in perform() di auK
                perform(i,h,j,2)=auK/(2*(csteps-1))
                auK=0 'auK torna a 0 per rientrare pulito nel ciclo seguente

                'calcolo delle medie sfumate relative alla partizione i
(necessario
                'per trovare poi otK)
                'la media sfumata centrata su ognuno dei valori di K in
everyK(t,i),
                'salvata in wmarK(k,i), è ottenuta moltiplicando tutti i valori
                'della colonna i di everyK() con tutti i valori della colonna k
                'di wmar(), sommandoli e dividendo il risultato per la somma
                'dei pesi contenuti nella colonna i di wmar()
                for k=0 to csteps
                    for t=0 to csteps
                        wmarK(k,i,h,j)+=wmar(t,k)*everyK(t,i,h,j)
                    next
                next
                for k=0 to csteps
                    wmarK(k,i,h,j)/=wmar(csteps+1,k)
                next
```

```
                'calcolo di otK
                temp1=0
                temp2=0
                for k=1 to csteps-1
                    if wmarK(k,i,h,j)>temp1 then
                        temp1=wmarK(k,i,h,j)
                        temp2=k
                    end if
                next
                perform(i,h,j,3)=everyK(temp2,i,h,j)
                perform(i,h,j,4)=thresholds(temp2)

            next

        'else i valori di performance non hanno significato e il caso (i,h)
        'viene ignorato

        end if

    next
next

erase OBS,PRED

print "   MAIN CYCLE COMPLETED"
print

erase wmarK
erase everyK

print "MEASURING PERFORMANCE COMPLETED"
print
print "FILLING PERFORMANCE ARRAYS"

for t=1 to part
    if label(t,2)=0 and label(t,3)=0 then
        cnt00+=1
        for j=1 to cD
            if cp(t,j)>0 then 'solo se la specie è presente nel test-set i
valori
                              'di performance vengono salvati, sennò jcnt00(j)
non
                              'cresce: il suo valore finale dice in quanti casi
                              'la specie j era presente nel test-set e si è
dunque
                              'potuto valutare la performance predittiva dei
modelli
                jcnt00(j)+=1
                K0100(cnt00,j)=perform(t,j,0,1)
                K0000(cnt00,j)=perform(t,j,1,1)
                auK100(cnt00,j)=perform(t,j,0,2)
                auK000(cnt00,j)=perform(t,j,1,2)
                K1100(cnt00,j)=perform(t,j,0,3)
                K1000(cnt00,j)=perform(t,j,1,3)
                thr100(cnt00,j)=perform(t,j,0,4)
                thr000(cnt00,j)=perform(t,j,1,4)
                CHP00(cnt00,j)=cp(t,j)
            end if
        next
    elseif label(t,2)=0 and label(t,3)=1 then
        cnt01+=1
        for j=1 to cD
            if cp(t,j)>0 then
```

```
                jcnt01(j)+=1
                K0101(cnt01,j)=perform(t,j,0,1)
                K0001(cnt01,j)=perform(t,j,1,1)
                auK101(cnt01,j)=perform(t,j,0,2)
                auK001(cnt01,j)=perform(t,j,1,2)
                K1101(cnt01,j)=perform(t,j,0,3)
                K1001(cnt01,j)=perform(t,j,1,3)
                thr101(cnt01,j)=perform(t,j,0,4)
                thr001(cnt01,j)=perform(t,j,1,4)
                CHP01(cnt01,j)=cp(t,j)
            end if
        next
    elseif label(t,2)=1 and label(t,3)=0 and label(t,1)<=cD then
        jcnt10(label(t,1))+=1
        K0110(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),0,1)
        K0010(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),1,1)
        auK110(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),0,2)
        auK010(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),1,2)
        K1110(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),0,3)
        K1010(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),1,3)
        thr110(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),0,4)
        thr010(jcnt10(label(t,1)),label(t,1))=perform(t,label(t,1),1,4)
        CHP10(jcnt10(label(t,1)),label(t,1))=cp(t,label(t,1))
    elseif label(t,2)=1 and label(t,3)=1 and label(t,1)<=cD then
        jcnt11(label(t,1))+=1
        K0111(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),0,1)
        K0011(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),1,1)
        auK111(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),0,2)
        auK011(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),1,2)
        K1111(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),0,3)
        K1011(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),1,3)
        thr111(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),0,4)
        thr011(jcnt11(label(t,1)),label(t,1))=perform(t,label(t,1),1,4)
        CHP11(jcnt11(label(t,1)),label(t,1))=cp(t,label(t,1))
    end if
next

erase perform

print "FILLING COMPLETED"
print
print "WRITING RAW PERFORMANCE FILES"

'scrittura dei file con le misure di performance
redim as integer card(1 to cD,1 to 8)
redim as double sortedK0(1 to rows0,1 to 8,1 to cD),sortedK1(1 to rows0,1 to
8,1 to cD),sortedauK(1 to rows0,1 to 8,1 to cD)

for i=1 to cD

    if i<10 then
        open "perfdist_0"+str(i)+"_"+species(i)+".csv" for output as #9
        print "   perfdist_0"+str(i)+"_"+species(i)+".csv"
    else
        open "perfdist_"+str(i)+"_"+species(i)+".csv" for output as #9
        print "   perfdist_"+str(i)+"_"+species(i)+".csv"

    end if

    print #9,
"K0000,K0001,K0010,K0011,K0100,K0101,K0110,K0111,K1000,K1001,K1010,K1011,K1100,
K1101,K1110,K1111,auK000,auK001,auK010,auK011,auK100,auK101,auK110,auK111,"

    redim as double temp(1 to rows0,1 to 1)
```

```
for j=1 to rows0
    temp(j,1)=K0000(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt00(i)
    sortedK0(j,1,i)=temp(j,1)
next
card(i,1)=jcnt00(i)
for j=1 to rows0
    temp(j,1)=K0001(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt01(i)
    sortedK0(j,2,i)=temp(j,1)
next
card(i,2)=jcnt01(i)

redim as double temp(1 to rows1,1 to 1)
for j=1 to rows1
    temp(j,1)=K0010(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK0(j,3,i)=temp(j,1)
next
card(i,3)=rows1
for j=1 to rows1
    temp(j,1)=K0011(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK0(j,4,i)=temp(j,1)
next
card(i,4)=rows1

redim as double temp(1 to rows0,1 to 1)
for j=1 to rows0
    temp(j,1)=K0100(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt00(i)
    sortedK0(j,5,i)=temp(j,1)
next
card(i,5)=jcnt00(i)
for j=1 to rows0
    temp(j,1)=K0101(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt01(i)
    sortedK0(j,6,i)=temp(j,1)
next
card(i,6)=jcnt01(i)

redim as double temp(1 to rows1,1 to 1)
for j=1 to rows1
    temp(j,1)=K0110(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK0(j,7,i)=temp(j,1)
next
card(i,7)=rows1
for j=1 to rows1
    temp(j,1)=K0111(j,i)
```

```
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK0(j,8,i)=temp(j,1)
next
card(i,8)=rows1


redim as double temp(1 to rows0,1 to 1)
for j=1 to rows0
    temp(j,1)=K1000(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt00(i)
    sortedK1(j,1,i)=temp(j,1)
next
for j=1 to rows0
    temp(j,1)=K1001(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt01(i)
    sortedK1(j,2,i)=temp(j,1)
next

redim as double temp(1 to rows1,1 to 1)
for j=1 to rows1
    temp(j,1)=K1010(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK1(j,3,i)=temp(j,1)
next
for j=1 to rows1
    temp(j,1)=K1011(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK1(j,4,i)=temp(j,1)
next

redim as double temp(1 to rows0,1 to 1)
for j=1 to rows0
    temp(j,1)=K1100(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt00(i)
    sortedK1(j,5,i)=temp(j,1)
next
for j=1 to rows0
    temp(j,1)=K1101(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt01(i)
    sortedK1(j,6,i)=temp(j,1)
next

redim as double temp(1 to rows1,1 to 1)
for j=1 to rows1
    temp(j,1)=K1110(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK1(j,7,i)=temp(j,1)
next
```

```
for j=1 to rows1
    temp(j,1)=K1111(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedK1(j,8,i)=temp(j,1)
next


redim as double temp(1 to rows0,1 to 1)
for j=1 to rows0
    temp(j,1)=auK000(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt00(i)
    sortedauK(j,1,i)=temp(j,1)
next
for j=1 to rows0
    temp(j,1)=auK001(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt01(i)
    sortedauK(j,2,i)=temp(j,1)
next

redim as double temp(1 to rows1,1 to 1)
for j=1 to rows1
    temp(j,1)=auK010(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedauK(j,3,i)=temp(j,1)
next
card(i,3)=rows1
for j=1 to rows1
    temp(j,1)=auK011(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
    sortedauK(j,4,i)=temp(j,1)
next

redim as double temp(1 to rows0,1 to 1)
for j=1 to rows0
    temp(j,1)=auK100(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt00(i)
    sortedauK(j,5,i)=temp(j,1)
next
for j=1 to rows0
    temp(j,1)=auK101(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to jcnt01(i)
    sortedauK(j,6,i)=temp(j,1)
next

redim as double temp(1 to rows1,1 to 1)
for j=1 to rows1
    temp(j,1)=auK110(j,i)
next
SenseInsertionSort(temp(),1,-1)
for j=1 to rows1
```

```
        sortedauK(j,7,i)=temp(j,1)
    next
    for j=1 to rows1
        temp(j,1)=auK111(j,i)
    next
    SenseInsertionSort(temp(),1,-1)
    for j=1 to rows1
        sortedauK(j,8,i)=temp(j,1)
    next


