Pattern Classification and Machine Learning: miniproject

Paul Wormser - Gevorg Poghosyan

June 2012

Abstract

The aim of this project is to implement two different machine learning methods: Error backpropagation gradient descent optimization for a multi-layer perceptron(MLP) and a support vector machine algorithm: SMO. We will apply those methods to the MNIST dataset which is composed of pictures of handwritten digits. For this report we assume that the reader knows the concepts of MLP and SVM.

1 The Dataset

For MLP, we will try to distinguish handwritten digits 3 from 5 and 4 from 9 using validation set and early stopping mechanism. For SVM, we will just solve the 4vs9 problem using cross validation for finding the optimal parameters.

1.1 Preprocessing the Data

We were given two .mat files (*mp_3-5_data.mat* and *mp_4-9_data.mat*). Each of those files consits of 4 sets (matrices): Xtest, Xtrain, Ytest and Ytrain. The X-sets consist of the images of digits (rows represents the images and columns the pixel values). The Y-sets are the corresping targets (column vector of values +1 and -1) for images from Xtest and Xtrain.

For both MLP and SVM we have normalized the values from Xtest and Xtrain using only pixel values from the Xtrain. This results of pixel values from Xtest and Xtrain $\in [0, 1]$. We would like to notice that our algorithmswill work also without normalization of datasets. The normalization aims to reduce computation cost.

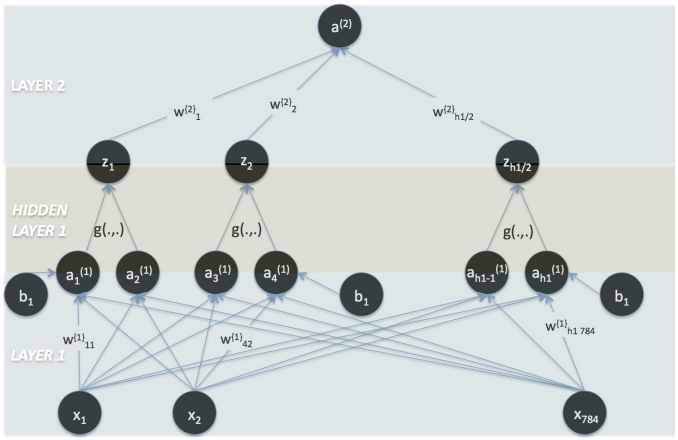
In the sets Xtrain and Xtest, the images are ordered so that the first half of the images has corresponding targets +1 and the second half -1. In order to have a random order of images we have randomized all the 4 datasets keeping the index correspondance between patterns and their corresponding targets.

For early stopping implementation in MLP, we will need a part of data for validation. So we have split the training set into 2 parts (2/3 for training and 1/3 for validation).

2 Multi-Layer Perceptron

2.1 Principle

A Multi-Layer Perceptron is a neural network classification algorithm. The MLP classifies the points by mapping them to multidimensional space and separating the patterns by a hyperplane in that space. In our project we applied MLP using online learning classification with stochastic gradient descent. The architecture have chosen is with one hidden layer. The interested reader is encouraged to try an architecture with more hidden layers.



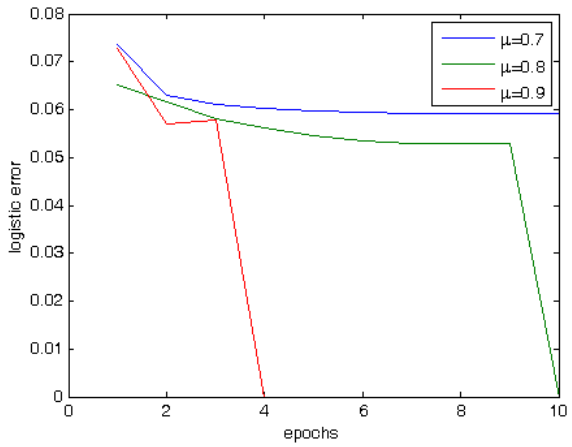
On the picture you can see the three layers, the input $\vec{X}(X_1$ to X_{784} are the pixel values of an image), the weight vectors $w^{(1)}$ and $w^{(2)}$, activations $a^{(1)}$ and $a^{(2)}$, first layer activation bias and the transfer function g. In our project we didn't use a bias for the second layer.

