MSc Machine Learning

DD2424 Deep Learning in Data Science

Assignment 1 Bonus: Image classification with 1-layer network

---

Author:
Alexandros Ferles
ferles@kth.se

Professor:
Josephine Sullivan

Contents

1 Exercise 1: Optimize the performance of the network

Try at least 3 improvements to help bump up performance and report your results

The following optimizations have been performed:

1. **Training for a larger number of epochs and using the validation set for early stopping**. Early stopping was performed after raining for 91 epochs.

   The following plot shows the cost evolution over each epoch for the training and validation datasets:

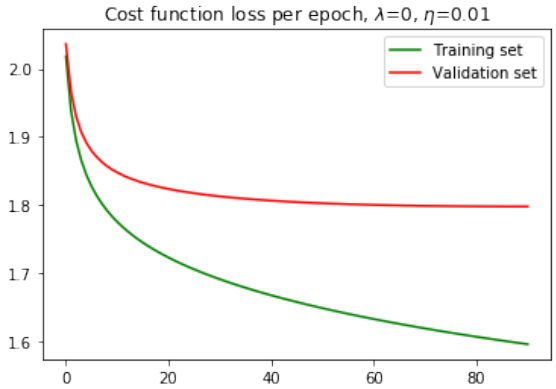


   Figure 1: Training and validation set cross entropy loss evolution when using extended training and early stopping

   The corresponding weights were learned from the usage of early stopping:

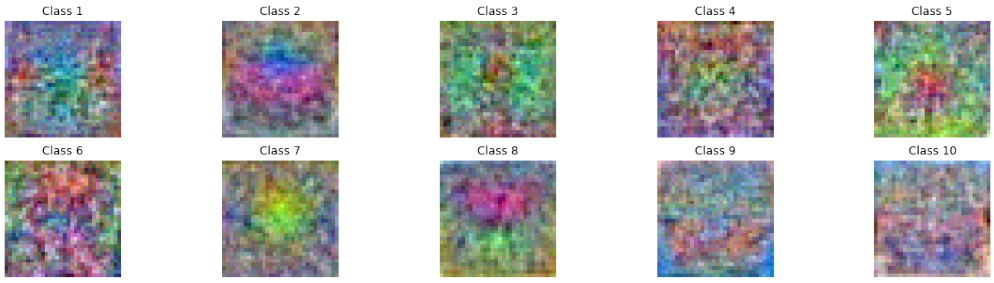


   Figure 2: Weights learned when using extended training and early stopping

   The accuracy derived when using early stopping is approximately 38.04%:

This time, we train under the same settings, but instead of applying early-stopping, we select the bias and weight matrices that led to highest validation-set accuracy performance during the training process:

The following plot shows the cost evolution over each epoch for the training and validation datasets:

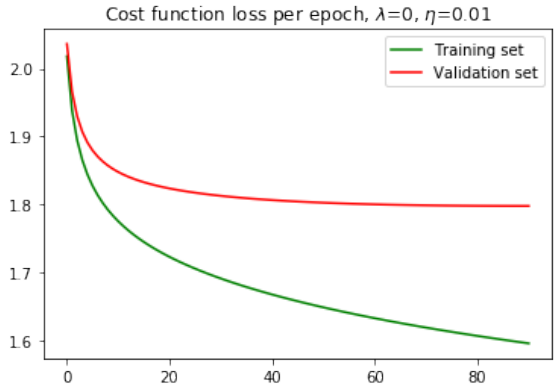


Figure 3: Training and validation set cross entropy loss evolution when using extended training and early stopping

The corresponding weights were learned from the usage of early stopping:

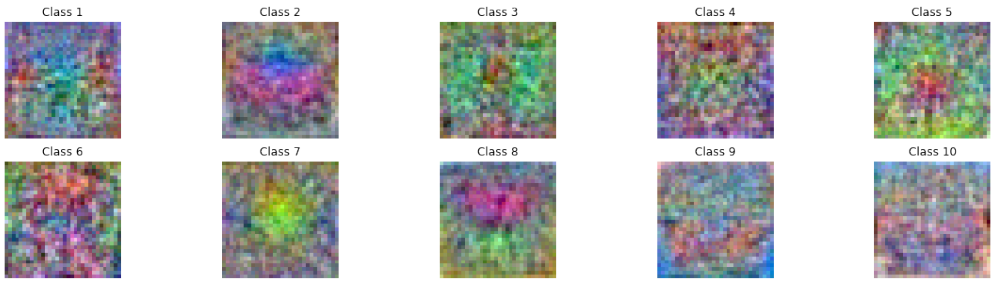


Figure 4: Weights learned when using extended training and early stopping

The accuracy derived when keeping track of the best model slightly increased to 38.11%:

A small trade-of can be reported between these 2 actions when training for a large number of epochs: On the first hand, the training algorithm is characterised by faster convergence, while on the latter case a small increase on the network's performance is reported, but more training time and resources have to be dedicated when training the network. This difference is a bit more obvious in the next setting that was applied.

2. **Using all the available datasets for training, and keeping only the last 1000 data from the second training batch (original training set) as the new validation set.**

   The evolution of cost when all the data except a small fraction (1000 in particular) is given by the following plot:

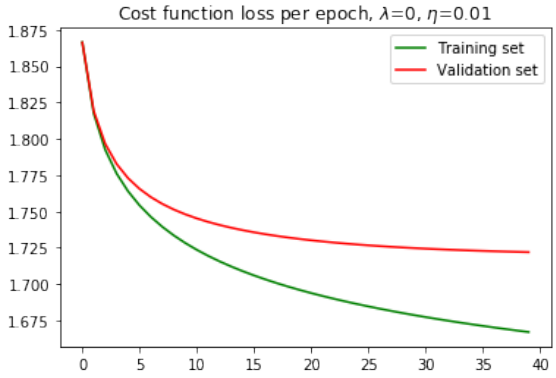


Figure 5: Training and validation set cross entropy loss evolution when using all of the data for training

The weights learned from using all the available data in the training set were:

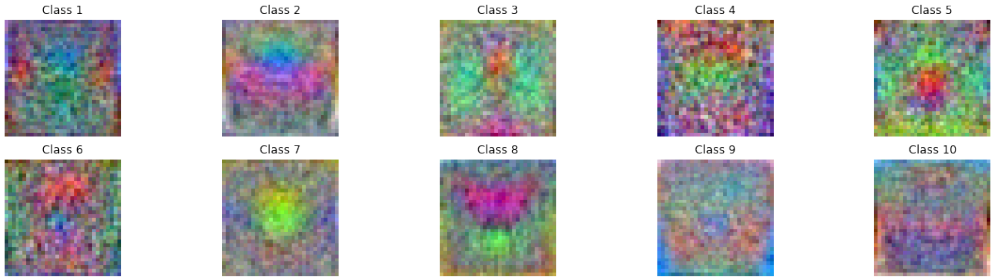


Figure 6: Weights learned when using all of the data for training

The accuracy when training on the full dataset is approximately 41%:

We then combine the extended dataset with both early stopping and keeping track of the best model, where we get our overall best results for the latter case.

The cross-entropy evolution on the training and validation set for these extended settings can be found placed next to each other in the following plot:

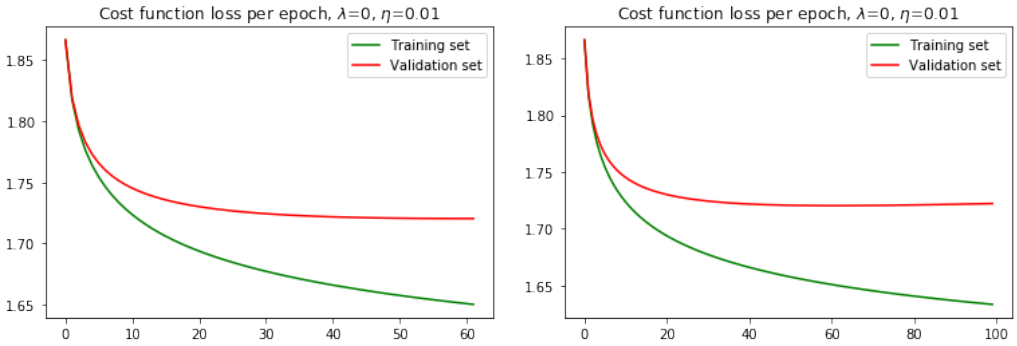


Figure 7: Cross-entropy loss evolution for the combination of the extended dataset with early stopping(**left**) and with keeping track of the best model (**right**)

