Mini-project Report

Samuel Humeau 222974, Benyounes Moumni 179848

May 24, 2013

Abstract

The goal of this project is to implement and train a multilayer perceptron to classify images from the small NORB dataset. The dataset has 10'800 images of objects in 5 different categories: four-legged animals, human figures, airplanes, trucks, and cars. The train set contains 5400 samples while the test set contains 5400. In Section 1 we describe some implementation decisions and discuss the result for the first task (binary classification). We describe the implementation of the MLP for multi way classification in section 2. We conclude in section 3.

1 Binary Classification

The first task is to implement a binary classifier using a 2-hidden layer perceptron with logistic error. the architecture of the network is described in the project description. We will first describe how we derived the gradient formulas then show the results.

1.1 Normalization

We use the normalization procedure as described in the project guidelines. We normalize the training set and save the mean and variance. These values are used to normalize the validation set and test set.

1.2 Early Stopping

After observing the evolution of the validation error after each epoch, we noticed that stopping after the validation error start increasing is not a good idea. Indeed the curve is highly variable during the first epochs. Instead we average the validation error over 4 epochs, that way, small fluctuations will be ignored. Since we have never observed an overfitting effect (increasing of validation error), we decided to stop when the improvement becomes insignificant to speed up the training time i.e. the new averaged error over 4 epochs is higher than 0.95 times the old one. $error(window) > 0.95 \times error(window - 1)$.

1.3 Results & Discussion

We tried the classifier with several configurations of number of layers and learning rate. We found that all instances of the validation set are correctly classified after the first epoch with a very low error. It seems that the two class are very easily separable, this explains this good performance. In this case there is not much optimization possible : since a logistic error is used, increasing the weight of the last layer makes tend to 0, no matters the hyper parameters. Therefore we arbitrarily chose, for the following results, a learning rate of 0.001 and a momentum of 0.05.

1.3.1 New dataset

In order to verify our assumption concerning the separability of the two classes, we created another dataset, using the samples of truck and cars. The idea is that some other classes are less separable and would allow

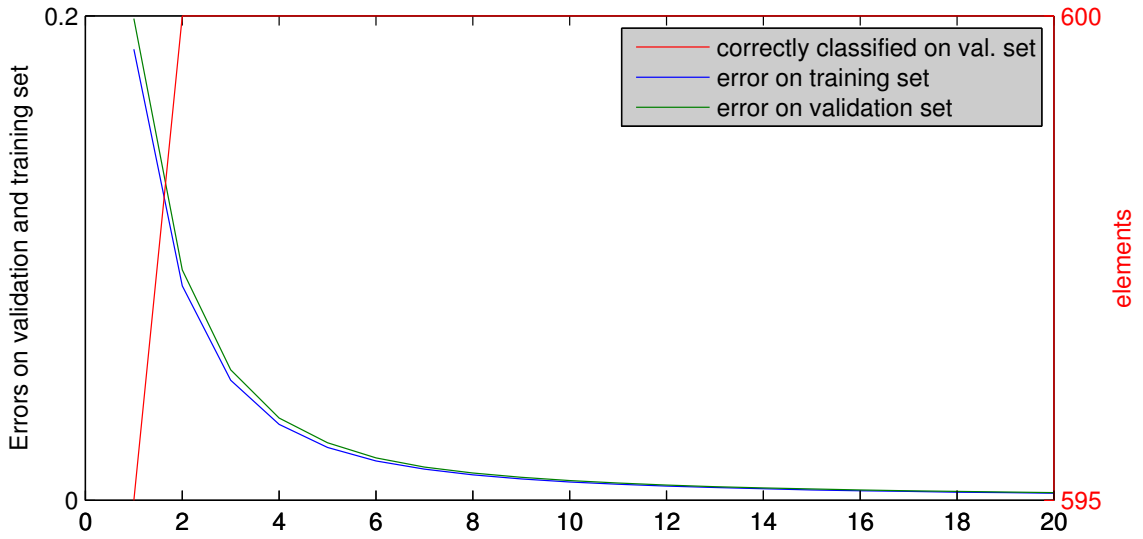

Figure 1: h1=10 h2=5 mom=0.05 learning_rate=0.001

Minimum	Mean	Standard deviation	Maximum
0.7%	3.5%	2.0%	7.5%

Table 1: Error rate on the test set with optimal parameters over 15 runs

us to have a better idea of the performance of our implementation. We have not driven formally a study of it, but we observed the classification on the validation set is higher than in the case Truck/Figurines.

