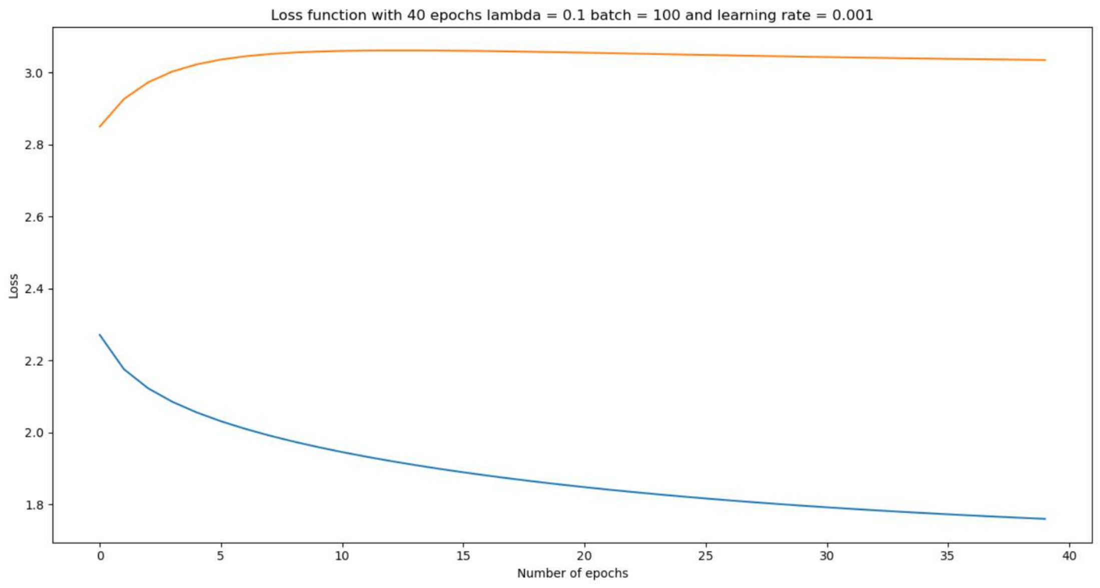
Assignment 1: Deep Learning in Data Science

Sebastián Barbas Laina

ssbl@kth.se

Part 1:

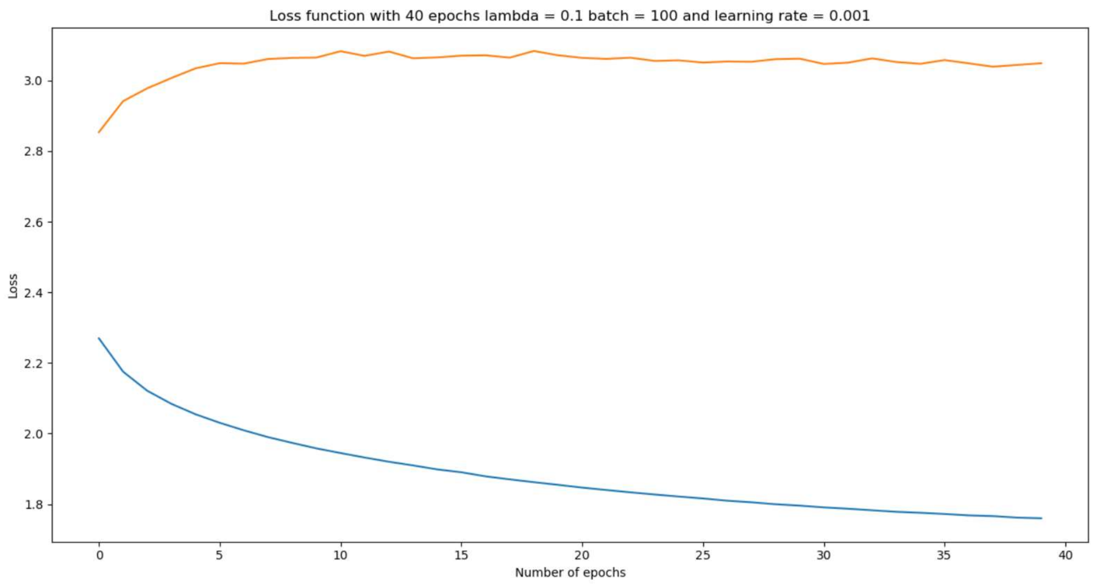
I will start by adding the graph from the mandatory part of the lab which gave less final cost, in order to be able to compare the different methods implemented in this part.



