Problem 1

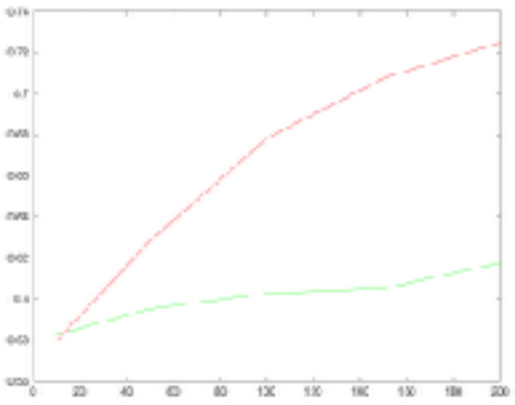
1.1
errors:

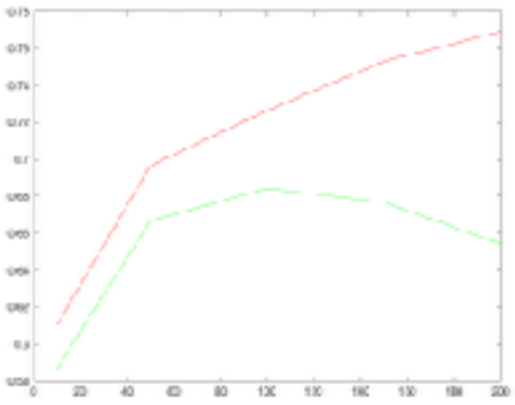
mean error of 2500 images(unit: 10^6)	**Train data set of women**	**Train data set of men**
Eigen face of women	2.7396	2.8044
Eigen face of men	2.7616	2.7839

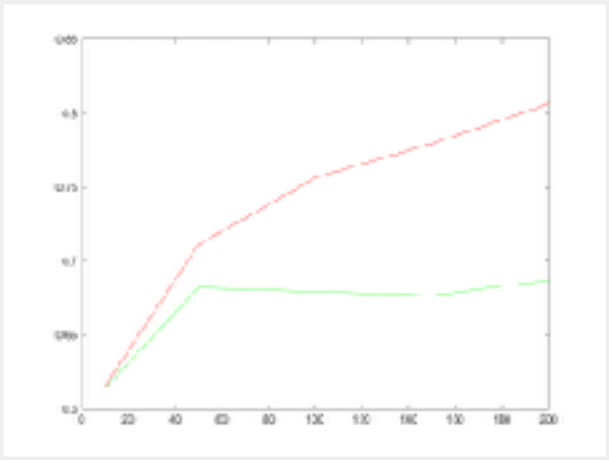
Two Eigen faces are saved in the ‘results’ folder.

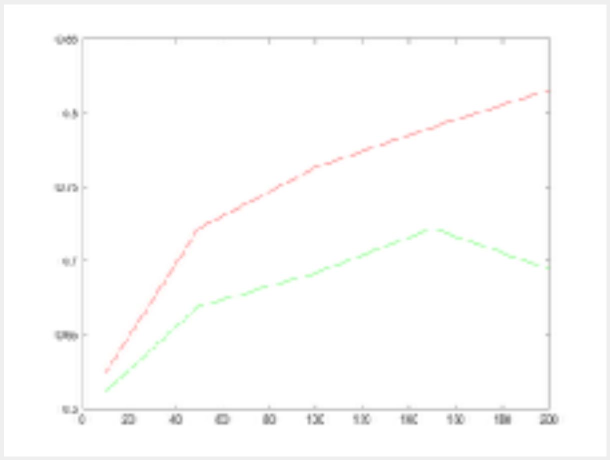
1.2 AdaBoost:
This implement of adaboost algorithm combines T decision stumps (1-level decision tree).
I have two helper functions: train_decision_stump and predict.
Each decision stump is trained by the training dataset X_tr, corresponding training labels Y_tr and data weights dweights. Among all dimensions of an image, only one dimension with the smallest error is chosen for deciding the decision criteria (the criteria includes a threshold value as long as a mark that indicates whether greater than the threshold means label +1 or -1). The decision stump is a pretty weak classifier and that is where I start. First, for one dimension, get all 5000 values of one dimension and sort them, then go through all 5000 possible threshold values (consider each threshold is inserted to the left of each value) and each of them comes with 2 directions (greater than threshold means +1 or -1). To speed up the training process, I initialize an error err_g, this error is calculated by considering all labels +1(that means set threshold smaller than the smallest value of all values and greater than threshold means +1), then error of the same threshold but of the other direction(less than threshold means +1) err_l = 1-err_g. As I gradually shift the threshold to right, err_g(i) = err_g(i-1) + dweights(i) * label(i). By doing addition instead of multiplication, the processed is speeded up. Find the minimum error among 5000 sets of threshold and direction, that is the error of this dimension, and then find the minimum error among all dimensions, and the final parameters of this decision stump are picked. Also new data weights for next weak classifier and this classifier’s weight α can be calculated. After training all T stumps, prediction can be made by combining them.

