Introduction to Machine Learning

Nizar Ghandri
Home Assignment 1
ENS ulm

April 20, 2020

1 Formal definition

1. Formalize the problem by defining the input space X , the output space Y and the training data set. What are their dimension?

Having preprocessed the data we are going to use the images as vectors of pixels with size $n = pixel\_width * pixel\_height$. Thus: X is $[0,1]^n$ and Y is $\{-1, 1\}$ and $D = (X \times Y)$

2 Loss functions

a) What are the empirical risk (training error) and the true risk associated with the 0-1 loss? Why is it complicated to minimize the empirical risk in this case ?

The empirical risk for the 0-1 loss is :

$$R_d(f) = \frac{1}{d} \sum_{i=1}^{d} \mathbb{I}_{f(X_i) \neq Y_i}$$

and the true risk is :

$$R(f) = \mathbb{E}\{\mathbb{I}_{f(X) \neq Y} | D\}$$

The empirical risk here is of the 0-1 loss function is non-convex and discontinuous, thus (sub)gradient methods cannot be applied. Any iterative or brute force method would be exponential in the input size thus the choice of the 0-1 loss would be inadequate.

b) Why should we use the test data to assess the performance ?

Test data should be from the same statistical distribution as the training data, and should not have been fed to the learning model before (otherwise there is no point given the model already adjusted to that data before).

c) Recall the definition of the optimization problems associated with the linear least square regression and the linear logistic regression.

We start firstly with linear regression:

$$f_n \in argmin_{f\in\mathbb{F}} R_n(f) := argmin_{f\in\mathbb{F}} \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i)$$

with l being the loss function associated to the problem. In linear regression, we usually choose the square loss, and as the name states we choose a linear model: $f(x) = \theta^T x$. Thus the linear regression problem becomes:

$$min_{\theta\in\mathbb{R}^n} R_n(\theta) := min_{\theta\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} (\theta^T x_i - y_i)^2$$

That can simply be rewritten to:

$$min_{\theta\in\mathbb{R}^n} R_n(\theta) := min_{\theta\in\mathbb{R}^n} \frac{1}{n}||Y - X\theta||_2^2$$

with $X \in \mathbb{R}^{d*n}$ being the design matrix having at the row i x_i^T, and $Y \in \mathbb{R}^d$ being the vector of outputs with y_i at the position i
For logistic regression we have:

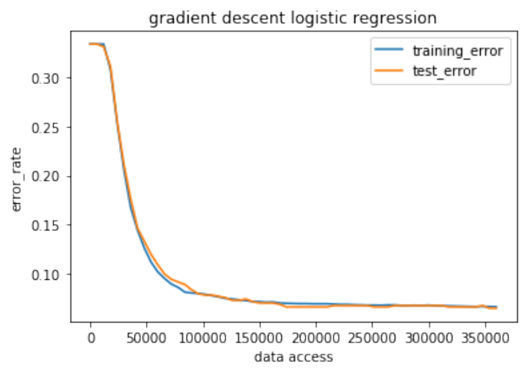
$$f_n \in argmin_{f\in\mathbb{F}} R_n(f) := argmin_{f\in\mathbb{F}} \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i)$$

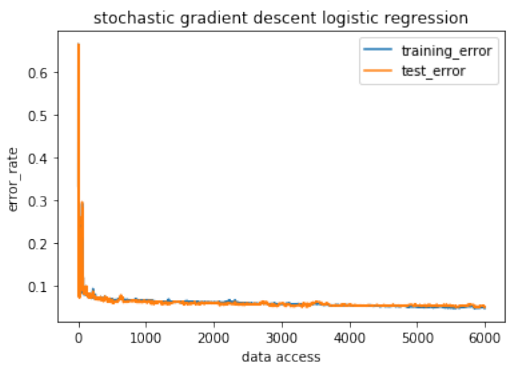
with l being the loss function associated to the problem. In logistic regression, we try to maximize the log-likelihood of an estimator and we end up with a minimization problem using the logistic loss, we choose a linear model: $f(x) = \theta^T x$. Thus the logistic regression problem becomes:

