Assignment 5: Supervised Learning

NEUR503: Computational Neuroscience

Solim Legris

9 février 2021

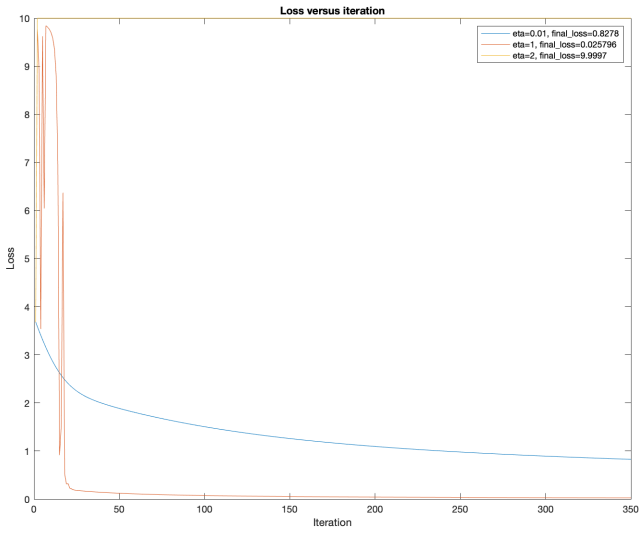
***Matlab Version**: 9.9*

***Collaboration**: Discussion of concepts and coding with Austin Cooper*

***Note**: The code to run tasks I, II and III is in the file A5. I have done it this way because it was much easier and cleaner to code that way. I only realized that you requested that every task be run in its own file when I got to task IV. Hopefully, this does not cause too much trouble. Please run A5 section by section.*

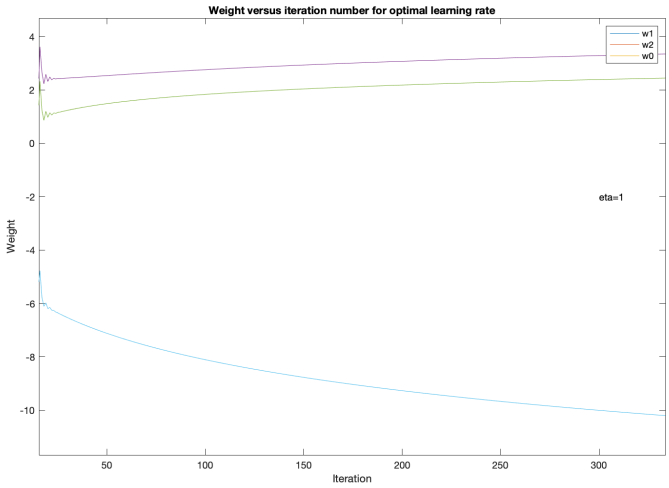
Task I: Single-layer Classifier

In the first section of this task, different values of the learning rate were systematically tried as can be seen in the file A5. When the learning rate was too small, the classifier tended not to converge on the best solution (i.e. the loss was larger than it could be). Conversely, when it was too large, the model did not learn at all and had a very large error. The largest value of learning rate that worked well for this classifier was eta = 1. This was determined by finding the minimal loss for all learning rates tested, loss = 0.02578.



This figure illustrates the loss as a function of the iteration number and the respective learning rates. The learning rate that was too small (eta = 0.001) did not converge to an optimal solution while the learning rate that was too large (eta = 2) did not learn at all since the error increases to ~10 and stays there for all other iterations.

