Wykład 2: Uczenie nadzorowane sieci neuronowych - I

Algorytmy uczenia sieci neuronowych

Na sposób działania sieci ma wpływ jej topologia oraz funkcjonowanie poszczególnych neuronów. Z reguły topologię sieci uznaje się za jej niezmienny składnik i aby wpływać na jej działanie modeluje się jedynie działanie poszczególnych neuronów, w szczególności zmieniając wartości wag je łączących.

Dwie metodologie uczenia:

- *nadzorowane (supervised)*:

  algorytm uczący adaptuje sieć stosownie do wymuszeń zewnętrznych, starając się możliwie wiernie (tj. z jak najmniejszym błędem) zrealizować zadaną funkcję;

- *nienadzorowane (unsupervised)*:

  neurony nie są "skrępowane" żadnym zewnętrznym kryterium (funkcją błedu), zadaniem sieci jest wykrycie pewnego "porządku" w danych podawanych na jej wejście (np. - w jednym z prostszych przypadków - analiza skupień przykładów w przestrzeni cech)

Z punktu widzenia Uczenia Maszynowego: w uczeniu nadzorowanym sieć "zna" przynależność do klas obiektów podawanych na jej wejście, w uczeniu nienadzorowanym informacja ta jest dla niej niedostępna.

Nadzorowane uczenie pojedynczego liniowego neuronu na przykładzie *reguły delta*.

Cel: nauczenie neuronu możliwie wiernej realizacji pewnego odwzorowania

$$z = F(\mathbf{x}) \tag{2.0}$$

o najczęściej nieznanej postaci analitycznej, danego w postaci zbioru uczącego, zawierającego przykłady w postaci par

$$\langle \mathbf{x}^j, z^j = F(\mathbf{x}^j) \rangle, \; j = 1 \ldots N \tag{2.0}$$

Różnica pomiędzy oczekiwaną wartością wyjścia neuronu z a i wartością otrzymaną y stanowi *błąd* popełniony przez neuron przy prezentacji j-tego przykładu

$$\delta^j = z^j - y^j \tag{2.0}$$

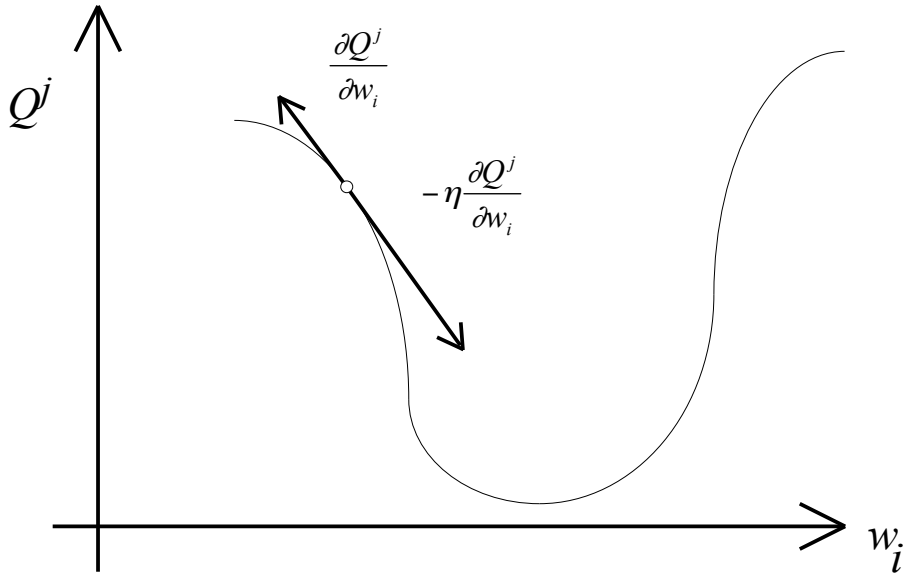
Zadaniem algorytmu uczącego jest minimalizacja tego błędu dla wszystkich przykładów zbioru uczącego. Błędy popełnione przez neuron dla poszczególnych przykładów da się zagregować w jedną wielkość przy pomocy *błędu średniokwadratowego:*

$$Q = \frac{1}{2}\sum_{j=1}^{N}\left(z^j - y^j\right)^2 = \sum_{j=1}^{N} Q^j, \quad Q^j = \frac{1}{2}\left(\delta^j\right)^2 \tag{2.0}$$