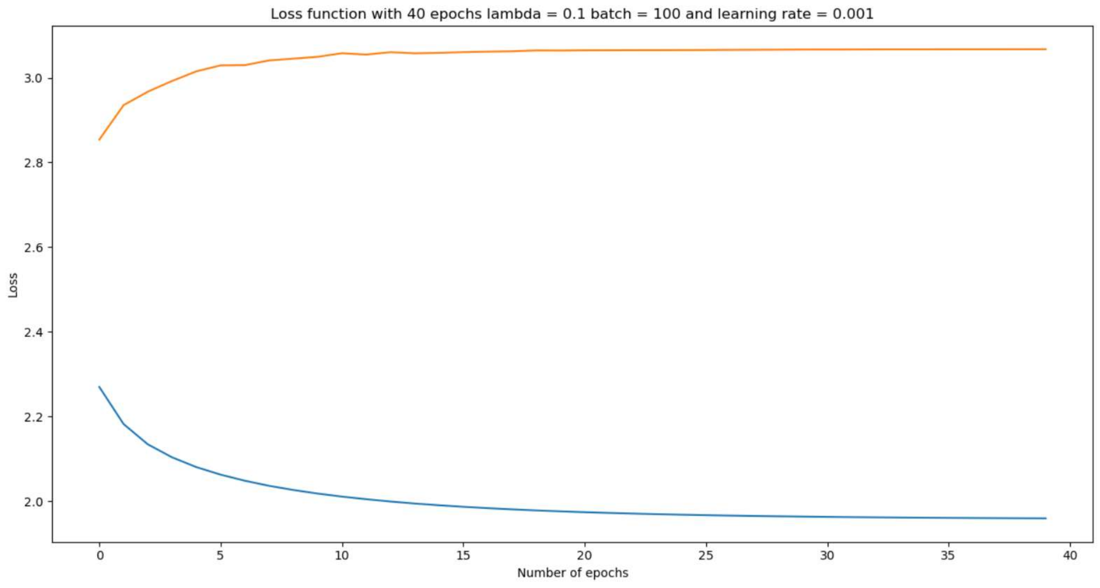
I tried several methods in order to improve the accuracy on the testing dataset. Some of them improved the accuracy whilst others did not have any effect on it.

The first method I tried was the ensemble method. This method did not yield any sort of improvement. The main issue with this method is that it is good with classifiers which have high variance and low bias errors. In our specific case, the bias is really high and the variance quite low, due to the fact that we are working with a very simple network. Due to this all the networks had similar outputs, yielding a similar accuracy (0.3741) as the one obtained with one network, just that now we used 10 classifiers.