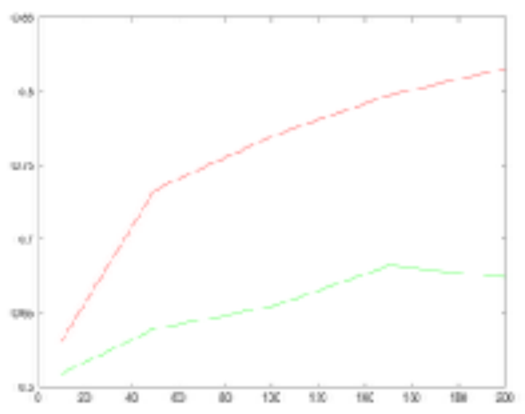
1.3
Following are the error plot for different Ks, the red line is train accuracy, and green one is test accuracy.


K = 10


K = 50


K = 100


K = 150


K = 200

The best accuracy of my AdaBoost algorithm on test dataset is 72%, when T = 150, K = 150. I think the main reason is that a decision stump classifier is too weak, as can be seen from figures, 10 decision stump classifiers together have an accuracy of about 60%. Leave out the poor accuracy, the relationship between error with T and K is worth analyzing.

For a certain K value, train error decreases monotonously as T increases; test error decreases as T increases but with fluctuation. T is the number of weak classifiers. AdaBoost is said to be more robust to overfit but more easily affected by outliers than other algorithms. There exists some extent of overfitting as I see, maybe caused by strange-looking faces in training dataset.

For a certain T, error decreases obviously when K goes from 10 to 50 to 100, but when K goes to 150 and 200, the improvement of accuracy is not so obvious (sometimes accuracy is worse when K gets larger). K is the number of features. That means too many features is not good for training and predicting, since not that much information is needed for a binary classification, and those redundant features may not be helpful or even violate training and predicting process and harm the performance of the AdaBoost classifier.

Problem 2

2.1

doNMF: update B and W by:

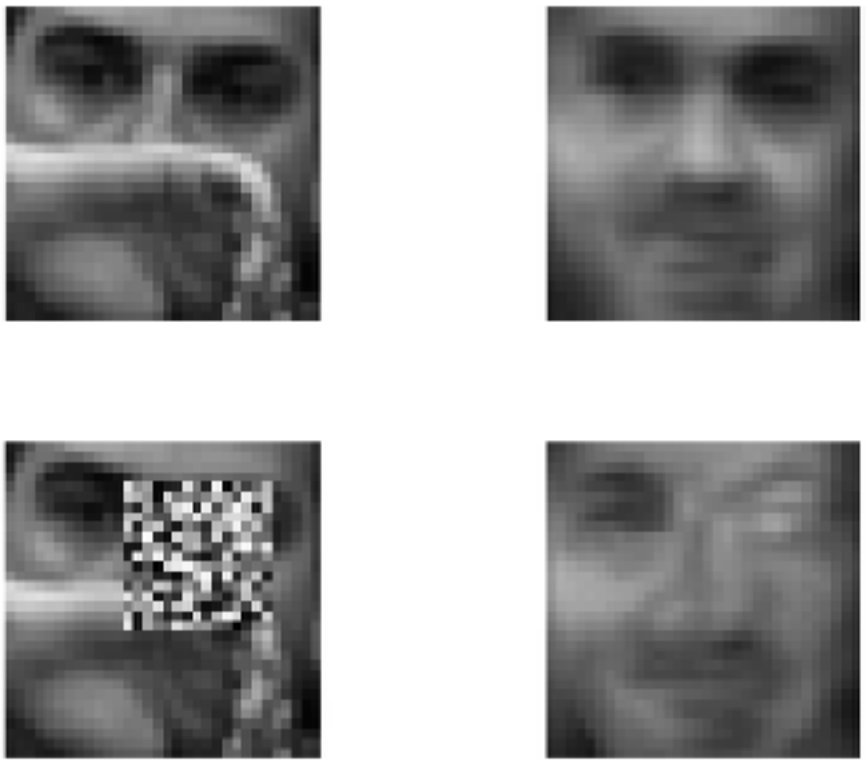
$$B = B\frac{(\frac{V}{BW})W^T}{1W^T}, W = W\frac{B^T(\frac{V}{BW})}{B^T1}$$

NMFreco: update W by:

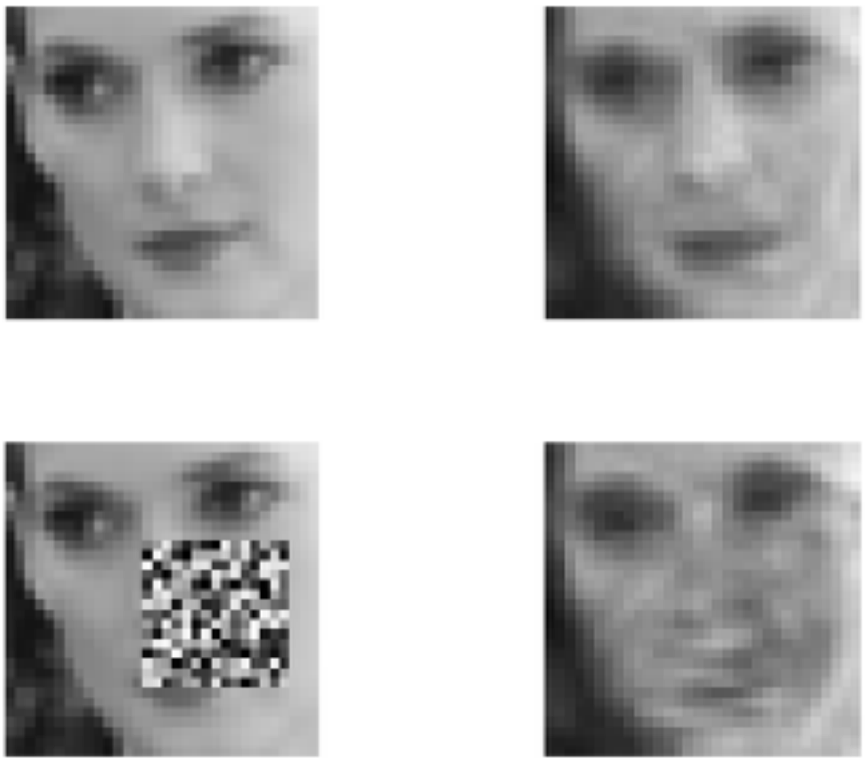
$$W = W\frac{B^T(\frac{V}{BW})}{B^T1}$$

error	100	200	300	400	500
org	2.813088714400324	1.851581364699330	1.422774386374684	1.202612383006221	1.070079077608456
cor	8.914224153307618	8.932655772480334	8.939432831515598	9.021745009370099	9.064722512842332