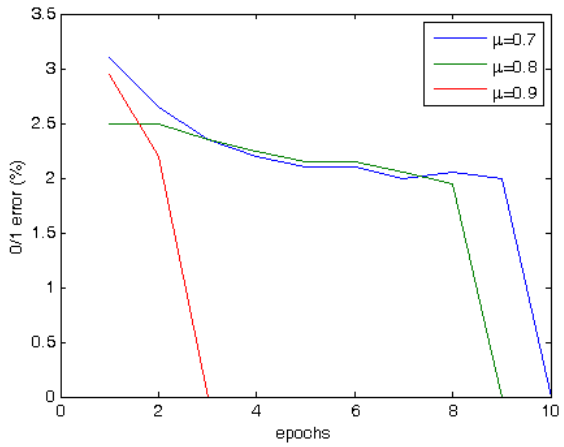
2.2 Methods

The free parameters for the MLP are the learning rate(η), the momentum term(μ) and the number of units in the hidden layers($h1$ for the first hidden layer, $h2$ for the second layer etc.). Also the initialization of the weight vectors' and biases' values must be done properly in order for the updates to be of the same order as the values being updated. We initialized weight vectors on the first and the second layers with normal distributions with means 0 and variances 0.1 and 0.5 respectively. $b^{(1)}$ values are initialized in the same way as $w^{(1)}$.

2.2.1 Final Setup

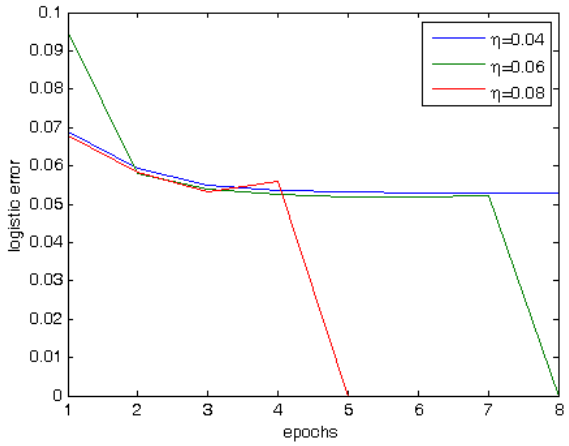
In this section we are describing the parameters' choice only for 3vs5 sub-task, because the heuristics for tuning the parameters for the other sub-task is the same. However we have got different "best" values for 2 sub-tasks and we talk more about this in the next section. So the best configuration we have achieved for the chosen architecture is with the following parameter values: $\mu = 0.8, \eta = 0.06, h1 = 64$. On the following plots you can see how the variation of the parameters influences the logistical error and the $0/1$ error rate both for training and validation sets.

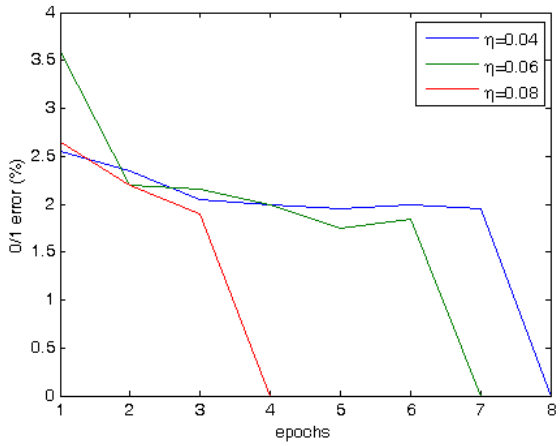




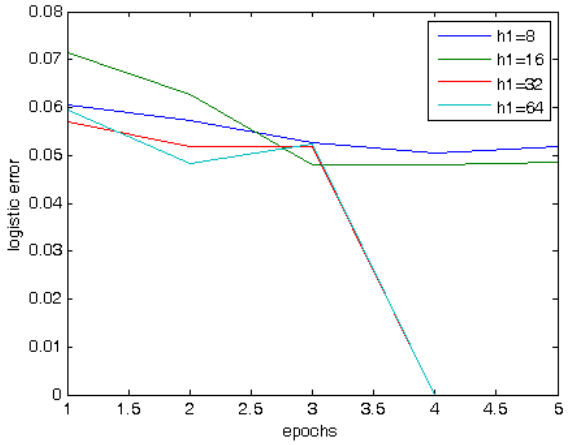
On each of the error plots above you see 3 curves for 3 different momentum term values: $\mu = 0.7, \mu = 0.8, \mu = 0.9$. The intuition about the momentum value must be given by the idea that it averages the last