The weights learned from these settings are the following:

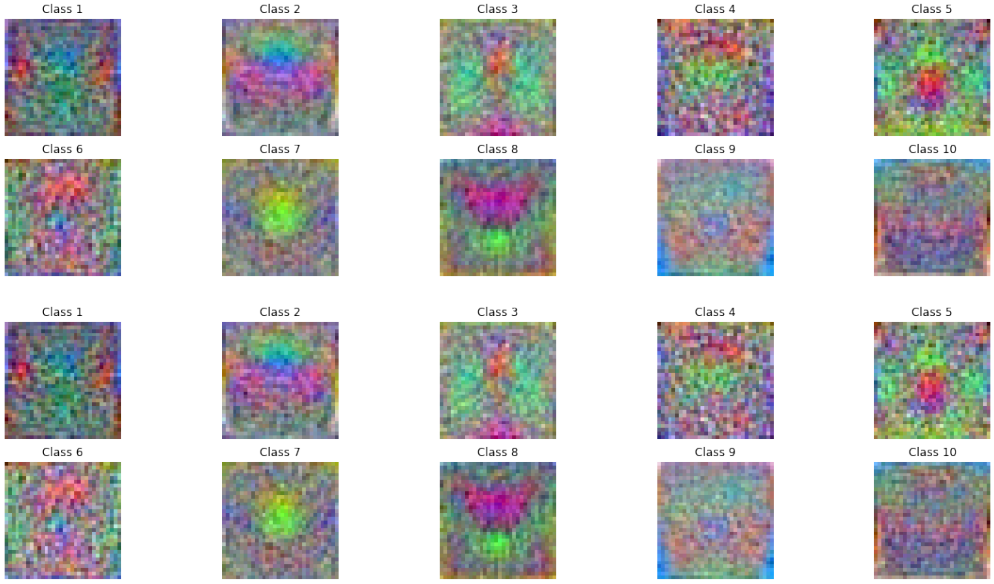


Figure 8: Weights learned for the combination of the extended dataset with early stopping(**up**) and with keeping track of the best model (**down**)

The test-set accuracy increased to approximately 40.1% for the first setting and reported its **overall-best performance** of 41.25% for the latter setting.

3. **Training 5 different classifiers (one from each dataset) and predicting the class of each data using the majority vote**. During the "voting" process, each classifier is rated according to its votes up to this point. When 2 or more classes share equal votes, the best classifier so far decides for the predicted class index.

