

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 (IV)

Backpropagation, convergence and second order methods

In a MLP of c hidden layers, let ω (or $\omega(t)$) represent a vector with all network's weights (at time t). This vector has length $\kappa = \sum_{l=1}^{c+1} (h_{l-1} + 1)h_l$, where $h_0 = n$ and $h_{c+1} = m$.

When training the MLP, the minimization of the error function E_{emp} using the Back-propagation algorithm (BPA) involves a sequence of weight iterates $\{\omega(t)\}_{t=0}^{\infty}$, where t indicates iterations through S (called *epochs*).

The algorithm finds the next weights using the relation:

$$\omega(t+1) \leftarrow \omega(t) - \alpha \nabla E_{emp}(\omega)|_{\omega=\omega(t)}$$

The role of the BPA is to compute the elements of $\nabla E_{emp}(\omega)|_{\omega=\omega(t)}$ recursively at each iteration (nothing more, nothing less!).

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

Define *global convergence* as convergence to a local minimizer of the error function from any starting point in $\mathbb{R}^\kappa$.

In order to ensure global convergence, the following assumptions are needed:

1. The error function is real-valued and continuous everywhere in $\mathbb{R}^\kappa$, and bounded below in $\mathbb{R}^\kappa$.
2. For any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^\kappa$, ∇E_{emp} satisfies the Lipschitz condition:

$$\|\nabla E_{emp}(\mathbf{x}) - \nabla E_{emp}(\mathbf{y})\| \leq L \cdot \|\mathbf{x} - \mathbf{y}\|$$

where $L > 0$ denotes the Lipschitz constant.

The BPA is globally convergent if we can determine a learning rate α such that E_{emp} is exactly minimized along the direction of the negative of the gradient in each iteration.

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

GENERAL CONVERGENCE THEOREM. Let λ_k be the eigenvalues of the matrix $\nabla^2 E_{emp}(\omega)|_{\omega=\omega(t)}$ for a given $\omega(t)$ (assumed PD). If $|1 - \lambda_k \alpha| < 1$ for all k then, as the number of iterations t tends to ∞ , $\omega(t)$ tends to a local minimum of $E_{emp}(\omega)$.

Observations:

- Recall $\nabla^2 E_{emp}(\omega)$ is the Hessian matrix, with elements $\frac{\partial^2 E_{emp}(\omega)}{\partial \omega_i \partial \omega_j}$
- It is straightforward to see that $\alpha < 2/\lambda_{max}$ is a sufficiently small α .
- Too large values of α show fast convergence but a tendency to oscillate
- Too small values of α show slow convergence

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

A generic (iterative) minimization algorithm is:

1. Choose an initial point x_0
2. Select a search direction p
3. Select a step size α , and set $x \leftarrow x + \alpha p$
4. Return to 2. unless a convergence criterion has been met

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

Suppose $f : \mathbb{R}^k \rightarrow \mathbb{R}$ we wish to minimize (assume that f is differentiable).

A first-order Taylor expansion around a point $\mathbf{x}_t$ is:

$$f(\mathbf{x}) \approx f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^t (\mathbf{x} - \mathbf{x}_t) = \hat{f}(\mathbf{x})$$

A simple minimization algorithm is to set $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)$ for $\alpha > 0$, since

$$\begin{aligned} \hat{f}(\mathbf{x}_{t+1}) &= f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^t (\mathbf{x}_{t+1} - \mathbf{x}_t) = f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^t (\mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t) - \mathbf{x}_t) = f(\mathbf{x}_t) - \\ &\alpha \nabla f(\mathbf{x}_t)^t \nabla f(\mathbf{x}_t) = f(\mathbf{x}_t) - \alpha \|\nabla f(\mathbf{x}_t)\|^2 < f(\mathbf{x}_t) \end{aligned}$$

provided $\nabla f(\mathbf{x}_t) \neq 0$ and $\alpha > 0$ is small enough. Therefore we derive a learning rule:

$$\boldsymbol{\omega}(t+1) \leftarrow \boldsymbol{\omega}(t) - \alpha(t) \nabla E_{emp}(\boldsymbol{\omega})|_{\boldsymbol{\omega}=\boldsymbol{\omega}(t)} = \boldsymbol{\omega}(t) - \alpha(t) \nabla E_{emp}(\boldsymbol{\omega}(t))$$

This method is known as **gradient descent** (a.k.a. steepest descent).

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

The Taylor expansion gives no clue on how to choose $\alpha(t) > 0$. In ANNs there are two common strategies:

1. Use a constant small $\alpha > 0$ or a variable $\alpha(t) = \frac{\alpha}{t+1}, \alpha > 0$.
2. Perform a line search to find $\alpha(t) = \arg \min_{\alpha} f[\mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)]$.

Convergence of steepest descent can be very slow, since it uses limited knowledge of the error function.

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

A better strategy is to use a second-order Taylor expansion around a point $\mathbf{x}_t$:

$$f(\mathbf{x}) \approx f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^t (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_t)^t \nabla^2 f(\mathbf{x}_t) (\mathbf{x} - \mathbf{x}_t) = \hat{f}(\mathbf{x})$$

where $\nabla^2 f(\mathbf{x}_t)$ is the Hessian of f evaluated at $\mathbf{x}_t$. Provided that $\nabla^2 f(\mathbf{x}_t)$ is positive definite, this time the minimum of $\hat{f}(\mathbf{x})$ occurs at the $\mathbf{x}$ satisfying:

$$\nabla f(\mathbf{x}_t) + \nabla^2 f(\mathbf{x}_t) (\mathbf{x} - \mathbf{x}_t) = 0$$

which leads to the minimization algorithm $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - (\nabla^2 f(\mathbf{x}_t))^{-1} \nabla f(\mathbf{x}_t)$

This second-order method is known as a **Newton method**.

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

Note there is no need for α , since the inverse of the Hessian determines both the step size and the search direction. However, this method has some practical drawbacks:

1. It requires $\nabla^2 f(\mathbf{x}_t)$ to be positive definite (otherwise there is no unique minimum)
2. It requires the (exact) computation of the Hessian at every iteration
3. It requires a matrix inversion at every iteration
4. The knowledge of the local curvature provided by the Hessian is only useful very close to $\mathbf{x}_t$

Artificial neural networks (IV)

Backpropagation, convergence and second order methods

- In *quasi-Newton methods*, the inverse of the Hessian matrix is directly approximated, leading to:

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \alpha_t B_t^{-1} \nabla f(\mathbf{x}_t), \text{ where } B_t^{-1} \approx (\nabla^2 f(\mathbf{x}_t))^{-1}$$

- The α_t parameter is optimized via line search (also ensuring that B_t^{-1} is positive definite)
- There exist several variations that prescribe different ways of generating and updating B_t^{-1}
- The most common quasi-Newton algorithm is the BFGS method (suggested independently by Broyden, Fletcher, Goldfarb, and Shanno)
- The BPA can also be used to compute the approximation to the Hessian at each iteration