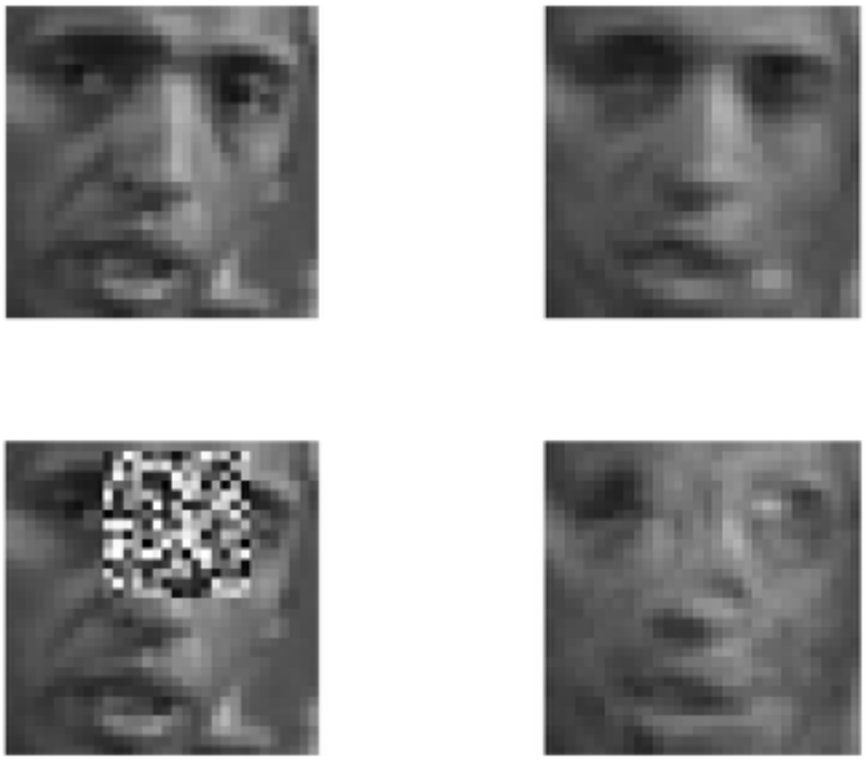
As K increases, reconstruction error of original face decreases, and the error of corrupted face increases, because the more iterations we do, the better the face is reconstructed toward the target we give, so the reconstructed original face is more close to org face, but the reconstructed corrupted face is more close to the corrupted face and less close to original face.



