Project-II by Group AUCKLAND

Marco Goretti **Pascal Lau** **Pierre Gabioud**

Abstract

This paper presents the results obtained for the second project of the Pattern Classification and Machine Learning course. This report contains two results. The first concerns a collaborative filtering system: We worked on the weak generalization, consisting on predicting listening counts for already present user and a strong generalization which predicts the listening counts of new users only having their friendship relationship with already existing users. On the second part of this report, we present a person detection system. We use and tune different models to detect whether a picture contains a person in it or not, such as Support Vector Machines, Neural Networks and Random Forest. We then compare this models and select one to detect people in an unlabeled test dataset.

1 Music Recommendation System

1.1 Introduction and Data Description

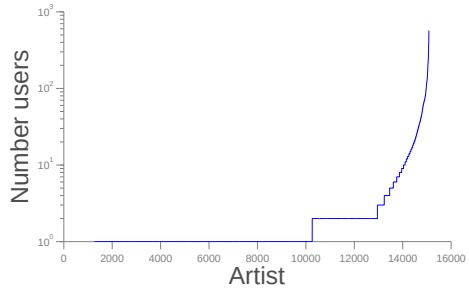
The first dataset consists of a friendship matrix, a listening count matrix linking users to artists. The friendship matrix $\mathbf{G_{train}} \in \mathbb{R}^{n_u \times n_u}$ is a binary valued matrix (n_u = 1774 is the number of users) where the entry (i, j) is 1 if users i and j are friend, 0 otherwise. The listening count graph $\mathbf{Y_{train}} \in \mathbb{R}^{n_u \times n_m}$ consists of a matrix where entry (i, j) represents the number of times user i has listened to artist j. The matrix $\mathbf{Y_{train}}$ is extremely sparse (has many zeros) so one of our task will be to estimate these zeroed values (weak generalization). Our second task is to, given a new user, predict his listening count for each artist (strong generalization). To achieve these two goals, we will apply several algorithms and compare their efficiency through cross validation.

1.2 Data Exploration

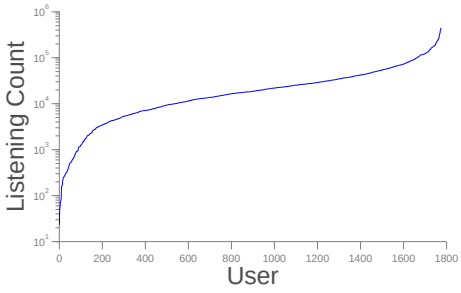
Our first task is to learn the data properly in order to devise an efficient learning algorithm. First, we found that there is a total of 57165963 hits. As mentionned above, $\mathbf{Y_{train}}$ is very sparse. That is, a great majority of the entries is zero: 99.74% of them. It means that we only have 0.26% of actual data. To better represent what it means, we tried to figure out the average listening count for each artist. Figure 1(a) illustrates it. We found that on average, each artist is listened by 4.6 users. We also observed that 10226 artists have been listened by 0 or 1 user while only 983 have been listened by more than 10. However, we observed that most of the users listened to more than 10 different artists, which allows us to conclude that the distribution of listening count is not uniform.

Moreover, we found some extreme values: some users have an unreasonable amount of listening count (451413 listenings !). However, these points follow nonetheless a logarithmic distribution and therefore are difficult to classify as outliers. we encountered some anomalies See figure 1(b). A user has on average 32224 listenings.

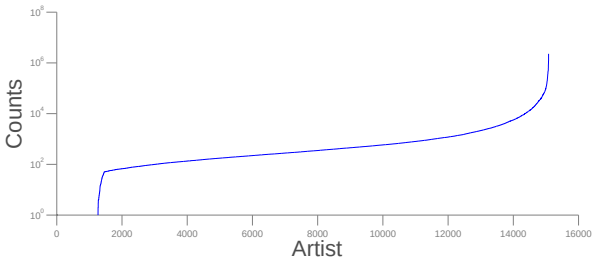
Figure 1(c) shows that some artists are quite popular. Indeed, a couple of them (723) have been listened more than 10000 times, and a few chosen ones amount to a million. They own 80.06% of the total listening count. This also shows that a majority of the artists have been listened less than 10000 times, and 8.37% of them (1262) have never been listened at all.



(a) Number of users that has listened to an artist (in log space)



(b) Listening count per user (in log space)



(c) Hits per artist.