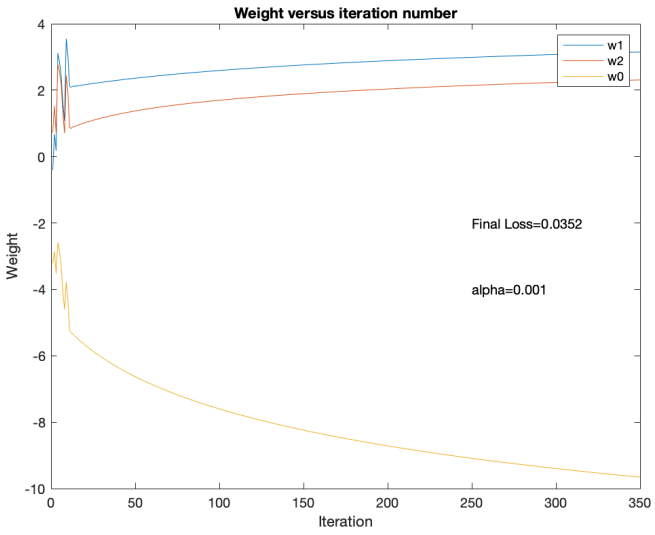
In the second section of this task, by observing the graph of the weights as a function of iteration number, we see that after ~25 iterations the weights diverge to increasingly high values (while the bias decreases a lot) which indicates that the classifier is potentially starting to overfit the data.



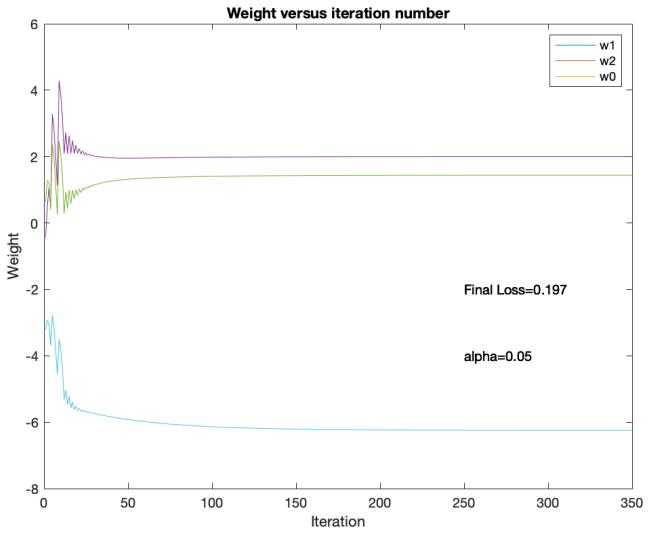
This figure illustrates the weights and bias plotted as a function of the iteration number with loss=0.02578

Task II: Single-layer classifier with regularization

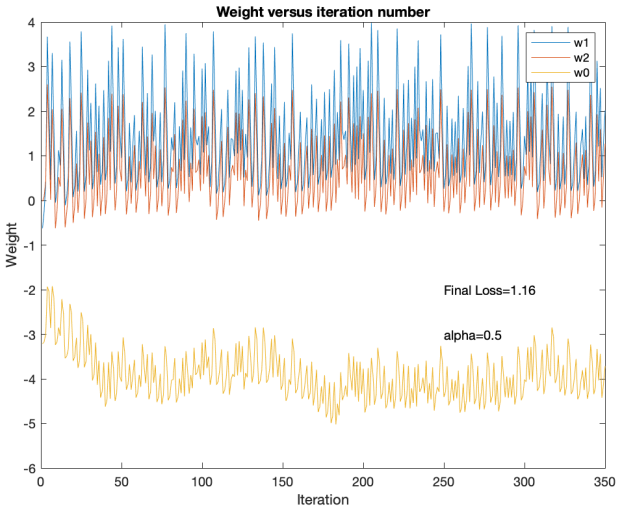
In the first section of this task (file A5), the code was modified to include regularization by updating weights with a penalty on the sum-squared weights. Next, I systematically tested different values of alpha to determine which one was best for regularization and lowered the learning rate to a value of 0.8 to account for wild behaviour of the classifier otherwise. As shown in the figures below, as the alpha value increases, the weights diverge less but the loss function also suffers a little. The optimal value of alpha was alpha = 0.05 with a loss of 0.197. When the alpha hyper-parameter becomes too large, the classifier does not learn since the penalty is too large as can be seen in the third figure below.