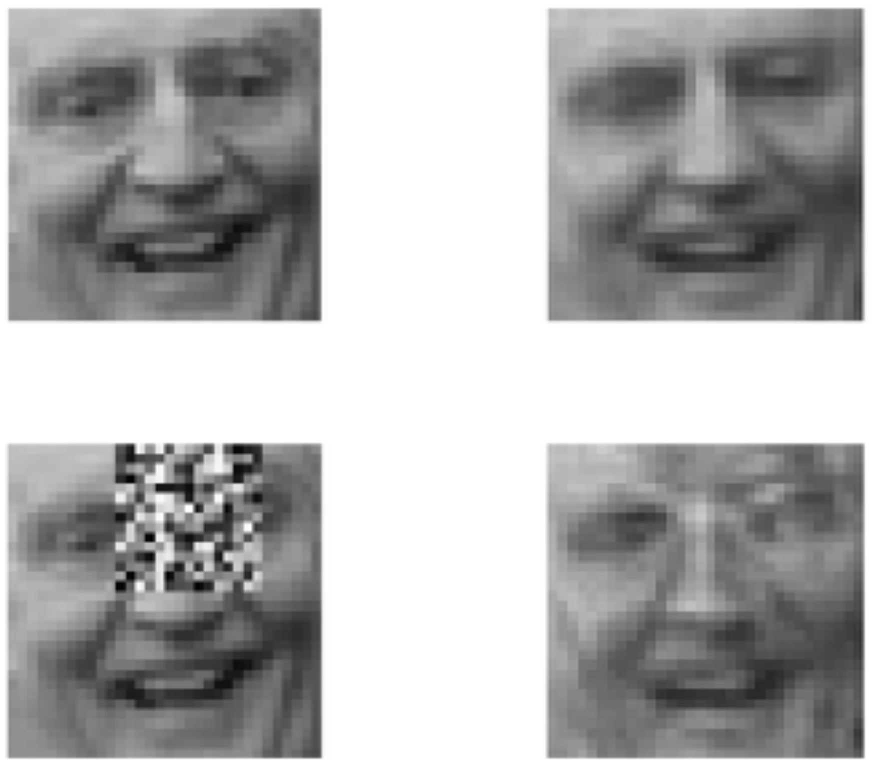
K = 100

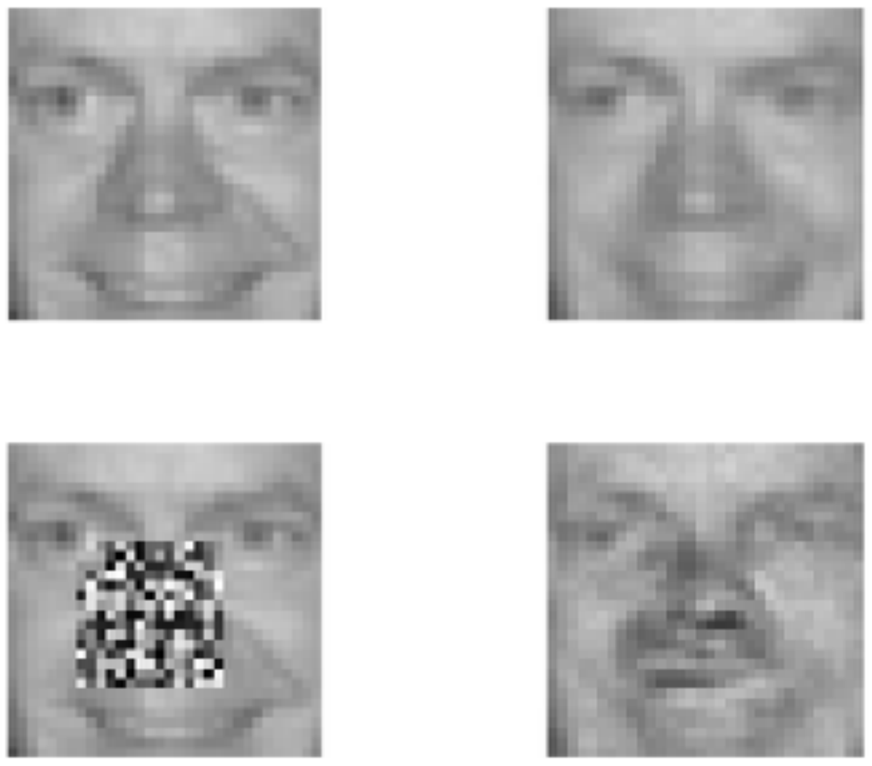


K = 200



K = 300


K = 400


K = 500

2.2
To separate speech and music, we need to express the mixed signal by two sets of bases, Bm and Bs. In function separate_signal, learn W by:
$W = W \frac{B^T(\frac{V}{BW})}{B^T 1}$, where B is [Bm,Bs], so we get W = $[\begin{matrix} W_m \\ W_s \end{matrix}]$, then separately calculate
$V_m = B_m W_m, V_s = B_s W_s$

Problem 3

3.1 loading data and centralize it.

3.2 function ICA is based on following equations:
H = AM, A = BC,
First do Eigen decomposition of MM^T,
$MM^T = ESE^T$, and $C = S^{-\frac{1}{2}} E^T$;
then, calculate D_X before B,
$D_X = \sum^k ||x_k||^2 x_k x_k^T$, where x_k is the kth column of CM,
do Eigen decomposition and get B, $D_X = B^T \Lambda B$;
finally, H = BCM, the two rows of H are separated signals we want.

I think ICA does a decent job in separating signals. Unlike NMF in problem 2.2, in this separation result, I can’t hear noises in speech nor human voice in noises.

3.3
To adjust results’ volume, I applied normalization on the two result audios by dividing the row of H by the max absolute value of this row. The normalized audios have higher volume than the original result.

3.4 errors

Based on my observation, error 1 is calculated by comparing S1 and R1, that is the error of voice; error 2 is the error of environment noise.

error1 = 571.3287207019507;
error2 = 2432.446711910501;