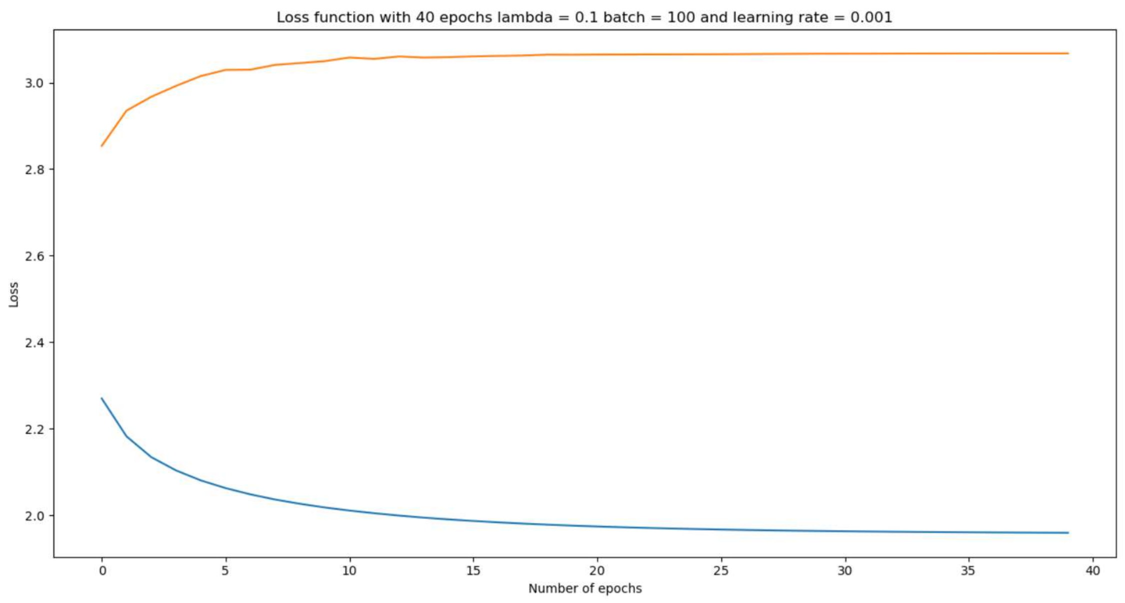
Afterwards, I tried improving the network by shuffling the dataset in each epoch. With this method I obtained the following cost curve and an accuracy of 0.3943:



Another improvement I implemented was the learning rate decay. In each epoch, the learning epoch was multiplied by 0.9. The following costs were obtained with an accuracy of 0.3809 if both the learning rate and the shuffling were incorporated:



If on top of this, we do a Xavier initialization, we obtain an accuracy of 0.3809:

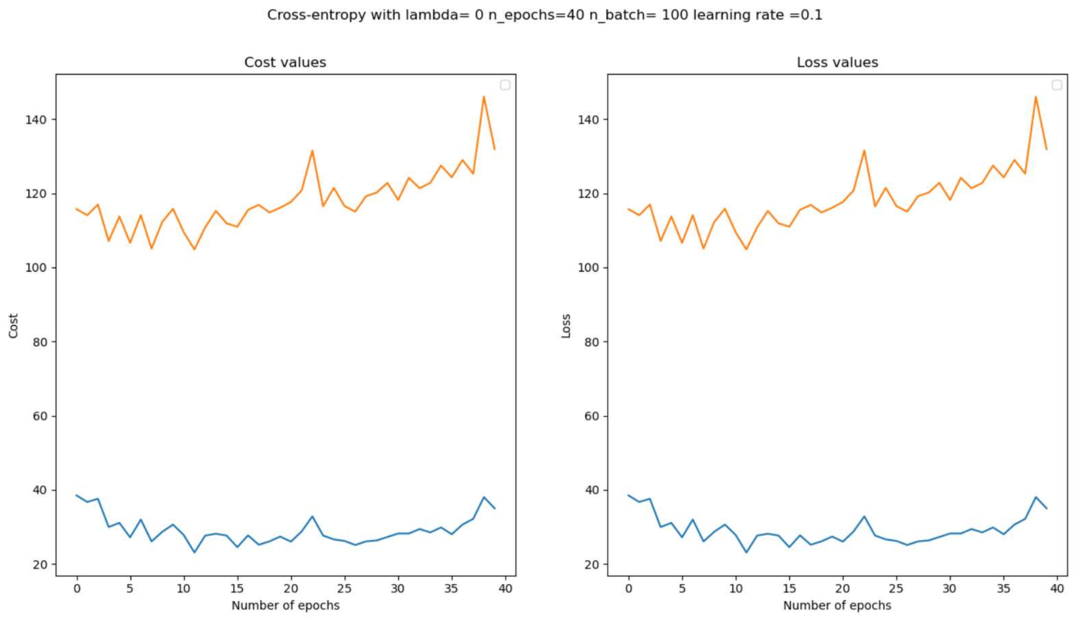


Finally a graph search was implemented in order to find the best possible combination of hyperparameters in order to achieve the best accuracy in the test set. The hyperparameters tested were: 0.1, 0.01 and 0.001 for the learning rate; 0.01, 0.1 and 0.5 for the shrinkage penalty; 20, 50 and 100 for the batch size and finally 40 and 70 for the number of epochs during the training phase. The best hyperparameters were: 0.1 for the learning rate, 0.1 for the shrinkage penalty, 100 for the batch size and 70 for the number of epochs. With these hyperparameters the accuracy obtained in the validation set was of 0.3987.

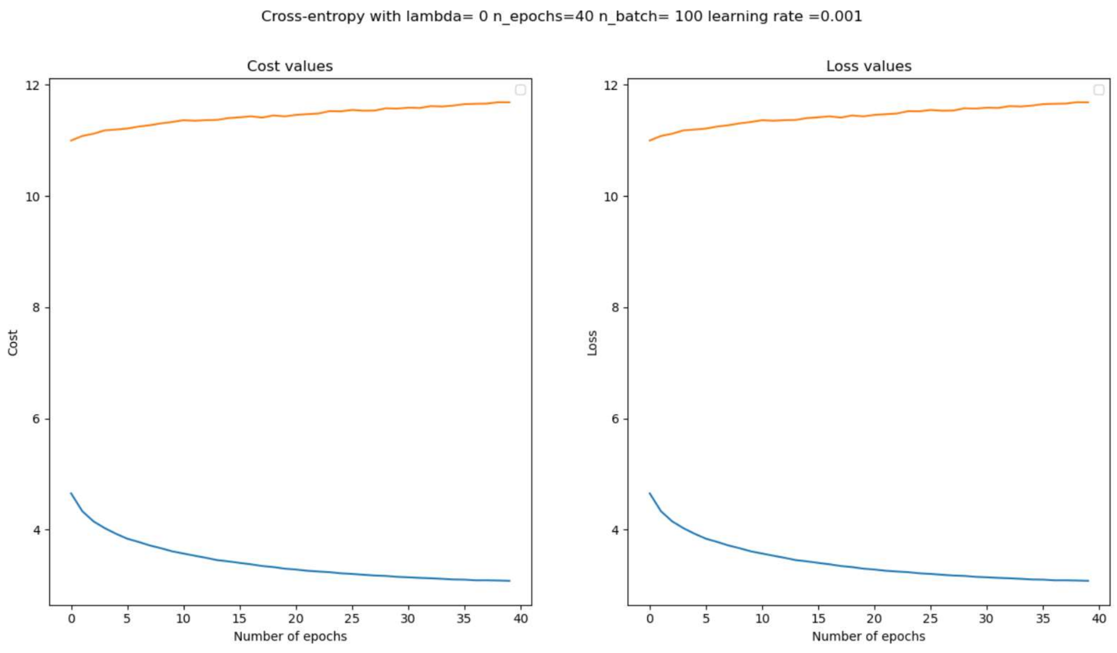
Part 2: SVM:

First, we will perform the same experiments as for the mandatory part of the lab. It is important to note that the cost and loss values cannot be compared with the ones with cross-entropy loss, as svm method gives higher penalties for mismatches, therefore, obtaining higher cost and loss values.

