Neural Networks (Part II)

Machine Learning, A.Y. 2022/23, Padova

Fabio Aiolli

October 26th, 2022

The Delta rule

- The Perceptron algorithm finds a weight vector (hyperplane) capable to separate the data (iff they are linearly separable);
- The Delta rule is a weight update rule different from the Perceptron rule that allows to obtain a best-fit solution approximating the target concept;
- In particular, it exploits gradient descent to explore the hypothesis space and select the hypothesis that best approximates the target concept (by minimizing an error function, appropriately defined).

The Delta rule

Consider a perceptron WITHOUT hard threshold (linear activation):

$$o(\mathbf{x}) = \sum_{i=0}^{n} w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

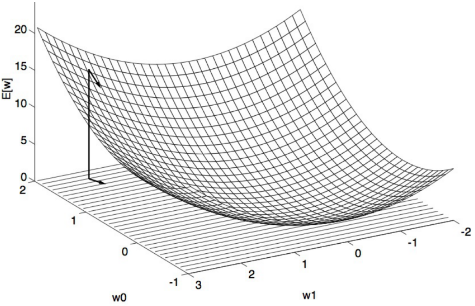
and let define a measure of the committed error (mean square error) given a specific weight vector $\mathbf{w}$ as:

$$E[\mathbf{w}] = \frac{1}{2N} \sum_{(\mathbf{x}^{(s)}, t^{(s)}) \in S} (t^{(s)} - o(\mathbf{x}^{(s)}))^2$$

where N is the cardinality of the training set S.

$\forall (\mathbf{x}^{(s)}, t^{(s)}) \in S, o(\mathbf{x}^{(s)}) = t^{(s)} \Rightarrow E[\mathbf{w}] = 0$ then... we need to MINIMIZE $E[\mathbf{w}]$ with respect to the parameters $\mathbf{w}$!

Minimization with gradient descent



Basic idea: start from a random **w** and update it in the opposite direction of the gradient (that indicates the direction of maximal increase of $E[\mathbf{w}]$).

$$\nabla E[\mathbf{w}] \equiv [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}], \quad \Delta\mathbf{w} = -\eta \nabla E[\mathbf{w}], \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}.$$

Gradient computation (linear activation)

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\left[\frac{1}{2N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})^2\right] \\
&= \frac{1}{2N}\sum_{s=1}^{N}\frac{\partial}{\partial w_i}\left[(t^{(s)} - o^{(s)})^2\right] \\
&= \frac{1}{2N}\sum_{s=1}^{N}2(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[t^{(s)} - o^{(s)}\right] \\
&= \frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[t^{(s)} - \mathbf{w}\cdot\mathbf{x}^{(s)}\right]
\end{aligned}
$$

Gradient computation (linear activation)

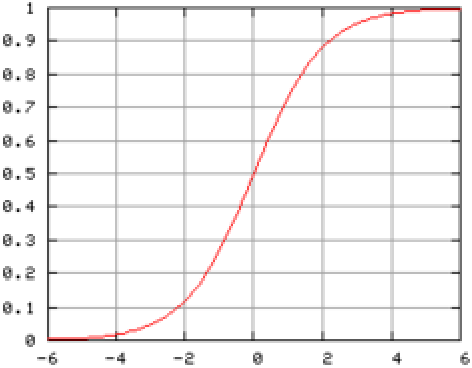
$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[t^{(s)} - \mathbf{w}\cdot\mathbf{x}^{(s)}\right] \\
&= \frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\left(-\frac{\partial}{\partial w_i}\left[\mathbf{w}\cdot\mathbf{x}^{(s)}\right]\right) \\
&= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[\sum_{j=1}^{n} w_j x_j^{(s)}\right] \\
&= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})x_i^{(s)}
\end{aligned}
$$

Algorithm with gradient descent

Gradient-Descent(S,η):
Each training example is a pair $(\mathbf{x}, t)$ where $\mathbf{x}$ is the vector of input values and t is the target value in output. η is the learning rate (that encompasses the constant term $1/N$).

1. Initialize the w_i's with small random values
2. Until the termination condition is met:
   a. $\Delta w_i \leftarrow 0$
   b. For each $(\mathbf{x}, t) \in S$:
      i. Present $\mathbf{x}$ to the neuron and compute the output $o(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
      ii. For each $i \in \{1, \dots, n\}$:
      $$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$
   c. For each $i \in \{1, \dots, n\}$:
   $$w_i \leftarrow w_i + \Delta w_i$$

Sigmoidal activation



Sigmoidal activation

Consider now a perceptron with sigmoidal activation function:

$$o(\mathbf{x}) = \sigma(\sum_{i=0}^{n} w_i x_i) = \sigma(\mathbf{w} \cdot \mathbf{x}) = \sigma(y)$$

where $y = \mathbf{w} \cdot \mathbf{x}$ and $\sigma(y) = \frac{1}{1+e^{-y}}$.

Following the same approach as before. We want to find the weight vector that minimizes the mean square error in the training set using a gradient-descent based algorithm.

To this aim, first we note that for the function $\sigma()$ defined above the following relation holds (verify as exercise):

$$\frac{\partial \sigma(y)}{\partial y} = \sigma(y)(1 - \sigma(y))$$

Gradient computation (sigmoidal activation)

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\left[\frac{1}{2N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})^2\right] \\
&= \frac{1}{2N}\sum_{s=1}^{N}\frac{\partial}{\partial w_i}\left[(t^{(s)} - o^{(s)})^2\right] \\
&= \frac{1}{2N}\sum_{s=1}^{N}2(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[t^{(s)} - o^{(s)}\right] \\
&= \frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[t^{(s)} - \sigma(y^{(s)})\right] \\
&= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[\sigma(y^{(s)})\right]
\end{aligned}
$$

Gradient computation (sigmoidal activation)

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial}{\partial w_i}\left[\sigma(y^{(s)})\right] \\
&= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial\sigma(y^{(s)})}{\partial y^{(s)}}\frac{\partial y^{(s)}}{\partial w_i} \\
&= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\frac{\partial\sigma(y^{(s)})}{\partial y^{(s)}}\frac{\partial}{\partial w_i}\left[\mathbf{w}\cdot\mathbf{x}^{(s)}\right] \\
&= -\frac{1}{N}\sum_{s=1}^{N}(t^{(s)} - o^{(s)})\sigma(y^{(s)})(1-\sigma(y^{(s)}))x_i^{(s)}
\end{aligned}
$$

Hence, the point 2.b.ii of the algorithm given above now becomes:

$$\Delta w_i \leftarrow \Delta w_i + \eta(t-o)\sigma(y)(1-\sigma(y))x_i$$

Final remarks about the Delta rule

- Note the similarity with the Perceptron rule!
- It converges to a valid solution even if the data are NOT linearly separable (provided a sufficiently small η)
- The objective function to minimize is convex. This implies that there is a unique global minimum!
- If η is too large, the gradient descent runs the risk of overstepping the minimum. If η is too small the convergence will be slow. One common modification is to gradually decrease η as the number of gradient descent steps grows.
- There exists a stochastic version (stochastic gradient descent)
- Other (equivalent) names from literature: LMS rule, Adaline rule, Widrow-Hoff rule.

Recap

Notions

- Gradient and minimization of vector functions
- The Delta rule
- Optimization with linear activation
- Optimization with sigmoidal activation

Exercises

- Show that $\sigma'(y) = \sigma(y)(1 - \sigma(y))$
- Implement the Delta rule optimization with linear/sigmoidal activation