Corso di Computational Mathematics for Learning and Data Analysis 2018-2019 Report Progetto n°2

Rudy Semola
Michele Fontana

January 21, 2019

Titolo

Sottotitolo

Abstract

L'ottimizzazione è al centro di quasi tutte le tecniche dell'apprendimento automatico e l'implementazione degli algoritmi risolutori necessita spesso di numerose tecniche e concetti avanzati di algebra lineare.

In questa relazione viene riportato il lavoro svolto per implementare differenti algoritmi di ottimizzazione in un contesto applicativo tipico del Machine Learning. Partendo da una Rete Neurale si esegue un'analisi comparativa tra la backpropagation con stepsize fissato, backpropagation con stepsize calcolato attraverso l' Armijo-Wolfe Line Search, infine due algoritmi della classe dei metodi quasi-newton a memoria limitata. Questi algoritmi verranno impiegati in modo differente per addestrare la Rete Neurale. Partendo da un'introduzione al problema in cui tali algoritmi sono impiegati e il dualismo presente con l'ottimizzazione, si prosegue illustrando il formalismo e la notazione matematica adottata. Si discutono i più importanti risultati teorici utilizzati come prime motivazioni delle scelte adottate. I vari risultati sperimentali ed i metodi di risoluzione adottati verranno mostrati e discussi utilizzando un approccio critico. Le conclusioni riportano infine una sintesi finale dei risultati ottenuti.

1 Introduzione e Formulazione Matematica

Introduzione

Il task sotto esame è classificabile come un problema di regressione non lineare. Per risolverlo si utilizza una tecnica di apprendimento di tipo supervisionato. Nello specifico, il modello utilizzato, oggetto di prove sperimentali, è una Feedforward NN costituita da un unico hidden layer e da due neuroni di output.

Le Feedforward Neural Network (NN) richiedono un processo di apprendimento per mappare un insieme di input x ad un insieme di output y (in Figure 1 $x = i$). Mediante tale processo si determina il valore ottimo dei pesi w da assegnare ad ogni singolo collegamento presente nella rete, in modo da minimizzare la funzione loss.

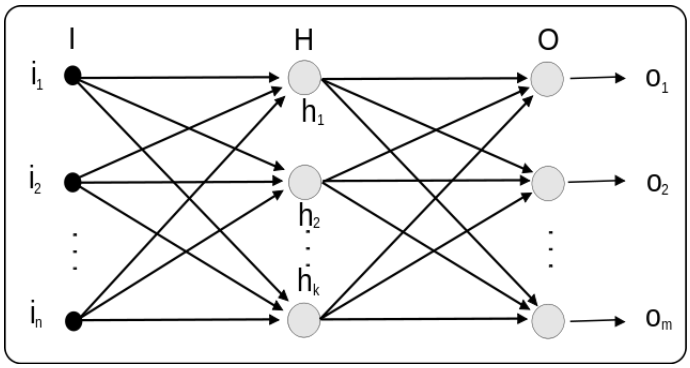

Figure 1: Illustrazione di una generica feedforward NN con un singolo hidden layer

In Figure 1 ne possiamo vedere la struttura grafica e seguire pertanto il dualismo matematico; il data set $\mathbf{X}$ (oppure input layer I in Figure 1), hidden layer $\mathbf{H}$ e l'output layer $\mathbf{O}$ sono noti quando anche il target $\mathbf{Y}$ (alle volte chiamato anche T) è noto. Abbiamo inoltre $\mathbf{W^L}$, che sono i pesi del livello L e la funzione di attivazione di un neurone per ogni livello $\mathbf{f^L}$. Nel nostro caso abbiamo $L = 1, 2$.

Formalismo matematico adottato

Dal punto di vista matematico (di maggior interesse) vengono visti come matrici cui indici e dimensioni si riportano a seguire. Abbiamo p numero di esempi (alle volte chiamati anche pattern) e n numero di features,

$$\mathbf{X}_{p\times n+1} = \begin{bmatrix} x_{11} & \cdot & \cdot & x_{1n+1} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_{p1} & \cdot & \cdot & x_{pn+1} \end{bmatrix}$$

k numero di neuroni nel hidden layer, m numero di neuroni nell'output layer,

$$\mathbf{H}_{p\times k} = \begin{bmatrix} h_{11} & \cdot & h_{1k} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ h_{p1} & \cdot & h_{pk} \end{bmatrix}$$

$$\mathbf{O}_{p\times m} = \begin{bmatrix} o_{11} & \cdot & o_{1m} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ o_{p1} & \cdot & o_{pm} \end{bmatrix}$$

i pesi presentano come indici h per indicare che è la matrice del hidden layer e *out* dell'output layer

$$\mathbf{W}^{L=1=h}_{k\times n+1} = \begin{bmatrix} w^h_{11} & \cdot & w^h_{1n+1} \\ \cdot & \cdot & \cdot \\ w^h_{k1} & \cdot & w^h_{kn+1} \end{bmatrix}$$

$$\mathbf{W}^{L=2=out}_{m\times k+1} = \begin{bmatrix} w^{out}_{11} & \cdot & w^{out}_{1k+1} \\ \cdot & \cdot & \cdot \\ w^{out}_{m1} & \cdot & w^{out}_{mk+1} \end{bmatrix}$$

Si presentano matrici in cui è stata aggiunta una colonna per inserire gli elementi del *bias* importante per gli scopi del Macchine Learning.