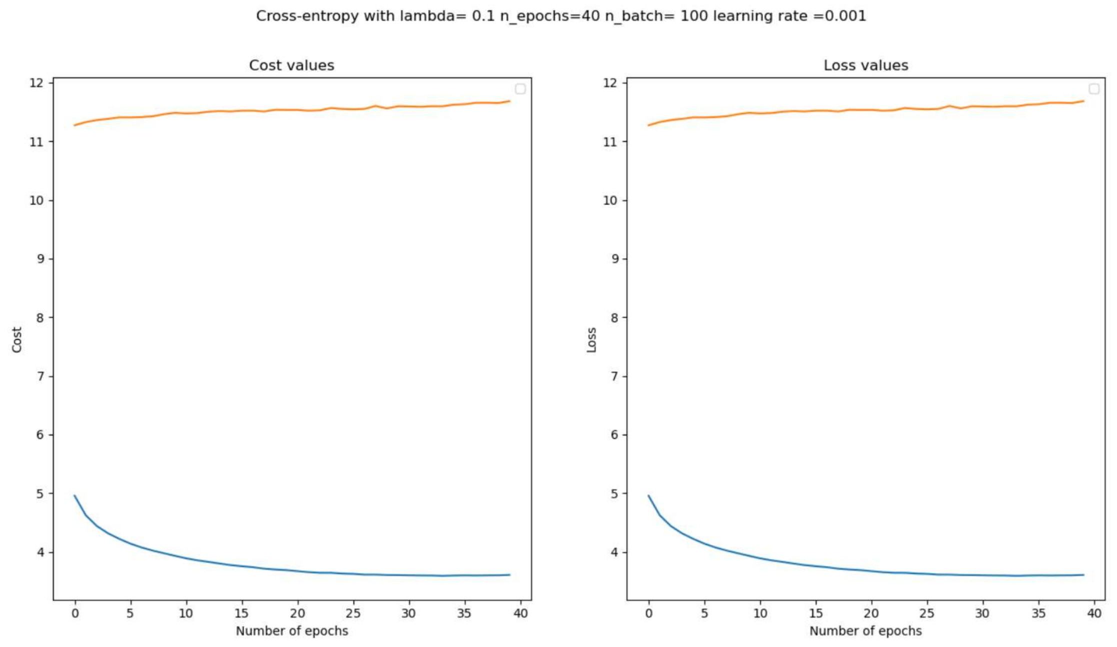
First we did the experiment with a shrinkage penalty of 0, 40 epochs, 100 images per batch and a learning rate of 0.1. With these settings these are the cost and loss functions:



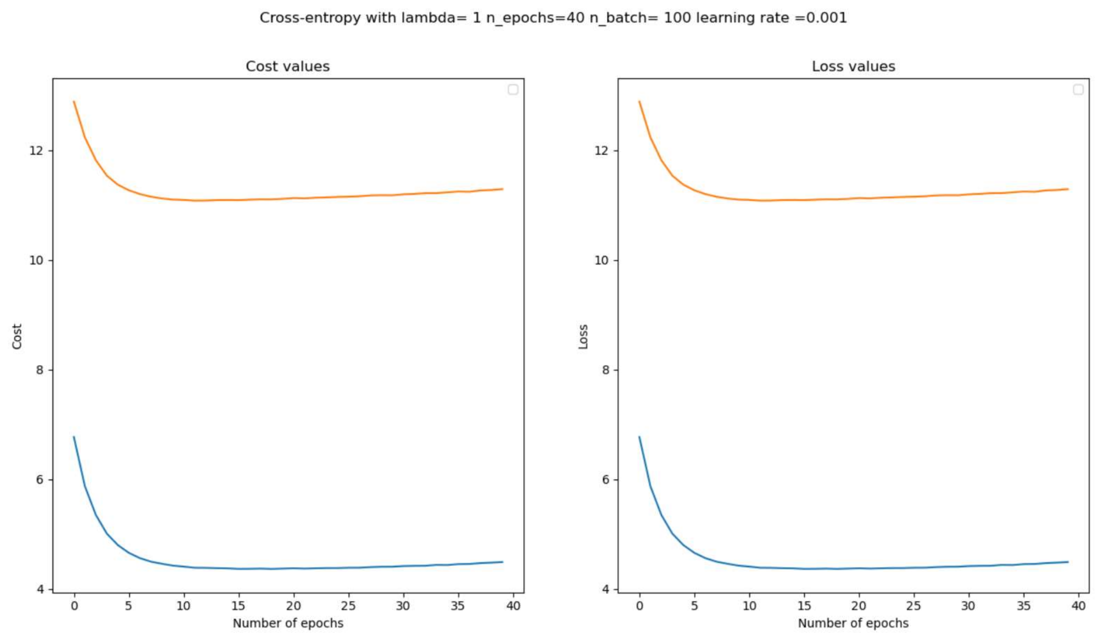
For the next experiment we had the same hyperparameters as before, but changing the learning rate to 0.001, achieving the following results:



Next, with changes the shrinkage penalty from 0 to 0.1 and left the other hyperparameters with the same values as before, obtaining the following graphs:



For the last experiment, I changed the lambda value from 0.1 to 1, achieving the following cost and loss functions:

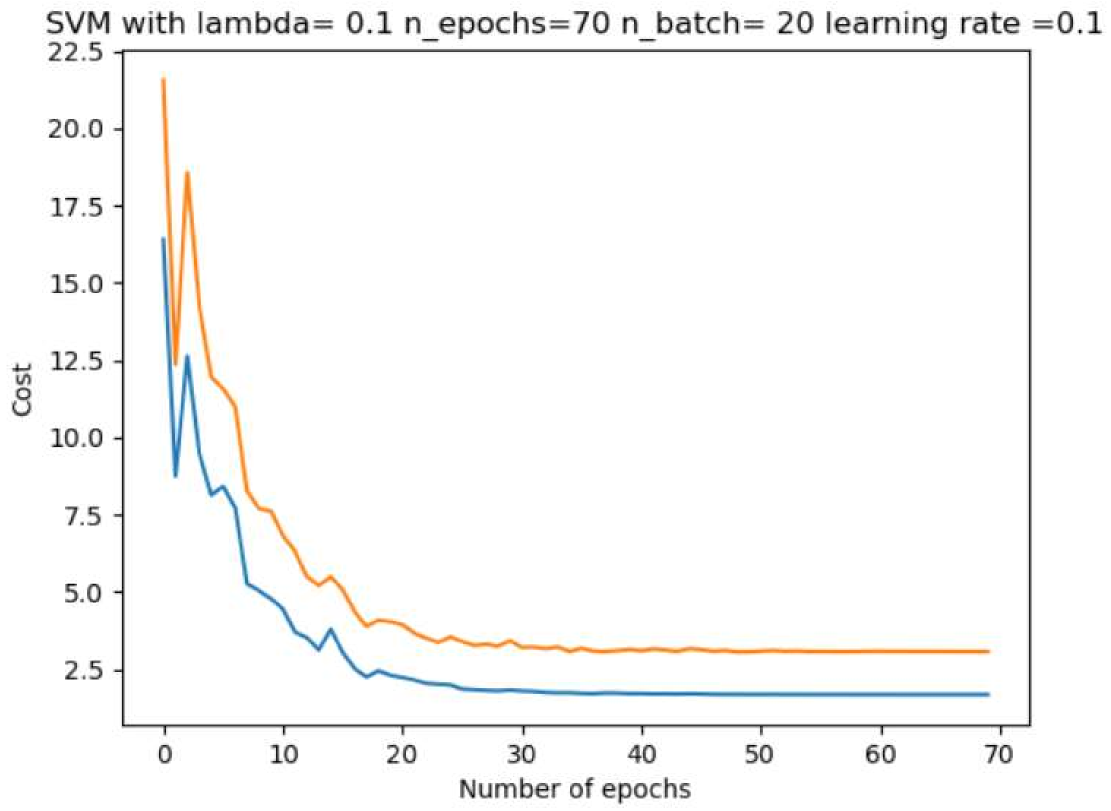


N_Epoch	Lambda	N_Batch	Eta	Accuracy
40	0	100	0,1	0,2092
40	0	100	0,001	0,342
40	0,1	100	0,001	0,3469
40	1	100	0,001	0,3465

