W05

Fekete Máté

2020 Április

Tartalomjegyzék

1 Neurális hálózatok - költség függvény

Mivel egy neurális hálózatban több kimeneti neuronunk is lehet ($h_\theta(x)_k$ jelöli a k-adik kimenetet), így változtatnunk kell logisztikus regressziónál használt költség függvényen.
Az eddig használt:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)}))+(1-y^{(i)})log(1-h_\theta(x^{(i)}))]+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Módosítva, ami már működik neurális hálózaton:

Jelölések:
L - a rétegek száma a hálózatban
S_l - az l-edik rétegben lévő neuronok száma (az eltoláson kívül)
K - a kimeneti neuronok/osztályok száma

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}[y_k^{(i)}log((h_\theta(x^{(i)}))_k)+(1-y_k^{(i)})log(1-(h_\theta(x^{(i)}))_k)]+$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{s_l+1}(\theta_{j,i}^{(l)})^2$$

2 Visszaterjesztés (backpropagation)

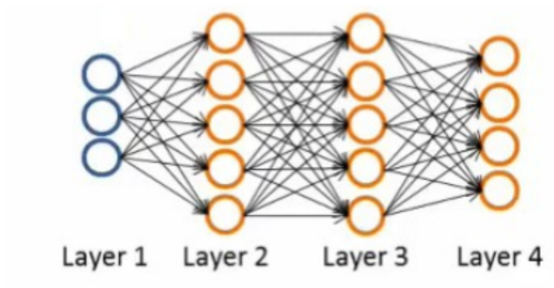
A visszaterjesztés a neurális hálók körében a költség függvény minimalizálását jelenti, a gradiens módszerhez hasonlóan a célunk most is a következő kiszámítása:

$$min_\theta J(\theta)$$

Tehát a költség függvény minimalizálása, az optimális θ paraméterek megadásával.
A függvény gradiensének kiszámítása:

$$\frac{\partial}{\partial\theta_{i,j}^{(l)}}J(\theta)$$

Ehhez a következő algoritmust használjuk:



Adott egy tanuló halmaz: $\{(x^{(1)}, y(1)) \ldots (x^{(m)}, y^{(m)})\}$
$\Delta_{i,j}^{(l)} := 0$ minden l,i,j értékre (tehát ez jelenleg egy teli 0 mátrix)

t=1-től m-ig minden tanuló példára:

1. $a^{(1)} := x^{(t)}$

2. Számoljuk ki $a^{(l)}$-t l=2,3,...,L
   $a^{(1)} = x$
   $z^{(2)} = \Theta^{(1)} a^{(1)}$
   $a^{(2)} = g(z^{(2)})$
   $z^{(3)} = \Theta^{(2)} a^{(2)}$
   $a^{(3)} = g(z^{(3)})$
   $z^{(4)} = \Theta^{(3)} a^{(3)}$
   $a^{(4)} = g(z^{(4)}) = h_\Theta(x)$

3. $\delta_j^{(l)}$ az l-edik réteg j-edik neuronjából származó hibát jelenti.
   $y^{(t)}$ segítségével $\delta^{(L)} = a^{(L)} - y^{(t)}$
   ahol L a rétegek száma és $a^{(L)}$ a kimeneti réteg, így az L-edik réteg hibaértéke pont a várt és a kapott értékek különbsége lesz.
   Innentől visszafele tudjuk a δ értékeket számolni:

4. $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ kiszámolása,
   a $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}). * a^{(l)}. * (1 - a^{(l)})$ képlet segítségével,
   tehát egy réteg hibaértékeit úgy számoljuk ki, hogy a következő réteg hibaértékeit megszorozzuk a jelenlegi Θ mátrixával, majd elemenként megszorozzuk a $[a^{(l)}. * (1 - a^{(l)})]$ taggal, ami a g függvényünk deriváltja.

5. $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ vagy vektorizálva $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$
A D mátrixban pedig összegyújtük az értékeket, ebben fogjuk megkapni a gradienst.

   - $D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)} + \lambda \Theta_{i,j}^{(l)}$ ha $j \neq 0$
   - $D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)}$ ha j = 0

   Tehát a tanuló példák végére érve $\frac{\partial}{\partial \theta_{i,j}^{(l)}} J(\theta) = D_{i,j}^{(l)}$