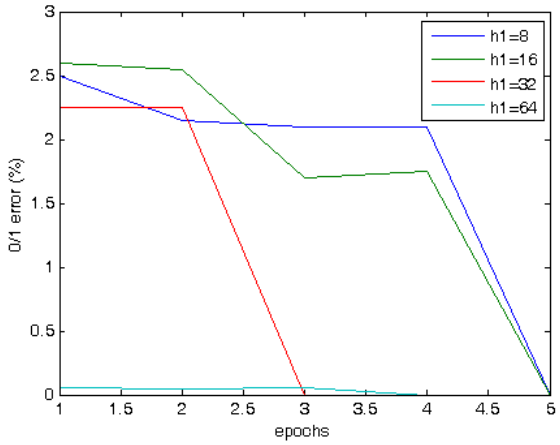
gradients with decay μ. We can see that for larger momentum terms the algorithm converges faster, but for that we pay with larger error. For smaller momentum term the convergence arrives later and not necessarily with better error rates.





On each of the error plots above you see 3 curves for 3 different learning rate values: $\eta = 0.02, \eta = 0.04, \eta = 0.08$. These plots clearly show that the learning rate is a tradeoff of convergence time and the least error that could be achieved.

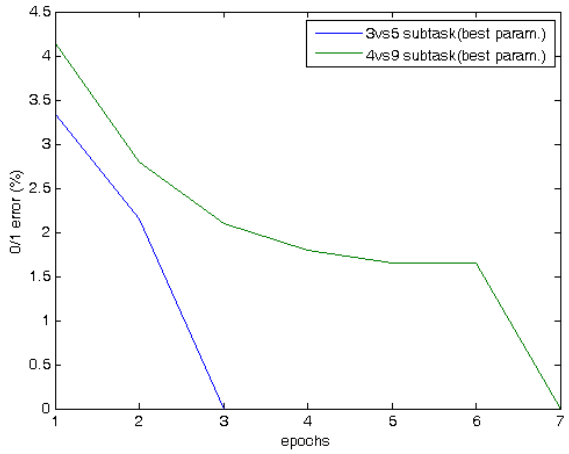


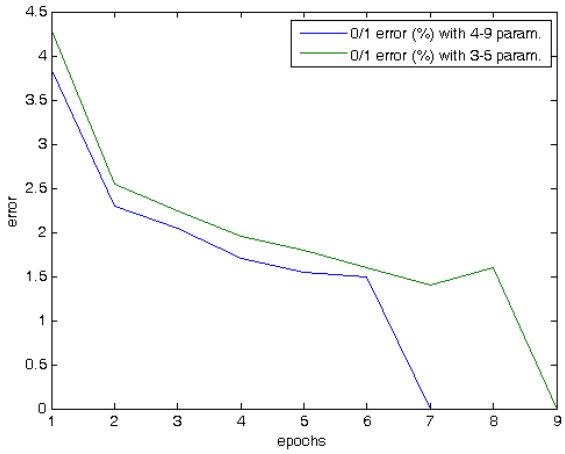


On each of the error plots above you see 4 curves for 4 different number of units in the hidden layer: $h1 = 8, h1 = 16, h1 = 32$ and $h1 = 64$. Here we get really interesting plots that show how fun is it to play with MLP parameters. The plot on the right hand side shows that the MLP gives just 3 misclassified patterns from 2000. This sounds pretty good and it converges as fast as the MLP with 32 units in the hidden layer. What we can see from the plot on the left is that the latter MLP has smaller logistic error, which means the separating hyperplane "better" fits the patterns. Nevertheless neither a smaller logistic error and nor a smaller $0/1$ error are not a warranty that the MLP will work as good also on another set, e.g. on the testing set that we were given.

2.2.2 Differences Between the Two Binary Sub-task Solutions

In this section we try to reveal the differences between 2 binary sub-task solutions(3vs5 and 4vs9). By the plot on the left below we illustrate the convergence rates and error values for each sub-task when running with the best chosen parameters for each. During our work, we found that the best values for 4vs9 sub-task MLP are $\mu = 0.9, \eta = 0.08, h1 = 64$. Also on the plot on the right side you can see the implementation of 4vs9 sub-task MLP with the chosen parameters from sub-task 3vs5. The conclusion is that for each new problem one must learn the "best" parameters from a scratch.

