Introduction to computational models

Lab Assignment 1. Implementation of the multilayer perceptron

Javier Sánchez Monedero (jsanchezm@uco.es)
Luca Romeo (luca.romeo@unimc.it)
Riccardo Rosati (r.rosati@pm.univpm.it)

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

27th September 2023

1. Contents
2. Notation and architecture
3. Data normalization
4. Pseudocode

Objectives of the lab assignment

- To familiarise the student with neural networks, in particular, with the multilayer perceptron.
- To implement the basic backpropagation algorithm for multilayer perceptrons.
- To check the effect of different parameters and data preprocessing:
  - Network architecture.
  - Learning rate.
  - Momentum factor.
  - Data normalization.
  - etc.

Backpropagation algorithm

- Please, read and analyse the theory notes.
- Pay special attention to the pseudo-code.

Stop condition

- Standard version, the algorithm stops if:
  - Training error does not decrease more than 0.00001 or increases, during 50 iterations (external loop).

Data normalization considerations I

- Normalization is an essential step in data pre-processing in any machine learning application and model fitting. This is essential to deal with the effect of different variable magnitudes and distribution of each variable.
- There are many normalization alternatives. Two widespread ones are:
    - Scalling: each feature is transformed within the range $[a, b]$:

$$X' = a + \frac{(X - X_{\text{min}})(b - a)}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

    - Standardization: each feature is transformed so that the normalized feature has mean equal to 0.0 and standard deviation equal to 1.0:

$$X' = \frac{X - \mu}{\sigma} \tag{2}$$

Data normalization considerations II

- It is important to estimate the normalization parameters using the training data, and then to apply data transformation to the train and test sets with these parameters.
- A typicall error is to perform data normalization on the test dataset calculating minimum and maximum values (X_{max} and X_{min}), or the mean and deviation (μ y σ), on the test dataset or before data splitting. In both cases we will be (wrongly) using testing information during our model building and calibration.
- For our work, we will implement feature scaling for input variables in $[-1, 1]$ and for output variables in $[0, 1]$ because we have a sigmoid transformation in the output layer.

Backpropagation algorithm

On-line backpropagation

Start

1. $w_{ji}^{h} \leftarrow U[-1,1]$ *// Random values between -1 and $+1$*
2. **Repeat**
   1. **For** each pattern with inputs $\mathbf{x}$ and outputs $\mathbf{d}$
      1. $\Delta w_{ji}^{h} \leftarrow 0$ *// Changes will be applied for each pattern*
      2. $out_{j}^{0} \leftarrow x_j$ *// Feed inputs*
      3. forwardPropagation() *// Forward propagation ($\Rightarrow\Rightarrow$)*
      4. backPropagation() *// Error backpropagation ($\Leftarrow\Leftarrow$)*
      5. accumulateChange() *// Obtain the weight update*
      6. weightAdjustment() *// Apply the calculated update*

      End for

   Until (StopCondition)
3. **Return** weight matrices.

End

Backpropagation algorithm

Off-line backpropagation

Start

1. $w_{ji}^h \leftarrow U[-1,1]$ *// Random values between -1 and $+1$*
2. **Repeat**
    1. $\Delta w_{ji}^h \leftarrow 0$ *// Changes will be applied at the end*
    2. **For** each pattern with inputs $\mathbf{x}$ and outputs $\mathbf{d}$
        1. $out_j^0 \leftarrow x_j$ *// Feed inputs*
        2. forwardPropagation() *// Forward propagation ($\Rightarrow\Rightarrow$)*
        3. backPropagation() *// Error backpropagation ($\Leftarrow\Leftarrow$)*
        4. accumulateChange() *// Obtain the weight update*

        End for
    3. weightAdjustment() *// Apply the calculated update*

    Until (StopCondition)
3. **Return** weight matrices.

End

Backpropagation algorithm: functions

forwardPropagation()

Start

1. **For** h from 1 to H *// For each layer (⇒⇒)*
    1. **For** j from 1 to n_h *// For each neuron of layer h*
        1. $net_j^h \leftarrow w_{j0}^h + \sum_{i=1}^{n_{h-1}} w_{ji}^h out_i^{h-1}$
        2. $out_j^h \leftarrow \frac{1}{1+\exp(-net_j^h)}$

        End For

    End For

End

Backpropagation algorithm: functions

backPropagation()

Start

1. **For** j from 1 to n_H *// For each output neuron*
    1. $\delta_j^H \leftarrow -\left(d_j - out_j^H\right) \cdot g'\left(net_j^H\right)$ *// We have eliminated the constant (2), the result should be similar*

    End For
2. **For** h from $H-1$ to 1 *// For each layer ($\Leftarrow\Leftarrow$)*
    1. **For** j from 1 to n_h *// For each neuron in layer h*
        1. $\delta_j^h \leftarrow \left(\sum_{i=1}^{n_{h+1}} w_{ij}^{h+1}\delta_i^{h+1}\right) \cdot out_j^h \cdot (1 - out_j^h)$ *// Navigate all neurons in layer $h+1$ connected with neuron j*

        End For

    End For

End

Backpropagation algorithm: functions

accumulateChange()

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 $h-1$*
         $\Delta w_{ji}^h \leftarrow \Delta w_{ji}^h + \delta_j^h \cdot out_i^{h-1}$
         End For
      2. $\Delta w_{j0}^h \leftarrow \Delta w_{j0}^h + \delta_j^h \cdot 1$ *// Bias*

      End For

   End For

End

Backpropagation algorithm: functions

weightAdjustment()

Start

1. **For** h from 1 to H *// For each layer (⇒⇒)*
    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 $h-1$*
           $w_{ji}^h \leftarrow w_{ji}^h - \eta \Delta w_{ji}^h - \mu \left(\eta \Delta w_{ji}^h (t-1) \right)$
           End For
        2. $w_{j0}^h \leftarrow w_{j0}^h - \eta \Delta w_{j0}^h - \mu \left(\eta \Delta w_{j0}^h (t-1) \right)$ *// Bias*

        End For

    End For

End

Introduction to computational models

Lab Assignment 1. Implementation of the multilayer perceptron

Javier Sánchez Monedero (jsanchezm@uco.es)
Luca Romeo (luca.romeo@unimc.it)
Riccardo Rosati (r.rosati@pm.univpm.it)

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

27th September 2023