Machine Learning–part I lez.8

Francesco Camastra

francesco.camastra@uniparthenope.it

Ricevimento

- Orario: Mercoledì 14.00-18.00
- Dove: Stanza 430, Quarto piano, Centro Direzionale, Isola C4
- Per il ricevimento è OBBLIGATORIO inviare una email di prenotazione almeno due giorni prima del giorno di ricevimento.
- email del Docente: francesco.camastra@uniparthenope.it
- Pagina del Docente: dist.uniparthenope.it/francesco.camastra

Artificial Neural Networks

- Sources
  - F.Camastra,A.Vinciarelli, Machine Learning for Audio, Image and Video Analysis, (8° Chapter)
  - Christopher Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995

Multilayer Networks Training

- The training procedure is based on the minimization of an error function, or ***empirical risk***, with respect to the parameter set of the algorithm under examination. In the case of MLPs, the parameter set $\vec{w}$ contains connection weights and neuron biases. The error function is a differentiable function of the network outputs y_k and these are a function of the network parameters, then the error function can be derived with respect to any single parameter in $\vec{w}$.

Error Backpropagation

- This enables to minimize the error function by applying different optimization algorithms such as gradient descent. The name ***error back-propagation*** comes from the fact that the derivation propagates the error from the output nodes to the input nodes.

Error Backpropagation

- In general the training algorithms are iterative and each iteration involves two steps that can be considered separately:
- ***Evaluation of error function derivatives***.
  - ✓The expression error back-propagation actually refers to this step, although it is used sometimes to define the whole training process. This stage depends on the particular network under examination because the functional expression corresponding to the network, changes for each architecture.

Error Backpropagation

- ***Parameters update***
  - ✓This stage modifies the network parameters with the goal of minimizing the error function. This stage is independent of the particular network used. In fact, once the derivatives are at disposition, the minimization techniques do not depend any more on the particular network or architecture used.

Error Backpropagation *for Feed-Forwards Networks*

- Since the training is supervised, we have a training set which is a collection of input-output patterns, i.e., $\mathcal{D} = \{(\overrightarrow{x_1}, \overrightarrow{t_1}), \dots, (\overrightarrow{x_\ell}, \overrightarrow{t_\ell})\} \in \mathbb{R}^n \times \mathcal{Y}$. In the regression problem $\mathcal{Y}$ is continuous i.e. $\mathcal{Y} \subseteq \mathbb{R}^O$. In the classification problem $\mathcal{Y}$ is discrete, i.e. $\mathcal{Y} = (\mathcal{C}_1, \dots, \mathcal{C}_O)$ that is not suitable to be used in a MLP.
- A more appropriate approach consists in representing $\mathcal{Y}$ as a discrete subset of $\mathbb{R}^O$, i.e. $\mathcal{Y} = \{+1, -1\}^O$, where the discrete values +1 and −1 corresponds to the membership and the non-membership to a given class, respectively. **Therefore if the *m*th component of the target $\vec{y}$ is +1 then the respective pattern $\vec{y}$ belongs to the class $\mathcal{C}_m$.**

Error Backpropagation *for Feed-Forwards Networks*

- Since the training is supervised, we have a training set which is a collection of input-output patterns, i.e., $\mathcal{D} = \{(\overrightarrow{x_1}, \overrightarrow{t_1}), \dots, (\overrightarrow{x_\ell}, \overrightarrow{t_\ell})\} \in \mathbb{R}^n \times \mathcal{Y}$. In the regression problem $\mathcal{Y}$ is continuous i.e. $\mathcal{Y} \subseteq \mathbb{R}^O$. In the classification problem $\mathcal{Y}$ is discrete, i.e. $\mathcal{Y} = (\mathcal{C}_1, \dots, \mathcal{C}_O)$ that is not suitable to be used in a MLP.
- A more appropriate approach consists in representing $\mathcal{Y}$ as a discrete subset of $\mathbb{R}^O$, i.e. $\mathcal{Y} = \{+1, -1\}^O$, where the discrete values +1 and −1 corresponds to the membership and the non-membership to a given class, respectively. **Therefore if the *m*th component of the target $\vec{y}$ is +1 then the respective pattern $\vec{y}$ belongs to the class $\mathcal{C}_m$.**

Error Backpropagation *for Feed-Forwards Networks*

- The functional form corresponding to a feed-forward network is: $y_k = g\left[\sum_{j=n+1}^{n+1+H} w_{kj}\, \tilde{g}\left(\sum_{l=0}^{n} w_{jl} x_l\right)\right]$
- where the biases are included in the summations through extra nodes with input fixed to 1 and do not need to be distinguished from connection weights.
- The error function (or empirical risk) has typically the following form: $E = \sum_{k=1}^{\ell} \epsilon_k$
- where ϵ_k is the error, i.e. **the *loss function***, of the network over the kth sample of the training set $\mathcal{D}$.