Metoda spadku gradientu

Posługując się metodą największego spadku gradientu (ang. *gradient descent*) dostajemy zależność pomiędzy modyfikacją *i*-tej wagi neuronu a zmianą wartości błędu przezeń popełnianego przy prezentacji *j*-tego przykładu:

$$\Delta w_i^j = -\eta \frac{\partial Q^j}{\partial w_i} \tag{2.0}$$



Funkcja Q^j jest funkcją złożoną, zatem jej pochodną cząstkową po w_i można przedstawić jako iloczyn dwóch pochodnych:

$$\frac{\partial Q^j}{\partial w_i} = \frac{\partial Q^j}{\partial y^j}\frac{\partial y^j}{\partial w_i} \tag{2.0}$$

co da się dalej rozwinąć (patrz wzory 2.3 i 2.4):

$$\frac{\partial Q^j}{\partial y^j} = -\left(z^j - y^j\right) = -\delta^j \tag{2.0}$$

oraz, pamiętając o liniowym charakterze funkcji $y=f(x)$:

$$\frac{\partial y^j}{\partial w_i} = x_i^j \tag{2.0}$$

Ostatecznie zatem, jedną z możliwości minimalizacji błędu popełnianego przez neuron jest modyfikacja jego wag przy prezentacji kolejnych przykładów zgodna z formułą:

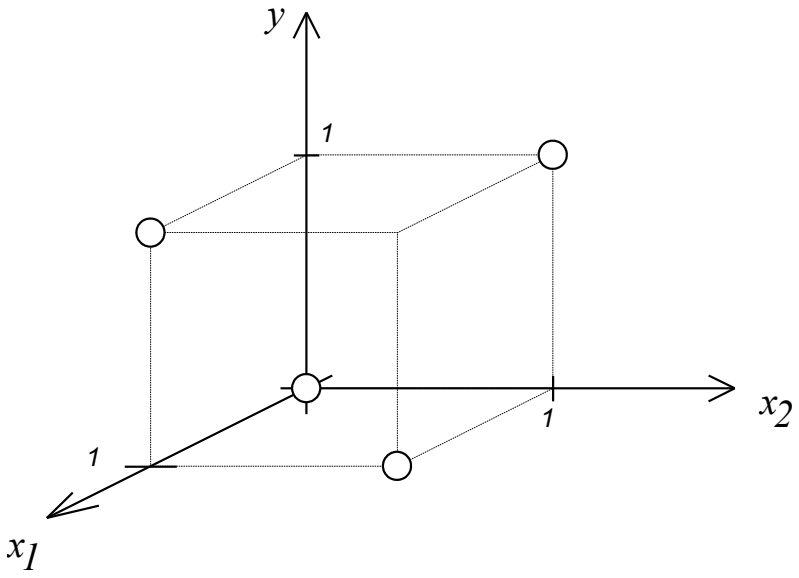
$$\Delta w_i^j = \eta \delta^j x_i^j \tag{2.0}$$

Jest to tzw. *reguła delta (delta rule)* , zaproponowana w [Widr60], stanowiąca jeden z pierwszych algorytmów uczących pojedynczy neuron.

Wady neuronu liniowego:

- jest w stanie skutecznie oddzielić od siebie tylko te klasy, które są *liniowo separowalne*; niestety, nawet proste problemy rzeczywiste do tej klasy nie należą, np.: problem różnicy symetrycznej (XOR); w szczególności dla przypadku dwuwymiarowego:

x1	x2	y=XOR(x1,x2)
0	0	0
0	1	1
1	0	1
1	1	0



Sieć złożona z neuronów liniowych

Tworzenie sieci wielowarstwowych z neuronów liniowych nie ma sensu: nie daje żadnej nowej jakości; kombinację dowolnej ilości warstwowo połączonych ze sobą neuronów liniowych da się zastąpić jednym neuronem liniowym:

$$y = f(e) = \sum(w_i x_i) = \sum\left(w_i \sum w_i x_i\right) = \ldots = \sum w_i' x_i$$

W związku z licznymi ograniczeniami neuron liniowy nie znajduje wielu zastosowań. Bardziej obiecujące okazały się neurony z nieliniową funkcją aktywacji.