2 Multi-way classification

In order to have elements of comparison for the multiway MLP we are going to implement, it is important to have other solutions as baselines. Indeed it is expected that a MLP outperforms all linear solutions. Therefore we have implemented 3 linear classifiers described in the next three sections.

2.1 Linear regression with squared error

For this part, each datapoint belongs to one of 5 class. Thus, the label attributed for each datapoint is no more a scalar but a binary 5-dimensional vector. For example if x belongs to the class number 2, its label will be $\widetilde{t_i} = [0, 1, 0, 0, 0]$. Let consider the classification error on the training set to be :

$$E_2(W) = \sum_{i=1}^{N} \left\| y_i - \tilde{t}_i \right\|^2$$

The linear classifier computes the 5-dimensional output : $y_i = W[x_i; 1]$ (we add a constant coordinate in each data points in order to incorporate the bias in the matrix W). Then the class is determined by $argmax_j y_i(j)$. In the case of the squared error above, the course shows that the matrix W that optimizes the regression on the training set is given by :

$W_{optimal} = (\Phi^T \Phi)^{-1} \Phi T$ where Φ is a matrix where each column is a datapoint, and T is the the matrix that regroup the corresponding labels.

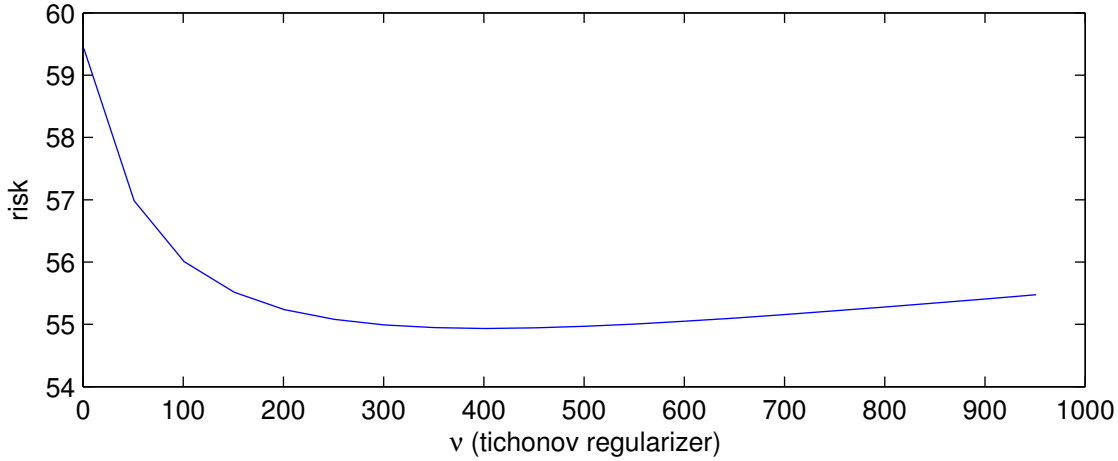

Figure 2: Evolution of the risk (defined in section 2.2) with the tichonov regularizer.

By computing this classifier, we obtain 1228 misclassified elements on the test set (over a total of 5400 data points), which means an accuracy of 77%.