    'stampa su file dei risultati grezzi ordinati
    for j=1 to rows0
        for k=1 to 8
            if card(i,k)>=j then
                print #9, sortedK0(j,k,i);",";
            else
                print #9, " ,";
            end if
        next
        for k=1 to 8
            if card(i,k)>=j then
                print #9, sortedK1(j,k,i);",";
            else
                print #9, " ,";
            end if
        next
        for k=1 to 7
            if card(i,k)>=j then
                print #9, sortedauK(j,k,i);",";
            else
                print #9, " ,";
            end if
        next
        if card(i,8)>=j then
            print #9, sortedauK(j,8,i)
        else
            print #9, " "
        end if
    next

    close(9)

next

print
print "start calculating..."
print

'matrici con gli indici sintetici delle distribuzioni di performance ottenute
'nei vari casi; la prima dimensione è la specie, la terza è il tipo di modello
'(8 e non 16 perché i modelli ottimizzati per la soglia hanno la matrice
'K1perf(), diversa da quelli non ottimizzati K0perf()); la seconda dimensione
'è il tipo di indice: 0) minimo, 1) primo quartile, 2) mediana, 3) terzo
'quartile, 4) massimo, 5) media, 6) frequenza dei positivi, 7) mediana dei
'non zeri, 8) 6*7=mediana ridotta (reduced median), 9) mediana sfumata (bmed);
'la decima dimensione è usata temporaneamente per salvare il numero assoluto
'di non zeri in una distribuzione
redim as double K0perf(1 to cD,0 to 10,1 to 8),auKperf(1 to cD,0 to 10,1 to
8),K1perf(1 to cD,0 to 10,1 to 8)

'calcolo della media e della frequenza dei positivi
for i=1 to cD
    for h=1 to 8
```

```
        for j=1 to card(i,h)
            K0perf(i,5,h)+=sortedK0(j,h,i)
            K1perf(i,5,h)+=sortedK1(j,h,i)
            auKperf(i,5,h)+=sortedauK(j,h,i)
            if sortedK0(j,h,i)>0 then
                K0perf(i,10,h)+=1
            end if
            if sortedK1(j,h,i)>0 then
                K1perf(i,10,h)+=1
            end if
            if sortedauK(j,h,i)>0 then
                auKperf(i,10,h)+=1
            end if
        next
        K0perf(i,5,h)/=card(i,h)
        K1perf(i,5,h)/=card(i,h)
        auKperf(i,5,h)/=card(i,h)
        K0perf(i,6,h)=K0perf(i,10,h)/card(i,h)
        K1perf(i,6,h)=K1perf(i,10,h)/card(i,h)
        auKperf(i,6,h)=auKperf(i,10,h)/card(i,h)
    next
next

'calcolo della mediana dei positivi e della mediana ridotta
for i=1 to cD
    for h=1 to 8

        if K0perf(i,10,h)=0 then
            K0perf(i,7,h)=0
            K0perf(i,8,h)=0
        else
            redim as double temp(1 to K0perf(i,10,h),1 to 1)
            for j=1 to K0perf(i,10,h)
                temp(j,1)=sortedK0(j,h,i)
            next
            K0perf(i,7,h)=QuantileStop(temp(),1,2,4,K0perf(i,10,h))
            K0perf(i,8,h)=K0perf(i,6,h)*K0perf(i,7,h)
        end if

        if K1perf(i,10,h)=0 then
            K1perf(i,7,h)=0
            K1perf(i,8,h)=0
        else
            redim as double temp(1 to K1perf(i,10,h),1 to 1)
            for j=1 to K1perf(i,10,h)
                temp(j,1)=sortedK1(j,h,i)
            next
            K1perf(i,7,h)=QuantileStop(temp(),1,2,4,K1perf(i,10,h))
            K1perf(i,8,h)=K1perf(i,6,h)*K1perf(i,7,h)
        end if

        if auKperf(i,10,h)=0 then
            auKperf(i,7,h)=0
            auKperf(i,8,h)=0
        else
            redim as double temp(1 to auKperf(i,10,h),1 to 1)
            for j=1 to auKperf(i,10,h)
                temp(j,1)=sortedauK(j,h,i)
            next
            auKperf(i,7,h)=QuantileStop(temp(),1,2,4,auKperf(i,10,h))
            auKperf(i,8,h)=auKperf(i,6,h)*auKperf(i,7,h)
        end if

    next
```

```
next

wrif=0.0001
dim as double weight,weightsum
'calcolo muM
for i=1 to cD
    for h=1 to 8
        weightsum=0
        if card(i,h)/2=int(card(i,h)/2) then
            for j=1 to card(i,h)
                if j<=card(i,h)/2 then
                    weight=exp(log(wrif)*((2*j-card(i,h))/(2-card(i,h)))^2)
                else
                    weight=exp(log(wrif)*((2*j-card(i,h)-2)/(2-card(i,h)))^2)
                end if
                weightsum+=weight
                K0perf(i,9,h)+=sortedK0(j,h,i)*weight
                K1perf(i,9,h)+=sortedK1(j,h,i)*weight
                auKperf(i,9,h)+=sortedauK(j,h,i)*weight
            next
            K0perf(i,9,h)/=weightsum
            K1perf(i,9,h)/=weightsum
            auKperf(i,9,h)/=weightsum
        else
            for j=1 to card(i,h)
                weight=exp(log(wrif)*((2*j-card(i,h)-1)/(1-card(i,h)))^2)
                weightsum+=weight
                K0perf(i,9,h)+=sortedK0(j,h,i)*weight
                K1perf(i,9,h)+=sortedK1(j,h,i)*weight
                auKperf(i,9,h)+=sortedauK(j,h,i)*weight
            next
            K0perf(i,9,h)/=weightsum
            K1perf(i,9,h)/=weightsum
            auKperf(i,9,h)/=weightsum
        end if
    next
next


'calcolo mediana delle distribuzioni di soglie ottimizzanti ottenute per ogni
'specie e per ogni modello
for j=1 to cD
    opth(j,1)=QuantileStop(thr000(),j,2,4,card(j,1))
    opth(j,2)=QuantileStop(thr001(),j,2,4,card(j,2))
    opth(j,3)=QuantileStop(thr010(),j,2,4,card(j,3))
    opth(j,4)=QuantileStop(thr011(),j,2,4,card(j,4))
    opth(j,5)=QuantileStop(thr100(),j,2,4,card(j,5))
    opth(j,6)=QuantileStop(thr101(),j,2,4,card(j,6))
    opth(j,7)=QuantileStop(thr110(),j,2,4,card(j,7))
    opth(j,8)=QuantileStop(thr111(),j,2,4,card(j,8))
next


redim as string comb(1 to 16)
comb(1)="'0000"
comb(2)="'0001"
comb(3)="'0010"
comb(4)="'0011"
comb(5)="'0100"
comb(6)="'0101"
comb(7)="'0110"
comb(8)="'0111"
comb(9)="'1000"
comb(10)="'1001"
```

```
comb(11)="'1010"
comb(12)="'1011"
comb(13)="'1100"
comb(14)="'1101"
comb(15)="'1110"
comb(16)="'1111"

print
print "WRITING 5-QUANTILES FILES"

open "K0000.csv" for output as #3
print "   K0000.csv"
print #3, "K0000,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K0perf(j,t,1)=QuantileStop(K0100(),j,t,4,jcnt00(j))
        print #3, K0perf(j,t,1);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,1);",";
    next
    print #3, K0perf(j,9,1)
next
close(3)

open "K0001.csv" for output as #3
print "   K0001.csv"
print #3, "K0001,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K0perf(j,t,2)=QuantileStop(K0000(),j,t,4,jcnt00(j))
        print #3, K0perf(j,t,2);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,2);",";
    next
    print #3, K0perf(j,9,2)
next
close(3)

open "K0010.csv" for output as #3
print "   K0010.csv"
print #3, "K0010,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K0perf(j,t,3)=QuantileStop(K0101(),j,t,4,jcnt01(j))
        print #3, K0perf(j,t,3);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,3);",";
    next
    print #3, K0perf(j,9,3)
next
close(3)

open "K0011.csv" for output as #3
print "   K0011.csv"
print #3, "K0011,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
```

```
        K0perf(j,t,4)=QuantileStop(K0001(),j,t,4,jcnt01(j))
        print #3, K0perf(j,t,4);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,4);",";
    next
    print #3, K0perf(j,9,4)
next
close(3)

open "K0100.csv" for output as #3
print "   K0100.csv"
print #3, "K0100,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K0perf(j,t,5)=QuantileStop(K0110(),j,t,4,jcnt10(j))
        print #3, K0perf(j,t,5);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,5);",";
    next
    print #3, K0perf(j,9,5)
next
close(3)

open "K0101.csv" for output as #3
print "   K0101.csv"
print #3, "K0101,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K0perf(j,t,6)=QuantileStop(K0010(),j,t,4,jcnt10(j))
        print #3, K0perf(j,t,6);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,6);",";
    next
    print #3, K0perf(j,9,6)
next
close(3)

open "K0110.csv" for output as #3
print "   K0110.csv"
print #3, "K0110,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K0perf(j,t,7)=QuantileStop(K0111(),j,t,4,jcnt11(j))
        print #3, K0perf(j,t,7);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,7);",";
    next
    print #3, K0perf(j,9,7)
next
close(3)

open "K0111.csv" for output as #3
print "   K0111.csv"
print #3, "K0111,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
```

```
        K0perf(j,t,8)=QuantileStop(K0011(),j,t,4,jcnt11(j))
        print #3, K0perf(j,t,8);",";
    next
    for t=5 to 8
        print #3, K0perf(j,t,8);",";
    next
    print #3, K0perf(j,9,8)
next
close(3)



open "auK000.csv" for output as #3
print "   auK000.csv"
print #3, "auK000,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,1)=QuantileStop(auK100(),j,t,4,jcnt00(j))
        print #3, auKperf(j,t,1);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,1);",";
    next
    print #3, auKperf(j,9,1)
next
close(3)

open "auK001.csv" for output as #3
print "   auK001.csv"
print #3, "auK001,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,2)=QuantileStop(auK000(),j,t,4,jcnt00(j))
        print #3, auKperf(j,t,2);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,2);",";
    next
    print #3, auKperf(j,9,2)
next
close(3)

open "auK010.csv" for output as #3
print "   auK010.csv"
print #3, "auK010,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,3)=QuantileStop(auK101(),j,t,4,jcnt01(j))
        print #3, auKperf(j,t,3);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,3);",";
    next
    print #3, auKperf(j,9,3)
next
close(3)

open "auK011.csv" for output as #3
print "   auK011.csv"
print #3, "auK011,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
```

