Aprenentatge Automàtic 1

GCED

Lluís A. Belanche

belanche@cs.upc.edu

Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)
Universitat Politècnica de Catalunya

2019-2020

LECTURE 8c: Artificial neural networks (I)

Artificial neural networks (I): the MLP

Relation between backpropagation and order methods

Let $\boldsymbol{\omega}(t)$ represent a vector with **all** network weights (at time t)

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

A basic algorithm finds the next weights using the relation:

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

The role of the BPA is to compute the elements of $\nabla E_{emp}(\boldsymbol{\omega}(t))$ at each iteration (recursively and rather efficiently).

Artificial neural networks (I): the MLP

Relation between backpropagation and order methods

General convergence theorem. Let λ_k be the eigenvalues of the matrix $\nabla^2 E_{emp}(\boldsymbol{\omega}(t))$ for a given $\boldsymbol{\omega}(t)$ (assumed p.d.). If $|1 - \lambda_k \alpha| < 1$ for all k then, as t tends to ∞, $\boldsymbol{\omega}(t)$ tends to a local minimum of $E_{emp}(\boldsymbol{\omega})$.

Observations:

- Recall $\nabla^2 E_{emp}(\boldsymbol{\omega})$ is the Hessian matrix, with elements $\dfrac{\partial^2 E_{emp}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}_i \partial \boldsymbol{\omega}_j}$
- 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 (I): the MLP

Relation between backpropagation and order methods

A generic (iterative) **minimization algorithm** is:

1. Choose an initial point $\boldsymbol{x}_0$; set $t = 0$
2. Select a search direction $\boldsymbol{p}_t$
3. Select a step size α_t, and set $\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \alpha_t \boldsymbol{p}_t$
4. Return to 2. (unless a convergence criterion has been met)

Artificial neural networks (I): the MLP

Suppose $f : \mathbb{R}^r \to \mathbb{R}$ we wish to minimize (assume that f is differentiable). A first-order Taylor expansion around the current point $\boldsymbol{x}_t$ is:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_t) + \nabla f(\boldsymbol{x}_t)^\top (\boldsymbol{x} - \boldsymbol{x}_t) = \widehat{f}(\boldsymbol{x})$$

A simple minimization algorithm is to set $\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \alpha_t \nabla f(\boldsymbol{x}_t)$ for $\alpha_t > 0$, since

$$\begin{aligned}
\widehat{f}(\boldsymbol{x}_{t+1}) = \\
f(\boldsymbol{x}_t) + \nabla f(\boldsymbol{x}_t)^\top (\boldsymbol{x}_{t+1} - \boldsymbol{x}_t) = \\
f(\boldsymbol{x}_t) + \nabla f(\boldsymbol{x}_t)^\top (\boldsymbol{x}_t - \alpha_t \nabla f(\boldsymbol{x}_t) - \boldsymbol{x}_t) = \\
f(\boldsymbol{x}_t) - \alpha_t \nabla f(\boldsymbol{x}_t)^\top \nabla f(\boldsymbol{x}_t) = \\
f(\boldsymbol{x}_t) - \alpha_t \|\nabla f(\boldsymbol{x}_t)\|^2 < \\
f(\boldsymbol{x}_t)
\end{aligned}$$

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

$$\boldsymbol{\omega}(t+1) := \boldsymbol{\omega}(t) - \alpha_t \nabla E_{emp}(\boldsymbol{\omega}(t))$$

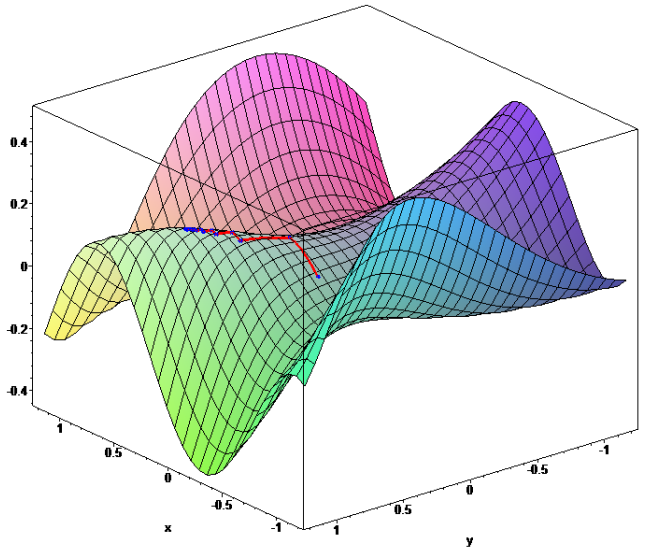
This method is known as **gradient descent**