2.2 Squared error with Tichonov regularizer

A classic improvement of linear regression is to constrain the weights using a tichonov regularizer. The error is now :

$$E_2(W) = \frac{1}{2}\sum_{i=1}^{N} \left\| y_i - \tilde{t}_i \right\|^2 + \frac{\nu}{2} \left\| W \right\|^2$$

In this case, for a fixed ν, the course gives us the optimal weights for the regression on the training set (using the same notation introduced above) :

$$W_{tichonov} = (\Phi^T \Phi + \nu I)^{-1} \Phi^T T$$

In order to find the best regularizer ν, we compute a 10-folds cross validation on the training set, for different values of ν. We have selected the regularizer that minimizes the risk, which is computed as follow : after splitting the training set in 10 equals parts, $risk = \frac{1}{10}\sum_{m=1}^{10} R^{(-m)}(W^{(-m)})$, where $R^{(-m)}$is the error E_2 computed only of the m^{th}part of the classifier trained with the 9 other parts of the training set.

Figure 2 shows the evolution of the risk with ν. Since minimum is obtained for $\nu \approx 400$, we have take this value to build the regularized classifier. This one makes a good result on the test set, with only 817 misclassified elements (85% accuracy).

2.3 Logistic regression

Another improvement is to consider a logistic error :

$$E_{log}(W) = \frac{1}{2}\sum_{i=1}^{N} \text{lsexp}(y(x_i)) - \tilde{t_i^T} y(x_i)$$

There is no analytic solution of this kind of error. We are forced to implement a gradient descent. The course p. 145 gives the value of the gradient :

$$\nabla_{y_i} E_i = \sigma(y_i) - \tilde{t_i} = [\frac{e^{y_i(k)}}{\sum_k e^{(k)}}]_{k=1...5}$$

μ	Averaged error on the validation set
0.01	0.9185
0.005	0.1700
0.001	0.3200
0.0005	0.4188

Table 2: Optimization of μ

μ	Mean	Standard deviation	Minimum	Maximum
0.005	19.6%	1.5%	17.2%	22.9%

Table 3: Error rate on the test set over 15 runs

For early stopping, we have simply taken a validation set of $1/3$ of the whole training set. Early stopping is triggered whenever the error on the validation does not decrease significantly ($error(epoch) > 0.95 * error(epoch-1)$). Table 2 and 3 gives the result of our optimization of the learning rate, and the result of the optimal classifier (learning rate = 0.005) on the test set.

2.4 Multi-Class MLP

2.4.1 Support for multiple classes

There are several ways to extend our binary MLP to support multiple classes. We can do pairwise binary classification or K one-vs-all binary classifiers and then have one linear layer to make a decision based on that. We chose however another solution which is to extend the last layer to support multiple classes by setting its size to K=5. The target values are therefore in the form $t_i = [0,0,1,0,0]$ for class $k = 3$ for example. As described in the guidelines, we used a squared error for training.

2.4.2 Hyper-parameters Search

Now that the MLP is working and the gradient has been tested, we need to find the optimal parameters – namely $H1, H2$, learning rate (μ) and momentum (ν) . We run two hyper-parameters search, one for $H1, H2$ and one for ν and μ. We assume naively that the sets of parameters are independent, although we found that it is a reasonable assumption in practice.

Number of neurons The first search has been done on a small cluster of 4 nodes and 32 cores during 14 hours. The values for both H1 and H2 are 10,20,30,40,60,80,100. We also try two values of learning rates and momentum term to increase the confidence in the result. Moreover, we average the results for a single configuration over 3 runs. The results are shown in Figure 3. Based on these number we decided to select $H1 = 60$ and $H2 = 40$. The difference between nearby values is relatively small and doesn't justify running another search around those values.

Learning rate and Momentum The second search have been done on a single quad-core computer and took a couple of hours. The value that were tested are for the learning rate: 0.001,0.005,0.01,0.05 and for the momentum: 0, 0.01, 0.05, 0.1 with H1=60, H2=40. The results are shown in Tableau 4.

2.5 Results & Discussion

Finally our best hyper parameters were $H1 = 60$, $H2 = 40$, $\mu = 0.01$, $\nu = 0$. Figure 4 shows the evolution of the error on the validation set for this case. The execution on the test set (Table 5), has given about 90% accuracy, which beats every linear classifiers we have implemented. The confusion matrix 5 shows that some classes are similar (like truck and car, and human and animals). In some cases, depending on the point

A.