```
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,4)=QuantileStop(auK001(),j,t,4,jcnt01(j))
        print #3, auKperf(j,t,4);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,4);",";
    next
    print #3, auKperf(j,9,4)
next
close(3)

open "auK100.csv" for output as #3
print "   auK100.csv"
print #3, "auK100,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,5)=QuantileStop(auK110(),j,t,4,jcnt10(j))
        print #3, auKperf(j,t,5);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,5);",";
    next
    print #3, auKperf(j,9,5)
next
close(3)

open "auK101.csv" for output as #3
print "   auK101.csv"
print #3, "auK101,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,6)=QuantileStop(auK010(),j,t,4,jcnt10(j))
        print #3, auKperf(j,t,6);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,6);",";
    next
    print #3, auKperf(j,9,6)
next
close(3)

open "auK110.csv" for output as #3
print "   auK110.csv"
print #3, "auK110,min,q1,Me,q3,max,aver,frn0,medn0,redmed,bmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,7)=QuantileStop(auK111(),j,t,4,jcnt11(j))
        print #3, auKperf(j,t,7);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,7);",";
    next
    print #3, auKperf(j,9,7)
next
close(3)

open "auK111.csv" for output as #3
print "   auK111.csv"
print #3, "auK111,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
```

```
    print #3, str(j);",";
    for t=0 to 4
        auKperf(j,t,8)=QuantileStop(auK011(),j,t,4,jcnt11(j))
        print #3, auKperf(j,t,8);",";
    next
    for t=5 to 8
        print #3, auKperf(j,t,8);",";
    next
    print #3, auKperf(j,9,8)
next
close(3)



open "K1000.csv" for output as #3
print "   K1000.csv"
print #3, "K1000,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,1)=QuantileStop(K1100(),j,t,4,jcnt00(j))
        print #3, K1perf(j,t,1);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,1);",";
    next
    print #3, K1perf(j,9,1)
next
close(3)

open "K1001.csv" for output as #3
print "   K1001.csv"
print #3, "K1001,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,2)=QuantileStop(K1000(),j,t,4,jcnt00(j))
        print #3, K1perf(j,t,2);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,2);",";
    next
    print #3, K1perf(j,9,2)
next
close(3)

open "K1010.csv" for output as #3
print "   K1010.csv"
print #3, "K1010,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,3)=QuantileStop(K1101(),j,t,4,jcnt01(j))
        print #3, K1perf(j,t,3);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,3);",";
    next
    print #3, K1perf(j,9,3)
next
close(3)

open "K1011.csv" for output as #3
print "   K1011.csv"
```

```
print #3, "K1011,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,4)=QuantileStop(K1001(),j,t,4,jcnt01(j))
        print #3, K1perf(j,t,4);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,4);",";
    next
    print #3, K1perf(j,9,4)
next
close(3)

open "K1100.csv" for output as #3
print "   K1100.csv"
print #3, "K1100,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,5)=QuantileStop(K1110(),j,t,4,jcnt10(j))
        print #3, K1perf(j,t,5);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,5);",";
    next
    print #3, K1perf(j,9,5)
next
close(3)

open "K1101.csv" for output as #3
print "   K1101.csv"
print #3, "K1101,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,6)=QuantileStop(K1010(),j,t,4,jcnt10(j))
        print #3, K1perf(j,t,6);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,6);",";
    next
    print #3, K1perf(j,9,6)
next
close(3)

open "K1110.csv" for output as #3
print "   K1110.csv"
print #3, "K1110,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,7)=QuantileStop(K1111(),j,t,4,jcnt11(j))
        print #3, K1perf(j,t,7);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,7);",";
    next
    print #3, K1perf(j,9,7)
next
close(3)

open "K1111.csv" for output as #3
print "   K1111.csv"
```

```
print #3, "K1111,min,q1,Me,q3,max,aver,frn0,medn0,redmed,Mmed"
for j=1 to cD
    print #3, str(j);",";
    for t=0 to 4
        K1perf(j,t,8)=QuantileStop(K1011(),j,t,4,jcnt11(j))
        print #3, K1perf(j,t,8);",";
    next
    for t=5 to 8
        print #3, K1perf(j,t,8);",";
    next
    print #3, K1perf(j,9,8)
next
close(3)


print
print "WRITING PRESENCE/FREQUENCE FILES"
print "   CHP00.csv"
open "CHP00.csv" for output as #4
print #4, "CHP00,datafreq,testfreq,meanfreq"
for j=1 to cD
    meanfreq=0
    for i=1 to rows0
        meanfreq+=CHP00(i,j)
    next
    print #4, str(j);",";freq(j);",";jcnt00(j)/rows0;",";meanfreq/jcnt00(j)
next
close(4)
print "   CHP01.csv"
open "CHP01.csv" for output as #4
print #4, "CHP01,datafreq,testfreq,meanfreq"
for j=1 to cD
    meanfreq=0
    for i=1 to rows0
        meanfreq+=CHP01(i,j)
    next
    print #4, str(j);",";freq(j);",";jcnt01(j)/rows0;",";meanfreq/jcnt01(j)
next
close(4)
print "   CHP10.csv"
open "CHP10.csv" for output as #4
print #4, "CHP10,datafreq,testfreq,meanfreq"
for j=1 to cD
    meanfreq=0
    for i=1 to rows1
        meanfreq+=CHP10(i,j)
    next
    print #4, str(j);",";freq(j);",";jcnt10(j)/rows1;",";meanfreq/rows1
next
close(4)
print "   CHP11.csv"
open "CHP11.csv" for output as #4
print #4, "CHP11,datafreq,testfreq,meanfreq"
for j=1 to cD
    meanfreq=0
    for i=1 to rows1
        meanfreq+=CHP11(i,j)
    next
    print #4, str(j);",";freq(j);",";jcnt11(j)/rows1;",";meanfreq/rows1
next
close(4)

print
print "WRITING PCA/ORDERED FILES"
```

```
redim as double order16(1 to 16,0 to 2),order8(1 to 8,0 to 2)
redim as string printK(1 to 16,1 to 6),printauK(1 to 8,1 to 6),juan(0 to 4)
redim as double Kord(1 to cD,0 to 9),auKord(1 to cD,0 to 9)

juan(0)="min"
juan(1)="q1"
juan(2)="med"
juan(3)="q3"
juan(4)="max"
juan(5)="aver"
juan(6)="frn0"
juan(7)="medn0"
juan(8)="redmed"
juan(9)="Mmed"

for t=0 to 9

    print "   ";str(juan(t));".csv"
    open juan(t)+".csv" for output as 6#

        'stampa matrice [specie(righe) X modelli(colonne)]
    print #6, "K-";juan(t);
    for j=1 to 16
        print #6, ",";comb(j);
    next
    print #6, ", ,";
    print #6, "auk-";juan(t);
    for j=1 to 7
        print #6, ",'";mid(comb(j),3);
    next
    print #6, ",'";mid(comb(8),3)
    for i=1 to cD
        print #6, str(i);",";
        for j=1 to 8
            print #6, K0perf(i,t,j);",";
        next
        for j=1 to 8
            print #6, K1perf(i,t,j);",";
        next
        print #6, " ,";
        print #6, str(i);",";
        for j=1 to 7
            print #6, auKperf(i,t,j);",";
        next
        print #6, auKperf(i,t,8)
    next
    print #6,

    'stampa tabelle ordinamento specie
    print #6, "K0-";juan(t);",values, ,K0-";juan(t);",rank, , , , , ,auK-
";juan(t);",values, ,auK-";juan(t);",rank"

    for i=1 to cD
        Kord(i,1)=i
        auKord(i,1)=i
        Kord(i,2)=0
        auKord(i,2)=0
        for j=1 to 8
            Kord(i,2)+=K0perf(i,t,j)
            Kord(i,2)+=K1perf(i,t,j)
            auKord(i,2)+=auKperf(i,t,j)
        next
        Kord(i,2)/=16
        auKord(i,2)/=8
```

```
    next
    SenseInsertionSort(Kord(),2,-1)
    SenseInsertionSort(auKord(),2,-1)
    for i=1 to cD
        Kord(i,0)=i
        auKord(i,0)=i
    next
    SenseInsertionSort(Kord(),1,1)
    SenseInsertionSort(auKord(),1,1)
    for i=1 to cD
        Kord(Kord(i,0),3)=Kord(i,1)
        Kord(Kord(i,0),4)=Kord(i,2)
        auKord(auKord(i,0),3)=auKord(i,1)
        auKord(auKord(i,0),4)=auKord(i,2)
    next

    for i=1 to cD
        print #6, Kord(i,1);",";Kord(i,2);", ,";Kord(i,3);",";Kord(i,4);", , ,
, , ,";auKord(i,1);",";auKord(i,2);", ,";auKord(i,3);",";auKord(i,4)
    next
    print #6,