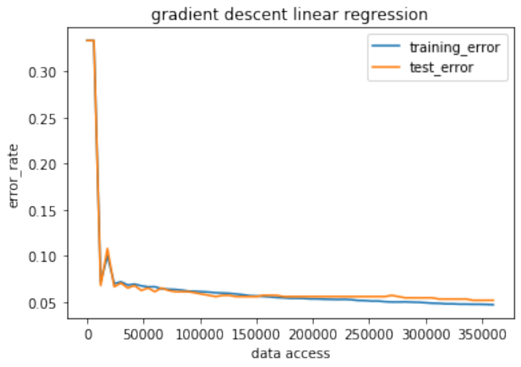
$$min_{\theta\in\mathbb{R}^n} R_n(\theta) := min_{\theta\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{y_i\theta^T x_i}))$$

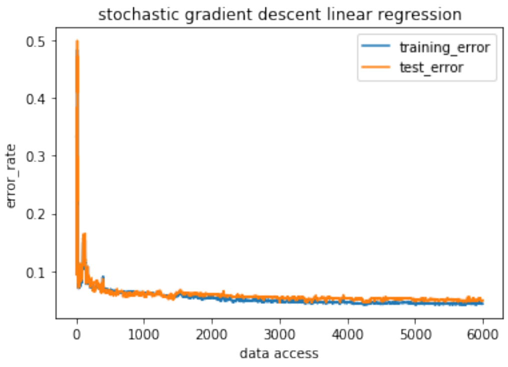
3 Implementation

a) Consider the logistic regression minimization problem. Plot the training errors and the test errors as functions of the number of access to the data points of GD and SGD for well-chosen (by hand) values of the step sizes.