μ=0.01, v=0							
H1\H2	10	20	30	40	60	80	100
10	0.0118						
20	0.0101	0.0092					
30	0.0120	0.0098	0.0104				
40	0.0132	0.0112	0.0083	0.0091			
60	0.0091	0.0083	0.0084	0.0079	0.0084		
80	0.0083	0.0091	0.0080	0.0079	0.0091	0.0085	
100	0.0086	0.0092	0.0082	0.0084	0.0087	0.0086	0.0087

μ=0.01, v=0.05							
H1\H2	10	20	30	40	60	80	100
10	0.0191						
20	0.0115	0.0103					
30	0.0120	0.0113	0.0110				
40	0.0095	0.0099	0.0102	0.0093			
60	0.0106	0.0096	0.0089	0.0096	0.0095		
80	0.0093	0.0091	0.0112	0.0084	0.0083	0.0082	
100	0.0097	0.0088	0.0081	0.0080	0.0079	0.0082	0.0087

μ=0.001, v=0							
H1\H2	10	20	30	40	60	80	100
10	0.035						
20	0.042	0.030					
30	0.031	0.023	0.028				
40	0.024	0.023	0.023	0.027			
60	0.024	0.024	0.030	0.021	0.031		
80	0.026	0.022	0.024	0.027	0.025	0.022	
100	0.024	0.023	0.021	0.026	0.022	0.025	0.023

μ=0.001, v=0.05							
H1\H2	10	20	30	40	60	80	100
10	0.039						
20	0.031	0.033					
30	0.028	0.026	0.027				
40	0.028	0.027	0.026	0.027			
60	0.027	0.023	0.026	0.025	0.025		
80	0.025	0.025	0.024	0.025	0.025	0.024	
100	0.026	0.024	0.024	0.025	0.024	0.023	0.024

B.

μ=0.01, v=0							
H1/H2	10	20	30	40	60	80	100
10	30.7						
20	34.7	36.0					
30	30.7	38.7	36.0				
40	26.7	28.0	38.7	36.0			
60	36.0	41.3	33.3	38.7	36.0		
80	37.3	36.0	34.7	38.7	32.0	37.3	
100	38.7	34.7	36.0	33.3	32.0	37.3	30.7

μ=0.01, v=0.05							
H1/H2	10	20	30	40.000	60	80	100
10	25.3						
20	34.7	38.7					
30	29.3	33.3	33.3				
40	36.0	38.7	32.0	37.3			
60	33.3	33.3	38.7	38.7	40.0		
80	37.3	30.7	29.3	41.3	37.3	41.3	
100	37.3	29.3	38.7	36.0	40.0	40.0	36.0

μ=0.001, v=0							
H1/H2	10	20	30	40	60	80	100
10	49.0						
20	40.0	40.0					
30	48.0	49.0	49.0				
40	48.0	49.0	49.0	48.0			
60	49.0	49.0	32.0	48.0	32.0		
80	49.0	48.0	44.0	48.0	40.0	49.0	
100	44.0	48.0	49.0	40.0	49.0	48.0	48.0

μ=0.001, v=0.05							
H1/H2	10	20	30	40	60	80	100
10	41.3						
20	43.0	47.3					
30	46.7	49.0	47.0				
40	44.0	46.0	47.3	48.3			
60	45.3	48.0	47.0	48.3	47.0		
80	46.7	44.3	48.7	46.0	47.0	48.0	
100	48.7	45.7	47.3	45.7	47.0	45.3	47.0

Figure 3: Hyper-parameter search for H1 and H2, 2 different values of learning rate and momentum. A. Shows the final error on the vaidation set B. Shows the number of epochs. The number of epochs is limited to 50 for speed of convergence reasons