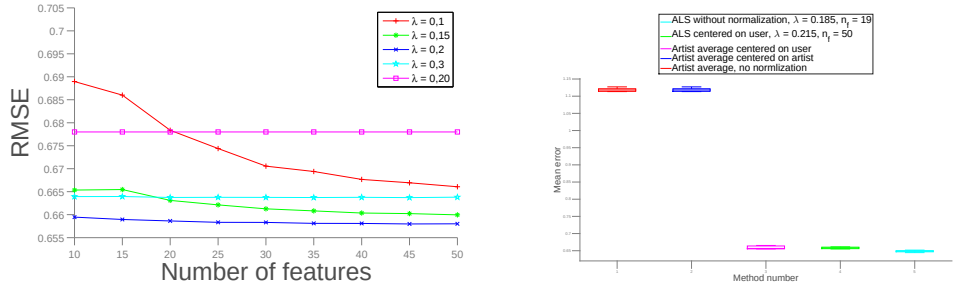
Figure 1: Data analysis

1.3 Cross Validation

Since $\mathbf{Y_{train}}$ is sparse, a problem regarding testing arises. Indeed, the lack of data makes it extensively difficult to assess how efficient our prediciton is. We had to come up with a way to cross validate a prediction with a K-fold. We start by computing a matrix $\mathbf{R} \in \mathbb{R}^{n_u \times n_u}$ where each entry $\mathbf{r_{(i,j)}}$ is a value $k \in 1...K$ where K is the number of fold for when a data is available in $\mathbf{Y_{train}}$ ($\neq 0$) and 0 otherwise. In our case, we set $K = 5$ splits. When an artist does not have at least K data available, we discard it. At the beginning of the K-fold, the entries in $\mathbf{R}$ are randomly but deterministically set (the seed is constant). At iteration k, the entries with value k form the test set $\mathbf{R_{Te}}$ while the entries with value $\neq$k (included entries with value zero) compose the training matrix $\mathbf{R_{Tr}}$.

1.4 Weak Generalization

For user i, i=1...n_u, we assign a feature vector $\theta^{\mathbf{i}} \in \mathbb{R}^{\mathbf{n_f}}$ and for each movie j, j=1...n_m, a feature vector $\mathbf{x^j} \in \mathbb{R}^{\mathbf{n_f}}$ where n_f is the number of features. A listening count prediction for user i on artist j is given by $\hat{y} = \theta^{\mathbf{T}}\mathbf{x}$. We call $\theta \in \mathbb{R}^{\mathbf{n_f} \times \mathbf{n_f}}$ and $\mathbf{X} \in \mathbb{R}^{\mathbf{n_{nf}} \times \mathbf{n_f}}$ the feature matrix for the users, respectively the artists. Thus, we have $n_f \times n_m + n_f \times n_u$ variables to predict. We do so by applying Alternative-Least-Squares with Weighted-λ-Regularization (ALS-WR) in order to predict both θ and $\mathbf{X}$ at the same time. We tried various values for our hyperparameters λ and n_f.



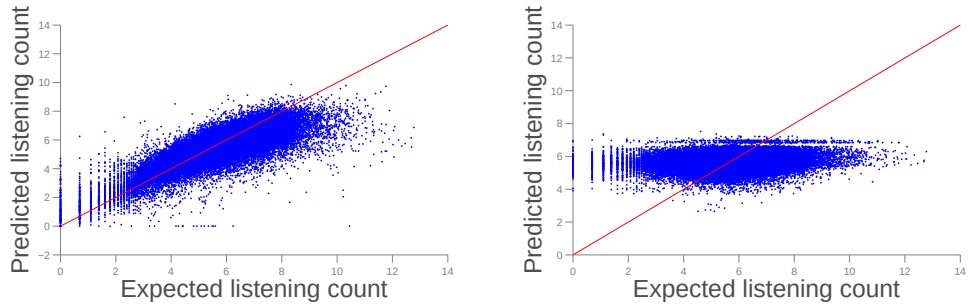
(a) Performance of ALS-WR with fixed λ and varying n_f

