Multilayer perceptron

Jaime López - May 2021

Abstract: In this note a model and algorithms are deduced for the multilayer perceptron (MLP), a feedforward neural network. In the first section, MLP's structure is shown, including the algorithm to transform the input $\mathbf{X}$ to the output $\hat{\mathbf{y}}$, i.e. the predicted values. In the second section, considerations to optimize MLP's parameters are defined. The backpropagation algorithm indicates how to adjust values for coefficients in each network connection. Finally, a gradient descent algorithm is developed to integrate forward and backpropagation operations.

Network Structure

The multilayer perceptron is a network composed of m layers with fully connected nodes. The layers are specified by a vector in which each element indicates the number of nodes by layer. $d^{(1)}$ is the number of nodes for the input layer and $d^{(m)}$ is the number of nodes for the output layer.

$$\mathbf{d} = [d^{(1)}, d^{(2)}, \dots, d^{(m)}]$$

Each layer operates on inputs by these sequential functions:

$$\mathbf{z}^{(l)} = \mathbf{a}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = \phi^{(l)}(\mathbf{z}^{(l)})$$

Notice that $\mathbf{a}^{(1)} = \mathbf{X}_{n\times d^{(1)}}$ and $\hat{\mathbf{y}}_{n\times d^{(m)}} = \mathbf{a}^{(m)}$. Besides that, $\mathbf{W}^{(l)}$ is a matrix of coefficients of size $d^{(l-1)} \times d^{(l)}$ and $b^{(l)}$ is a vector of constants of size $d^{(l)}$ for the linear function $\mathbf{z}^{(l)}$. $\mathbf{W}^{(l)}$, $\mathbf{b}^{(l)}$, and $\mathbf{z}^{(l)}$ are defined for $l = 2, \dots, m$. $\phi^{(l)}$ is a non-linear function that transforms $\mathbf{z}^{(l)}$. Algorithm 1 shows the process to map $\hat{\mathbf{y}}$ from $\mathbf{X}$.

Algorithm 1: Forward

Data: $\mathbf{X}$

Result: $\hat{\mathbf{y}}$

$\mathbf{a}^{(1)} \leftarrow \mathbf{X}$

for $l \leftarrow 2 \dots m$ **do**

  $\mathbf{z}^{(l)} \leftarrow \mathbf{a}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}$

  $\mathbf{a}^{(l)} \leftarrow \phi^{(l)}(\mathbf{z}^{(l)})$

$\hat{\mathbf{y}} \leftarrow \mathbf{a}^{(m)}$

Optimal parameters

In scalar notation, a cost function is defined:

$$C = \frac{1}{nd^{(m)}} \sum_{i=1}^{d^{(m)}} \sum_{j=1}^{n} (\hat{y}_{i,j} - y_{i,j})^2$$

The cost function in vectorial notation is presented below. Observe that $\otimes$ is the element-wise product of matrices.

$$C = \frac{1}{nd^{(m)}} \mathbf{1}_n^T [(\hat{\mathbf{y}} - \mathbf{y}) \otimes (\hat{\mathbf{y}} - \mathbf{y})] \mathbf{1}_{d^{(m)}}$$

The optimal values of parameters $\mathbf{W}$ and $\mathbf{b}$ are those at the minimum of the function C

$$\mathbf{J}_C = \mathbf{0}$$

Where $\mathbf{J}_C$ is the Jacobian of the function C

$$\mathbf{J}_C = [\partial C / \partial \mathbf{W}, \partial C / \partial \mathbf{b}]$$

For the last layer, applying the chain rule for derivatives, the elements of the Jacobian are:

$$\frac{\partial C}{\partial \mathbf{W}^{(m)}} = \frac{\partial C}{\partial \mathbf{a}^{(m)}} \frac{\partial \mathbf{a}^{(m)}}{\partial \mathbf{z}^{(m)}} \frac{\mathbf{z}^{(m)}}{\partial \mathbf{W}^{(m)}}$$

$$\frac{\partial C}{\partial \mathbf{b}^{(m)}} = \frac{\partial C}{\partial \mathbf{a}^{(m)}} \frac{\partial \mathbf{a}^{(m)}}{\partial \mathbf{z}^{(m)}} \frac{\mathbf{z}^{(m)}}{\partial \mathbf{b}^{(m)}}$$

