lions@epfl

EE-556: MATHEMATICS OF DATA: FROM THEORY TO COMPUTATION
LABORATORY FOR INFORMATION AND INFERENCE SYSTEMS
FALL 2018

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

INSTRUCTOR
PROF. VOLKAN CEVHER

HEAD TA
YA-PING HSIEH

RECITATION 1

The first part of this recitation covers the terminology and some basic notions from mathematical optimization theory, and in the second part we apply will apply gradient descent to solve binary logistic regression problem. In this part, we will also investigate the effect of the step-size on the convergence characteristics.

PROBLEM 1: TERMINOLOGY

The first and the foremost step for solving an optimization problem is to correctly understand the main objective and the restrictions of the problem. This will provide you the key elements to develop the basic reasoning that you need to evaluate and compare the performance of different methods during this course. More importantly, it will help you improve your intuition and your knowledge on choosing the best solver for a given problem a priori, which is the key challenge for a practitioner.

In optimization theory, objectives and the restrictions are expressed in a mathematical problem formulations. Below, we present a simple problem formulation and some simple definitions. Match the terms given in the right, with the elements in this formulation.

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned}$$

$$\mathbf{x}^\star \in \mathcal{X}^\star = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq f(\mathbf{y}), \forall \mathbf{y} \in \mathcal{X}\}$$

$$f^\star = f(\mathbf{x}^\star)$$

- decision variable
- objective function
- constraint
- feasible set
- solution
- solution set
- optimal value

PROBLEM 2: LOGISTIC REGRESSION WITH BREAST-CANCER DATA

Logistic regression is an important problem in statistics and it has many applications, especially in data mining, machine learning, computer vision and bioinformatics. In this problem, we will implement a gradient descent method for solving the binary logistic regression model.

We can briefly describe logistic regression as follows: Suppose that we are given a set of data points from two distinct classes with the associated class labels. The aim is to design a classifier, that can accurately estimate the class of the new data points.

Let $\mathbf{a} \in \mathbb{R}^p$ be a sample and $b \in \{-1, +1\}$ be the associated (binary) class label. The logistic regression model for the label b, given sample $\mathbf{a}$, is:

$$\mathbb{P}(b = +1; \mathbf{a}) = \frac{1}{1 + \exp(-(\mathbf{a}^T\mathbf{x} + \mu))}, \quad \mathbb{P}(b = -1; \mathbf{a}) = 1 - \mathbb{P}(b = +1; \mathbf{a}). \tag{1}$$

where $\mathbf{x}$ and μ are the parameters. It is clear that $\mathbf{a}^T\mathbf{x} + \mu = 0$ defines a hyperplane in $\mathbb{R}^p$. The probability $\mathbb{P}(b; \mathbf{a})$ is above 0.5 if $\mathbf{a}^T\mathbf{x} + \mu$ has the same sign as b, and below otherwise.

We assume that we are given a collection of n independent samples (or training data points) $\mathcal{D} := \{(\mathbf{a}_{(1)}, b_1); (\mathbf{a}_{(2)}, b_2); \cdots; (\mathbf{a}_{(n)}, b_n)\}$. The main assumption is that there are parameters $\mathbf{x}^\natural$ and $\mu^\natural$ that accurately estimates the class labels of the samples, and the aim is to approximate these parameters. If we can find a good approximation, then we can predict the class of the new data points accurately. We denote these approximations by $\hat{\mathbf{x}}$ and $\hat{\mu}$, which we can estimate by solving the following logistic regression problem:

$$\underset{\mathbf{x}\in\mathbb{R}^p,\ \mu\in\mathbb{R}}{\text{minimize}} \quad f(\mathbf{x}, \mu) := -\sum_{i=1}^{n} \log\Big(h(b_i(\mathbf{a}_{(i)}^T\mathbf{x} + \mu))\Big)$$

where $h(t)$ is the sigmoid function, i.e. $h(t) = 1/(1 + e^{-t})$.

Perform the following experiments in MATLAB or Python.