(b) Boxplot of various results with different methods and parameters

Figure 2: Weak generalization results

1.5 Weak Generalization Discussion

Figure 2(a) shows the evolution of the RMSE with respect to the number of features and λ (the two hyperparameters). λ is the regularization term. A big value will penalize large features thus pulling them towards zero and (hopefully) avoid overfit but a value that is too big will produce overfit by generating previsions very close to zero. Concerning n_f, a large amout of features would theoretically lead to overfitting. However, in the case of ALS-WR, given the userwise centering, we observe that an increasing value for n_f does not overfit but instead reduces the error by a decreasing amount. Thus, we can conclude that our regularization is performing well. By extensively increasing n_f (we tried with a value of 500), we observed a slightly better performance (a few bps). Figure 2(b) shows various results with data normalization with respect to dimension 1 or 2 (user, respectively artist) or no normalization at all. From this figure, we observe that we get a new optimal solution without normalizing, with $\lambda = 0.815$ and $n_f = 19$. This means that in this case the regularization from the paper (that takes into account the number of informations for each user/artist) does not prevent overfitting. Figure 3 shows graphically how our predictions are doing compared to the expectation. Figure 3(a) shows the result for our best prediction. When it comes to artists with bigger listening count, we denote a small bias that pulls the values toward the bottom thus our prediction underestimate the expected value.



(a) Performance of our best result: ALS without normalization (method 6 on figure 2(b))

(b) Performance of the method number 1 on figure 2(b))

Figure 3: Expected listening count vs predicted listening count, red line denotes the true mean.

1.6 Strong Generalization

The cross-validation was done by assigning all the known data for each user to the same category K. The objective is to generate previsions based only on the friendship relationship of the new tested user with the already existing users. The method studied here consists in generating an X and Θ features matrices (respectively for the users and artists) by training the ALS-WR method described

above on the known data (the original hits matrix) and then trying several transformations to infer the features of the new user based on the features of his friends.

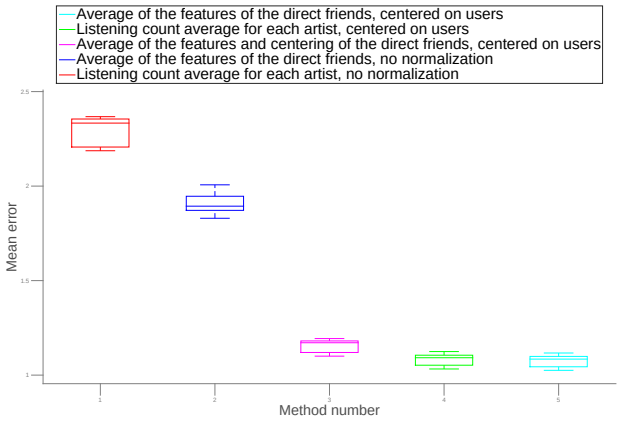


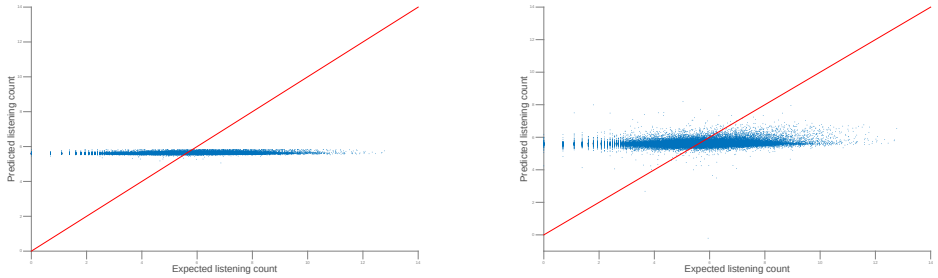
Figure 4: Strong generalization results

1.7 Strong Generalization Discussion

Figure 4 shows the cross-validated test error of a few implementations of our algorithm, compared to the naive average method which predicts the current average count for each artist. The main difference in performance comes from the way regularization is done: The best result was obtained by supposing that the new users were centered on the global average (1.08 test error) while a slightly worse result (1.2 test error) was observed when trying to infer the real behaviour of the user by averaging over his friends. Since a high user average means he listens more than the mean of the population while a low user average means that he has less listen counts than the rest of the population, we were expecting to find a correlation between the listening behaviour of friends but our results indicate otherwise since a worse result was found when correlating the mean of direct friends. In the weak problem the best result was obtained when there was no centering of the training data performed which made ALS take those differences into account. The result in this case is far from optimal and we suppose that it comes from the lack of prediction potential given by our processing of the friendship information. This processing still amounted to some information since we were able to reduce the mean test error by 1%.

