Intro To ML – HW4

students:

Dor Bank - 301380416 - [dorbank@gmail.com](mailto:dorbank@gmail.com),

Avi Caciularu - 203056585 - <avi.c33@gmail.com>

**Theory Questions**

**Question 1:**

1. Let be hypotheses and let be their majority:

.

Assume that for every feature , there are exactly hypotheses which return the correct label of , meaning: .

Then for every , the majority function will return the incorrect label, meaning .

Let be the hypotheses: . Explicitly:

Let we have 3 samples:

. Thus thus . We get that but (where ).

1. Assume now that . Then . In total, we get that .

**Question 2:**

1. Not submitted.
2. The error of the current hypothesis relative to the new hypothesis can be expressed by:

where .

Thus, we will now show that :

1. Proof by contradiction that . Then:

.

But where is a class of weak learners, by the definition of AdaBoost.

In addition, in one AdaBoost iteration, we choose the best (which gives the minimal error probability), meaning that for every weak learner chosen by the AdaBoost algorithm the parameter is effectively zero. Hence should fulfill- . Thus,

**Question 3:**

1. Denote and *.*

Hence: *.*

Denote the matrix which all its elements are equal to as . Thus, generalizing the discussion to matrices, we get: and since and are symmetrical (by definition of ), then and we get: **.** Assuming we are given and that matrix multiplication takes for matrices dimension’s , we conclude that the time complexity to compute in total is .

1. We will show that any where can be expressed by a linear combination of :

Denote as the covariance matrix formed of , meaning: . We note that every is an eigen-vector of , with eigenvalue (by the definition of principal component). Thus:

.

Hence:

(note that we didn’t prove that , which is easy to show). Notice that is just a scalar! Then we can write - and finding all eigenvectors is equivalent to finding the coefficients . The calculation of :

First, we will substitute our result of to get

Which can be rewritten as:

Multiply both members by from the left:

Which can be rewritten as:

Which equals to:

We can obviously remove a factor of from both sides. This would only affect the eigenvectors with eigenvalues of , but they won't be chosen as principle components anyway. So, we remain with:

Recall our orthonormality condition: . Thus, we get:

By multiplying the equation above by and using our orthonormality condition, we get:

Once we find the vectors’ eigenvalue, deriving the coefficients is trivial.

1. . Calculating and depends on the number of features, meaning it takes . Repeating this times (as the number of samples), and in total, this phase’s time complexity is .

**Question 4:**

Consider the following optimization problem:

Where is a matrix of the variables, is a column vector of size , and is a column vector of size of the coefficients.

Consider the SVD of , we get the constraint:

We want now to recover . From the recitation, denote the pseudo-inverse matrix of . Then we get the solution: . Denote this solution by .

Assume is singular. Denote by a vector in the null space of . Namely a vector such that . Suppose other solution to the optimization problem: , so fulfills . Thus, we get: .

Hence any solution of the form contains all possible solutions to .

From the recitation, we know that the projection matrix, can be calculated by . We would like to project to the null space of so we can get the minimum distance vector from . Then the null space projection would be: , and for we get: .

Then, from the definition of projection, the closest vector to in this null space is **.**

We know that the null space is linear, hence as well.

Now we can solve the optimization problem:

.

Note: can be of the form where is non-zeroed diagonal matrix. In this case, we define for the above calculations.

