Metody Inżynieri Wiedzy
15/12/2020

Wsteczna propagacja

Teoria

Wsteczna propagacja to algorytm uczenia nadzorowanego jednokierunkowych sieci neuronowych. Polega na znalezieniu gradientu funkcji błędu względem wag sieci i przesunięciu ich w kierunku największego spadku, zaczynając od ostatniej warstwy.

Dla uproszczenia, na początku wyprowadzę wzory dla pojedyńczego neuronu wyjściowego.

Neuron wyjściowy

Oznaczenia dla *j*-tego neuronu z bierzącej warstwy:

w_{jk} - waga dla k-tego neuronu z poprzedniej warstwy
b_j - wartość odchylenia
net_j - wejście do neuronu, obliczone na podstawie wyjść poprzedniej warstwy
out_j - wyjście neuronu z bierzącej warstwy
$o\bar{u}t_j$ - wyjście neuronu z poprzedniej warstwy
t_j - poprawna wartość wyjścia
J - zbiór indeksów neuronów bierzącej warstwy
K - zbiór indeksów neuronów poprzedniej warstwy

Funkcja aktywacji i jej pochodna:

$$\phi(x)=\frac{1}{(1+e^{-x})}$$

$$\phi'(x)=\left(\frac{1}{(1+e^{-x})}\right)'=\frac{-1}{(1+e^{-x})^2}(1+e^{-x})'=\frac{e^{-x}}{(1+e^{-x})^2}=\left(\frac{1}{(1+e^{-x})}\right)\left(1-\frac{1}{(1+e^{-x})}\right)=\phi(x)(1-\phi(x))$$

Podstawowe wzory:

$$net_j=\left(\sum_{k\in K} w_{jk}\, o\bar{u}t_k\right)+b_j$$

$$out_j=\phi(net_j)=\frac{1}{(1+e^{-net_j})}$$

Funkcja błędu:

$$E=\frac{1}{2}\sum_{j\in J}(t_j-out_j)^2$$

Celem algorytmu jest minimalizacja funkcji E przez odpowiednie dopasowanie wartości w_{jk} i b_j.

Do określenia, jaki wpływ na E mają zmiany tych wartości, trzeba znaleść $\frac{\partial E}{\partial w_{ik}}$ oraz $\frac{\partial E}{\partial b_k}$.

Ponieważ E jest różniczkowalną funkcją out_j, out_j jest różniczkowalną funkcją net_j, oraz net_j jest różniczkowalną funkcją w_{jk} i b_j, to można tu zastosować wzór na pochodną funkcji złożonej:

$$\frac{\partial E}{\partial w_{jk}}=\frac{\partial E}{\partial out_j}\frac{\partial out_j}{\partial net_j}\frac{\partial net_j}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial b_j}=\frac{\partial E}{\partial out_j}\frac{\partial out_j}{\partial net_j}\frac{\partial net_j}{\partial b_j}$$

Każdy z tych czynników jest już łatwy do policzenia. Warto zauważyć, że w obydwu sumach tylko jeden składnik jest zależny od zmiennej po której różniczkujemy, pozostałe można pominąć.

$$\frac{\partial E}{\partial out_j}=\frac{\partial}{\partial out_j}\left(\frac{1}{2}\sum_{i\in J}(t_i-out_i)^2\right)=\frac{\partial}{\partial out_j}\left(\frac{1}{2}(t_j-out_j)^2\right)=(out_j-t_j)$$

$$\frac{\partial out_j}{\partial net_j}=\frac{\partial}{\partial net_j}(\phi(net_j))=\phi'(net_j)=\phi(net_j)(1-\phi(net_j))=out_j(1-out_j)$$

$$\frac{\partial net_j}{\partial w_{jk}}=\frac{\partial}{\partial w_{jk}}\left((\sum_{i\in K} w_{ji}\bar{out}_i)+b_j\right)=\frac{\partial}{\partial w_{jk}}(w_{jk}\bar{out}_k+b_j)=\bar{out}_k$$

$$\frac{\partial net_j}{\partial b_j}=\frac{\partial}{\partial b_j}\left((\sum_{i\in K} w_{ji}\bar{out}_i)+b_j\right)=\frac{\partial}{\partial b_j}(b_j)=1$$

Po podstawieniu otrzymujemy:

$$\frac{\partial E}{\partial w_{jk}}=(out_j-t_j)out_j(1-out_j)\bar{out}_k$$

$$\frac{\partial E}{\partial b_j}=(out_j-t_j)out_j(1-out_j)$$

Dla uproszczenia zapisu wprowadzę następujące oznaczenie:

$$\sigma_j:=\frac{\partial E}{\partial out_j}\frac{\partial out_j}{\partial net_j}=(out_j-t_j)out_j(1-out_j)$$

Ostatecznie:

$$\frac{\partial E}{\partial w_{jk}}=\sigma_j\bar{out}_k$$

$$\frac{\partial E}{\partial b_j}=\sigma_j$$

Więc żeby zmniejszyć wartość funkcji E, wartości w_{jk} i b_j zmieniają się o:

$$\Delta w_{jk}=-\mu\sigma_j\bar{out}_k$$
$$\Delta b_j=-\mu\sigma_j$$

gdzie μ to stała określająca szybkość uczenia się.

