

Metoda gradient descent

Modelarea coeficienților β :

la iterația 0: valori random (sau 0)

la iterația $t+1$ ($t=0,1,2,\dots$)

$$\beta_k(t+1) = \beta_k(t) - \text{learning_rate} * \text{error}(t) * x_k, \quad k=1,2,\dots,d$$

$$\beta_0(t+1) = \beta_0(t) - \text{learning_rate} * \text{error}(t)$$

Unde

$$\text{error}(t) = \text{computed} - \text{realOutput}$$

$$\text{error}(t) = \beta_0(t) + \beta_1(t)*x_1 + \beta_2(t)*x_2 + \dots + \beta_d(t)*x_d - y$$

BATCH gradient descent:

Eroarea se calculează pentru fiecare exemplu de antrenament

Modelul se updatează după ce toate exemplele de antrenament au fost evaluate (la finalul unei epoci)

Algoritm evolutiv

Initializare populație $P(0)$

Evaluare $P(0)$

$g := 0$;

Cât timp (not condiție_stop) execută

Repetă

 Selectează 2 părinți p_1 și p_2 din $P(g)$

 Încrucișare(p_1, p_2) $\Rightarrow o_1$ și o_2

 Mutație(o_1) $\Rightarrow o_1^*$

 Mutație(o_2) $\Rightarrow o_2^*$

 Evaluare(o_1^*)

 Evaluare(o_2^*)

 adăugare o_1^* și o_2^* în $P(g+1)$

Până când $P(g+1)$ este completă

$g := g + 1$

Sf Cât timp

Algoritm neural network

Se inițializează ponderile

 Cât timp nu este îndeplinită condiția de oprire

 Pentru fiecare exemplu

 Se activează fiecare neuron al rețelei

 Se propagă informația înainte și se calculează ieșirea

corespunzătoare fiecărui neuron al rețelei

 Se ajustează ponderile

 Se stabilește și se propagă eroarea înapoi

 Se stabilesc erorile corespunzătoare neuronilor din stratul de ieșire

 Se modifică ponderile între nodurile de pe stratul ascuns și stratul de ieșire

 Se propagă erorile nodurilor de pe stratul de ieșire înapoi în toată rețeaua \rightarrow se distribuie erorile pe toate conexiunile existente în rețea proporțional cu valorile ponderilor asociate acestor conexiuni

 Se modifică ponderile între nodurile de pe stratul de intrare și stratul ascuns