(a) First, we conduct some experiments with a synthetic toy example in 2 dimensions where we can visualize data. For this, complete and run the script `generateData`. This script first draws random parameters $\mathbf{x}^\natural \in \mathbb{R}^2$ and $\mu^\natural \in \mathbb{R}$. Then, it generate n samples in $\mathbb{R}^2$, and assigns these samples to one of the classes so that they are seperable by the line $\mathbf{a}^T\mathbf{x}^\natural + \mu^\natural = 0$. Choose $n = 100$.

SOLUTION: Classifier predicts the class of a random sample $\mathbf{a}$ as -1 if $\mathbb{P}(b = -1; \mathbf{a}) < 0.5$, and as $+1$ if $\mathbb{P}(b = +1; \mathbf{a}) \geq 0.5$. We want to assign labels of the n samples, so that the line $\mathbf{a}^T\mathbf{x}^\natural + \mu^\natural = 0$ perfectly separates the samples of these two classes. We can do this as follows, since the probability $\mathbb{P}(b; \mathbf{a})$ is above 0.5 if $\mathbf{a}^T\mathbf{x} + \mu$ has the same sign as b, and below otherwise:

```
labels = sign(features*x_true + mu_true);
```

(b) We want to solve the logistic regression problem using the gradient descent method. Implement the gradient descent completing the function `GD`.

SOLUTIONS: Recall that given some initial iterate $\mathbf{x}_0$, the gradient descent methods iterates as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k), \quad k = 1, 2, \ldots.$$

An implementation in MATLAB is as follows.

```
x = x0;
for iter = 1:maxit
    x = x - stepsize * gradf(x);
    obj(iter, 1) = fx(x);

    % The following is not necessary: Print the information.
    fprintf('Iter = %4d,  f(x) = %5.3e\n', iter,  obj(iter, 1));
end
```

(c) Solve the logistic regression problem using the function you have implemented and with the data you have generated, using the `logisticToy` script. Note that you are given a function handle that returns the gradient of the logistic function. Use different step-sizes and observe its effects on the convergence.

SOLUTIONS: Notice that if we set $\tilde{\mathbf{a}}_{(i)}^T = [\tilde{\mathbf{a}}_{(i)}^T, 1] \in \mathbb{R}^{p+1}$, and $\tilde{\mathbf{x}}^T = [\mathbf{x}^T, \mu]$, then we have

$$\mathbf{a}_{(i)}^T\mathbf{x} + \mu = \tilde{\mathbf{a}}_{(i)}\tilde{\mathbf{x}},$$

and the function to be minimized can be written as

$$f(\mathbf{x}, \mu) = \tilde{f}(\tilde{\mathbf{x}}),$$

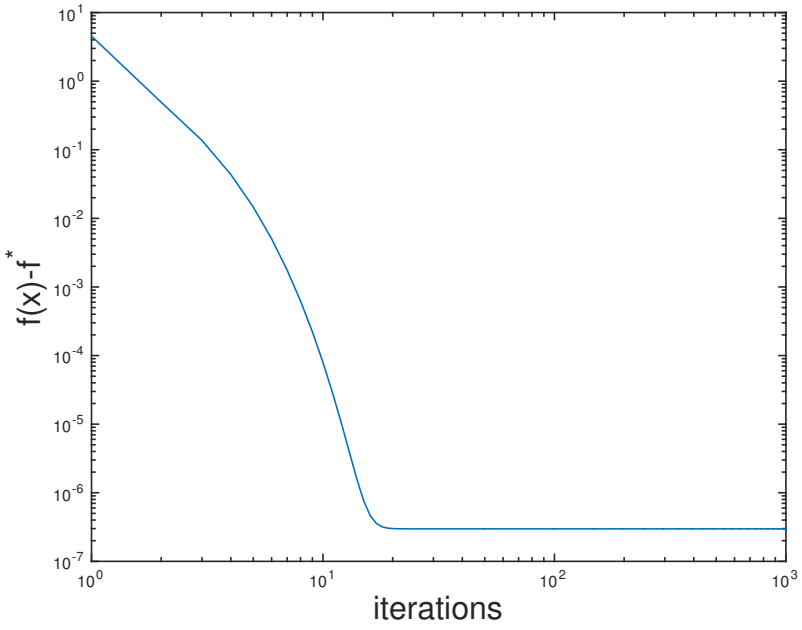
for the corresponding function $\tilde{f}$. This representation is convenient, as then we only need to minimize over all possible $\tilde{\mathbf{x}} \in \mathbb{R}^{p+1}$. This is why we appended a column of 1's to the feature matrix (called 'A' in the code).