    'stampa tabelle ordinamento modelli
    for i=1 to cD

        print #6, juan(t);",";str(i);"-K,norm, ,";juan(t);",";str(i);"-
K(ord),norm, , , ,";juan(t);",";str(i);"-auK,norm, ,";juan(t);",";str(i);"-
auK(ord),norm"

        for j=1 to 8
            order16(j,0)=j
            order16(j+8,0)=j+8
            order16(j,1)=K0perf(i,t,j)
            order16(j+8,1)=K1perf(i,t,j)
        next
        SenseInsertionSort(order16(),1,-1)
        for j=1 to 16
            order16(j,2)=(order16(j,1)-order16(16,1))/(order16(1,1)-
order16(16,1))
        next
        for j=1 to 16
            printK(order16(j,0),1)=str(order16(j,1))
            printK(order16(j,0),2)=comb(order16(j,0))
            printK(order16(j,0),3)=str(order16(j,2))
            printK(j,4)=str(order16(j,1))
            printK(j,5)=comb(order16(j,0))
            printK(j,6)=str(order16(j,2))
        next

        for j=1 to 8
            order8(j,0)=j
            order8(j,1)=auKperf(i,t,j)
        next
        SenseInsertionSort(order8(),1,-1)
        for j=1 to 8
            order8(j,2)=(order8(j,1)-order8(8,1))/(order8(1,1)-order8(8,1))
        next
        for j=1 to 8
            printauK(order8(j,0),1)=str(order8(j,1))
            printauK(order8(j,0),2)=comb(order8(j,0))
            printauK(order8(j,0),3)=str(order8(j,2))
            printauK(j,4)=str(order8(j,1))
            printauK(j,5)=comb(order8(j,0))
            printauK(j,6)=str(order8(j,2))
```

```
        next

        for j=1 to 16
            if j<9 then
                print #6, printK(j,1);",";printK(j,2);",";printK(j,3);",
,";printK(j,4);",";printK(j,5);",";printK(j,6);", , , ,";
                print #6, printauK(j,1);",";printauK(j,2);",";printauK(j,3);",
,";printauK(j,4);",";printauK(j,5);",";printauK(j,6)
            else
                print #6, printK(j,1);",";printK(j,2);",";printK(j,3);",
,";printK(j,4);",";printK(j,5);",";printK(j,6)
            end if
        next

        print #6,
        print #6,

    next

    for j=1 to 7
        print #6, " ,";juan(t);"-";comb(j);",";juan(t);"-";comb(j+8);" ,";
    next
    print #6, " ,";juan(t);"-";comb(8);",";juan(t);"-";comb(8+8)
    for i=1 to cD
        for j=1 to 7
            print #6, species(i);",";K0perf(i,t,j);",";K1perf(i,t,j);" ,";
        next
        print #6, species(i);",";K0perf(i,t,8);",";K1perf(i,t,8)
    next

    close(6)

next

print "  hyp2.csv"
open "hyp2.csv" for output as #6
print #6, " ,";
for j=1 to 7
    print #6, "freq,opthr_";comb(j);", ,";
next
print #6, "freq,opthr_";comb(8)
for i=1 to cD
    print #6, species(i);",";
    for j=1 to 7
        print #6, freq(i);",";opth(i,j);", ,";
    next
    print #6, freq(i);",";opth(i,8)
next
close(6)




print
print "FINISHED WRITING PERFORMANCE FILES"
print
print
print "STARTING TESTS"
print

redim as double pK(1 to 16,1 to 16,1 to cD)
redim as double pauK(1 to 8,1 to 8,1 to cD)

redim as double LOGpK(1 to 16,1 to 16,1 to cD)
```

```
redim as double LOGpauK(1 to 8,1 to 8,1 to cD)

for j=1 to cD

    print "Species ";j;"/";str(cD)

    pK(1,2,j)=ProWMW(jcnt00(j),jcnt01(j),K0000(),j,K0001(),j)
    pK(1,3,j)=ProWMW(jcnt00(j),jcnt10(j),K0000(),j,K0010(),j)
    pK(1,4,j)=ProWMW(jcnt00(j),jcnt11(j),K0000(),j,K0011(),j)
    pK(1,5,j)=ProWMW(jcnt00(j),jcnt00(j),K0000(),j,K0100(),j)
    pK(1,6,j)=ProWMW(jcnt00(j),jcnt01(j),K0000(),j,K0101(),j)
    pK(1,7,j)=ProWMW(jcnt00(j),jcnt10(j),K0000(),j,K0110(),j)
    pK(1,8,j)=ProWMW(jcnt00(j),jcnt11(j),K0000(),j,K0111(),j)
    pK(1,9,j)=ProWMW(jcnt00(j),jcnt00(j),K0000(),j,K1000(),j)
    pK(1,10,j)=ProWMW(jcnt00(j),jcnt01(j),K0000(),j,K1001(),j)
    pK(1,11,j)=ProWMW(jcnt00(j),jcnt10(j),K0000(),j,K1010(),j)
    pK(1,12,j)=ProWMW(jcnt00(j),jcnt11(j),K0000(),j,K1011(),j)
    pK(1,13,j)=ProWMW(jcnt00(j),jcnt00(j),K0000(),j,K1100(),j)
    pK(1,14,j)=ProWMW(jcnt00(j),jcnt01(j),K0000(),j,K1101(),j)
    pK(1,15,j)=ProWMW(jcnt00(j),jcnt10(j),K0000(),j,K1110(),j)
    pK(1,16,j)=ProWMW(jcnt00(j),jcnt11(j),K0000(),j,K1111(),j)

    pK(2,3,j)=ProWMW(jcnt01(j),jcnt10(j),K0001(),j,K0010(),j)
    pK(2,4,j)=ProWMW(jcnt01(j),jcnt11(j),K0001(),j,K0011(),j)
    pK(2,5,j)=ProWMW(jcnt01(j),jcnt00(j),K0001(),j,K0100(),j)
    pK(2,6,j)=ProWMW(jcnt01(j),jcnt01(j),K0001(),j,K0101(),j)
    pK(2,7,j)=ProWMW(jcnt01(j),jcnt10(j),K0001(),j,K0110(),j)
    pK(2,8,j)=ProWMW(jcnt01(j),jcnt11(j),K0001(),j,K0111(),j)
    pK(2,9,j)=ProWMW(jcnt01(j),jcnt00(j),K0001(),j,K1000(),j)
    pK(2,10,j)=ProWMW(jcnt01(j),jcnt01(j),K0001(),j,K1001(),j)
    pK(2,11,j)=ProWMW(jcnt01(j),jcnt10(j),K0001(),j,K1010(),j)
    pK(2,12,j)=ProWMW(jcnt01(j),jcnt11(j),K0001(),j,K1011(),j)
    pK(2,13,j)=ProWMW(jcnt01(j),jcnt00(j),K0001(),j,K1100(),j)
    pK(2,14,j)=ProWMW(jcnt01(j),jcnt01(j),K0001(),j,K1101(),j)
    pK(2,15,j)=ProWMW(jcnt01(j),jcnt10(j),K0001(),j,K1110(),j)
    pK(2,16,j)=ProWMW(jcnt01(j),jcnt11(j),K0001(),j,K1111(),j)

    pK(3,4,j)=ProWMW(jcnt10(j),jcnt11(j),K0010(),j,K0011(),j)
    pK(3,5,j)=ProWMW(jcnt10(j),jcnt00(j),K0010(),j,K0100(),j)
    pK(3,6,j)=ProWMW(jcnt10(j),jcnt01(j),K0010(),j,K0101(),j)
    pK(3,7,j)=ProWMW(jcnt10(j),jcnt10(j),K0010(),j,K0110(),j)
    pK(3,8,j)=ProWMW(jcnt10(j),jcnt11(j),K0010(),j,K0111(),j)
    pK(3,9,j)=ProWMW(jcnt10(j),jcnt00(j),K0010(),j,K1000(),j)
    pK(3,10,j)=ProWMW(jcnt10(j),jcnt01(j),K0010(),j,K1001(),j)
    pK(3,11,j)=ProWMW(jcnt10(j),jcnt10(j),K0010(),j,K1010(),j)
    pK(3,12,j)=ProWMW(jcnt10(j),jcnt11(j),K0010(),j,K1011(),j)
    pK(3,13,j)=ProWMW(jcnt10(j),jcnt00(j),K0010(),j,K1100(),j)
    pK(3,14,j)=ProWMW(jcnt10(j),jcnt01(j),K0010(),j,K1101(),j)
    pK(3,15,j)=ProWMW(jcnt10(j),jcnt10(j),K0010(),j,K1110(),j)
    pK(3,16,j)=ProWMW(jcnt10(j),jcnt11(j),K0010(),j,K1111(),j)

    pK(4,5,j)=ProWMW(jcnt11(j),jcnt00(j),K0011(),j,K0100(),j)
    pK(4,6,j)=ProWMW(jcnt11(j),jcnt01(j),K0011(),j,K0101(),j)
    pK(4,7,j)=ProWMW(jcnt11(j),jcnt10(j),K0011(),j,K0110(),j)
    pK(4,8,j)=ProWMW(jcnt11(j),jcnt11(j),K0011(),j,K0111(),j)
    pK(4,9,j)=ProWMW(jcnt11(j),jcnt00(j),K0011(),j,K1000(),j)
    pK(4,10,j)=ProWMW(jcnt11(j),jcnt01(j),K0011(),j,K1001(),j)
    pK(4,11,j)=ProWMW(jcnt11(j),jcnt10(j),K0011(),j,K1010(),j)
    pK(4,12,j)=ProWMW(jcnt11(j),jcnt11(j),K0011(),j,K1011(),j)
    pK(4,13,j)=ProWMW(jcnt11(j),jcnt00(j),K0011(),j,K1100(),j)
    pK(4,14,j)=ProWMW(jcnt11(j),jcnt01(j),K0011(),j,K1101(),j)
    pK(4,15,j)=ProWMW(jcnt11(j),jcnt10(j),K0011(),j,K1110(),j)
    pK(4,16,j)=ProWMW(jcnt11(j),jcnt11(j),K0011(),j,K1111(),j)
```