Neuron nieliniowy

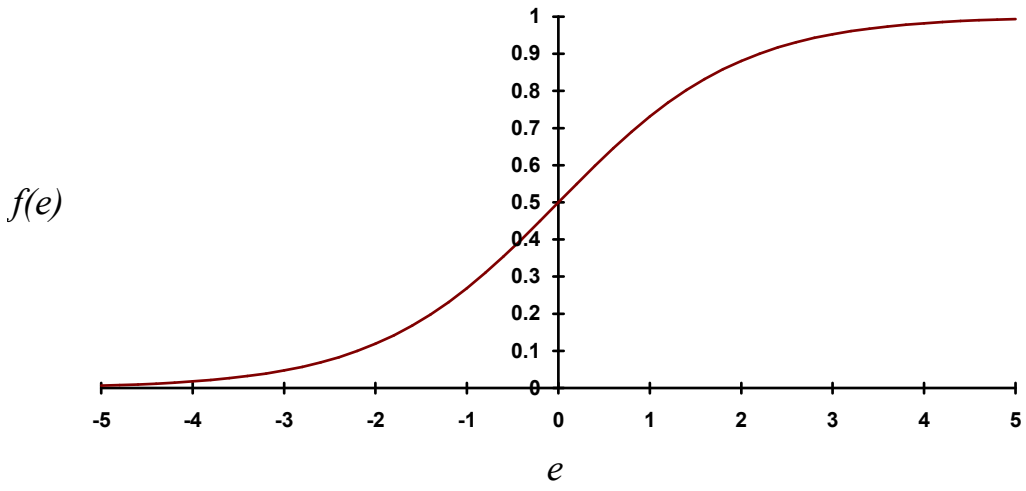
Funkcja przenosząca neuronu zwana jest często w tym przypadku funkcją "ściskającą" (*squeezing function*), jako że jej zadaniem jest odwzorować (w ogólności nieograniczony) zakres możliwych wartości pobudzenia neuronu e w ograniczony przedział, z reguły [0,1] lub [-1,1].

Najczęściej stosowane funkcje aktywacji neuronów nieliniowych:

- funkcja sigmoidalna

$$f(e) = \frac{1}{1+\exp(-2\beta e)}$$

β - współczynnik stromości



- tangens hiperboliczny

$$f(e) = \tanh(\beta e)$$

- funkcja progowa (skoku jednostkowego)

$$f(e) = \begin{cases} 1, & dla\ e \geq \Theta \\ 0, & dla\ e < \Theta \end{cases}$$

Uczenie pojedynczego nieliniowego neuronu

Podobnie jak w przypadku neuronu liniowego możemy wykorzystać metodę największego spadku gradientu (wzór (2.5)):

$$\Delta w_i^j = -\eta \frac{\partial Q^j}{\partial w_i} \tag{2.0}$$

Pochodną błędu popełnianego przez neuron da się przedstawić jako iloczyn dwóch pochodnych:

$$\frac{\partial Q^j}{\partial w_i} = \frac{\partial Q^j}{\partial y^j} \frac{\partial y^j}{\partial w_i} \tag{2.0}$$

Wartość pierwszego czynnika pozostaje bez zmian w porównaniu ze wzorem (2.7):

$$\frac{\partial Q^j}{\partial y^j} = -\delta^j \tag{2.0}$$

Natomiast prawy wyraz zmieni postać, z racji nieliniowości funkcji aktywacji; w ogólności:

$$\frac{\partial y^j}{\partial w_i} = \frac{\partial y^j}{\partial e} \frac{\partial e}{\partial w_i} \tag{2.0}$$

Oczywiście:

$$\frac{\partial e}{\partial w_i} = x_i^j \tag{2.0}$$

natomiast postać pochodnej $\frac{\partial y^j}{\partial e}$ zależy od zastosowanej funkcji aktywacji.

