

Data Mining (Mineria de Dades)

Lluís A. Belanche
belanche@lsi.upc.edu

Soft Computing Research Group

Dept. de Llenguatges i Sistemes Informàtics (Software department)

Universitat Politècnica de Catalunya

2011-2012

Data Mining (Part II)

Contents

1. Association Rules
2. Bayesian decision theory
3. Bayesian classifiers: LDA/QDA
4. Bayesian classifiers: Naïve Bayes
5. Bayesian classifiers: Nearest Neighbours
6. Neural Networks: Multilayer Perceptrons
7. Support Vector Machines
8. Ensemble Learning

Artificial neural networks (III)

A gentle derivation of backpropagation

A MLP of c hidden layers is a function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ made up of pieces $F_1, \dots, F_m$ of the form:

$$F_k(\mathbf{x}) = g \left(\sum_{j=0}^{h_c} w_{kj}^{(c+1)} \phi_j^{(c)}(\mathbf{x}) \right), k = 1, \dots, m$$

where, for every $l = 1, \dots, c$, $W^{(l)} = (w_{ki}^{(l)})$ is the matrix of weights connecting layers $l - 1$ and l , h_l is the size of hidden layer l and

$$\phi_j^{(l)}(\mathbf{x}) = g \left(\sum_{i=0}^{h_{l-1}} w_{ji}^{(l)} \phi_i^{(l-1)}(\mathbf{x}) \right), \text{ for } l = 1, \dots, c$$

with $\phi_i^{(0)}(\mathbf{x}) = x_i, \phi_0^{(l)}(\mathbf{x}) = 1$ (in particular, $x_0 = 1$) and $h_0 = n$.

Artificial neural networks (III)

A gentle derivation of backpropagation

Recall the goal in regression is to minimize the empirical error of the network on the training data sample $S = \{(\mathbf{x}_\mu, \mathbf{t}_\mu)\}_{\mu=1,\dots,p}$, where $\mathbf{x}_\mu \in \mathbb{R}^n, \mathbf{t}_\mu \in \mathbb{R}^m$.

$$E_{emp}(\mathcal{W}) = \frac{1}{2} \sum_{\mu=1}^p \sum_{k=1}^m (t_{\mu k} - F_k(\mathbf{x}_\mu))^2$$

where $\mathcal{W} = \{W^{(1)}, \dots, W^{(c+1)}\}$ is the set of all network weights.

Artificial neural networks (III)

A gentle derivation of backpropagation

- Note that, if g admits a derivative everywhere, $E_{emp}(\mathcal{W})$ is a differentiable function of every weight $w_{ji}^{(l)}$
- If we want to apply gradient descent, we need to compute the partial derivative of the error w.r.t. every weight, the *gradient vector*:

$$\nabla E_{emp}(\mathcal{W}) = \left(\frac{\partial E_{emp}(\mathcal{W})}{\partial w_{ji}^{(l)}} \right)_{l,j,i}$$

- There exists a reasonably efficient algorithm for computing this gradient vector: the backpropagation algorithm

Artificial neural networks (III)

A gentle derivation of backpropagation

Consider a MLP where, for notational simplicity, we define:

$$z_j^{(l)} \equiv g(a_j^{(l)}) \equiv g \left(\sum_i w_{ji}^{(l)} z_i^{(l-1)} \right), \quad z_j^{(0)} = x_j$$

- Note that E_{emp} is the sum of the (independent) errors for every input/output example $(\mathbf{x}_\mu, \mathbf{t}_\mu)$:

$$E_{emp}(\mathcal{W}) = \sum_{\mu=1}^p \frac{1}{2} \sum_{k=1}^m (t_{\mu k} - F_k(\mathbf{x}_\mu))^2 \equiv \sum_{\mu=1}^p E_{emp}^{(\mu)}(\mathcal{W})$$

Artificial neural networks (III)

A gentle derivation of backpropagation

Suppose we present x_μ to the network and compute all the neuron's outputs $z_j^{(l)}$ (this is known as the **forward propagation**). Now,

$$\frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial w_{ji}^{(l)}} = \frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} \cdot z_i^{(l-1)}$$

where we have defined $\delta_j^{(l)} \equiv \frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial a_j^{(l)}}$.