The below figures show the behaviour of the predictions against the expected values. It is obvious that the predictions are highly biased toward the global mean while having a very low correlation with the truth.

An interesting result is that the friends help predict the features of the user (which kind of artists they will enjoy) but not their mean (how much they will globally listen to music compared to the general population).



(a) Plot of results from artist mean against expected (b) Plot of results from ALS against expected

Figure 5: Expected listening count vs predicted listening count, red line denotes the true mean.

2 Person Detection

2.1 Introduction and Data description

In this section, we use different methods to classify images into two classes: those with a person in it and those without. We use a few methods that give us different results: Neural Networks, SVM and Random Forest which will be described and compared in the next sections. Our data consists of 8545 images and 8545 labels describing whether the corresponding image contains a person or not: $+1$ if it does, -1 if it does not. In our dataset, 1237 of them have a person in it and 7308 do not. In addition to these, we have a set of features extracted from each image using *Histogram of oriented gradients* (HOG) descriptors. These features are particularly suited for human detection in pictures when they are in upright poses. Figure 6 shows an example of picture containing a person and its corresponding HOG features cells. We can already visually detect a shape that corresponds to the person in the picture. Each cell basically corresponds to the direction in which the gradient points the most intensely.

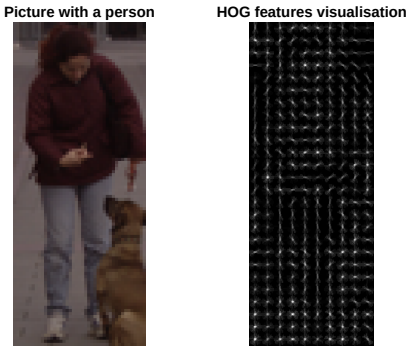


Figure 6: HOG Feature visualisation

2.2 Testing our results

To test our results, we compute a 10-fold cross validation. We plot a Receiver Operating Characteristic curve for each method and compare them in section 2.4.

2.3 Model Creation and Training

2.3.1 Neural Network

We explore here our experiments and results using a feed-forward Neural Network for classification. After many tests we estimate that a good number of sweeps through the network to avoid overfitting and excessive computation time is at about 15 with our settings. Figure 7 shows the evolution of the training error after n sweeps through the network. We see a convergence after about 15 steps. We then did different things to improve our results. First, we use two hidden layers with 200 hidden neurons for the first layer and 20 for the second, improving the accuracy or our training error and our final performance depicted in the ROC curve. Another considerable improvement was performed when using a sigmoid function as activation function of the neural network instead of a tan. After running a 10-fold we get an average True Positive Rate for a False Positive Rate between 10^{-3} and 10^{-2} of 0.845 with a standard deviation of 0.037. The ROC curve displayed in Figure 8(a) shows the average ROC curve of our 10-fold which will be used to compare this model with the others.