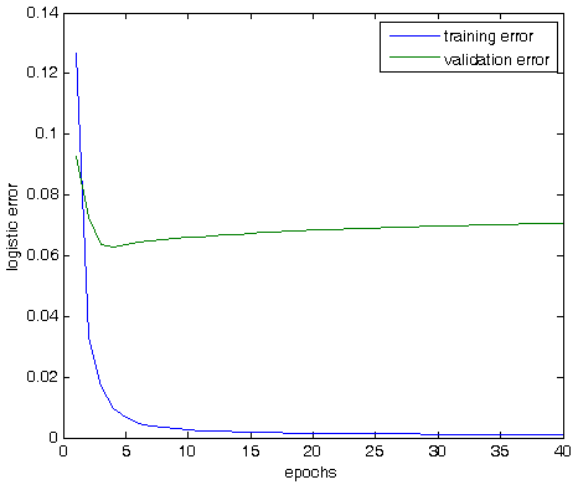


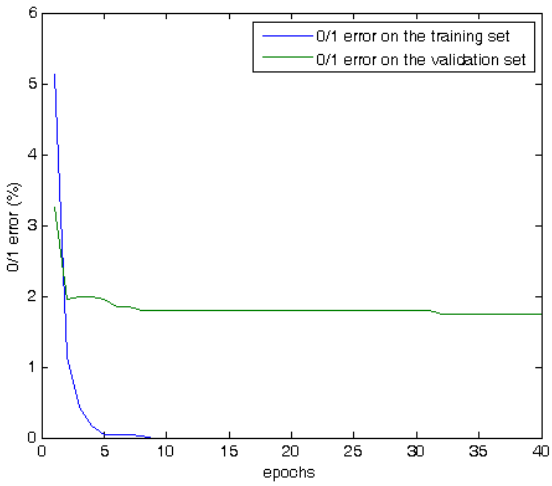


Here we present the results only for 0/1 error. For logistic error both MLPs with the best parameters reach the same results. For the graph on the right, logistic error behavior was the same as it is for 0/1 error.

2.2.3 Example of Overfitting

For any given dataset, if you leave the algorithm to learn on it too much, it will overfit the data. This results in worse error rates on new datasets, than it was possible. This is the motivation to use the validation set and to monitor the classification error on it during the learning process. The growth of logistic error on the validation set is the number 1 indicator of overfitting. Below we illustrate 2 plots showing the logistic error and 0/1 error both on training and validation sets.



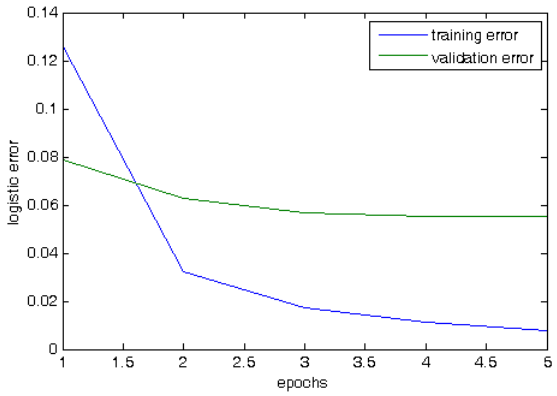


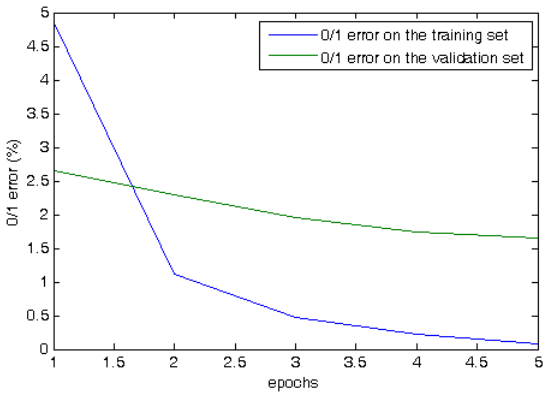
After the 3rd epoch logistic error on validation set starts to grow. Regardless the fact that the 0/1 error on the validation set decreases, we may get worse result with this weight vectors on the test set. The training error must always decrease during the learning process(and that's what you can see), if it's not the case, it means there is a bug in the code.

2.3 Results and Discussion

2.3.1 Model Evaluation

On the graphs below we evaluate the performance of our multilayer perceptron showing the minimum error levels that we have achieved.