In general, the elements for the Jacobian for other layers are

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} = \frac{\partial C}{\partial \mathbf{a}^{(m)}} \frac{\partial \mathbf{a}^{(m)}}{\partial \mathbf{a}^{(m-1)}} \cdots \frac{\partial \mathbf{a}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \frac{\mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l)}}$$

$$\frac{\partial C}{\partial \mathbf{b}^{(l)}} = \frac{\partial C}{\partial \mathbf{a}^{(m)}} \frac{\partial \mathbf{a}^{(m)}}{\partial \mathbf{a}^{(m-1)}} \cdots \frac{\partial \mathbf{a}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \frac{\mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}}$$

The derivatives of elements in the Jacobian are shown below. $\phi^{(l)\prime}$ is the derivative of the function $\phi^{(l)}$.

$$\frac{\partial C}{\partial \mathbf{a}^{(m)}} = \frac{2}{nd^{(m)}} (\mathbf{a}^{(m)} - \mathbf{y})$$

$$\frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{a}^{(l-1)}} = \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \frac{\mathbf{z}^{(l)}}{\partial \mathbf{a}^{(l-1)})}$$

$$\frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} = \phi^{(l)\prime}(\mathbf{z}^{(l)})$$

$$\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{a}^{(l-1)}} = \mathbf{W}^{(l)}$$

$$\frac{\mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l)}} = \mathbf{a}^{(l-1)}$$

$$\frac{\mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} = \mathbf{1}_{d^l}$$

Algorithm 2 shows how to update $\mathbf{W}$ and $\mathbf{b}$ departing from approximate values, using the derivatives of the cost function. In that, δ is a matrix that keeps the product of previous derivatives and the scalar η is the learning rate, used to regulate how fast values are updated. Besides that, vector $\mathbf{1}$ must have the same number of elements than $\mathbf{b}^{(l)}$.

Algorithm 2: Backpropagation

Data: $\mathbf{y}$, η (rate learning)
Result: $\mathbf{W}$, $\mathbf{b}$

```
δ ← 2/(n d^(m)) (a^(m) − y)
for l ← m ... 2 do
    δ ← δ ⊗ φ^(l)′(z^(l))
    W^(l) ← W^(l) − η a^(l−1)T δ
    b^(l) ← b^(l) − η 1 δ
    if m > 2 then
        δ ← δ W^(l)T
```

Given forward and backpropagation algorithms, a gradient descent algorithm can adjust values of $\mathbf{W}$ and $\mathbf{b}$ near to the optimal values. Below is algorithm 3 in which κ indicates the maximun number of iterations and ϵ represents the maximun accepted error. **rnd** is a function that returns matrices of random numbers.

Algorithm 3: Gradient descent

Data: $\mathbf{X}$, $\mathbf{y}$, m, $\mathbf{d}$, η (rate learning), κ (maximun iterations), ϵ (maximun accepted error)

Result: $\mathbf{W}$, $\mathbf{b}$

for l *in* $\mathit{2} \ldots m$ **do**

  $\mathbf{W}^{(l)} \leftarrow \mathbf{rnd}()_{d^{(l-1)} \times d^{(l)}}$

  $\mathbf{b}^{(l)} \leftarrow \mathbf{rnd}()_{1 \times d^{(l)}}$

  $\mathbf{z}^{(l)} \leftarrow \mathbf{0}_{1 \times d^{(l)}}$

  $\mathbf{a}^{(l)} \leftarrow \mathbf{0}_{1 \times d^{(l)}}$

$forward(\mathbf{X})$

k $\leftarrow$ 1

repeat

  backpropagate$(\mathbf{X}, \mathbf{y}, \eta)$

  $\hat{\mathbf{y}} \leftarrow forward(\mathbf{X})$

  $C \leftarrow \frac{1}{nd^{(m)}} \mathbf{1}_n^T ((\hat{\mathbf{y}} - \mathbf{y}) \otimes (\hat{\mathbf{y}} - \mathbf{y})) \mathbf{1}_{d(m)}$

  k $\leftarrow$ k + 1

until $k \leq \kappa \wedge C > \epsilon$

return $\mathbf{W}$, $\mathbf{b}$