Artificial neural networks (III)

A gentle derivation of backpropagation

What have we done?

We have found that, in order to compute the desired derivative $\frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial w_{ji}^{(l)}}$,

we only need to find the $\delta_j^{(l)}$:

Let us concentrate on an arbitrary neuron k . Suppose first that k is an **output neuron**, then

$$\delta_k^{(c+1)} = \frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial a_k^{(c+1)}} = -g'(a_k^{(c+1)}) \cdot (t_{\mu k} - F_k(\mathbf{x}_\mu)) = g'(a_k^{(c+1)}) \cdot (z_k^{(c+1)} - t_{\mu k})$$

where we have made use of the identity $F_k(\mathbf{x}_\mu) = g(a_k^{(c+1)}) = z_k^{(c+1)}$.

Artificial neural networks (III)

A gentle derivation of backpropagation

When g is the logistic function $l_\beta(z) = \frac{1}{1+e^{-\beta z}} \in (0, 1)$, we obtain:

$$g'(a_k^{(c+1)}) = \beta g(a_k^{(c+1)})[1 - g(a_k^{(c+1)})] = \beta z_k^{(c+1)}(1 - z_k^{(c+1)})$$

Therefore

$$\frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial w_{ji}^{(l)}} = \beta z_j^{(c+1)}(1 - z_j^{(c+1)})(z_j^{(c+1)} - t_{\mu j})z_i^{(c)}$$

Artificial neural networks (III)

A gentle derivation of backpropagation

Suppose now that k is a **hidden neuron**, located in a layer $l \in \{1, \dots, c\}$:

$$\begin{aligned}\delta_k^{(l)} &= \frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial a_k^{(l)}} = \sum_q \frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial a_q^{(l+1)}} \cdot \frac{\partial a_q^{(l+1)}}{\partial a_k^{(l)}} = \sum_q \delta_q^{(l+1)} \cdot \frac{\partial a_q^{(l+1)}}{\partial a_k^{(l)}} \\ &= \sum_q \delta_q^{(l+1)} \cdot \frac{\partial a_q^{(l+1)}}{\partial z_k^{(l)}} \cdot \frac{\partial z_k^{(l)}}{\partial a_k^{(l)}} = \sum_q \delta_q^{(l+1)} w_{qk}^{(l+1)} g'(a_k^{(l)}) = g'(a_k^{(l)}) \sum_q \delta_q^{(l+1)} w_{qk}^{(l+1)}\end{aligned}$$

Again, when g is the logistic, $g'(a_k^{(l)}) = \beta g(a_k^{(l)})[1 - g(a_k^{(l)})] = \beta z_k^{(l)}(1 - z_k^{(l)})$

Artificial neural networks (III)

A gentle derivation of backpropagation

Therefore

$$\frac{\partial E_{emp}(\mathcal{W})}{\partial w_{ji}^{(l)}} = \sum_{\mu=1}^p \frac{\partial E_{emp}^{(\mu)}(\mathcal{W})}{\partial w_{ji}^{(l)}} \text{ and we have } \nabla E_{emp}(\mathcal{W}) = \left(\frac{\partial E_{emp}(\mathcal{W})}{\partial w_{ji}^{(l)}} \right)_{l,j,i}$$

The updating formula for the weights is:

$$w_{ji}^{(l)}(t+1) \leftarrow w_{ji}^{(l)}(t) - \alpha \left. \frac{\partial E_{emp}(\mathcal{W})}{\partial w_{ji}^{(l)}} \right|_{\mathcal{W}=\mathcal{W}(t)}$$