Weights plotted against the iteration number for a small value of alpha. There was no significant effect on the weights compared to the classifier without regularization. The penalty being too small, they still diverged and the classifier is still overfitting.



Weights plotted against the iteration number for optimal value of alpha. The weights settle after ~25 iterations, preventing overfitting.

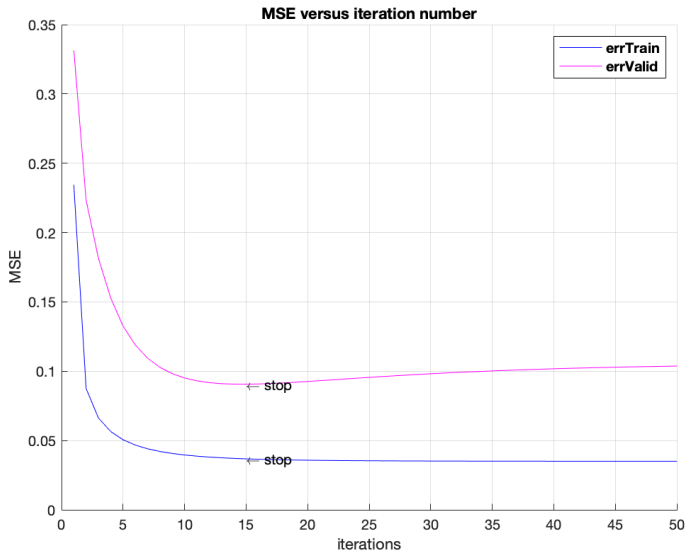


Weights plotted against the iteration number for an alpha that is too large. The loss is high and the weights never converge to some value indicating that the classifier is not learning (underfitting) as well as for lower values of alpha.

Task III: Receptive-field estimation using regression with early stopping

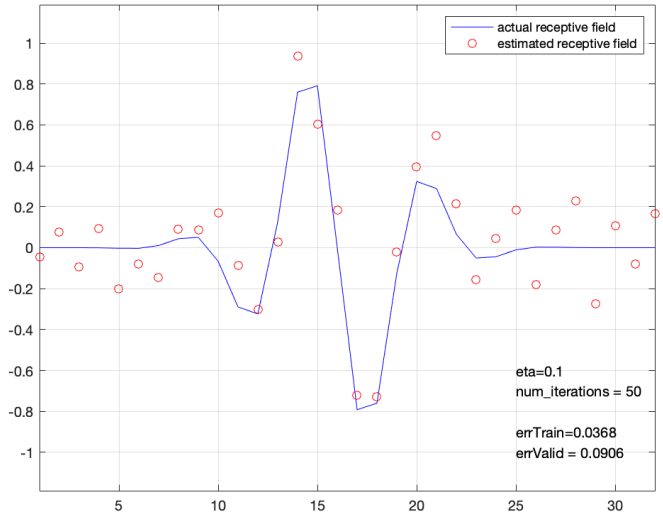
In the first section of this task (file A5), I modified the code in order to monitor the validation set loss. As soon as the loss starts increasing again (usually we would

define what is called a patience in order to account for slight variations), the algorithm stops since the model at this point is starting to overfit the training set data. The model gets better at predicting values from the training set but it loses its ability to generalize outside of it.



MSE plotted against iteration number for the training and validation sets. The optimal number of iterations is 15 where errTrain = 0.0351 and errValid=0.1037

In the second section of this task, early stopping is implemented and the model stops training once it reaches 15 iterations. The error for the training set is almost the same, errTrain= 0.0368, as for 50 iterations so the model learnt just as well the training set but it can also generalize better as indicated by the improved error on the validation set. The learning rate was eta=0.1 and the validation set error is errValid=0.0906.