Zaletą stosowania funkcji sigmoidalnej jest dogodna postać jej pochodnej

$$f'(e) = f(e)(1 - f(e)) \tag{2.0}$$

Jak widać, da się ją wyrazić przy pomocy wartości funkcji aktywacji - upraszcza to znacznie obliczenia (podobnie jest w przypadku tangensa hiperbolicznego). W takim przypadku

$$\frac{\partial y^j}{\partial e} = y^j\left(1 - y^j\right) \tag{2.0}$$

Co ostatecznie prowadzi do następującej formuły określającej wartość modyfikacji i-tej wagi

$$\Delta w_i^j = \eta \delta^j x_i^j y^j\left(1 - y^j\right) \tag{2.0}$$

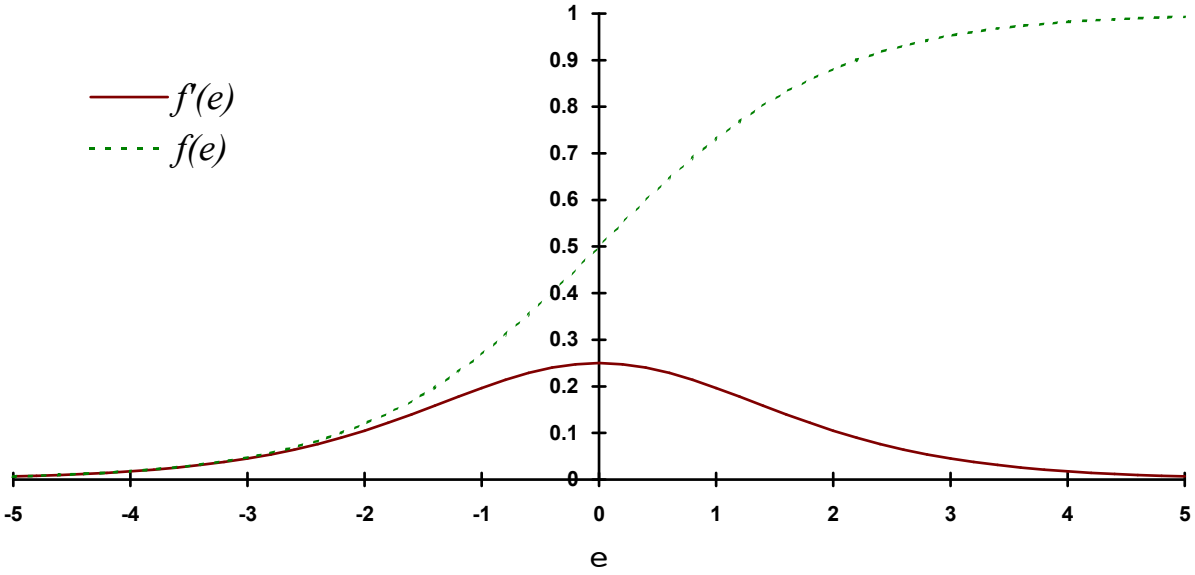
Analiza formuły definiującej modyfikację wagi

Formuła (2.17)

$$\Delta w_i^j = \eta \delta^j x_i^j y^j \left(1 - y^j\right)$$

mówi, że poprawka i-tej wagi jest proporcjonalna przez współczynnik prędkości uczenia η do:

- δ^j - błędu popełnianego przez neuron; neuron nie popełniający żadnego błędu nie będzie ulegał modyfikacjom
- x_i^j - sygnału podawanego na i-te wejście neuronu; synapsę "obarcza się winą" za błędne działanie neuronu w takim stopniu, w jakim przyczyniła się do jego pobudzenia
- $y^j\left(1-y^j\right)$ - pochodnej funkcji aktywacji; w przypadku funkcji sigmoidalnej ma ona następujący kształt:



W konsekwencji poprawki dokonywane przez formułę są silniejsze, gdy tzw. *punkt pracy* neuronu znajduje się blisko punktu przegięcia funkcji *f(e)*.