

Reti Neurali (Parte II)

Corso di AA, anno 2018/19, Padova



Fabio Aioli

12 Novembre 2018



- L'algoritmo **Perceptron** trova un vettore dei pesi (iperpiano) capace di separare i dati (se questi sono linearmente separabili)
- La **regola Delta** è una regola di aggiornamento dei pesi diversa da quella del Perceptron che ci permette di ottenere una soluzione che approssima "al meglio" il concetto target
- In particolare, essa usa la **discesa di gradiente** per esplorare lo spazio delle ipotesi e selezionare l'ipotesi che approssima meglio il concetto target (minimizzando una funzione di errore di apprendimento, opportunamente definita)



Consideriamo un Perceptron **SENZA** la hard-threshold (attivazione lineare):

$$o(\mathbf{x}) = \sum_{i=0}^n w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

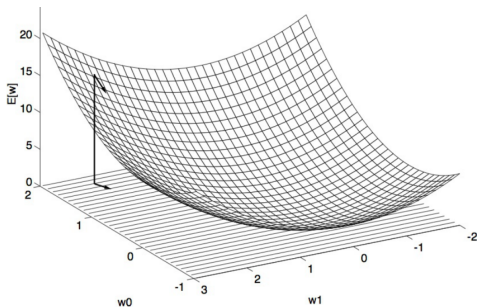
e definiamo una misura dell'errore commesso (scarto quadratico medio) da un particolare vettore dei pesi \mathbf{w} come:

$$E[\mathbf{w}] = \frac{1}{2N} \sum_{(\mathbf{x}^{(s)}, t^{(s)}) \in S} (t^{(s)} - o(\mathbf{x}^{(s)}))^2$$

dove N è la cardinalità dell'insieme di apprendimento S

$\forall (\mathbf{x}^{(s)}, t^{(s)}) \in S, o(\mathbf{x}^{(s)}) = t^{(s)} \Rightarrow E[\mathbf{w}] = 0$ quindi... bisogna **MINIMIZZARE** $E[\mathbf{w}]$ rispetto a \mathbf{w} !

Minimizzazione con discesa di gradiente



Idea base: partire da un \mathbf{w} random e modificarlo nella direzione contraria al gradiente (che indica la direzione di crescita di $E[\mathbf{w}]$).

$$\nabla E[\mathbf{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right], \quad \Delta \mathbf{w} = -\eta \nabla E[\mathbf{w}], \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}.$$



$$\begin{aligned}\frac{\partial E}{\partial w_j} &= \frac{\partial}{\partial w_j} \left[\frac{1}{2N} \sum_{s=1}^N (t^{(s)} - o^{(s)})^2 \right] \\ &= \frac{1}{2N} \sum_{s=1}^N \frac{\partial}{\partial w_j} \left[(t^{(s)} - o^{(s)})^2 \right] \\ &= \frac{1}{2N} \sum_{s=1}^N 2(t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_j} \left[t^{(s)} - o^{(s)} \right] \\ &= \frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_j} \left[t^{(s)} - \mathbf{w} \cdot \mathbf{x}^{(s)} \right]\end{aligned}$$



$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_i} [t^{(s)} - \mathbf{w} \cdot \mathbf{x}^{(s)}] \\ &= \frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \left(-\frac{\partial}{\partial w_i} [\mathbf{w} \cdot \mathbf{x}^{(s)}] \right) \\ &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_i} \left[\sum_{j=1}^n w_j x_j^{(s)} \right] \\ &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) x_i^{(s)}\end{aligned}$$



Algoritmo con discesa di gradiente

Gradient-Descent(S, η):

Ogni esempio di apprendimento è una coppia (\mathbf{x}, t) dove \mathbf{x} è il vettore di valori in input e t è il valore desiderato in output (target). η è il coefficiente di apprendimento (che ingloba il termine costante $1/N$)

