CentraleSupélec

Machine Learning Avancé

TP2 : Stochastic approximation algorithms

Auteurs :
Lorenzo Gasparollo
Tommaso Chiara

10 octobre 2018

Table des matières

1 Batch and online solutions

We will consider : n as the size of the training set, $w \in \mathrm{R}^d$, $\{x\}_{i=1}^n \in \mathrm{R}^d$.

1.1 1. Express the gradient of F, and implement the gradient descent algorithm for minimizing it.

$$F(w) = \sum_{i=1}^{n} \log(1 + \exp -y_i w^T x_i) + \frac{\lambda}{2}\|w\|^2 \tag{1}$$

We introduce the matrix L, defined as :

$$\begin{cases} L = -Diag(y)X \\ X = \begin{bmatrix} — & x_1^T & — \\ — & x_2^T & — \\ \vdots & \vdots & \\ — & x_n^T & — \end{bmatrix} \end{cases}$$

(2)

We can re-express the eq. (1) as :

$$F(w) = \sum_{i=1}^{n} \log(1 + \exp{[Lw]_i}) + \frac{\lambda}{2}\|w\|^2 \tag{3}$$

Thus the gradient reads as follow :

$$\nabla F(w) = L^T \frac{\exp{(Lw)}}{1 + \exp{(Lw)}} + \lambda w \tag{4}$$

1.2 What is the influence of λ parameter on the quality of the classification results ?

In general we have that λ small leads to overfitting. For λ big : leads to underfitting. For nIt = 250, accuracy of 88, alpha = 30

We can use the Cross validation technique and find the value that leads to better accuracy in the test samples.

1.3 Show that F can be written in the form :

$$F(w) = \sum_{i=1}^{n} f_i(w) \tag{5}$$

f()$dependingon$(x$_i, y_i$). Trivially we get :

$$F(w) = \sum_{i=1}^{n} \log(1 + \exp{[Lw]_i}) + \frac{\lambda}{2}\|w\|^2 = \sum_{i=1}^{n} \left(\log(1 + \exp{[Lw]_i}) + \frac{\lambda}{2}\frac{\|w\|^2}{n} \right) \tag{6}$$

Thus :

$$f_i(w) = \log(1 + \exp{[Lw]_i}) + \frac{\lambda}{2}\frac{\|w\|^2}{n} \tag{7}$$

1.3.1 Deduce a stochastic gradient algo- rithm to minimize F, and implement it.

Let's see how to express f_i(x) in closed form.

$$\begin{aligned} L^T &= \big(-Diag(y)X\big)^T && (8)\\ &= -\big(Diag(y)X\big)^T && (9)\\ &= -X^T Diag(y)^T && (10)\\ &= -X^T Diag(y) && (11) \end{aligned}$$

L^T has the following form :

$$L^T = -\left[y_1 x_1 | y_2 x_2 | \dots | y_n x_n\right]$$

By inspecting jointly eq. (2) and eq. (8) we get :

$$\nabla f_i(w) = \left(y_i \frac{\exp\left([Lw]_i\right)}{1+\exp\left([Lw]_i\right)}\right)\left[x_i\right] + \frac{\lambda}{n} w \tag{13}$$

We then apply the well known stochastic descent algorithm. As we're optimizing a convex function (F(w) is a linear combination of convex functions, which is convex), in a convex set (R^d is convex), there exists a minimum, wich is unique. The gradient stochastic descent will, in principle, not have problem of convergence.

1.3.2 Display the evolution of the cost function along iterations, for several strategies for the setting of the learning rate, with and without averaging, and comment the results.

We've set the initial condition $w^0 = 0$. For lr small : the algorithm does not approach the minimum. Conversely. for lr big the algorithm diverges. If we avarage the w we observe a smoothing effect on the cost fonction. We get a result much closer to the optimum, than the non-avaraging case.

1.4 Deduce a mini-batch version of the previous stochastic gradient approach, and implement it. Study the influence of m on the convergence speed.

1.4.1 Proof of the expression

Let be m,k $\in N^*$ s.t. n = mk. By exploiting eq. (2) and by factorizing proporerly :

$$F(w) = \sum_{i=1}^{n} f_i(w) = \sum_{j=1}^{k} \sum_{i=1+m(j-1)}^{jm} f_i(w) \tag{14}$$

Thus :

$$\begin{cases} F(w) = \sum_{j=1}^{k} F_j(w) \\ F_j(w) = \sum_{i=1+m(j-1)}^{jm} f_i(w) \end{cases} \tag{15}$$

1.4.2 Study the influence of m on the convergence speed.

The more we increase the value of m :
— the less number of iterations are required for the convergence
— the higher computation time for each iteration.

1.5 Try the acceleration strategies seen in the course.

Thanks to the simulations that has been carried out, we have noticed the improvement of the speed of convergence (The abs value of the slope of the loss function is much bigger than the non-accelerated case). That leads to a problem in the calibration of the learning rate : since the loss function decrease very fast and the learning rate decays in the same way as $\frac{1}{t}$ (t, number of iteration), the algorithm diverges, as it surpass the minimum value. One simple strategy to overcome this problem consists in selecting a learning rate that decreases faster.