```
pK(5,6,j)=ProWMW(jcnt00(j),jcnt01(j),K0100(),j,K0101(),j)
pK(5,7,j)=ProWMW(jcnt00(j),jcnt10(j),K0100(),j,K0110(),j)
pK(5,8,j)=ProWMW(jcnt00(j),jcnt11(j),K0100(),j,K0111(),j)
pK(5,9,j)=ProWMW(jcnt00(j),jcnt00(j),K0100(),j,K1000(),j)
pK(5,10,j)=ProWMW(jcnt00(j),jcnt01(j),K0100(),j,K1001(),j)
pK(5,11,j)=ProWMW(jcnt00(j),jcnt10(j),K0100(),j,K1010(),j)
pK(5,12,j)=ProWMW(jcnt00(j),jcnt11(j),K0100(),j,K1011(),j)
pK(5,13,j)=ProWMW(jcnt00(j),jcnt00(j),K0100(),j,K1100(),j)
pK(5,14,j)=ProWMW(jcnt00(j),jcnt01(j),K0100(),j,K1101(),j)
pK(5,15,j)=ProWMW(jcnt00(j),jcnt10(j),K0100(),j,K1110(),j)
pK(5,16,j)=ProWMW(jcnt00(j),jcnt11(j),K0100(),j,K1111(),j)

pK(6,7,j)=ProWMW(jcnt01(j),jcnt10(j),K0101(),j,K0110(),j)
pK(6,8,j)=ProWMW(jcnt01(j),jcnt11(j),K0101(),j,K0111(),j)
pK(6,9,j)=ProWMW(jcnt01(j),jcnt00(j),K0101(),j,K1000(),j)
pK(6,10,j)=ProWMW(jcnt01(j),jcnt01(j),K0101(),j,K1001(),j)
pK(6,11,j)=ProWMW(jcnt01(j),jcnt10(j),K0101(),j,K1010(),j)
pK(6,12,j)=ProWMW(jcnt01(j),jcnt11(j),K0101(),j,K1011(),j)
pK(6,13,j)=ProWMW(jcnt01(j),jcnt00(j),K0101(),j,K1100(),j)
pK(6,14,j)=ProWMW(jcnt01(j),jcnt01(j),K0101(),j,K1101(),j)
pK(6,15,j)=ProWMW(jcnt01(j),jcnt10(j),K0101(),j,K1110(),j)
pK(6,16,j)=ProWMW(jcnt01(j),jcnt11(j),K0101(),j,K1111(),j)

pK(7,8,j)=ProWMW(jcnt10(j),jcnt11(j),K0110(),j,K0111(),j)
pK(7,9,j)=ProWMW(jcnt10(j),jcnt00(j),K0110(),j,K1000(),j)
pK(7,10,j)=ProWMW(jcnt10(j),jcnt01(j),K0110(),j,K1001(),j)
pK(7,11,j)=ProWMW(jcnt10(j),jcnt10(j),K0110(),j,K1010(),j)
pK(7,12,j)=ProWMW(jcnt10(j),jcnt11(j),K0110(),j,K1011(),j)
pK(7,13,j)=ProWMW(jcnt10(j),jcnt00(j),K0110(),j,K1100(),j)
pK(7,14,j)=ProWMW(jcnt10(j),jcnt01(j),K0110(),j,K1101(),j)
pK(7,15,j)=ProWMW(jcnt10(j),jcnt10(j),K0110(),j,K1110(),j)
pK(7,16,j)=ProWMW(jcnt10(j),jcnt11(j),K0110(),j,K1111(),j)

pK(8,9,j)=ProWMW(jcnt11(j),jcnt00(j),K0111(),j,K1000(),j)
pK(8,10,j)=ProWMW(jcnt11(j),jcnt01(j),K0111(),j,K1001(),j)
pK(8,11,j)=ProWMW(jcnt11(j),jcnt10(j),K0111(),j,K1010(),j)
pK(8,12,j)=ProWMW(jcnt11(j),jcnt11(j),K0111(),j,K1011(),j)
pK(8,13,j)=ProWMW(jcnt11(j),jcnt00(j),K0111(),j,K1100(),j)
pK(8,14,j)=ProWMW(jcnt11(j),jcnt01(j),K0111(),j,K1101(),j)
pK(8,15,j)=ProWMW(jcnt11(j),jcnt10(j),K0111(),j,K1110(),j)
pK(8,16,j)=ProWMW(jcnt11(j),jcnt11(j),K0111(),j,K1111(),j)

pK(9,10,j)=ProWMW(jcnt00(j),jcnt01(j),K1000(),j,K1001(),j)
pK(9,11,j)=ProWMW(jcnt00(j),jcnt10(j),K1000(),j,K1010(),j)
pK(9,12,j)=ProWMW(jcnt00(j),jcnt11(j),K1000(),j,K1011(),j)
pK(9,13,j)=ProWMW(jcnt00(j),jcnt00(j),K1000(),j,K1100(),j)
pK(9,14,j)=ProWMW(jcnt00(j),jcnt01(j),K1000(),j,K1101(),j)
pK(9,15,j)=ProWMW(jcnt00(j),jcnt10(j),K1000(),j,K1110(),j)
pK(9,16,j)=ProWMW(jcnt00(j),jcnt11(j),K1000(),j,K1111(),j)

pK(10,11,j)=ProWMW(jcnt01(j),jcnt10(j),K1001(),j,K1010(),j)
pK(10,12,j)=ProWMW(jcnt01(j),jcnt11(j),K1001(),j,K1011(),j)
pK(10,13,j)=ProWMW(jcnt01(j),jcnt00(j),K1001(),j,K1100(),j)
pK(10,14,j)=ProWMW(jcnt01(j),jcnt01(j),K1001(),j,K1101(),j)
pK(10,15,j)=ProWMW(jcnt01(j),jcnt10(j),K1001(),j,K1110(),j)
pK(10,16,j)=ProWMW(jcnt01(j),jcnt11(j),K1001(),j,K1111(),j)

pK(11,12,j)=ProWMW(jcnt10(j),jcnt11(j),K1010(),j,K1011(),j)
pK(11,13,j)=ProWMW(jcnt10(j),jcnt00(j),K1010(),j,K1100(),j)
pK(11,14,j)=ProWMW(jcnt10(j),jcnt01(j),K1010(),j,K1101(),j)
pK(11,15,j)=ProWMW(jcnt10(j),jcnt10(j),K1010(),j,K1110(),j)
pK(11,16,j)=ProWMW(jcnt10(j),jcnt11(j),K1010(),j,K1111(),j)

pK(12,13,j)=ProWMW(jcnt11(j),jcnt00(j),K1011(),j,K1100(),j)
```

```
    pK(12,14,j)=ProWMW(jcnt11(j),jcnt01(j),K1011(),j,K1101(),j)
    pK(12,15,j)=ProWMW(jcnt11(j),jcnt10(j),K1011(),j,K1110(),j)
    pK(12,16,j)=ProWMW(jcnt11(j),jcnt11(j),K1011(),j,K1111(),j)

    pK(13,14,j)=ProWMW(jcnt00(j),jcnt01(j),K1100(),j,K1101(),j)
    pK(13,15,j)=ProWMW(jcnt00(j),jcnt10(j),K1100(),j,K1110(),j)
    pK(13,16,j)=ProWMW(jcnt00(j),jcnt11(j),K1100(),j,K1111(),j)

    pK(14,15,j)=ProWMW(jcnt01(j),jcnt10(j),K1101(),j,K1110(),j)
    pK(14,16,j)=ProWMW(jcnt01(j),jcnt11(j),K1101(),j,K1111(),j)

    pK(15,16,j)=ProWMW(jcnt10(j),jcnt11(j),K1110(),j,K1111(),j)


    pauK(1,2,j)=ProWMW(jcnt00(j),jcnt00(j),auK000(),j,auK001(),j)
    pauK(1,3,j)=ProWMW(jcnt00(j),jcnt01(j),auK000(),j,auK010(),j)
    pauK(1,4,j)=ProWMW(jcnt00(j),jcnt01(j),auK000(),j,auK011(),j)
    pauK(1,5,j)=ProWMW(jcnt00(j),jcnt10(j),auK000(),j,auK100(),j)
    pauK(1,6,j)=ProWMW(jcnt00(j),jcnt10(j),auK000(),j,auK101(),j)
    pauK(1,7,j)=ProWMW(jcnt00(j),jcnt11(j),auK000(),j,auK110(),j)
    pauK(1,8,j)=ProWMW(jcnt00(j),jcnt11(j),auK000(),j,auK111(),j)

    pauK(2,3,j)=ProWMW(jcnt01(j),jcnt10(j),auK001(),j,auK010(),j)
    pauK(2,4,j)=ProWMW(jcnt01(j),jcnt11(j),auK001(),j,auK011(),j)
    pauK(2,5,j)=ProWMW(jcnt01(j),jcnt00(j),auK001(),j,auK100(),j)
    pauK(2,6,j)=ProWMW(jcnt01(j),jcnt01(j),auK001(),j,auK101(),j)
    pauK(2,7,j)=ProWMW(jcnt01(j),jcnt10(j),auK001(),j,auK110(),j)
    pauK(2,8,j)=ProWMW(jcnt01(j),jcnt11(j),auK001(),j,auK111(),j)

    pauK(3,4,j)=ProWMW(jcnt10(j),jcnt11(j),auK010(),j,auK011(),j)
    pauK(3,5,j)=ProWMW(jcnt10(j),jcnt00(j),auK010(),j,auK100(),j)
    pauK(3,6,j)=ProWMW(jcnt10(j),jcnt01(j),auK010(),j,auK101(),j)
    pauK(3,7,j)=ProWMW(jcnt10(j),jcnt10(j),auK010(),j,auK110(),j)
    pauK(3,8,j)=ProWMW(jcnt10(j),jcnt11(j),auK010(),j,auK111(),j)