   The accuracy of the ensembling method is 39.48%

4. **Fine tuning of parameters η and λ to define some "optimal" values that perform better than the best performance so far ($\eta = 0.01$ and $\lambda = 0$)**

```
Lambda: 0, eta: 0.001, accuracy:0.3597, BEST SO FAR
Lambda: 0.0001, eta: 0.001, accuracy:0.37139999999999995, BEST SO FAR
Lambda: 0.001, eta: 0.001, accuracy:0.37860000000000005, BEST SO FAR
Lambda: 0.01, eta: 0.001, accuracy:0.3819, BEST SO FAR
Lambda: 0.1, eta: 0.001, accuracy:0.3781
Lambda: 0, eta: 0.002, accuracy:0.38270000000000004, BEST SO FAR
Lambda: 0.0001, eta: 0.002, accuracy:0.3852, BEST SO FAR
Lambda: 0.001, eta: 0.002, accuracy:0.3879, BEST SO FAR
Lambda: 0.01, eta: 0.002, accuracy:0.389, BEST SO FAR
Lambda: 0.1, eta: 0.002, accuracy:0.37760000000000005
Lambda: 0, eta: 0.003, accuracy:0.38249999999999995
Lambda: 0.0001, eta: 0.003, accuracy:0.38780000000000003
Lambda: 0.001, eta: 0.003, accuracy:0.38980000000000004, BEST SO FAR
Lambda: 0.01, eta: 0.003, accuracy:0.3909, BEST SO FAR
Lambda: 0.1, eta: 0.003, accuracy:0.369
Lambda: 0, eta: 0.004, accuracy:0.38470000000000004
Lambda: 0.0001, eta: 0.004, accuracy:0.38859999999999995
Lambda: 0.001, eta: 0.004, accuracy:0.38949999999999996
Lambda: 0.01, eta: 0.004, accuracy:0.39170000000000005, BEST SO FAR
Lambda: 0.1, eta: 0.004, accuracy:0.36519999999999997
Lambda: 0, eta: 0.005, accuracy:0.387
Lambda: 0.0001, eta: 0.005, accuracy:0.3903
Lambda: 0.001, eta: 0.005, accuracy:0.38970000000000005
Lambda: 0.01, eta: 0.005, accuracy:0.39070000000000005
Lambda: 0.1, eta: 0.005, accuracy:0.3611
Lambda: 0, eta: 0.006, accuracy:0.38759999999999994
Lambda: 0.0001, eta: 0.006, accuracy:0.39049999999999996
Lambda: 0.001, eta: 0.006, accuracy:0.3899
Lambda: 0.01, eta: 0.006, accuracy:0.3901
Lambda: 0.1, eta: 0.006, accuracy:0.357
Lambda: 0, eta: 0.007, accuracy:0.3863
Lambda: 0.0001, eta: 0.007, accuracy:0.39059999999999995
Lambda: 0.001, eta: 0.007, accuracy:0.38980000000000004
Lambda: 0.01, eta: 0.007, accuracy:0.3882
Lambda: 0.1, eta: 0.007, accuracy:0.3547
Lambda: 0, eta: 0.008, accuracy:0.38480000000000003
Lambda: 0.0001, eta: 0.008, accuracy:0.39049999999999996
Lambda: 0.001, eta: 0.008, accuracy:0.39039999999999997
Lambda: 0.01, eta: 0.008, accuracy:0.3883
Lambda: 0.1, eta: 0.008, accuracy:0.3517
Lambda: 0, eta: 0.009000000000000001, accuracy:0.3842
```

```
Lambda: 0.0001, eta: 0.009000000000000001, accuracy:0.38859999999999995
Lambda: 0.001, eta: 0.009000000000000001, accuracy:0.38649999999999995
Lambda: 0.01, eta: 0.009000000000000001, accuracy:0.38580000000000003
Lambda: 0.1, eta: 0.009000000000000001, accuracy:0.3488
Lambda: 0, eta: 0.010000000000000002, accuracy:0.3852
Lambda: 0.0001, eta: 0.010000000000000002, accuracy:0.38770000000000004
Lambda: 0.001, eta: 0.010000000000000002, accuracy:0.38639999999999997
Lambda: 0.01, eta: 0.010000000000000002, accuracy:0.38470000000000004
Lambda: 0.1, eta: 0.010000000000000002, accuracy:0.3477
Lambda: 0, eta: 0.011, accuracy:0.3843
Lambda: 0.0001, eta: 0.011, accuracy:0.38749999999999996
Lambda: 0.001, eta: 0.011, accuracy:0.3853
Lambda: 0.01, eta: 0.011, accuracy:0.3831
Lambda: 0.1, eta: 0.011, accuracy:0.3456
Lambda: 0, eta: 0.012, accuracy:0.3832
Lambda: 0.0001, eta: 0.012, accuracy:0.3853
Lambda: 0.001, eta: 0.012, accuracy:0.3831
nd output; double click to hide output .012, accuracy:0.382
Lambda: 0.1, eta: 0.012, accuracy:0.3435
Lambda: 0, eta: 0.013000000000000001, accuracy:0.3829
Lambda: 0.0001, eta: 0.013000000000000001, accuracy:0.3833
Lambda: 0.001, eta: 0.013000000000000001, accuracy:0.38170000000000004
Lambda: 0.01, eta: 0.013000000000000001, accuracy:0.38029999999999997
Lambda: 0.1, eta: 0.013000000000000001, accuracy:0.3417
Lambda: 0, eta: 0.014000000000000002, accuracy:0.3811
Lambda: 0.0001, eta: 0.014000000000000002, accuracy:0.3822
Lambda: 0.001, eta: 0.014000000000000002, accuracy:0.38039999999999996
Lambda: 0.01, eta: 0.014000000000000002, accuracy:0.37839999999999996
Lambda: 0.1, eta: 0.014000000000000002, accuracy:0.3399
```