Per completezza si riporta anche la matrice dei target con dimensione analoga a O e le funzioni di attivazione per i neuroni del hidden layer $f^{L=h} = tanh(z)$ che è una tangente iperbolica e dell'output layer che è una semplice funzione lineare.

$$\mathbf{Y}_{p\times m} = \begin{bmatrix} y_{11} & \cdot & y_{1m} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ y_{p1} & \cdot & y_{pm} \end{bmatrix}$$

Spesso le notazioni possono aumentare ai fini dell'esposizione ad esempio per rendere le matrici sopra citate a blocchi; nel corso del testo verranno esplicitamente menzionati.

Funzione obiettivo e problema ottimizzazione

Possiamo vedere il problema dell' apprendimento sopra citato come un problema non lineare di ottimizzazione non vincolata.

In particolare, tale processo di learning si può formalizzare come un algoritmo che vuole risolvere il seguente problema:

$$min\left\{E(\mathbf{w}), \mathbf{w} \in R^j\right\}$$

con j numero di pesi totali della rete incluso i bias; come sopra menzionato, i pesi avranno una rappresentazione matriciale per ogni livello per facilitare le operazioni computazionali.

La funzione obiettivo *E(*$\mathbf{w}$*)* in Macchine Learning viene comunemente chiamata *Loss*; in forma già matriciale adottata poi nei calcoli abbiamo:

$$E(\mathbf{w}) = \frac{1}{2}\left[\frac{1}{p}\left\|\mathbf{Y}-\mathbf{O}\right\|_F^2 + \lambda\left(\left\|\mathbf{W^h}\right\|_F^2 + \left\|\mathbf{W^{out}}\right\|_F^2\right)\right]$$

la norma adottata è $\|\cdot\|$ quella di *Frobenius*, λ è il coefficiente di regolarizzazione e con $\Omega(\mathbf{w}) = \left\|\mathbf{W^h}\right\|_F^2 + \left\|\mathbf{W^{out}}\right\|_F^2$ la funzione di regolarizzazione.

Tutte le proprietà del particolare problema verranno mostrati in seguito; comunque vale la pena evidenziare fin da subito che la funzione obiettivo è differenziabile e limitata inferiormente almeno a 0 (infatti è una somma di norme moltiplicate per coefficienti diversi da 0 e positivi). Da notare inoltre che se il coefficiente di regolarizzazione viene impostato a 0, allora il secondo contributo della somma che in Macchine learning prende il nome di termine di penalità si annulla.

Osservazioni sul calcolo Gradiente ed approssimata Hessiana

Definendo la funzione obiettivo come sopra, il calcolo del gradiente di questa funzione può non essere banale (nel successivo capitolo verrà data una motivazione più esauriente scomponendola nelle sue parti); tuttavia tramite l'algoritmo della backpropagation il calcolo delle derivate parziali rispetto ai pesi $\mathbf{w}$ su una rete neurale può essere calcolato in modo ormai definito standard. Tutto questo ci ha suggerito di determinare il gradiente della funzione obiettivo con tale tecnica sia per i primi due algoritmi (da cui prendono il nome) confrontati assieme ai metodi *Quasi-Newton a memoria limitata*; infatti quest'ultima classe di algoritmi lascia liberi riguardo il calcolo del gradiente mettendo enfasi invece sulla determinazione del Hessiana approssimata.

Il gradiente con le sue derivate parziali può essere visto così come L matrici di dimensioni pari alle matrici dei pesi corrispondenti; nel nostro esempio avremo quindi due matrici, una per il livello hidden $\nabla\mathbf{E}^{L=h}(\mathbf{w})$ ed una per il livello finale $\nabla\mathbf{E}^{L=out}(\mathbf{w})$; ogni posizione di un elemento, ovvero la derivata parziale $\frac{\partial E^L(\mathbf{w})}{\partial w_{ir}^L}$ corrisponde alla posizione che l'elemento w_{ir}^L.

Le due matrici hanno quindi questa forma:

$$\mathbf{G}^{h}_{k\times n+1} = \nabla\mathbf{E}^{L=1=h}_{k\times n+1} = \begin{bmatrix} \frac{\partial E}{\partial w}{}^{h}_{11} & \cdot & \frac{\partial E}{\partial w}{}^{h}_{1n+1} \\ \cdot & \cdot & \cdot \\ \frac{\partial E}{\partial w}{}^{h}_{k1} & \cdot & \frac{\partial E}{\partial w}{}^{h}_{kn+1} \end{bmatrix}$$

$$\mathbf{G}^{out}_{m\times k+1} = \nabla\mathbf{E}^{L=2=out}_{m\times k+1} = \begin{bmatrix} \frac{\partial E}{\partial w}{}^{out}_{11} & \cdot & \frac{\partial E}{\partial w}{}^{out}_{1n+1} \\ \cdot & \cdot & \cdot \\ \frac{\partial E}{\partial w}{}^{out}_{k1} & \cdot & \frac{\partial E}{\partial w}{}^{out}_{kn+1} \end{bmatrix}$$

Ogni dettaglio rilevante ai fini dei risultati teorici utilizzati per le scelte implementative, verranno discusse nei successivi capitoli.

2 Risultati Teorici

TODO...

3 Metodi di Soluzione Implementati

TODO... Vengono riportati dettagli problema ML per AN Vengono riportati gli iperparametri ML e la configurazione scelta come modello in cui vengono fatti gli esperimenti di CM Scelte implementative e loro motivazione critica

4 Risultati sperimentali

TODO...

Conclusioni

TODO...