    pauK(4,5,j)=ProWMW(jcnt11(j),jcnt00(j),auK011(),j,auK100(),j)
    pauK(4,6,j)=ProWMW(jcnt11(j),jcnt01(j),auK011(),j,auK101(),j)
    pauK(4,7,j)=ProWMW(jcnt11(j),jcnt10(j),auK011(),j,auK110(),j)
    pauK(4,8,j)=ProWMW(jcnt11(j),jcnt11(j),auK011(),j,auK111(),j)

    pauK(5,6,j)=ProWMW(jcnt00(j),jcnt01(j),auK100(),j,auK101(),j)
    pauK(5,7,j)=ProWMW(jcnt00(j),jcnt10(j),auK100(),j,auK110(),j)
    pauK(5,8,j)=ProWMW(jcnt00(j),jcnt11(j),auK100(),j,auK111(),j)

    pauK(6,7,j)=ProWMW(jcnt01(j),jcnt10(j),auK101(),j,auK110(),j)
    pauK(6,8,j)=ProWMW(jcnt01(j),jcnt11(j),auK101(),j,auK111(),j)

    pauK(7,8,j)=ProWMW(jcnt10(j),jcnt11(j),auK110(),j,auK111(),j)

    for i=1 to 8
        for t=i to 8
            pauK(t,i,j)=-pauK(i,t,j)
        next
        pauK(i,i,j)=1
    next

    for i=1 to 16
        for t=i to 16
            pK(t,i,j)=-pK(i,t,j)
        next
        pK(i,i,j)=1
    next

next
```

```
for i=1 to 8
    for j=1 to 8
        for t=1 to cD
            LOGpauK(i,j,t)=sgn(pauK(i,j,t))*(-log(abs(pauK(i,j,t)))/log(10))
        next
    next
next

for i=1 to 16
    for j=1 to 16
        for t=1 to cD
            LOGpK(i,j,t)=sgn(pK(i,j,t))*(-log(abs(pK(i,j,t)))/log(10))
        next
    next
next




print "   TEST FINISHED"
print
print
print "WRITING OUTCOME FILES"
print

redim as integer bonf(1 to 5)
bonf(0)=4 'singola ipotesi con singola specie con AUK
bonf(1)=8 'singola ipotesi con singola specie con K
bonf(2)=32 'tutte le ipotesi con singola specie
bonf(3)=120 'tutti i confronti con singola specie
bonf(4)=4*32 '=128 singola ipotesi con tutte le specie con AUK
bonf(5)=8*32 '=256 singola ipotesi con tutte le specie con K
bonf(6)=28*32 '=896 tutte le ipotesi con tutte le specie
bonf(7)=120*32 '=3840 tutte le ipotesi con tutte le specie


print "   pK.csv"
open "pK.csv" for output as #5
for t=1 to cD
    print #5, str(t);"_pK";
    for i=1 to 16
        print #5, ",";comb(i);
    next
    print #5, ", ,";
    print #5, str(t);"_-log(0-4)";
    for i=1 to 16
        print #5, ",";comb(i);
    next
    print #5, ", ,";
    print #5, str(t);"_-log10";
    for i=1 to 15
        print #5, ",";comb(i);
    next
    print #5, ",";comb(16)

    for i=1 to 16
        print #5, comb(i);
        for j=1 to 16
            print #5, ",";str(pK(i,j,t));
        next
        print #5, ", ,";
        print #5, comb(i);
```

```
        for j=1 to 16
            if abs(LOGpK(i,j,t))>=-log(0.0001)/log(10) then
                print #5, ",";4*sgn(pK(i,j,t));
            elseif abs(LOGpK(i,j,t))<-log(0.05)/log(10) then
                print #5, ",0";
            else
                print #5, ",";str(LOGpK(i,j,t));
            end if
        next
        print #5, ", ,";
        print #5, comb(i);
        for j=1 to 15
            print #5, ",";str(LOGpK(i,j,t));
        next
        print #5, ",";str(LOGpK(i,j,t))
    next

    print #5,

next
close(5)

print "   pauK.csv"
open "pauK.csv" for output as #5
for t=1 to cD
    print #5, str(t);"_pauK";
    for i=1 to 8
        print #5, ",";comb(i);
    next
    print #5, ", ,";
    print #5, str(t);"_-log(0-4)";
    for i=1 to 8
        print #5, ",";comb(i);
    next
    print #5, ", ,";
    print #5, str(t);"_-log10";
    for i=1 to 7
        print #5, ",";comb(i);
    next
    print #5, ",";comb(8)

    for i=1 to 8
        print #5, comb(i);
        for j=1 to 8
            print #5, ",";str(pauK(i,j,t));
        next
        print #5, ", ,";
        print #5, comb(i);
        for j=1 to 8
            if abs(LOGpauK(i,j,t))>=-log(0.0001)/log(10) then
                print #5, ",";4*sgn(pauK(i,j,t));
            elseif abs(LOGpauK(i,j,t))<-log(0.05)/log(10) then
                print #5, ",0";
            else
                print #5, ",";str(LOGpauK(i,j,t));
            end if
        next
        print #5, ", ,";
        print #5, comb(i);
        for j=1 to 7
            print #5, ",";str(LOGpauK(i,j,t));
        next
        print #5, ",";str(LOGpauK(i,j,t))
```

```
    next

    print #5,
next
close(5)

redim as double printh1(1 to cD,1 to 8)
redim as double printh3(1 to cD,1 to 8)
redim as double printh4(1 to cD,1 to 8)
redim as double printh5(1 to cD,1 to 8)
redim as double printh1auK(1 to cD,1 to 4)
redim as double printh3auK(1 to cD,1 to 4)
redim as double printh4auK(1 to cD,1 to 4)
redim as double printh5auK(1 to cD,1 to 4)

print "   hyp1.csv"
open "hyp1.csv" for output as #5
print #5, "0vs1,";
for j=1 to 8
    print #5, "#";mid(comb(j),3);",";
next
print #5, " ,opth,";
for j=1 to 7
    print #5, "'";mid(comb(j),3);",";
next
print #5, "'";mid(comb(8),3)
for h=1 to cD
    print #5, species(h);
    for j=1 to 8
        if abs(LOGpK(j,j+8,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+8,h));
        elseif abs(LOGpK(j,j+8,h))<-log(0.05)/log(10) then
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+8,h));
        end if
    next
    print #5, ", ,";species(h);",";
    for j=1 to 7
        print #5, str(opth(h,j));",";
    next
    print #5, str(opth(h,8))
next

for h=1 to cD
    for j=1 to 8
        printh1(h,j)=-pK(j,j+8,h)
    next
next

print #5,
print #5,
print #5, "0vs1,";
for j=1 to 7
    print #5, "#";mid(comb(j),3);",";
next
print #5, "#";mid(comb(8),3)
for h=1 to cD
    print #5, species(h);
    for j=1 to 8
        print #5, ",";str(printh1(h,j));
    next
    print #5, ", "
next
```

```
for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    for j=1 to 7
        print #5, "#";mid(comb(j),3);",";
    next
    print #5, "#";mid(comb(8),3)
    for h=1 to cD
        print #5, species(h);
        for j=1 to 8
            if abs(printh1(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh1(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh1(h,j)));
            elseif abs(printh1(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh1(h,j)));
            elseif abs(printh1(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh1(h,j)));
            else
                print #5, ",";str(0.0001*sgn(printh1(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

close(5)

print "   hyp3.csv"
open "hyp3.csv" for output as #5
print #5, "SSPvsMSP,";
print #5, "'0#00,'0#01,'0#10,'0#11,'1#00,'1#01,'1#10,'1#11"
for h=1 to cD
    print #5, species(h);
    for j=1 to 4
        if abs(LOGpK(j,j+4,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+4,h));
        elseif abs(LOGpK(j,j+4,h))<-log(0.05)/log(10) then
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+4,h));
        end if
    next
    for j=9 to 11
        if abs(LOGpK(j,j+4,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+4,h));
        elseif abs(LOGpK(j,j+4,h))<-log(0.05)/log(10) then
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+4,h));
        end if
    next
```

```
    if abs(LOGpK(12,12+4,h))>=-log(0.0001)/log(10) then
        print #5, ",";4*sgn(pK(12,12+4,h))
    elseif abs(LOGpK(12,12+4,h))<-log(0.05)/log(10) then
        print #5, ",0"
    else
        print #5, ",";str(LOGpK(12,12+4,h))
    end if
next

for h=1 to cD
    printh3(h,1)=-pK(1,1+4,h)
    printh3(h,2)=-pK(2,2+4,h)
    printh3(h,3)=-pK(3,3+4,h)
    printh3(h,4)=-pK(4,4+4,h)
    printh3(h,5)=-pK(9,9+4,h)
    printh3(h,6)=-pK(10,10+4,h)
    printh3(h,7)=-pK(11,11+4,h)
    printh3(h,8)=-pK(12,12+4,h)
    printh3auK(h,1)=-pauK(1,1+4,h)
    printh3auK(h,2)=-pauK(2,2+4,h)
    printh3auK(h,3)=-pauK(3,3+4,h)
    printh3auK(h,4)=-pauK(4,4+4,h)
next

print #5,
print #5,
print #5, "SSPvsMSP,";
print #5, "'0#00,'0#01,'0#10,'0#11,'1#00,'1#01,'1#10,'1#11"
for h=1 to cD
    print #5, species(h);
    for j=1 to 8
        print #5, ",";str(printh3(h,j));
    next
    print #5, ", "
next

for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    print #5, "'0#00,'0#01,'0#10,'0#11,'1#00,'1#01,'1#10,'1#11"
    for h=1 to cD
        print #5, species(h);
        for j=1 to 8
            if abs(printh3(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh3(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh3(h,j)));
            elseif abs(printh3(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh3(h,j)));
            elseif abs(printh3(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh3(h,j)));
            else
                print #5, ",";str(0.0001*sgn(printh3(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
```

