Introduction to computational models

Lab assignment 2. Multilayer perceptron for classification problems

Pedro Antonio Gutiérrez
pagutierrez@uco.es

Module "Introduction to computational models"
4th year of "Grado en Ingeniería Informática"
Especialidad Computación
Escuela Politécnica Superior
(Universidad de Córdoba)

19th October 2022

Objectives of the lab assignment

- To implement the *off-line* version of the error backpropagation algorithm for the multilayer perceptron.
- To adapt the formulation in classification problems by interpreting the outputs using a probabilistic perspective (*softmax* function).
- To use a probabilistic error function to train the network (cross entropy).
- To check whether these modifications improve the results.

Classification

- Please, read and analyse the theory notes.
- We have studied how to adapt MLP to classification problems:
  - Representation of the class label using a 1-of-J coding.
  - Use of multiple neurons in the output layer and the softmax activation function.
  - During training, use of the cross-entropy cost function as an alternative to MSE.
  - For checking the goodness-of-fit, use of the CCR evaluation function.

Summary of the modifications to be performed

- We must make the program show information about the *CCR*.
- We must incorporate the *softmax* function in the output layer, that is, change the way the inputs are propagated (according to the definition of the *softmax*) and the way error is backpropagated (according to the new expression of δ_j^h).
- We must incorporate the L error function (cross entropy), calculating it in the functions that have to obtain an error and modifying the way in which the error is backpropagated for δ_j^H (only output layer).
- We must incorporate the *off-line* version of the algorithm (previous lab assignment).

Obtaining δ_j^h

- Derivatives for sigmoid neurons:
  - Output layer:
    - *MSE*:
      $\delta_j^H \leftarrow -\left(d_j - out_j^H\right) \cdot out_j^H \cdot \left(1 - out_j^H\right)$
    - Cross-entropy:
      $\delta_j^H \leftarrow -\left(d_j / out_j^H\right) \cdot out_j^H \cdot \left(1 - out_j^H\right)$
  - Hidden layers:
    $\delta_j^h \leftarrow \left(\sum_{i=1}^{n_{h+1}} w_{ij}^{h+1} \delta_i^{h+1}\right) \cdot out_j^h \cdot \left(1 - out_j^h\right)$
- Derivatives for *softmax* functions:
  - Only output layer:
    - *MSE*:
      $\delta_j^H \leftarrow -\sum_{i=1}^{n_H} \left(\left(d_i - out_i^H\right) \cdot out_j^H (I(i = j) - out_i^H)\right)$
    - Cross-entropy:
      $\delta_j^H \leftarrow -\sum_{i=1}^{n_H} \left(\left(d_i / out_i^H\right) \cdot out_j^H (I(i = j) - out_i^H)\right)$

Adjustment of derivatives for *off-line* mode

- When using the *off-line* mode, derivatives are accumulated for all the patterns and their magnitude can be very high.
- As we are using an averaged error, it is a good idea to divide the derivative by the number of patterns (N).

Adjustment of derivatives for *off-line* mode

weightAdjustment()

Start

1. **For** h from 1 to H *// For each layer ($\Rightarrow\Rightarrow$)*
   1. **For** j from 1 to n_h *// For each neuron of layer h*
      1. **For** i from 1 to n_{h-1} *// For each neuron of layer $h-1$*
         $w_{ji}^h \leftarrow w_{ji}^h - \frac{\eta\Delta w_{ji}^h}{N} - \frac{\mu\left(\eta\Delta w_{ji}^h(t-1)\right)}{N}$
         End For
      2. $w_{j0}^h \leftarrow w_{j0}^h - \frac{\eta\Delta w_{j0}^h}{N} - \frac{\mu\left(\eta\Delta w_{j0}^h(t-1)\right)}{N}$ *// Bias*

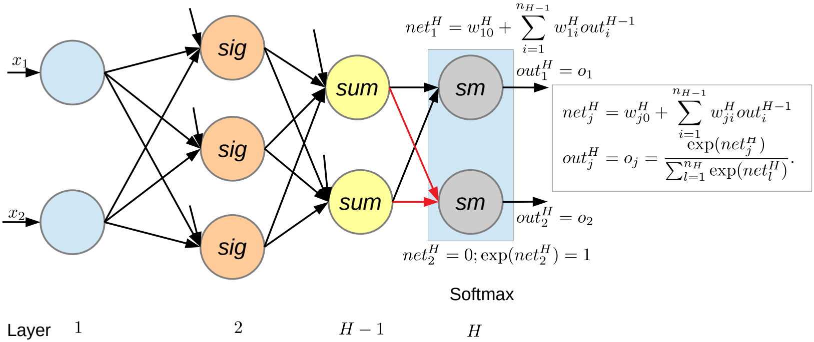
      End For

   End For

End

Optimization

An optimization technique consist of removing the last output neuron of the last layer (softmax) to avoid unnecesary computations associated to that neuron.



Introduction to computational models

Lab assignment 2. Multilayer perceptron for classification problems

Pedro Antonio Gutiérrez
pagutierrez@uco.es

Module "Introduction to computational models"
4th year of "Grado en Ingeniería Informática"
Especialidad Computación
Escuela Politécnica Superior
(Universidad de Córdoba)

19th October 2022