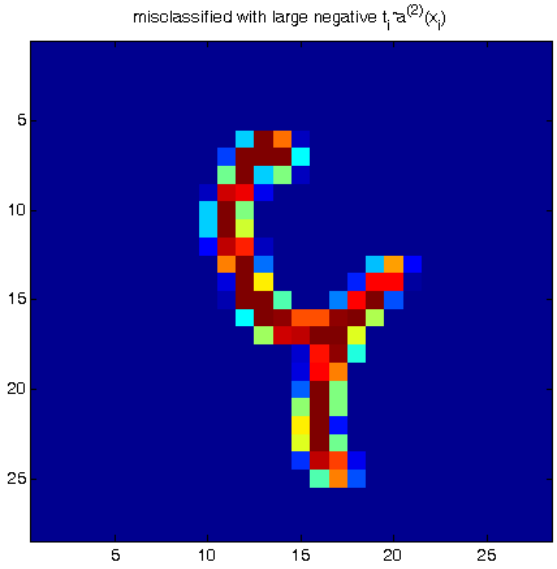
Note that the plots are taken when the early stopping mechanism was implemented and at the 5th epoch the algorithm converged. during the 6th epoch the logistic error on the validation set started to grow.

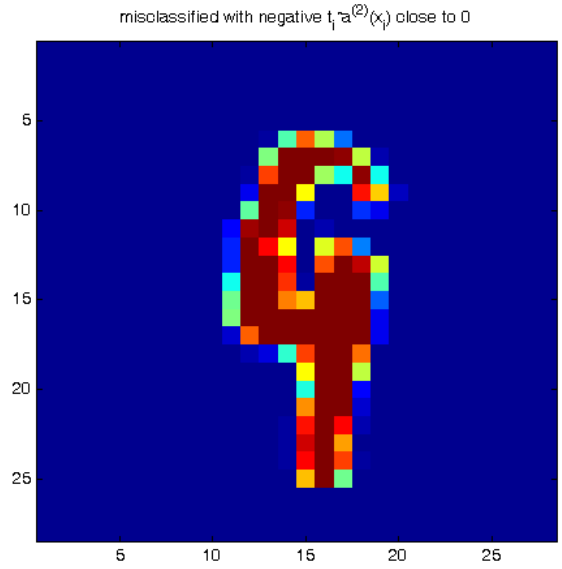
2.3.2 Results on the Testing Set

After learning on the training set with the best chosen parameters we tested our obtained weight vectors and biases on the testing set. The interesting thing for the 3vs5 problem was that achieved 0/1 error was 1.73% (33 misclassified patterns from 1902) which is as good result as it was for the validation set, but the logistic error was infinitely large. An explanation for this can be that the image is very difficult to classify even for a human. This kind of situation can be avoided by bounding a large logistic error for a pattern. For the 4vs9 sub-task we have achieved 2% of 0/1 error (41 misclassified patterns from 1991) and the logistic error was again very very large.

2.3.3 Examples of Misclassified Patterns

Below we show two misclassified patterns from 4vs9 sub-task. The first one has a large negative $t_i a^{(2)}(x_i)$ value and the second has a $t_i a^{(2)}(x_i)$ value close to 0.





3 Support Vector Machine

3.1 Principle

SVM is a classification problem of maximizing the margin between the separating hyperplane and the nearest datapoints from it. The aim is to make the classifier resistant to a noise in the data. Choosing the margin size is a tradeoff of number of misclassified points and robustness to a noise. SMO is one of the solutions for the SVM classification problem.

3.2 Choice of C and τ

The free parameters for the SMO algorithm are C and τ. The larger the C, the less is classification error, but we may pay with larger error on the test set. τ is the kernel parameter which defines the feature space mapping coefficients. To choose the best possible C and τ we have run the algorithm on 10 different values of each parameter and evaluated the risk for each of them. On the matrix below you can see the risk values for chosen parameters and our choice highlighted.