```
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

print #5,
print #5,
print #5, "SSPvsMSP,";
print #5, "'X#00,'X#01,'X#10,'X#11"
for h=1 to cD
    print #5, species(h);
    for j=1 to 4
        print #5, ",";str(printh3auK(h,j));
    next
    print #5, ", "
next

for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    print #5, "'X#00,'X#01,'X#10,'X#11"
    for h=1 to cD
        print #5, species(h);
        for j=1 to 4
            if abs(printh3auK(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh3auK(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh3auK(h,j)));
            elseif abs(printh3auK(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh3auK(h,j)));
            elseif abs(printh3auK(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh3auK(h,j)));
            else
                print #5, ",";str(0.0001*sgn(printh3auK(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

close(5)

print "   hyp4.csv"
open "hyp4.csv" for output as #5
print #5, "PPvsRP,";
print #5, "'00#0,'00#1,'01#0,'01#1,'10#0,'10#1,'11#0,'11#1"
for h=1 to cD
    print #5, species(h);
    for j=1 to 2
        if abs(LOGpK(j,j+2,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+2,h));
        elseif abs(LOGpK(j,j+2,h))<-log(0.05)/log(10) then
```

```
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+2,h));
        end if
    next
    for j=5 to 6
        if abs(LOGpK(j,j+2,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+2,h));
        elseif abs(LOGpK(j,j+2,h))<-log(0.05)/log(10) then
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+2,h));
        end if
    next
    for j=9 to 10
        if abs(LOGpK(j,j+2,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+2,h));
        elseif abs(LOGpK(j,j+2,h))<-log(0.05)/log(10) then
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+2,h));
        end if
    next
    if abs(LOGpK(13,13+2,h))>=-log(0.0001)/log(10) then
        print #5, ",";4*sgn(pK(j,j+2,h));
    elseif abs(LOGpK(13,13+2,h))<-log(0.05)/log(10) then
        print #5, ",0";
    else
        print #5, ",";str(LOGpK(13,13+2,h));
    end if
    if abs(LOGpK(14,14+2,h))>=-log(0.0001)/log(10) then
        print #5, ",";4*sgn(pK(14,14+2,h))
    elseif abs(LOGpK(14,14+2,h))<-log(0.05)/log(10) then
        print #5, ",0"
    else
        print #5, ",";str(LOGpK(14,14+2,h))
    end if
next

for h=1 to cD
    printh4(h,1)=-pK(1,1+2,h)
    printh4(h,2)=-pK(2,2+2,h)
    printh4(h,3)=-pK(5,5+2,h)
    printh4(h,4)=-pK(6,6+2,h)
    printh4(h,5)=-pK(9,9+2,h)
    printh4(h,6)=-pK(10,10+2,h)
    printh4(h,7)=-pK(13,13+2,h)
    printh4(h,8)=-pK(14,14+2,h)
    printh4auK(h,1)=-pauK(1,1+2,h)
    printh4auK(h,2)=-pauK(2,2+2,h)
    printh4auK(h,3)=-pauK(5,5+2,h)
    printh4auK(h,4)=-pauK(6,6+2,h)
next

print #5,
print #5,
print #5, "PPvsRP,";
print #5, "'00#0,'00#1,'01#0,'01#1,'10#0,'10#1,'11#0,'11#1"
for h=1 to cD
    print #5, species(h);
    for j=1 to 8
        print #5, ",";str(printh4(h,j));
    next
    print #5, ", "
```

```
next

for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    print #5, "'00#0,'00#1,'01#0,'01#1,'10#0,'10#1,'11#0,'11#1"
    for h=1 to cD
        print #5, species(h);
        for j=1 to 8
            if abs(printh4(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh4(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh4(h,j)));
            elseif abs(printh4(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh4(h,j)));
            elseif abs(printh4(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh4(h,j)));
            else
                print #5, ",";str(0.0001*sgn(printh4(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

print #5,
print #5,
print #5, "PPvsRP,";
print #5, "'X0#0,'X0#1,'X1#0,'X1#1"
for h=1 to cD
    print #5, species(h);
    for j=1 to 4
        print #5, ",";str(printh4auK(h,j));
    next
    print #5, ", "
next

for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    print #5, "'X0#0,'X0#1,'X1#0,'X1#1"
    for h=1 to cD
        print #5, species(h);
        for j=1 to 4
            if abs(printh4auK(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh4auK(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh4auK(h,j)));
            elseif abs(printh4auK(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh4auK(h,j)));
            elseif abs(printh4auK(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh4auK(h,j)));
```

```
            else
                print #5, ",";str(0.0001*sgn(printh4auK(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

close(5)

print "   hyp5.csv"
open "hyp5.csv" for output as #5
print #5, "EPvsRP,";
print #5, "'000#,'001#,'010#,'011#,'100#,'101#,'110#,'111#"
for h=1 to cD
    print #5, species(h);
    for j=1 to 13 step 2
        if abs(LOGpK(j,j+1,h))>=-log(0.0001)/log(10) then
            print #5, ",";4*sgn(pK(j,j+1,h));
        elseif abs(LOGpK(j,j+1,h))<-log(0.05)/log(10) then
            print #5, ",0";
        else
            print #5, ",";str(LOGpK(j,j+1,h));
        end if
    next
    if abs(LOGpK(15,15+1,h))>=-log(0.0001)/log(10) then
        print #5, ",";4*sgn(pK(15,15+1,h))
    elseif abs(LOGpK(15,15+1,h))<-log(0.05)/log(10) then
        print #5, ",0"
    else
        print #5, ",";str(LOGpK(15,15+1,h))
    end if
next

for h=1 to cD
    for j=1 to 8
        printh5(h,j)=-pK(2*j-1,2*j,h)
        if j<5 then
            printh5auK(h,j)=-pK(2*j-1,2*j,h)
        end if
    next
next

print #5,
print #5,
print #5, "EPvsRP,";
print #5, "'000#,'001#,'010#,'011#,'100#,'101#,'110#,'111#"
for h=1 to cD
    print #5, species(h);
    for j=1 to 8
        print #5, ",";str(printh5(h,j));
    next
    print #5, ", "
next
```

```
for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    print #5, "'000#,'001#,'010#,'011#,'100#,'101#,'110#,'111#"
    for h=1 to cD
        print #5, species(h);
        for j=1 to 8
            if abs(printh5(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh5(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh5(h,j)));
            elseif abs(printh5(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh5(h,j)));
            elseif abs(printh5(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh5(h,j)));
            else
                print #5, ",";str(0.0001*sgn(printh5(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

print #5,
print #5,
print #5, "EPvsRP,";
print #5, "'X00#,'X01#,'X10#,'X11#"
for h=1 to cD
    print #5, species(h);
    for j=1 to 4
        print #5, ",";str(printh5auK(h,j));
    next
    print #5, ", "
next

for i=0 to 7
    print #5,
    print #5,
    print #5, str(bonf(i));",";
    print #5, "'X00#,'X01#,'X10#,'X11#"
    for h=1 to cD
        print #5, species(h);
        for j=1 to 4
            if abs(printh5auK(h,j))>0.05/bonf(i) then
                print #5, ",0";
            elseif abs(printh5auK(h,j))>0.01/bonf(i) then
                print #5, ",";str(0.05*sgn(printh5auK(h,j)));
            elseif abs(printh5auK(h,j))>0.001/bonf(i) then
                print #5, ",";str(0.01*sgn(printh5auK(h,j)));
            elseif abs(printh5auK(h,j))>0.0001/bonf(i) then
                print #5, ",";str(0.001*sgn(printh5auK(h,j)));
            else
```

```
                print #5, ",";str(0.0001*sgn(printh5auK(h,j)));
            end if
        next
        print #5, ", "
    next
    print #5,
    print #5,
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.05 -->
";0.05/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.01 -->
";0.01/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.001 -->
";0.001/bonf(i)
    print #5, "bonferroni correction ";str(bonf(i));" - alpha 0.0001 -->
";0.0001/bonf(i)
next

close(5)

print

print "END WRITING"

sleep
end



'SUBROUTINE E FUNZIONI

  'versione modificata di InsertionSort: in base ai valori contenuti nella
colonna 'key',
  'ordina le righe della matrice array in modo crescente o decrescente a
seconda che 'sense'
  'venga passato con valore, rispettivamente, positivo o negativo (se sense=0,
la matrice
  'resta invariata); inoltre la sub è generalizzata a matrici con indici di
riga o
  'colonna inferiori a 1
Sub SenseInsertionSort(array() as double,key as integer,sense as double)
    If sense <> 0 Then
        Dim As Integer i,j,k,fr,lr,fc,lc
        'determina i limiti della matrice da ordinare
        fr = Lbound(array,1) 'first row
        lr = Ubound(array,1) 'last row
        fc = Lbound(array,2) 'first column
        lc = Ubound(array,2) 'last column
        'vettore temporaneo (tanti elementi quante colonne nella matrice da
ordinare)
        Dim As Double tmp(fc to lc)
        For i = fr + 1 To lr
            For j = fc to lc
                tmp(j) = array(i,j)
            Next
            For k = i - 1 To fr Step -1
                If sense > 0 Then
                    If tmp(key) < array(k,key) Then
                        For j = fc To lc
                            array(k+1,j) = array(k,j)
                        Next
                    Else
                        Exit For
                    End If
                Else
```

```
                    If tmp(key) > array(k,key) Then
                        For j = fc To lc
                            array(k+1,j) = array(k,j)
                        Next
                    Else
                        Exit For
                    End If
                End If
            Next
            For j = fc to lc
                array(k+1,j) = tmp(j)
            Next
        Next
    End If
End Sub