Neuron wewnętrzny

Wprowadźmy dodatkowe oznaczenia. Wielkości z daszkiem $(\text{np. } \hat{w}_{jk},\ o\hat{u}t_j)$ będą odnosiły sie do analogicznych wartości z następnej warstwy neuronów. *L* to zbiór indeksów następnej warstwy neuronów.

W przypadku neuronu wewnętrznego zmienia się tylko jeden czynnik: $\frac{\partial E}{\partial out_j}$.

E nie można obliczyć już bezpośrednio z out_j jak przy neuronie wyjściowym. Teraz sygnał out_j wpływa na następną warstwę neuronów, które z kolei wpływają na *E*. A więc funkcję *E* można potraktować jako funkcję wielu zmiennych $E(o\hat{u}t_1, o\hat{u}t_2, \ldots, o\hat{u}t_l)$, w której każdy z argumentów jest zależny od out_j. Korzystając ze wzoru na pochodną funkcji złożonej dla funkcji wielu zmiennych otrzymujemy:

$$\frac{\partial E}{\partial out_j} = \sum_{l \in L} \frac{\partial E}{\partial o\hat{u}t_l} \frac{\partial o\hat{u}t_l}{\partial out_j}$$

Podobnie jak wcześniej, $o\hat{u}t_l$ jest różniczkowalną funkcją $n\hat{e}t_l$, a $n\hat{e}t_l$ jest różniczkowalną funkcją out_j, więc można zastosować wzór na pochodną funkcji złożonej.

Dla każdego $l \in L$ mamy:

$$\frac{\partial E}{\partial o\hat{u}t_l} \frac{\partial o\hat{u}t_l}{\partial out_j} = \frac{\partial E}{\partial o\hat{u}t_l} \frac{\partial o\hat{u}t_l}{\partial n\hat{e}t_l} \frac{\partial n\hat{e}t_l}{\partial out_j}$$

$$\frac{\partial E}{\partial o\hat{u}t_l} \frac{\partial o\hat{u}t_l}{\partial n\hat{e}t_j} = \hat{\sigma}_l$$

$$\frac{\partial n\hat{e}t_l}{\partial out_j} = \frac{\partial}{\partial out_j}\left(\left(\sum_{i \in L} \hat{w}_{li}\, out_i\right) + \hat{b}_l \right) = \frac{\partial}{\partial out_j}\left(\hat{w}_{lj}\, out_j + \hat{b}_l \right) = \hat{w}_{lj}$$

Ostatecznie:

$$\frac{\partial E}{\partial out_j} = \sum_{l \in L} \hat{\sigma}_l \hat{w}_{jl}$$

Co pociąga za sobą zmianę wzoru:

$$\sigma_j = \frac{\partial E}{\partial out_j} \frac{\partial out_j}{\partial net_j} = \left(\sum_{l \in L} \hat{\sigma}_l \hat{w}_{jl}\right) out_j (1 - out_j)$$

Pozostałe wzory pozostają takie same.

$$\frac{\partial E}{\partial w_{jk}} = \sigma_j\, o\bar{u}t_k$$

$$\frac{\partial E}{\partial b_j} = \sigma_j$$

$$\Delta w_{jk} = -\mu \sigma_j\, o\bar{u}t_k$$
$$\Delta b_j = -\mu \sigma_j$$

Zapis macierzowy

W celu zoptymalizowania skryptu w Pythonie, trzeba zapisać powyższe wzory w postaci operacji na macierzach.

Oznaczenia:

$w=\begin{pmatrix} w_{11} & \dots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{j1} & \cdots & w_{jk} \end{pmatrix}$ - macierz wag

$b=\begin{pmatrix} b_1 & \dots & b_k \end{pmatrix}$ - wektor wartości odchylenia

$o=\begin{pmatrix} o_1 & \dots & o_j \end{pmatrix}$ - wektor wyjść neuronów

$t=\begin{pmatrix} t_1 & \dots & t_j \end{pmatrix}$ - wektor prawidłowych wartości wyjść

$A \circ B$ - iloczyn Hadamarda

Tak jak poprzednio, kreska nad symbolem oznacza poprzednią warstwę, a daszek - następną.

Wzory mają wtedy postać:

$$o=\phi(\bar{o}\, w^T+b)$$

$$\sigma=\begin{cases} (o-t)\circ o\circ(1-o) & \text{dla warstwy wyjściowej} \\ (\hat{\sigma}\, \hat{w})\circ o\circ(1-o) & \text{dla warstw wewnętrznych} \end{cases}$$

$$\Delta w=-\mu \sigma^T \bar{o}$$

$$\Delta b=-\mu \sigma$$