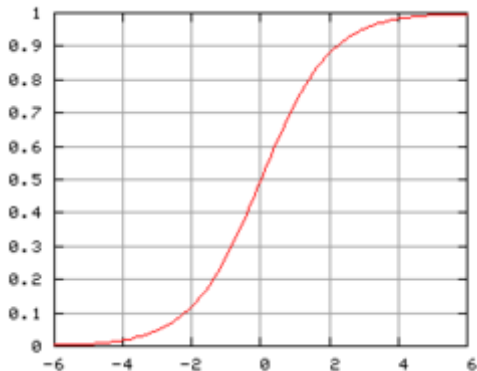
- 1 Assegna a w_i valori piccoli random
- 2 Finché la condizione di terminazione non è verificata:
 - (a) $\Delta w_i \leftarrow 0$
 - (b) Per ogni $(\mathbf{x}, t) \in S$:
 - (i) Presenta \mathbf{x} al neurone e calcola l'output $o(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
 - (ii) Per ogni $i \in \{1, \dots, n\}$:

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)\mathbf{x}_i$$

- (c) Per ogni $i \in \{1, \dots, n\}$:

$$w_i \leftarrow w_i + \Delta w_i$$

Attivazione con sigmoide





Attivazione con sigmoide

Consideriamo ora un Perceptron con **funzione di attivazione sigmoide**:

$$o(\mathbf{x}) = \sigma\left(\sum_{i=0}^n w_i x_i\right) = \sigma(\mathbf{w} \cdot \mathbf{x}) = \sigma(y)$$

dove $y = \mathbf{w} \cdot \mathbf{x}$ and $\sigma(y) = \frac{1}{1+e^{-y}}$.

Seguiamo lo stesso approccio di prima, ovvero vogliamo trovare il vettore dei pesi che minimizza lo scarto quadratico medio sul training set utilizzando un algoritmo basato su discesa di gradiente.

A questo scopo si nota intanto che per la funzione $\sigma()$ definita sopra vale la seguente relazione (verificare per esercizio):

$$\frac{\partial \sigma(y)}{\partial y} = \sigma(y)(1 - \sigma(y))$$

Calcolo del gradiente (attivazione sigmoidale)



$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \left[\frac{1}{2N} \sum_{s=1}^N (t^{(s)} - o^{(s)})^2 \right] \\ &= \frac{1}{2N} \sum_{s=1}^N \frac{\partial}{\partial w_i} \left[(t^{(s)} - o^{(s)})^2 \right] \\ &= \frac{1}{2N} \sum_{s=1}^N 2(t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_i} \left[t^{(s)} - o^{(s)} \right] \\ &= \frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_i} \left[t^{(s)} - \sigma(\mathbf{y}^{(s)}) \right] \\ &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_i} \left[\sigma(\mathbf{y}^{(s)}) \right]\end{aligned}$$

Calcolo del gradiente (attivazione sigmoidale)



$$\begin{aligned}\frac{\partial E}{\partial w_i} &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial}{\partial w_i} [\sigma(\mathbf{y}^{(s)})] \\ &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial \sigma(y^{(s)})}{\partial y^{(s)}} \frac{\partial y^{(s)}}{\partial w_i} \\ &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \frac{\partial \sigma(y^{(s)})}{\partial y^{(s)}} \frac{\partial}{\partial w_i} [\mathbf{w} \cdot \mathbf{x}^{(s)}] \\ &= -\frac{1}{N} \sum_{s=1}^N (t^{(s)} - o^{(s)}) \sigma(y^{(s)}) (1 - \sigma(y^{(s)})) x_i^{(s)}\end{aligned}$$

Quindi il punto 2.b.ii dell'algoritmo dato sopra ora diventa:

$$\Delta w_i \leftarrow \Delta w_i + \eta (t - o) \sigma(y) (1 - \sigma(y)) x_i$$



Nozioni

- Gradiente e minimizzazione di funzioni vettoriali
- La regola Delta
- Ottimizzazione con attivazione lineare
- Ottimizzazione con attivazione sigmoideale

Esercizi

- Dimostrare $\sigma'(y) = \sigma(y)(1 - \sigma(y))$
- Implementare ottimizzazione regola Delta con attivazione lineare/sigmoideale