	τ=0.0005	τ=0.001	τ=0.002	τ=0.004	τ=0.008	τ=0.016	τ=0.032	τ=0.064	τ=0.128	τ=0.256
C=1	5.8333	5.1333	4.5833	3.8833	2.85	2.1667	1.3	0.9833	1.0667	6.9833
C=2	5.1667	4.6833	4.1167	3.2333	2.55	1.7333	1.1333	**0.9167**	1	6.9333
C=3	4.9833	4.4	3.8333	2.9667	2.2833	1.55	1.1333	0.95	1	6.9333
C=4	4.7667	4.25	3.5833	2.8333	2.1667	1.4667	1.1167	0.9333	1	6.9333
C=5	4.5833	4.1	3.4667	2.7	2.0833	1.3833	1.1667	0.9333	1	6.9333
C=6	4.4667	4.0167	3.3333	2.6667	2.0333	1.3667	1.15	0.95	1	6.9333
C=7	4.3167	3.8333	3.2667	2.5833	1.95	1.3167	1.15	0.95	1	6.9333
C=8	4.3333	3.8333	3.2	2.5167	1.8833	1.3	1.1333	0.95	1	6.9333
C=9	4.2	3.75	3.1667	2.4667	1.8	1.25	1.15	0.95	1	6.9333
C=10	4.1333	3.65	3.1	2.35	1.7167	1.3167	1.1167	0.95	1	6.9333

So you can see that we have chosen C=2 and $\tau = 0.064$ as the best parameters. The risks were estimated using cross-validation on the training dataset. The full dataset was divided into 10 folds and for each pair of C and τ the algorithm was learning on data from 9 folds and validating on the 10th. For each C and τ we were iteratively choosing all the 10 folds as a validation set for the rest of the learning data. Then, the squared error was averaged over those 10 small learning-validation runs.

3.3 Results and Discussion

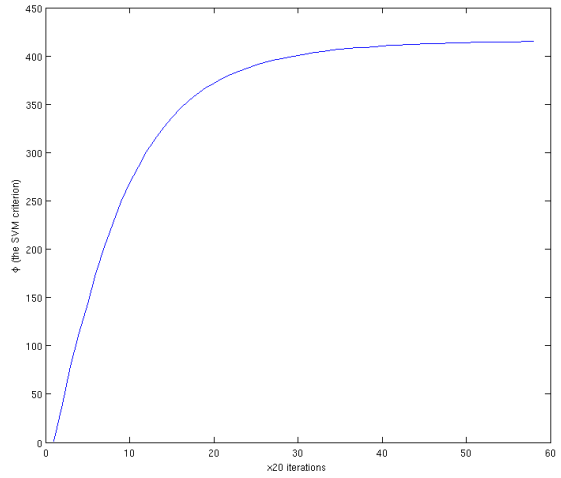
3.3.1 Reasoning the Parameter Choice

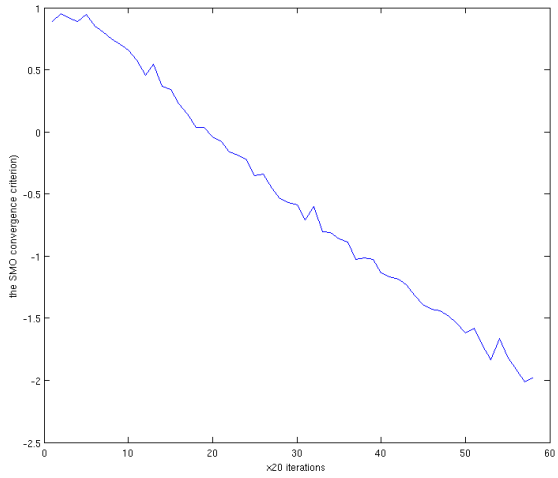
Checking the 10-fold cross-validation scores in the matrix from the previous section you can see that good results are also achieved for other combinations of C and τ but the one we have chosen has the smallest parameters, which means it's the most robust to the noise.

3.3.2 Training and Testing

After we have obtained the "best" C and τ from the cross validation, we run SMO on the full training set with these parameters. On the plots below we illustrate the SVM criterion (Φ) maximization and the conver-

gence criterion based on KKT condition violations. The horizontal axis show the number of iterations(1:20 iterations) of SMO algorithm till convergence. The vertical axis for the second graph has a logarithmic scale. Note that the smallest value of the convergence criterion has a value $exp(-2) \approx 2 * \tau = 0.128$.





3.3.3 0/1 error

As an output from the training over the full dataset we take vector α and the bias. With those values we classify the testing patterns. The 0/1 error we have obtained for training and testing datasets are both 0! Yes, 0! This means that the SMO works perfectly and solves the SVM classification problem.

4 MLP vs SVM

So to conclude, in this project we have implemented 2 powerful algorithms for pattern classification. We have used the same datasets of 4s and 9s for both algorithms and the difference is explicit. Not to much to comment here, and not too much surprise. As we have learned during the course, SVM performs better than the MLP. For comparison see the bar plots.