Optimization for Computer Science
Assignment 4

December 13, 2016

Submission: Upload your implementation and report zipped (`MatrNr.zip`) to the TU-Graz TeachCenter using the MyFiles extension.
Deadline: January 10, 2017 at 23:58h.

1 Logistic Regression

Logistic regression is one of the most famous models for binary classification in machine learning. The goal is to classify elements of a set into two distinct groups. Typically, the elements to be classified are *feature vectors* in a n-dimensional feature space. Consider a feature vector $x \in \mathbb{R}^n$. The probability that x is drawn from the class $y = +1$ is given by

$$p(y = +1|x) = \ell(w^T x + b), \quad \text{with } \ell(u) = \frac{1}{1 + e^{-u}} \tag{1}$$

where $w^T x + b = 0$ is a hyperplane with parameters $w \in \mathbb{R}^n, b \in \mathbb{R}$ and $\ell(\cdot)$ is the logistic function. Similar, the probability that x is drawn from the class $y = -1$ is given by

$$p(y = -1|x) = 1 - \ell(w^T x + b) = \ell(-(w^T x + b)) \tag{2}$$

Assume that we have a training database of C feature vectors $x^{(c)} \in \mathbb{R}^n, c = 1 \dots C$ along with their corresponding class labels $y^{(c)} \in \{-1, 1\}$. In order to find the optimal parameters w, b for the hyperplane, consider the following risk minimization problem

$$\min_{w,b} f(w, b) = \frac{a}{2}\|w\|^2 - \sum_{c=1}^{C} \log \ell(y^{(c)}(w^T x^{(c)} + b)), \tag{3}$$

where $a > 0$ is a regularization parameter to prevent overfitting.

1.1 Tasks

1. Generate training data $\{x^{(c)}, y^{(c)}\}$ with $x^{(c)} \in \mathbb{R}^2$, $y^{(c)} \in \{-1, +1\}$ by drawing from two partially overlapping 2D Gaussian distributions with appropriate means and covariances (not uniform!). Use at least 100 data samples per class.

   Visualize your training data.

2. Compute analytically the gradient ∇f and the Hessian $\nabla^2 f$ with respect to the parameters w, b. Describe all your steps in the report.

   Hint: Consider w and b as a single unknown vector $p = \begin{bmatrix} w, b \end{bmatrix}^T \in \mathbb{R}^3$ and rewrite eq. (3) in terms of p. To that end, introduce an appropriate matrix $S \in \mathbb{R}^{3\times 3}$ to obtain $\|w\|^2 = \|Sp\|^2$ and use an augmented version $\hat{x} = \begin{bmatrix} x, 1 \end{bmatrix}^T$ of the feature vectors.

3. Implement steepest descent, Newtons method and Nesterov accelerated gradient descent to optimize eq. (3). Steepest descent and Newtons method are obtained by choosing $d^k = -\nabla f(p^k)$ and $d^k = -(\nabla^2 f(p^k))^{-1} \nabla f(p^k)$ respectively in Alg. 1. Nesterov accelerated gradient descent is given by Alg. 2.

4. Visualize the optimal hyperplane together with the training data.

5. Choose a suitable regularization parameter $a > 0$. Find for each algorithm a fixed stepsize α such that the optimization converges.

6. Fix the training dataset and run the best performing method for a large number of iterations to obtain groundtruth values p^* and $f^* = f(p^*)$. Evaluate the performance of the algorithms by counting the number of iterations until the algorithm finds a solution close to the groundtruth. Use the distance of the objective function to the optimal value f^* as a stopping criterion.

   During the iterations, store the values $f(p^k) - f^*$ as well as $\|p^k - p^*\|$. At the end, plot the data. What do you find? How does Nesterov compare to steepest descent in terms of convergence speed? Include your findings in the report!

7. Generate new test examples from the same Gaussian distributions and evaluate the classification error. Include your findings in the report.

1.2 Framework

We provide a framework with the basic structure. It contains functionality to generate training data as well as a few function definitions you might find useful. You are free to change the script to you liking, as long as you use the same function for generating data samples.

You must not import any additional python modules besides the ones that are already present!

Submission Upload your implementation and report zipped (filename: `MatrNr.zip`) to the TeachCenter. Don't forget to delete your old submission.

Data: Differentiable function $f(p) : \mathbb{R}^n \to \mathbb{R}$. Choose $p^0 \in \mathbb{R}^n$ and iterate for $k \geq 0$ and a stepsize $\alpha > 0$

while *True* **do**
  compute a descent direction d^k;
  $p^{k+1} = p^k + \alpha d^k$;
  if *stopping criterion* **then**
    exit
  end
end

Algorithm 1: Gradient method.

Data: Differentiable function $f(p) : \mathbb{R}^n \to \mathbb{R}$. Choose $p^0 = p^{-1} \in \mathbb{R}^n$, $t_0 = 0$ and iterate for $k \geq 0$ and a stepsize $\alpha > 0$

while *True* **do**

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \quad \beta^k = \frac{t_k - 1}{t_{k+1}}$$
$$q^k = p^k + \beta^k (p^k - p^{k-1})$$
$$p^{k+1} = q^k - \alpha \nabla f(q^k)$$

  if *stopping criterion* **then**
    exit
  end
end

Algorithm 2: Nesterov accelerated gradient algorithm.