The best accuracy derived for η and λ was 39.17% for (η, λ)= (0.004, 0.01)

5. **Decay the learning rate by a factor of 0.9 after each epoch, staring value of η is 0.1.**

   The training and validation cost evolution when decaying η after each epoch are plotted in the following graph:

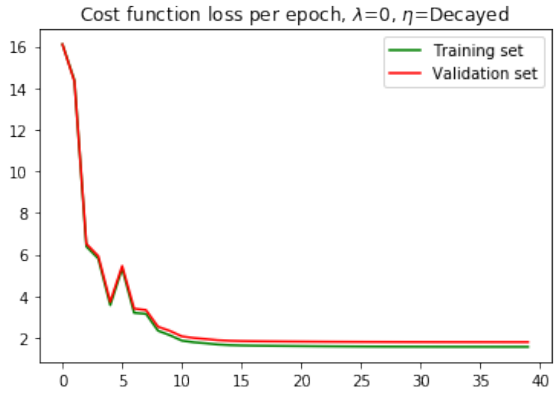


Figure 9: Training and validation set cross entropy loss evolution when using factor decaying

The weights learned from this procedure are:

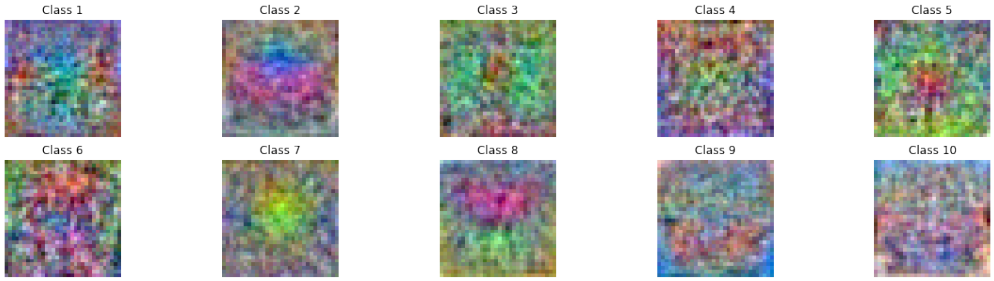


Figure 10: Weights learned when using factor decaying

The accuracy of this method is 38.57%

6. **Augment the training data, to create more data, and also represent the same class in various transformations**.

   During this setting, we create more data, and more diverse data, by either applying some transformations in the original images (including random rotation in a range of 90 degrees, horizontal flipping of the images, and/or a small shift on the height and width of the image) or by introducing some noise on our data.

   Some of the augmented images can be seen below:

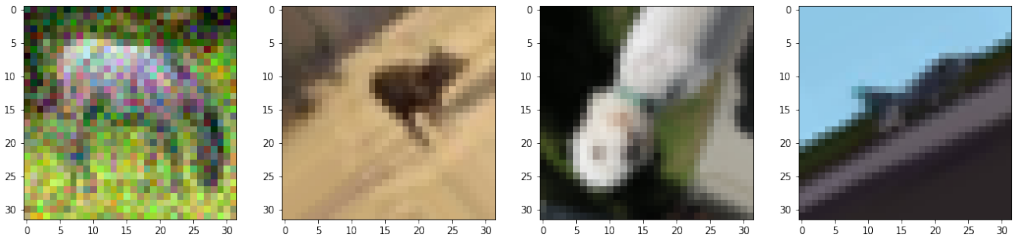


Figure 11: Samples of augmented images

For every single batch of dataset (consisting of 10000 images), we create another 20000 images. We train our neural network under 3 different settings:

- **Case 1**: Training by extending data_batch_1 with augmented images and no modification in the standards settings η, λ=0.01, 0.
- **Case 2**: Training by extending data_batch_1 with augmented images and for 100 epochs by keeping track of the best model in terms of validation set accuracy.
- **Case 3**: Training by extending the whole dataset (all but 1000 images used for validation) with augmented images and for 100 epochs by keeping track of the best model in terms of validation set accuracy.