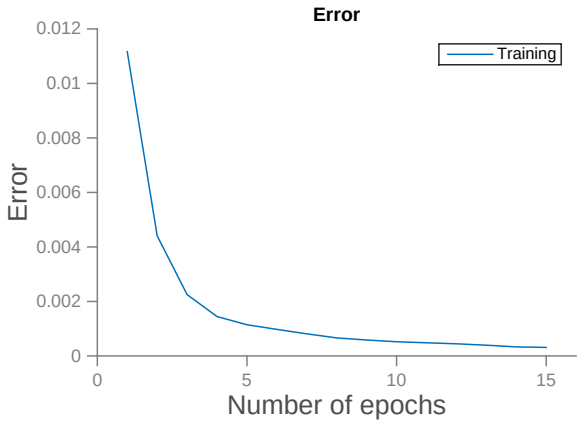


Figure 7: Training Neural Network

2.3.2 Support Vector Machine

We use SVM to perform our detection. SVM is also well suited for human detection using HOG features. We first tested a very basic linear kernel using SVM that performed fairly well with an average TPR of 0.796 for FPR varying between 10^{-3} to 10^{-2}. We then decided to use a radial basis function (RBF) kernel and found that this kernel was well suited to improve our original results, according to our cross validation and our readings on the subject. However, the sigma of the kernel has to be finely adjusted in order to have satisfactory results. An efficient way to do so is to run the Matlab train method with automatic kernelscale parameter and use different factors of 10 of this kernelscale for our sigma. We had an original kernelscale of about 0.08 and found an ideal sigma at 81 when running our tests with a 10-fold cross validation. Another improvement was made when changing the box constraint parameter. This parameter allows to adjust the penalty for observations violating the margin and reduce the number of Support Vectors. By testing different values from 10^{-5} to 10^{3} and taking the best (10), we not only improved our model precision but also reduced potential overfitting, allowing a better generalization for later predictions. Finally, after running a 10-fold we get an average True Positive Rate of 0.896 with a standard deviation of 0.045. The average ROC curve of our 10-fold is displayed in Figure 8(a) along with the ROC curves of our other models for comparison.

2.3.3 Random Forest

The last model we developed was using a Random Forest classifier. We tested different values for the number of decision trees used for classification, the minimum number of observations per tree leaf and the number of variable to sample for each decision. The results of this model were however never as good as those using Neural Networks and SVM and we decided to keep our computation

time to tune and cross-validate the parameters of our other satisfying models. We get our best results with 120 decision trees and by keeping the number of variables to sample for each decision to about the square root of the number of variables in X, here 96. We show the average ROC curve for the 10-fold of our random forest model along the ROC curves of the other models in Figure 8(a). The average TPR for a FPR between 10^{-3} and 10^{-2} is 0.698 with a standard deviation of 0.06, which is fairly high compared to our other models. However, at a FPR of 10^{-2} we get a satisfying result of about 0.86.

2.4 Classifier Comparison and Model selection

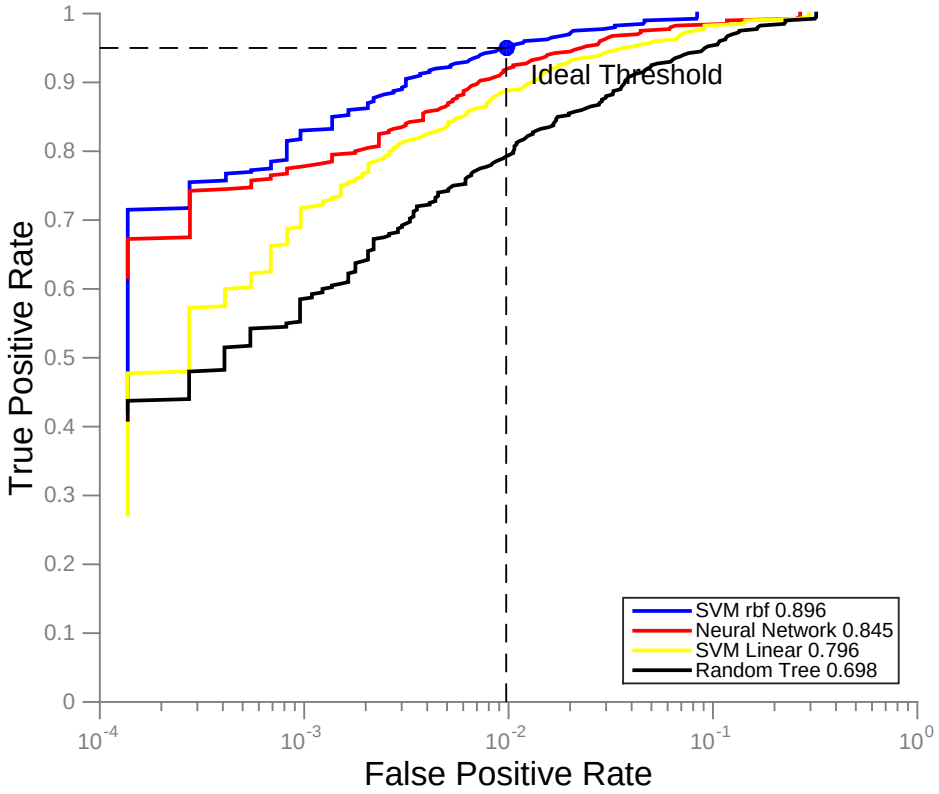
We select here a model to classify and give scores for the given test data of 8743 feature sets. Figure 8(a) displays the ROC curves of all our models. It gives us useful information to select our best model. Using different threshold, we compute the *True Positive Rate* (TPR) and the *False Positive Rate* (FPR) for all our classifiers. We also compute the average TPR for a FPR between 10^{-3} and 10^{-2}, displayed in the legend. This result gives a first insight of the performance of our predictor. However, we need to compare the ROC curves for different False Positive Rates to decide which model is best suited for our classification task. In our dataset, we have about 10% of the pictures with a person in it and the rest without. Our purpose is to get a TPR as high as possible with a reasonably low FPR. The ROC curves show that SVM gives better results than the others for every FPR so it is the model we use it to predict our test data and compute the scores. As we can see, the ROC curve of our SVM model is satisfactory and at a FPR of about 10^{-2} we get an excellent True Positive Rate of 0.95. We estimate that having a high TPR of 0.95 for a relatively low FPR of 1 out of 100 gives an ideal classifier for our dataset: using this threshold, we would detect about 95% of the people in pictures containing someone and falsly detect people in 1 out of 100 picture without a person in it. Figure 8(b) shows the boxplot of our the average values of each 10-fold in our computation. We see that SVM with a gaussian kernel has the best results with the lowest variance among all our models, thus giving us more confidence in selecting this model for our predictions.