There are only three lines to fill in. There does not exist a "correct answer" for these three lines; you can try any possible values, and observe the corresponding convergence behavior of the gradient descent method. Here we provide a standard setting:

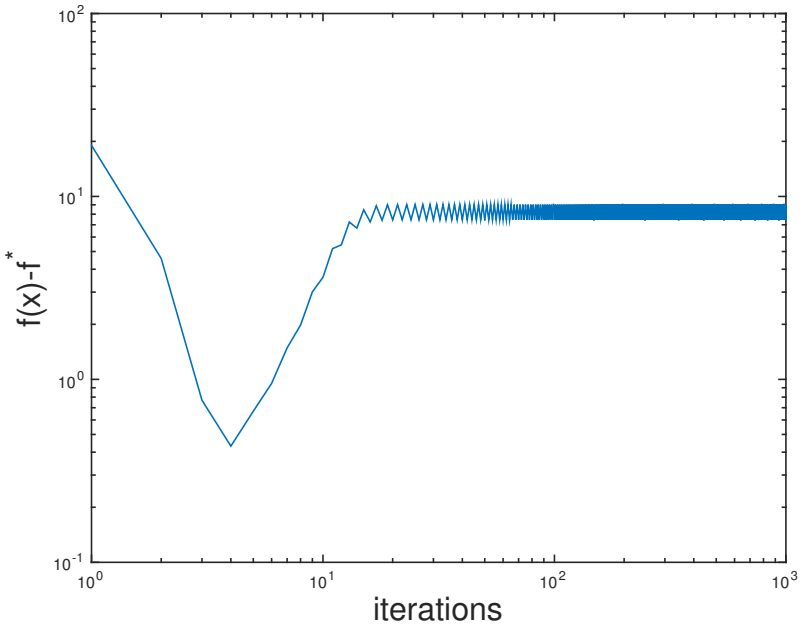
```
maxit = 1000;
stepsize = 1/(0.25*max(eig(A'*A)));
x0 = zeros(size(A,2),1);
```

You will learn soon in the course why we set the value of stepsize by such a mysterious way.

Notice that if you set the values properly, you should be able to see that the algorithm converges, as shown in the following figure. (You might want to comment out the codes specifying the limits of the y-axis of the figure.)



Moreover, you should see that if the value of "stepsize" becomes smaller, the algorithm becomes slower; if the value of "stepsize" is very large, the algorithm may not converge, as shown in the following figure.



(d) In practical situations the samples are usually not perfectly separable by a hyperplane, due to the imperfections like model mismatch, noise and outliers. Invert 20 of the class labels in your dataset, so that the data becomes noisy. Repeat the same experiments with this noisy dataset.

SOLUTION: You may make the data noisy, by changing as:

```
labels = sign(features*x_true + mu_true);
labels(end-20:end) = -labels(end-20:end);
```

(e) Finally we will solve the logistic regression with the `breast-cancer` dataset [1]. We randomly split this dataset into training and test partitions. We use the samples in the first partition to train our classifier, and use the second partition to evaluate its accuracy. Complete and run `testLogistic` script. Observe the effects of the stepsize on the convergence.

SOLUTION: Similarly to Problem 2(c), try different stepsize values and observe different behaviors. You can also use the standard setting provided for Problem 2(c):

```
maxit = 1000;
stepsize = 1/(0.25*max(eig(A'*A)));
x0 = zeros(size(A,2),1);
```

You can use the following lines to print predicted labels for the test samples

```
labels_pred = sign(features_test*xGD(2:end) + xGD(1))
```

References

[1] LICHMAN, M. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. (2013)