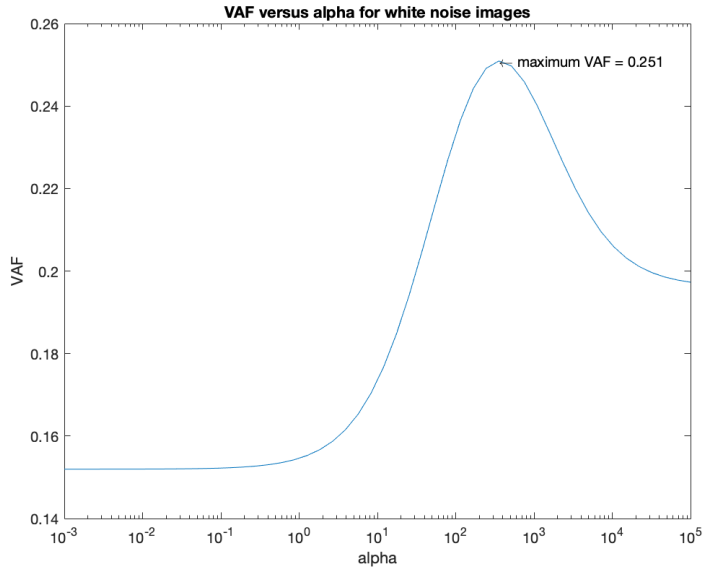


Here the actual receptive field is plotted in comparison to the learned receptive field.

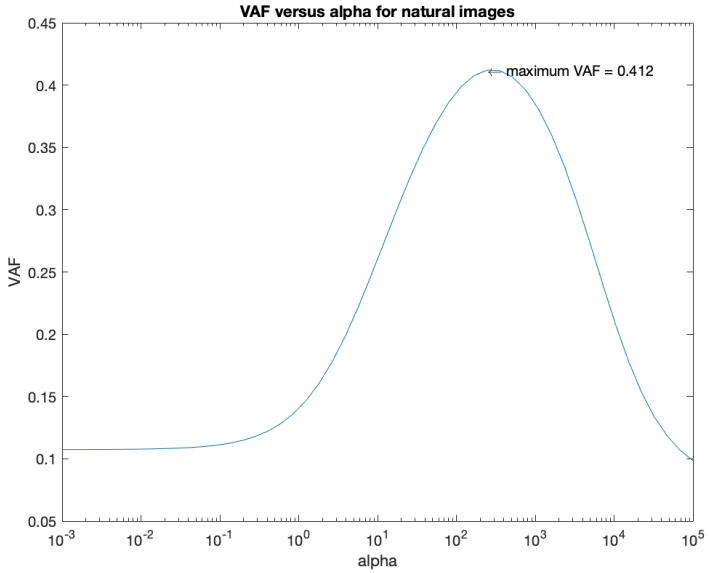
Task IV: RF estimation, comparing regression versus correlation for different stimuli

In the first section of this task (assignment file), the code was modified to create a third dataset (a test set) using the remainder of the data. Since the test set should not used at all for training or hyper-parameter tuning, performance on it only tested once training and tuning is done.

In the second section of this task, the code was modified so as to find the best alpha value to use for regularization (using a for loop) with scg-regression both for white noise images and natural images. This was done using logscale values of alpha between 10^{-3} and 10^{5}.



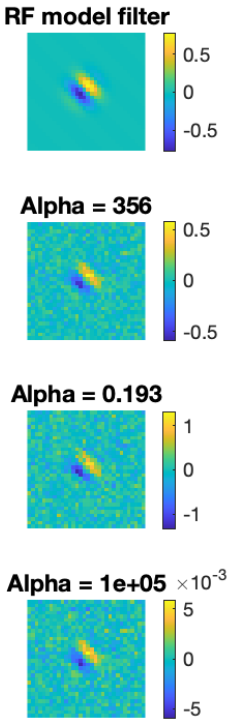
VAF plotted against logscale alpha values on semilogx plot. Maximum VAF is achieved at alpha=356.