2.5 Principal Component Analysis (PCA)

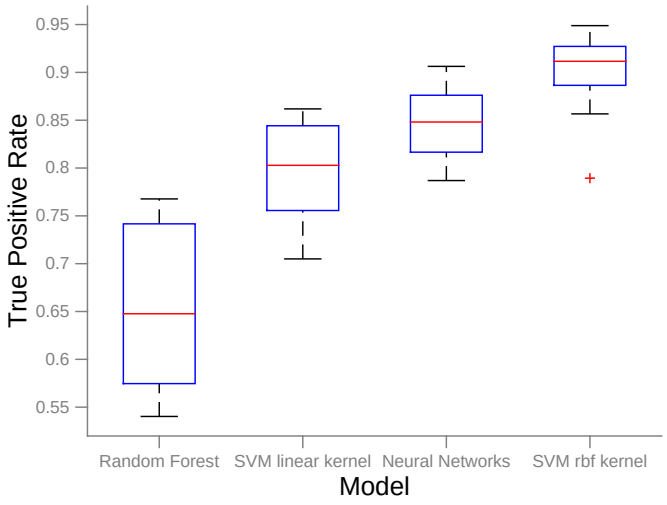
In order to reduce computation time, we also tried to perform dimensionality reduction with PCA. We performed a singular value decomposition on the feature matrix $\mathbf{X}$ in order to get the matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$. By inspecting $\mathbf{S}$, we realized that a great majority of the variance of the features are held by the first 1000 components of $\mathbf{U}$. We therefore tried to reduce $\mathbf{X} \in \mathbb{R}^{8545 \times 9360}$ into a matrix $\mathbf{Z} \in \mathbb{R}^{8545 \times 1000}$. Using this matrix $\mathbf{Z}$, we indeed improved the computation time but the results have been worsened from a significant amount. By tried to reduce the dimentionsality of $\mathbf{X}$ by even a bigger amount (we kept 100 features) and suprisingly obtained better results (still below the ones gotten without performing PCA). This can be explained by the fact that we actually avoid overfitting by throwing away features. Since our original results were satisfying enough, we did not dig deeper into dimensionality reduction, even know some greater results could have been found because it would allow much faster features tuning for our methods.

2.6 Conclusion

Detecting humans in pictures is not an easy task. However, with advanced classification techniques taught in class and Histogram of Oriented Gradients descriptors, the tasks becomes feasible. After testing and cross-validating different models and many parameters they contain, we were able to find an efficient classifier for our task using Support Vector Machines with a gaussian kernel. Using this techniques, we were able to get satisfying results, with a ROC curve proving it. Using this technique, we could detect an average of 95% of the people in pictures containing someone while having only 1 out of a 100 which were falsly detected in our training data. When taking the average of the *True Positive Rate* for a *False Positive Rate* between 10^{-3} and 10^{-2}, we found a result of 0.89. Other models such as Neural Networks and linear-kernel SVM gave us satisfying results but we were unfortunately not able to tune the parameters to make the random forest model satisfying.



(a) ROC curve for Neural Networks, SVM and Random Trees



(b) Boxplot for our different models

Figure 8: Model comparison: ROC curves and Boxplots

Implemented methods

The main methods we implemented can be found in the following files: *CVweak.m*, *CVstrong.m*, *ALSWR.m* and *update.m*.