'trasforma i valori contenuti nella colonna FColumn della matrice FromMatrix()
 'nei corrispondenti ranghi e li copia nella colonna TColumn della matrice'
 'ToMatrix(); usa la subroutine [SenseInsertionSort]
Sub ColumnRank(FromMatrix() as double,FColumn as integer,ToMatrix() as
double,TColumn as integer)

    dim as integer r,i,j,k,m,cnt 'variabili di lavoro
    dim as single rankmean 'variabili di lavoro
    r=ubound(FromMatrix,1)-lbound(FromMatrix,1)+1 'calcolo numero righe di
FromMatrix()
    redim as double work(1 to r,1 to 2) 'dichiarazione matrice temporanea di
lavoro
    'riempimento colonna 1 di work() con i naturali da 1 a r
    for i=1 to r
        work(i,1)=i
    next
    'riempimento colonna 2 di work() con valori presi da FromMatrix();
    for i=1 to r
        work(i,2)=FromMatrix(i,FColumn)
    next
    'InsertionSort ordina le righe di work in base ai valori della colonna 2;
    'i naturali nella prima colonna conservano il riferimento alla riga
    'di FromMatrix() cui appartengono i valori sulla colonna 2
    SenseInsertionSort(work(),2,1)
    'ogni ciclo attribuisce il rango ad un singolo valore (se non ripetuto)
    'o ad un gruppo di valori (se ripetuti) della colonna 2 di work();
    'i ranghi vengono scritti su ToMatrix() in corrispondenza delle posizioni
    'dei valori originali su FromMatrix()
    k=1
    do
        cnt=1
        do while work(k+1,2)=work(k,2) and k<r 'rileva e conta eventuali
ripetizioni
            cnt+=1
            k+=1
        loop
        rankmean=(2*k-cnt+1)/2 'media dei ranghi dei valori ripetuti
        'tale media è copiata sulla colonna j di ToMatrix(), alla riga indicata
        'dal numero naturale sulla prima colonna di work()
        for m=(k-cnt+1) to k
            ToMatrix(work(m,1),TColumn)=rankmean
        next
        k+=1 'definisce la posizione del nuovo valore di cui calcolare il rango
    loop until k>r
end sub
```

```
function Combination(n as uinteger,k as uinteger) as uinteger

    if k>n or n<0 or k<0 then
        return 0
    else
        if k=0 or k=n then
            return 1
        else
            if k=1 or k=n-1 then
                return n
            else
                dim as uinteger i
                dim as double comb
                comb=1
                if k>=n/2 then
                    for i=0 to n-k-1
                        comb/=n-k-i
                        comb*=n-i
                    next
                else
                    for i=0 to k-1
                        comb/=k-i
                        comb*=n-i
                    next
                end if
                return comb
            end if
        end if
    end if

end function

function pNorm(z as double) as double

    dim as double p
    dim as uinteger i,loops
    const epsilon as double = 0.0001
    const coeff as double = epsilon/(2*sqr(2*3.1415926535897932))

    z=abs(z)
    loops=70000

    for i=1 to loops
        p+=exp(-(z^2)/2)+exp(-((z+epsilon)^2)/2)
        z+=epsilon
    next
    if p>=2.51E-296 then
        return p*coeff
    else
        return 1E-300
    end if

end function


 'calcola il k-esimo t-ile nella distribuzione degli istop valori più elevati
 'contenuti nella c-esima colonna della matrice array(); per calcolare la
mediana
 'basta passare k=t/2; passando k=0 o k=t, rispettivamente, si ottengono il
  'minimo e il massimo della stessa distribuzione; richiede la sub
SenseInsertionSort
function QuantileStop(array() as double,c as integer,k as integer,t as
integer,istop as integer) as double
   dim as double p 'position
```

```
   dim as integer u,l,r,i,ip 'integer position
   u=ubound(array,1)
   l=lbound(array,1)
   r=u-l+1
   redim as double temp(l to u,1 to 1)
   if istop<=r and istop>=0 then
       for i=l to u
           temp(i,1)=array(i,c)
       next
       SenseInsertionSort(temp(),1,1)
       if k=0 then
           return temp(u-istop+1,1)
       elseif k=t then
           return temp(u,1)
       else
           p=k*((istop-1)/t)+r-istop+l
           ip=int(p)
           return temp(ip,1)+(p-ip)*(temp(ip+1,1)-temp(ip,1))
       end if
   else
       return -2.2E-307
   end if
end function



function WMW(A() as double,cA as integer,B() as double,cB as integer) as double
    dim as uinteger n1,n2
    n1=ubound(A,1)-lbound(A,1)+1
    n2=ubound(B,1)-lbound(B,1)+1

    if n1<3 or n2<3 then
        return 0.7
    else
        dim as uinteger n,i,j,cnt,cntpos,sum,nmin
        dim as double z,T,p
        dim as byte psign
        dim as single expmean
        n=n1+n2
        redim as double tabel(1 to n,1 to 3)
        'si assegna l'etichetta 1 ai dati della serie meno numerosa (colonna 2
di tabel())
        if n1>=n2 then
            nmin=n2
            psign=1
            expmean=n2*(n1+n2+1)/2
            for i=n1+1 to n
                tabel(i,2)=1
            next
        else
            nmin=n1
            psign=-1
            expmean=n1*(n1+n2+1)/2
            for i=1 to n1
                tabel(i,2)=1
            next
        end if
        'la prima colonna di tabel() è riempita con i dati di entrambe la serie
        for i=1 to n1
            tabel(i,1)=A(i,cA)
        next
        for i=1 to n2
            tabel(i+n1,1)=B(i,cB)
        next
```

```
'i dati sono trasformati nei relativi ranghi
ColumnRank(tabel(),1,tabel(),3)
'tabel() è riordinata nel senso crescente dei ranghi
SenseInsertionSort(tabel(),3,1)
'somma dei ranghi

for i=1 to n
    T+=tabel(i,2)*tabel(i,3)
next
z=T-expmean
if z>=0 then
    psign*=-1
    z-=0.5
else
    z+=0.5
end if
if n1>15 or n2>15 then
    dim as uinteger Q
    dim as double DS
    cnt=1
    for i=1 to n-1
        if tabel(i+1,3)=tabel(i,3) then
            cnt+=1
        else
            Q+=cnt^3-cnt
            cnt=1
        end if
    next
    DS=((n1*n2/(n*(n-1)))*((n^3-n-Q)/12))^(1/2)
    z/=DS
    return psign*pNorm(z)
else
    dim as uinteger istart,istop
    istart=2^n
    istop=2^(n+1)-1
    redim as double distrib(1 to 1,1 to 2)
    dim as single sum
    dim as ubyte ctr
    distrib(1,1)=0
    distrib(1,2)=0
    cnt=1
    for i=istart to istop
        ctr=0
        sum=0
        for j=2 to n+1
            sum+=cbyte(mid(str(bin(i)),j,1))
            if sum>nmin then
                ctr=1
                sum=0
                exit for
            end if
        next
        if sum<nmin then
            ctr=1
            sum=0
        end if
        if ctr=0 then
            sum=0
            for j=2 to n+1
                sum+=cbyte(mid(str(bin(i)),j,1))*tabel(j-1,3)
            next
            for j=1 to cnt
                if sum=distrib(j,1) then
                    distrib(j,2)+=1
```

```
                        exit for
                    else
                        if j=cnt then
                            cnt+=1
                            redim preserve as double distrib(1 to cnt,1 to
2)
                            distrib(cnt,1)=sum
                            distrib(cnt,2)=1
                        end if
                    end if
                next
                sum=0
            end if
        next
        SenseInsertionSort(distrib(),1,1)
        i=1
        j=cnt
        sum=0
        if z<0 then
            do
                sum+=distrib(i,2)
                i+=1
            loop until distrib(i,1)>T
        else
            do
                sum+=distrib(j,2)
                j-=1
            loop until distrib(j,1)<T
        end if

        return psign*(sum/Combination(n,nmin))
    end if
end if
end function

 'assiste la sub WMW quando non tutti i valori di una o di entrambe le
 'distribuzioni vanno considerati nell'analisi
function ProWMW(astop as integer,bstop as integer,A() as double,cA as
integer,B() as double,cB as integer) as double
    dim as integer rA,rB,cnt,i
    rA=ubound(A,1)-lbound(A,1)+1
    rB=ubound(B,1)-lbound(B,1)+1

    if rA<>astop and rB<>bstop then

        redim as double tempA(1 to rA,1 to 2),tempB(1 to rB,1 to 2)

        for i=1 to rA
            tempA(i,1)=A(i,cA)
        next
        for i=1 to rB
            tempB(i,1)=B(i,cB)
        next

        for i=1 to rA
            if abs(tempA(i,1))>0 then
                cnt+=1
                tempA(cnt,2)=tempA(i,1)
            end if
        next
        cnt=0
        for i=1 to rB
            if abs(tempB(i,1))>0 then
                cnt+=1
```

```
                tempB(cnt,2)=tempB(i,1)
            end if
        next

        redim preserve as double tempA(1 to astop,1 to 2),tempB(1 to bstop,1 to
2)

        return WMW(tempA(),2,tempB(),2)

    elseif rA<>astop and rB=bstop then

        redim as double tempA(1 to rA,1 to 2)

        for i=1 to rA
            tempA(i,1)=A(i,cA)
        next

        for i=1 to rA
            if abs(tempA(i,1))>0 then
                cnt+=1
                tempA(cnt,2)=tempA(i,1)
            end if
        next

        redim preserve as double tempA(1 to astop,1 to 2)

        return WMW(tempA(),2,B(),cB)

    elseif rB<>bstop and rA=astop then

        redim as double tempB(1 to rB,1 to 2)

        for i=1 to rB
            tempB(i,1)=B(i,cB)
        next

        for i=1 to rB
            if abs(tempB(i,1))>0 then
                cnt+=1
                tempB(cnt,2)=tempB(i,1)
            end if
        next

        redim preserve as double tempB(1 to bstop,1 to 2)

        return WMW(A(),cA,tempB(),2)

    else

        return WMW(A(),cA,B(),cB)

    end if

end function
```

Stampato a Roma
Maggio 2010