The cross-entropy training and validation set loss evolution for each case is shown for each case respectively in the following, unified plot:

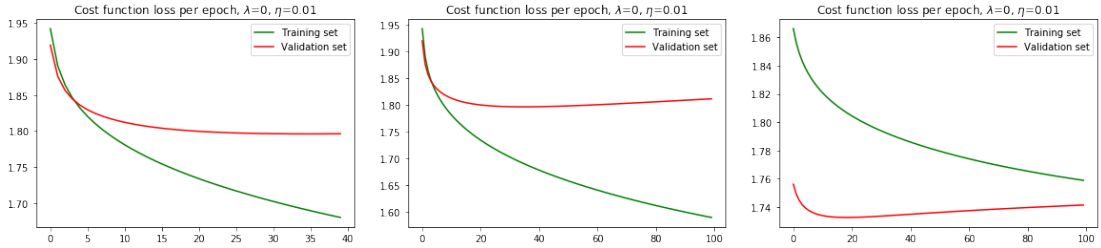


Figure 12: Training and validation set cross entropy loss evolution for case 1 (**left**), case 2 (**center**) and case 3 (**right**)

The weights learned from each training case are the following:

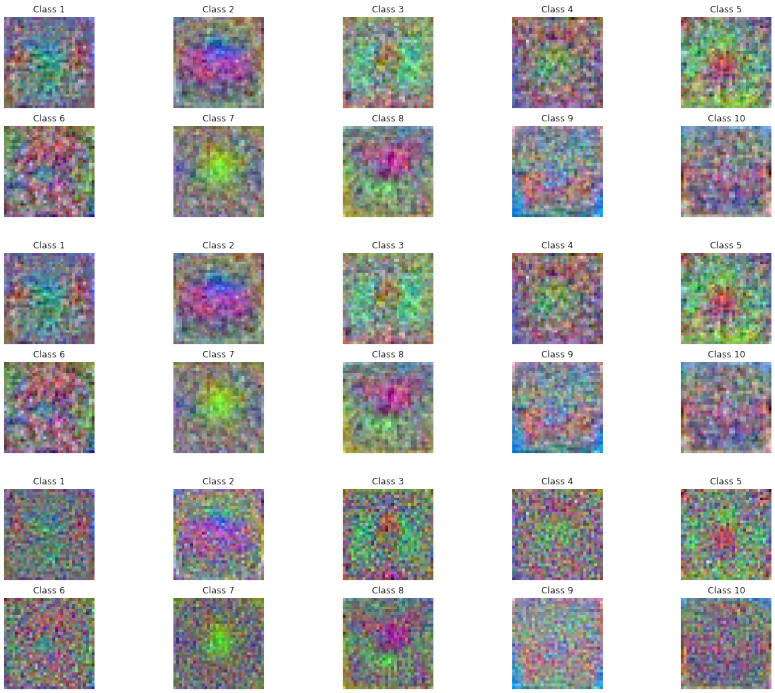


Figure 13: Weights learned from case 1 (**up**), case 2 (**center**) and case 3(**down**)

The test-set accuracy performance of these settings can be found in the following table:

Case no.#	**Test-set accuracy**
1	38.36%
2	38.4%
3	40.67%

This time, we observe a slightly different behavior of the cross-entropy loss evolution. Validation set loss is not always greater than the training set one, especially in the third case where it is always smaller. This different behavior can be credited to the alterations made in the training set; data are more diverse and thus the loss between the predictions and their true labels increases, while on validation set we have not tampered with data, thus it is easier to predict correct labels with high probabilities. However, this setting helps the network learn features well, scoring well in terms of test set performance. In fact, in the third case where the loss evolution seems strange, the network learns weight and bias matrices that lead to the overall second best performance of 40.67%.

