Wstęp do uczenia maszynowego

AdaLine i maszyna liniowa

Tomasz Derek

KMS

Listopad 20, 2019

Krótkie przypomnienie

Na ostatnich zajęciach omawialiśmy perceptron. Powiedzieliśmy sobie również o funkcji aktywacji, a także o problemie minimalizacji funkcji błędu. Model perceptronu uczyliśmy za pomocą jednego z trzech algorytmów: simple learning algorithm, pocket learning algorithm, pocket learning algorithm with ratchet.

Ogólny zapis perceptronu progowego

$$O(x_1, ..., x_n) = f(\sum_{i=1}^{n} x_i w_i + w_0)$$

lub też

$$O(x_1, ..., x_n) = f(\sum_{i=1}^{n} x_i w_i - \theta)$$

Funkcja progowa

$$f(x) = \begin{cases} -1 & x < 0 \\ +1 & x \geq 0 \end{cases}$$

AdaLine

AdaLine - Adaptive Linear Neuron

Główne różnice między Perceptronem a AdaLine

- Funkcja aktywacji
- Algorytm uczenia

Zapis AdaLine

$$O(x_1, ..., x_n) = \sum_{i=1}^{n} x_i w_i + w_0$$

lub też

$$O(x_1, ..., x_n) = \sum_{i=1}^{n} x_i w_i - \theta$$

Błąd

$$\varepsilon = d - s$$

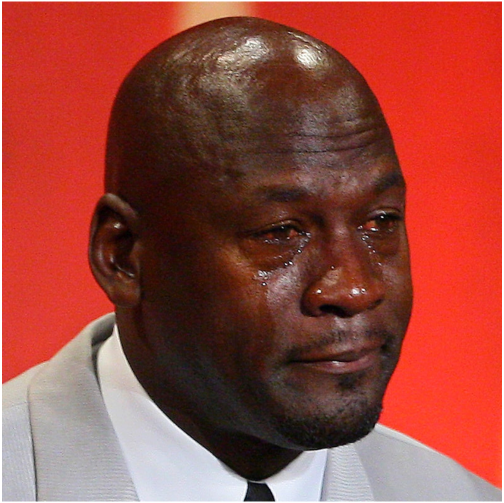
gdzie **d** oznacza nasz oczekiwany sygnał wyjściowy, a **s** oznacza wyjście naszej sieci

Dobór wag i problem minimalizacji błędu

$$E(w) = \frac{1}{2}\varepsilon^2 = \frac{1}{2}[d - s]^2 = \frac{1}{2}\left[d - (\sum_{i=1}^{n} x_i w_i - \theta)\right]^2$$

Dobór wag i problem minimalizacji błędu

WOW I CO TERAZ?



Algorytm spadku gradientowego

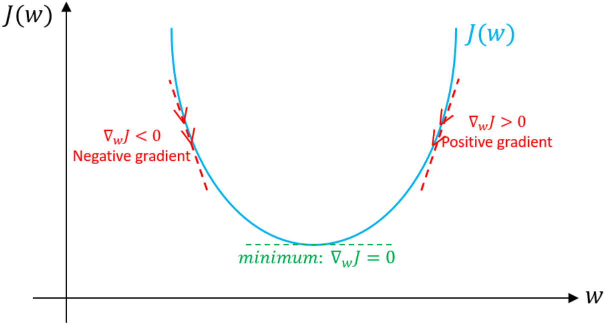
- Dana jest funkcja $f : R^n \to R$, która jest różniczkowalna
- Chcemy znaleźć jej minimum lokalne w sposób numeryczny

Pochodna cząstkowa

Pochodną cząstkową danej funkcji $f : R^n \to R$ po x_i określamy jako:

$$\frac{\partial f}{\partial x_i}(x_1, ..., x_n) = \lim_{h \to 0} \frac{f(x_1, ..., x_{i-1}, x_i + h, x_{i+1}, ..., x_n) - f(x_1, ..., x_n)}{h}$$

Dobór wag i problem minimalizacji błędu



Co to jest gradient?

Gradientem pewnej funkcji sklarnej $f(x_1, ..., x_n)$ w układzie współrzędnych kartezjańskich nazywamy wektor, którego składowymi są pochodne cząstkowe funkcji f.

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n}\right]$$

Innymi słowy gradient wskazuje w którą stronę i funkcja f rośnie w wybranym przez nas punkcie.

Algorytm spadku gradientowego

Z angielskiego *Gradient Descent Algorithm* w skrótcie GDA

- Chcemy znaleźć minimum funkcji f
- Obliczamy gradient pochodnych cząstkowych
- Robimy krok w przeciwnym kierunku

Algorytm uczenia AdaLine

Ze względu na fakt, że nasza funkcja jest różniczkowalna, zastosujemy do niej metodę największego spadku gradientu:

$$w_i = w_i - \eta \frac{\partial E(w_i)}{\partial w_i} = w_i + \eta(d - s)x_i$$

Możemy to zapisać w powyższy sposób ze względu na fakt, iż:

$$\frac{\partial E(w_i)}{\partial w_i} = \frac{\partial E(w_i)}{\partial s} * \frac{\partial s}{\partial w_i}$$

Ponieważ s jest funkcją liniową względem wektora wag, więc możemy zapisać:

$$\frac{\partial s}{\partial w_i} = x_i$$

Algorytm uczenia AdaLine c.d.

Ponadto:

$$\frac{\partial E(w_i)}{\partial s} = -(d - s)$$

$\delta = (d - s)$ Wtedy wzór przyjmuje postać:

$$w_i = w_i + \eta \delta x_i$$

Model neuronu sigmoidalnego

Model neuronu sigmoidalnego nie różni się dużo od modeli AdaLine czy też Perceptronu. Różnica polega na zastosowaniu innej funkcji aktywacji, a dokładniej funkcji sigmoidalnej. Model ten uczymy na zasadzie algorytmu największego spadku gradientu:

$$w_i = w_i - \eta \frac{\partial E(w_i)}{\partial w_i} = w_i + \eta(d - f(s))f'(s)x_i$$

Algorytm

- Losujemy wagi
- Losujemy przykład
- Obliczamy pobudzenie wagi
- Korygujemy wagi
- Kończymy przechodząc przez odpowiednią ilość iteracji. W przeciwnym wypadku wracamy do 2.