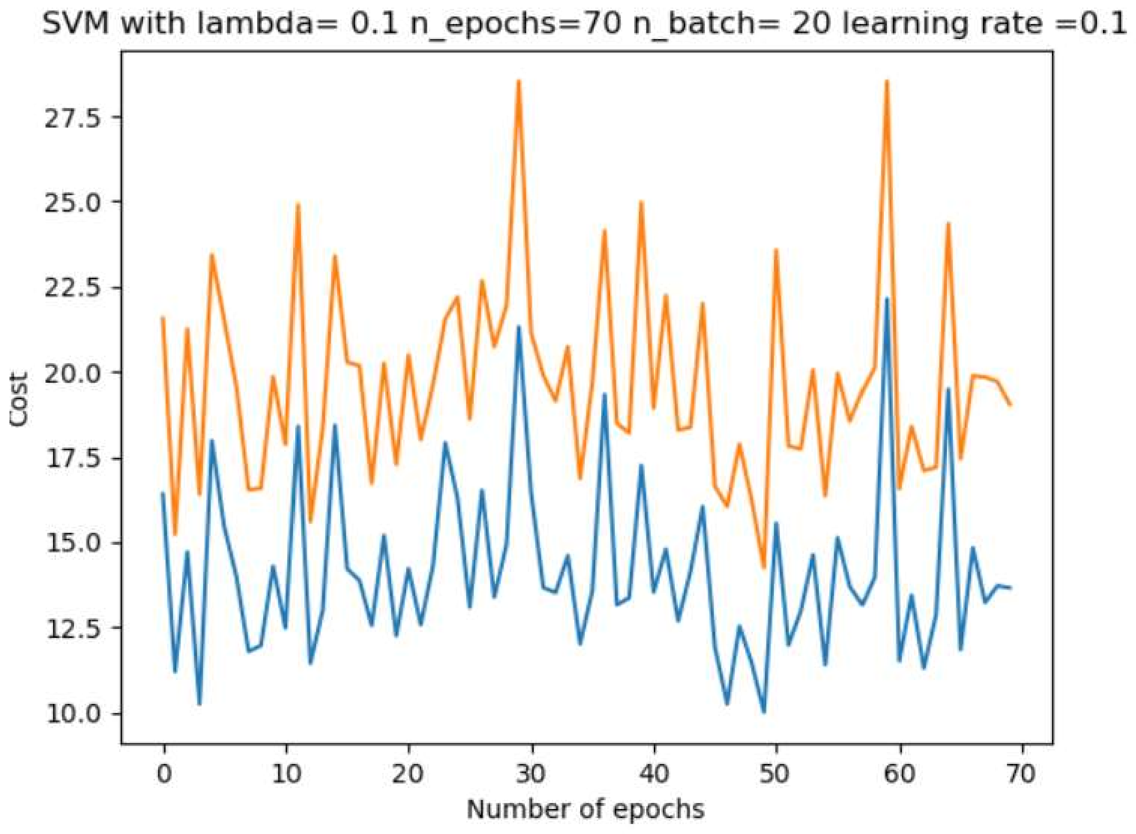
As it can be seen, the accuracy is lower with svm than with cross-entropy, making it a worse method to train a neural network.

Now, I am going to apply graph search with the same hyperparameters as before (except for the epoch which I will leave at 70, with Xavier initialization, shuffling and with decay in the learning rate.

For the graph search, it was obtained that the best hyperparameter were: 0.1 for the learning rate, 0.1 for the shrinking penalty and 20 for the batch size, yielding an accuracy of 0.3973 in the test set and the following cost function:



If there is no decay in the learning rate we obtain the following graph with an accuracy of 0.203, so for svm having a decrease in the learning rate is crucial.


SVM with lambda= 0.1 n_epochs=70 n_batch= 20 learning rate =0.1
Cost
Number of epochs