$\mu \backslash \nu$	0	0.01	0.05	0.1
0.001	0.0251	0.0251	0.0224	0.0242
0.005	0.0106	0.0141	0.0114	0.0109
0.01	**0.0075**	0.0082	0.0095	0.0095
0.05	0.0123	0.0082	0.0078	0.0107

Table 4: Optimization of learning rate and momentum, with H1=60, H2=40

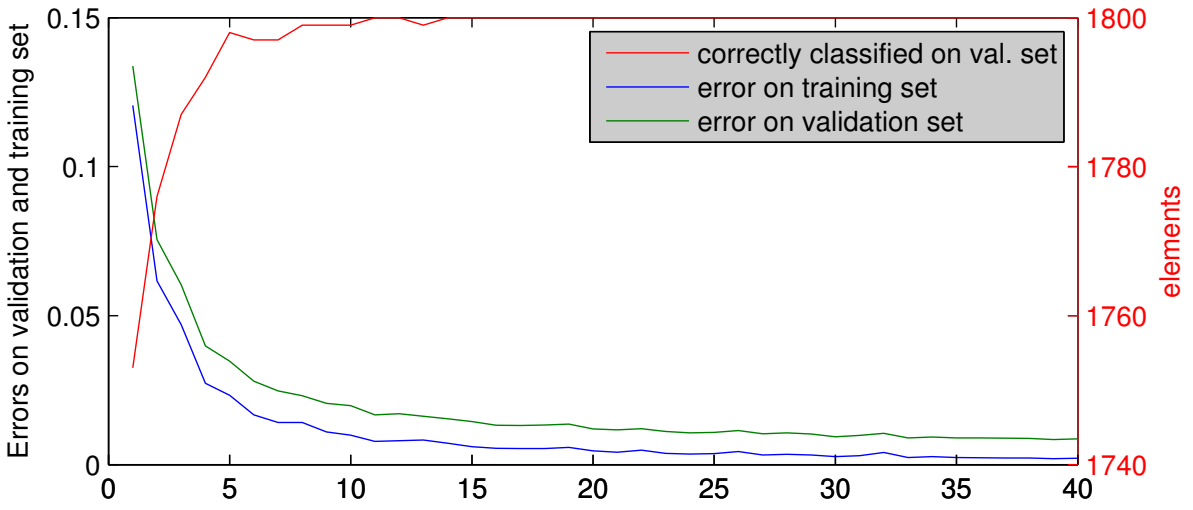

Figure 4: Error on multi-class classification task with parameters: $H1 = 60$, $H2 = 40$, $\mu = 0.01$, $\nu = 0$

of view where the picture has been token, the ambiguity can be high. For an example, figure 6shows one pattern that is not well classified by the binary classifier, and we notice that even with a human eye it is hard to determine which class it belongs.

3 Conclusion

In general we have achieved the main goal of the project which is to train an MLP for multiway classification of the NORB dataset. We have finally achieve a system that is 90% accurate, which is good and on par with reported results for the NORB dataset (even if recent research on deep learning outperforms largely any other methods). The project was helpful to teach us the subtlety and caveats of training neural networks.

Minimum	Mean	Standard deviation	Maximum
9.0%	10.7%	1.0%	12.3%

Table 5: Error 0/1 on the test set with optimal parameters over 15 runs

Confusion matrix (averaged over 15 launches)						
		Interpreted as				
		animal	human	plane	truck	car
real class	animal	1000.5	66.6	11.7	0.0	1.2
	human	89.8	954.4	4.7	0.0	31.1
	plane	72.0	4.9	976.7	0.5	26.0
	truck	0.8	0.0	0.0	1065.1	14.1
	car	9.3	0.0	5.7	237.9	827.1

Figure 5: Confusion matrix on the test set with optimal parameters

Figure 6: A. Large negative $\boldsymbol{t}_i \boldsymbol{a}_i^{(3)}$ -10 B. Small negative $\boldsymbol{t}_i \boldsymbol{a}_i^{(3)}$ -0.2 C. Well classified 9.8 (on the binary classifier)