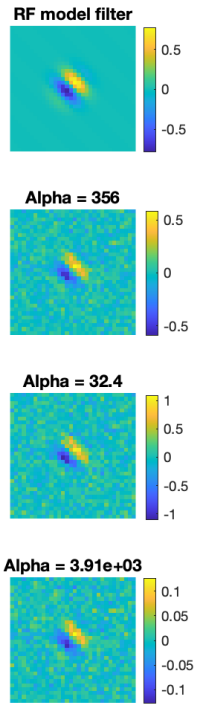
VAF plotted against logscale alpha values on semilogx plot. Maximum VAF is achieved at alpha=356.

As can be seen in these plots, the model learns better with natural images as the VAF is higher than for noisy images. Below are the learned RFs for various values of alpha. For lower values of alpha, the scale demonstrates that the model is

not learning the appropriate response values as they are too high and conversely for higher values of alpha the response values are much smaller than they should be when compared to the actual RF model filter. In both cases, the VAF is very low meaning the model does not generalize well.

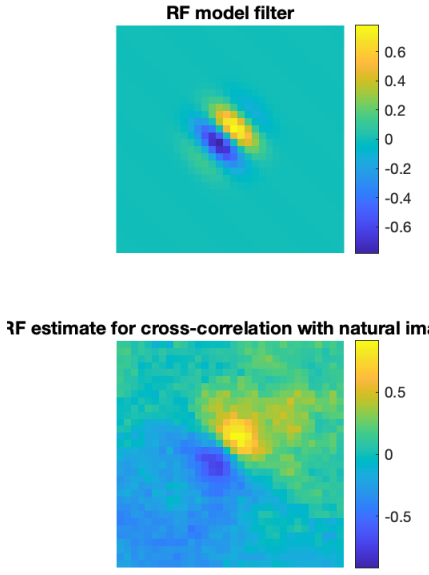


RF model filters of white noise images for actual RF, optimal alpha predicted RF, too small alpha predicted RF and too large alpha predicted RF respectively. Results obtained with scg-regression.

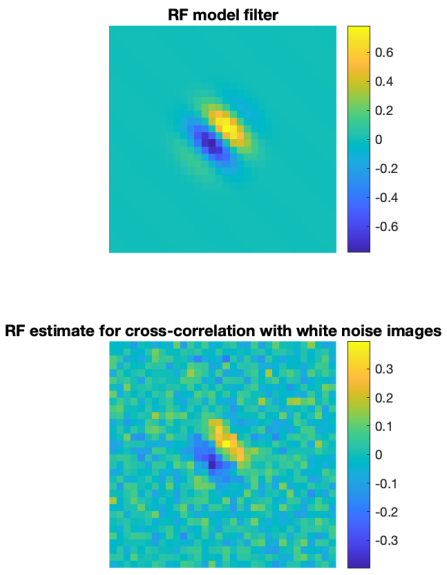


RF model filters of natural images for actual RF, optimal alpha predicted RF, too small alpha predicted RF and too large alpha predicted RF respectively. Results obtained with scg-regression.

Lastly, the results were evaluated for cross-correlation instead of ridge regression.



RF model filters of natural images for actual RF and predicted RF obtained through cross-correlation.



RF model filters of white noise images for actual RF and predicted RF obtained through cross-correlation.

As can be seen, the cross-correlation learns the RF relatively for both natural images and white noise images. In the case of natural images, the scale is relatively accurate but the receptive field does not have well-defined boundaries as obtained with scg-regression. Conversely, the boundaries of the RF were learnt relatively well for white noise images with cross-correlation but the scale is off. Whereas cross-correlation does not require grid-search of hyperparamaters and is obtained by modelling the dynamics of real cell's receptive fields mathematically, with the appropriate hyperparameters, scg-regression models learn the RF with better precision for both white noise and natural images.