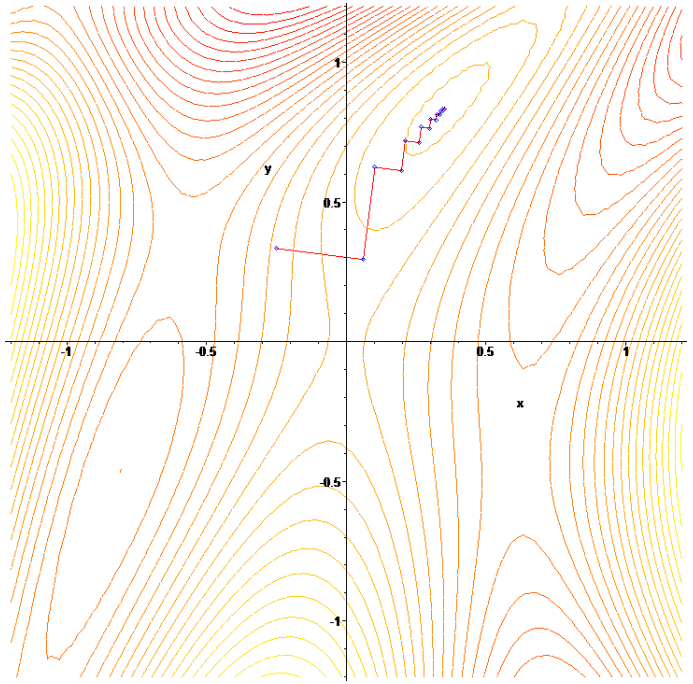
Artificial neural networks (I): the MLP

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** $\alpha_t = \frac{\alpha_0}{t+1}, \alpha_0 > 0$

2. Perform a costly **line search** to find $\alpha_t = \arg\min_{\alpha} f[\boldsymbol{x}_t - \alpha \nabla f(\boldsymbol{x}_t)]$





In the curved flat valley the method zig-zags slowly towards the minimum. Convergence can then be very slow, since it uses limited knowledge of the error function

Artificial neural networks (I): the MLP

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

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

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

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

which leads to the minimization step $\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \left(\nabla^2 f(\boldsymbol{x}_t)\right)^{-1} \nabla f(\boldsymbol{x}_t)$

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

Artificial neural networks (I): the MLP

Note there is no need for α_t, 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 $\nabla^2 f(\boldsymbol{x}_t)$ to be (strictly) p.d. (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 $\boldsymbol{x}_t$

Artificial neural networks (I): the MLP

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

$$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \alpha_t M_t \nabla f(\boldsymbol{x}_t), \text{ where } M_t \approx \left(\nabla^2 f(\boldsymbol{x}_t)\right)^{-1}$$

- The most common variant is the **BFGS method** (suggested independently by Broyden, Fletcher, Goldfarb, and Shanno)

In BFGS, the p.d. estimate of the inverse Hessian does not require matrix inversion and uses only gradient information (supplied by the BPA):

1. a p.d. matrix $M_0 = I$ is chosen
2. the search direction is set to $M_t \nabla f(\boldsymbol{x}_t)$
3. a line search is performed along this direction to find α_t
4. $\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \alpha_t M_t \nabla f(\boldsymbol{x}_t)$
5. M_{t+1} is generated using $M_t, \boldsymbol{x}_t - \boldsymbol{x}_{t-1}$ and $\nabla f(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_{t-1})$