Error Backpropagation *for Feed-Forwards Networks*

- The derivative of E with respect to any parameter w_{ij} can then be expressed as: $\frac{\partial E}{\partial w_{ij}} = \sum_{k=1}^{\ell} \frac{\partial \epsilon_k}{\partial w_{ij}}$
- and in the following we can focus on a single $\frac{\partial \epsilon}{\partial w_{ij}}$ (the index k is omitted whenever possible).
- The derivative of ϵ with respect to a weight of the ***first layer*** can be obtained as follows: $\frac{\partial \epsilon}{\partial w_{ij}} = \frac{\partial \epsilon}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}$
- where a_i is the input of node i in the hidden layer, $i = n + 1, \dots, n + 1 + H$ and $j = 0, \dots, n$.

Error Backpropagation *for Feed-Forwards Networks*

- The first term of the above product is called ***error*** and it is denoted by δ_i: $\delta_i = \frac{\partial \epsilon}{\partial a_i}$.
- Since $a_i = \sum_{l=0}^{n} w_{jl} x_l$, the second term $\frac{\partial a_i}{\partial w_{ij}}$ of the same product is simply, $\frac{\partial \sum_{l=0}^{n} w_{jl} x_l}{\partial w_{ij}} = x_j$.
- As a result, the derivative of ϵ with respect to a weight in the first layer can be written as follows: $\frac{\partial \epsilon}{\partial w_{ij}} = \frac{\partial \epsilon}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} = \delta_i x_j$.

Error Backpropagation *for Feed-Forwards Networks*

- Using the same approach, the derivative of ϵ with respect to a weight w_{kl} in the second layer, i.e., $k = n + H + 1, \dots, n + H + O$ and $l = n + 1, \dots, n + 1 + H$, can be written as: $\frac{\partial \epsilon}{\partial w_{kl}} = \delta_k z_k$.
- where $\delta_k = \frac{\partial \epsilon}{\partial z_k}$.

Error Backpropagation *for Feed-Forwards Networks*

- The expression of the errors δ_i is different for hidden and output nodes. The input nodes are not considered because their activation function is the identity. For the output nodes the error δ_i is:

  $$\delta_i = \frac{\partial \epsilon}{\partial z_i} = \frac{\partial \epsilon}{\partial y_i}\frac{\partial y_i}{\partial z_i} = \acute{g}(z_i)\,\frac{\partial \epsilon}{\partial y_i}\,.$$

- where $\acute{g}(z)$ is simply the first derivative of the activation function of the output nodes $g(z)$.

Error Backpropagation *for Feed-Forwards Networks*

- For the hidden nodes we have to take into account the fact that they are connected to all of the output nodes, then it is necessary to sum over all of these:
$$\delta_k = \frac{\partial \epsilon}{\partial a_k} = \sum_{l=1}^{O} \frac{\partial \epsilon}{\partial z_l} \frac{\partial z_l}{\partial a_k} = \sum_{l=1}^{O} \delta_l \frac{\partial z_l}{\partial a_k}$$
- where the expression δ_l corresponds to the previous equation $\delta_l = \acute{g}(z_l) \frac{\partial \epsilon}{\partial y_l}$ because the sum is made over the output neurons.

Error Backpropagation *for Feed-Forwards Networks*

- The last missing element is then $\frac{\partial z_l}{\partial a_k}$ which corresponds to the following expression:
$$\frac{\partial z_l}{\partial a_k} = \frac{\partial}{\partial a_k}\left(\sum_{i=1}^{H} \tilde{g}(a_i) w_{lj}\right) = \acute{\tilde{g}}(a_k) w_{lk}.$$
- By plugging the last expression into previous equation, the result for the hidden nodes errors is:
$$\delta_k = \tilde{g}(a_k)\ \sum_{l=1}^{O} w_{lk}\ \acute{g}(z_l)\,\frac{\partial \epsilon}{\partial y_l}.$$

Error Backpropagation *for Feed-Forwards Networks*

- The above results enable one to write the derivative of ϵ_n with respect to any network parameter by simply plugging the expression of the activation functions $g(z)$ and $\tilde{\acute{g}}(a)$ as well as of the loss ϵ_n. The derivative of E can then be obtained by simply summing over the errors of all the training set samples.