2 Exercise 2: Train network by minimizing the SVM multi-class loss

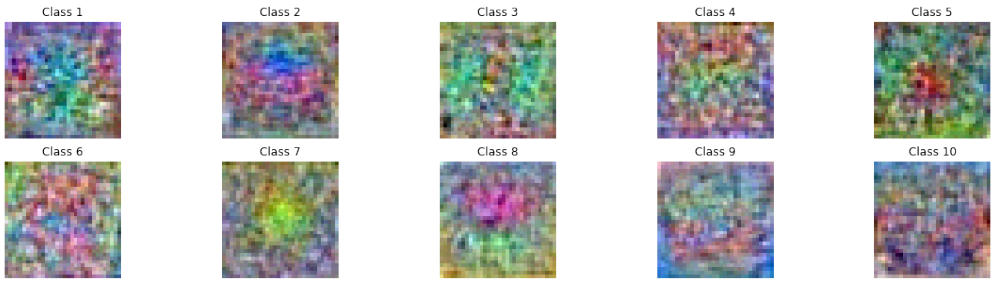
Replace the cross-entropy loss with the SVM loss, and report your results.

The reader is strongly advised to refer to `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf` for the formulas used to perform the computations of this part.

After using the SVM loss and gradient descent update, and with $\eta = 0.01$ and no regularization applied, the cost evolution on the training and validation set is as follows:

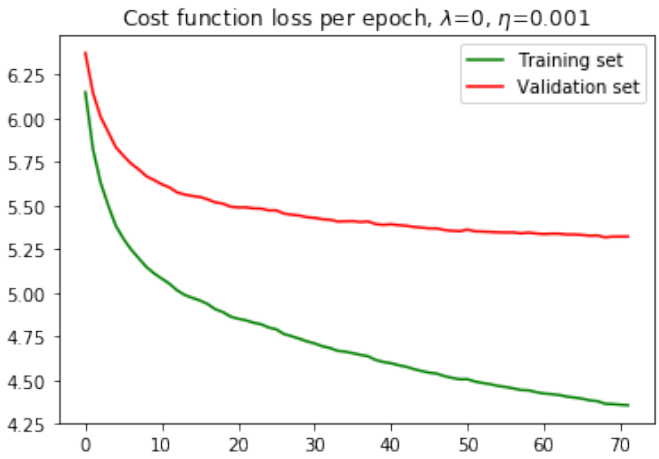
The performance of this setting is more unstable in comparison with using the cross-entropy loss and gradients update that use the same hyperparameters.

Nevertheless, the weights learned by this setting are the following:

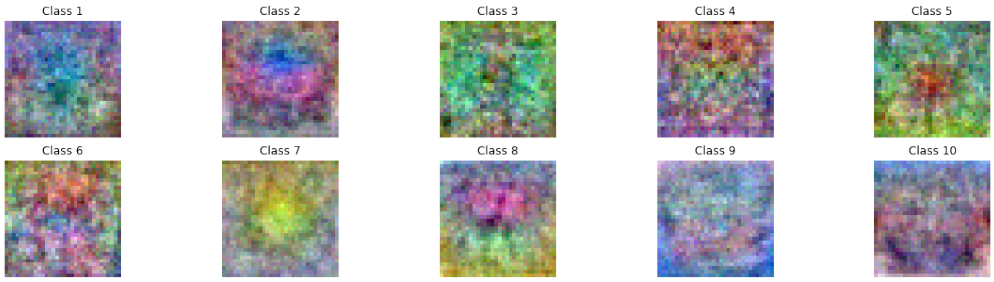


The accuracy performance on the test set was 31.19%

We also test the network based on the SVM loss, by applying a smaller value for the learning rate ($\eta = 0.001$) and a small amount of regularization ($\lambda = 0.01$). Then, the loss evolution on the training and validation set get smoother:



The weights learned by this setting are the following:



The accuracy performance on the test set was 34.34%

To recapitulate, and comparing SVM loss with the Croos-Entropy loss, the latter has more stable behavior and better performance in terms of test-set accuracy