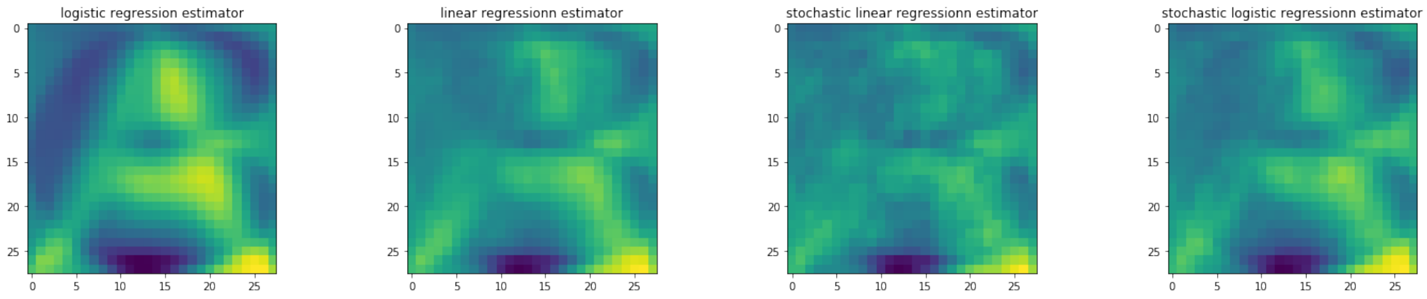






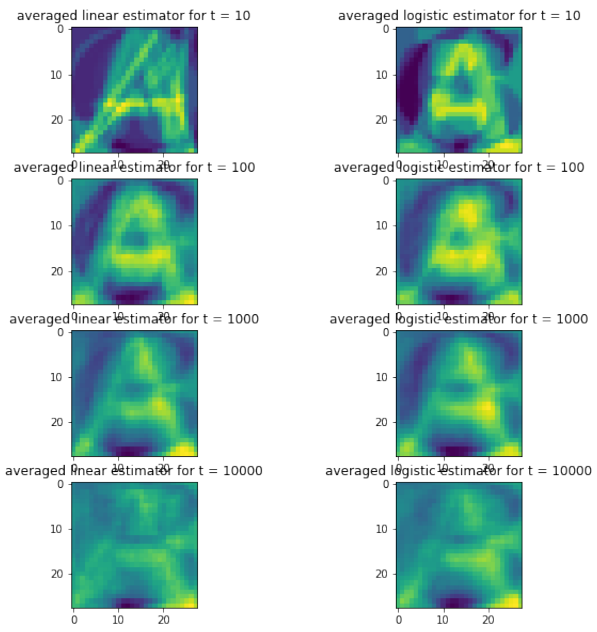
Notes:
- SGD converges a lot faster then GD.
- For SGD to get even a faster and smoother convergence one could actually increase the batch size from 1 to 10 for example (in practice around 120 in our case should work fine).
- The learning rate could be picked by hand, but a constant learning rate is not optimal we could use algorithms such as the backtracking line search to find a good value for the learning rate in each iteration.
- The number of iterations can be also decided depending on the precision we require. Thus we add a stopping condition to the algorithm. One very common one is $||\nabla f(x)||_2 \leq \epsilon$ with ϵ being the precision hyper parameter.

b) Plot the estimators $\beta_n^{logist} \in \mathbb{R}^{28x28}$ and $\beta_n^{lin} \in \mathbb{R}^{28x28}$ respectively associated with the logistic and linear regression as two images of size 28 x 28.



We can see an A getting drawn. This normal because each estimator during training is trying to pick up the features of an A.

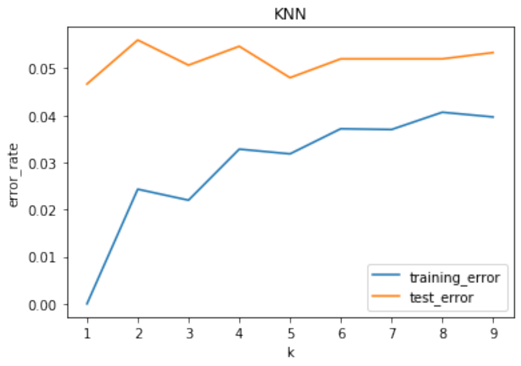
c) Denote by $\beta_n^{logist} \in \mathbb{R}^{28x28}$ the estimator of logistic regression after t gradient iterations of SGD. Plot as images the averaged estimators $\beta_{avg}^{logist} = \frac{1}{t}\sum_t \beta_n^{logist} \in \mathbb{R}^{28x28}$for $t \in \{10, 100, 1000, 10000\}$.Repeat for the linear regression estimator.



4 KNN

a) Recall the definition of the k-nearest neighbors classification rule with l_2 metric.
In k-NN classification, the output is a class membership. An object is classified to the class to which most of its k nearest neighbors belong (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. Here we define nearest to be $argmin||X - Y||_2$ with X being the data set, y being the vector of reference and Y being the matrix with the vector y as its columns.

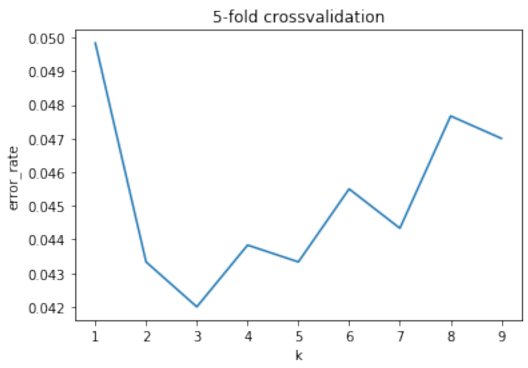
b) Implement it and plot as a function of k, its training and test errors.



Notes:
- The implementation of the function KNN can be heavily parallelised, with each thread executing the knn_one_example.
- An interesting idea would be not to use all of the data set for training but instead take all the samples of A and randomly take only 1000 sample of B and 1000 sample C. This would balance the training set (2000 A, 2000 non A). Training on this yields better results. However as I didn't know if I was allowed to do that for this assignment I switched back to

the old data set.

c) Calibrate k using K-fold cross-validation with K=5.



5 Comparison

	logistic regression	linear regression	KNN
Empirical error	0.07	0.05	0
Test error	0.07	0.05	0.04