Reti Neurali Artificiali per lo studio del mercato

Università degli studi di Brescia - Dipartimento Metodi Quantitativi
Marco Sandri (sandri.marco@gmail.com)

Regressione non lineare con un modello neurale di tipo radial basis

Obiettivi:

(1) Imparare a costruire una rete neurale radial basis con Matlab e a stimarne i pesi al fine di affrontare un problema di regressione non lineare. Anche in questo caso con Matlab è tutto piuttosto semplice. I comandi **`newrb`** e **`sim`** sono i due comandi fondamentali con cui, rispettivamente, si crea la rete e si stimano i suoi parametri a partire dai dati e poi si effettuano le previsioni.
Attenzione perché qui non viene utilizzato il comando **`train`** dato che la stima dei pesi della rete avviene già direttamente in **`newrb`**.
Le reti radial basis offrono un grande vantaggio rispetto alle reti neurali:
- la stima dei parametri della rete (parametri di localizzazione e pesi fra strato nascosto ed output) risulta molto più semplice e rapida rispetto ad una rete feedforward; l'impegno richiesto da un punto di vista computazione è quindi molto ridotto.

Permangono però alcuni problemi:
- le difficoltà connesse alla costruzione della rete, in particolare alla scelta della sua complessità ottimale (che è legata al numero di neuroni dello strato nascosto) e alla scelta del parametro di scala σ_k; se questa complessità è insufficiente, la rete può non essere in grado di rappresentare adeguatamente il problema e quindi può non essere in grado di fare buone previsioni; se invece è eccessiva, la rete rischia di cadere in overfitting, iperspecializzandosi sui dati, approssimando non solo la componente deterministica ma anche quella stocastica e quindi perdendo capacità di generalizzazione.

(2) Fare una serie di esperimenti numerici per comprendere il ruolo di uno dei parametri più importanti per una rete neurale: il numero di neuroni nello strato nascosto e lo **`spread`**, cioè il parametro di scala σ_k per le funzioni di attivazione. Variando **`nhid`** e **`spread`** potremo creare situazioni di under- e over-fitting, cioè situazioni in cui la complessità della rete è insufficiente o eccessiva per il problema che le si chiede di risolvere. Variando il numero di input **`ninp`** si potrà anche sperimentare il ruolo che hanno le variabili non importanti sulla capacità previsiva della rete. Sarà interessante anche provare a lavorare con campioni piccoli o grandi (variando il parametro **`N`**) e con dati affetti da dosi più o meno marcate di stocasticità (variando il parametro **`devstd`**).

Osservazioni:

(1) Nell'esempio che abbiamo qui considerato non analizziamo dei dati reali, ma dei dati simulati. Si tratta cioè di dati artificiali, da noi generati utilizzando un processo (non lineare) generatore dei dati piuttosto semplice:

$$y = -0.6 + 1.2 \cdot x_1^3 + \sin(3.46 \cdot x_1) + \varepsilon$$

dove ε è la componente stocastica distribuita come una normale di media nulla e deviazione standard s, cioè $\varepsilon \sim N(0, s)$. Si vede quindi che y è influenzata solo dalla variabile x_1. Tutte le altre variabili osservate (x_2, x_3, etc.) sono quindi variabili non importanti ai fini previsivi. Anzi, sono variabili che, se utilizzate in fase di stima, possono peggiorare la qualità della previsione del modello.

(2) Grazie al fatto che si tratta di dati simulati, noi conosciamo tutto di essi, anche la componente deterministica (indicata con una linea rossa), che nei problemi reali non è mai nota e che di fatto è il vero oggetto di indagine. Quindi in questo esperimento numerico abbiamo anche la possibilità di valutare se la rete riesce ad approssimare bene la componente deterministica. Nella realtà, lo ribadiamo, questa verifica ovviamente non è mai possibile e l'unica strada percorribile per valutare la bontà del modello è quella dell'analisi dei residui.
(3) Nelle reti RBF la capacità di generalizzazione è legata non solo al numero di nodi nascosti ma anche al valore del parametro di scala σ_k. E' interessante provare a simulare situazioni limite di overfitting in cui il numero di nodi nascosti è vicino al numero di esempi del training set (con centri dati dai vettori di input stessi) e i parametri σ_k assumono valori così piccoli da consentire al neurone di attivarsi solo per l'osservazione esattamente corrispondente al suo centro.

Costruzione della rete
Matlab offre un interessante metodo per la costruzione delle reti radial basis. Si tratta di un metodo iterativo, che cioè avviene per passi successivi.

(1) Supponiamo di avere un campione formato da N casi, su ciascuno dei quali vengano osservate p variabili indipendenti (X ha quindi dimensioni N x p). L'algoritmo parte con una rete priva di neuroni nello strato nascosto e stima per ognuna delle N unità campionarie (quindi per ognuna delle righe di X) la previsione di y della rete (**`yhat`**), calcolando poi gli N errori di previsione commessi.

(2) A questo punto l'algoritmo trova il massimo di questi errori. Supponiamo che questo errore massimo avvenga in corrispondenza della i-esima osservazione X_i. L'algoritmo aggiunge quindi un neurone nello strato nascosto ponendo X_i come suo parametro di localizzazione.

(4) Vengono poi ristimati i pesi delle connessioni fra strato nascosto ed output (mediante minimizzazione dell'errore quadratico medio).

(5) I punti (2) e (3) vengono ripetuti finche' l'errore complessivo commesso dalla rete scende al di sotto della quantità definita con **`goal`** oppure finchè il numero massimo di neuroni **`MaxNeu`** è stato raggiunto.