**Programming Questions**

**The code source files stay under /specific/a/home/cc/students/cs/avicaciularu/ML/HW4/:**

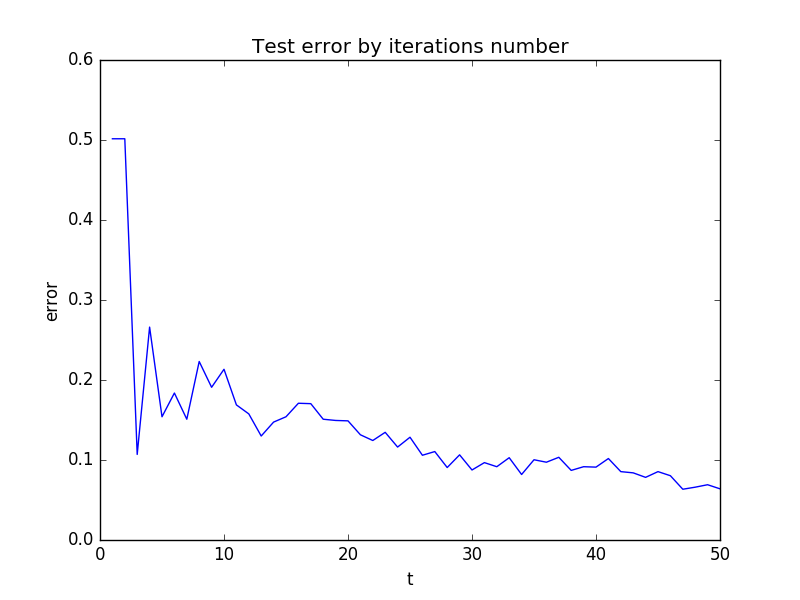
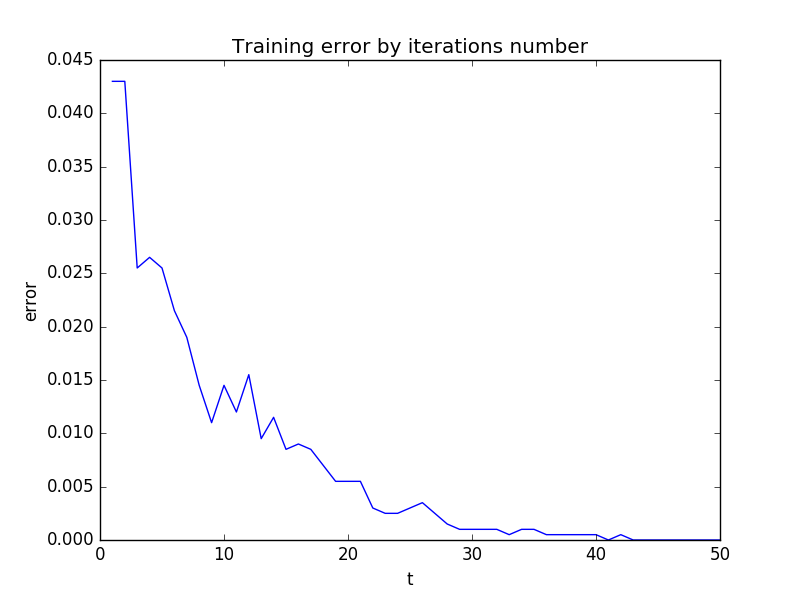
**Q5- Q5.py**

**Q6- Q6.py**

**Question 5:**

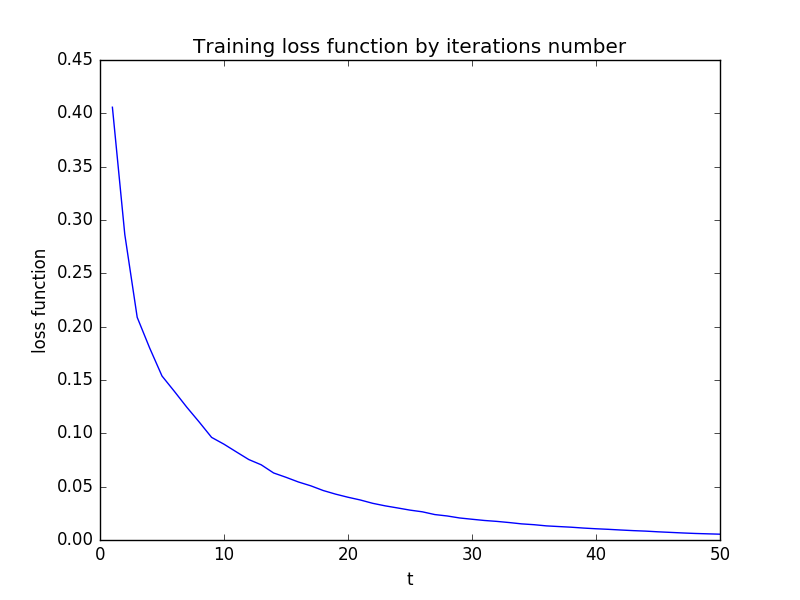
General comments: For our weak learners, we scanned all pixels. For each one, we took all values appeared at the training data (for that pixel) as thresholds. The code takes about 15 minutes to run so we added prints after each iteration containing the results for both sections.

1. Plot of the training and test error with respect to the iteration number :



As seen in the graphs, the errors improve as is bigger (exponentially, as in theory). The test error is bigger than the train error (as in theory), and we still have not started to over fit the train data (since the test error is still decreasing).

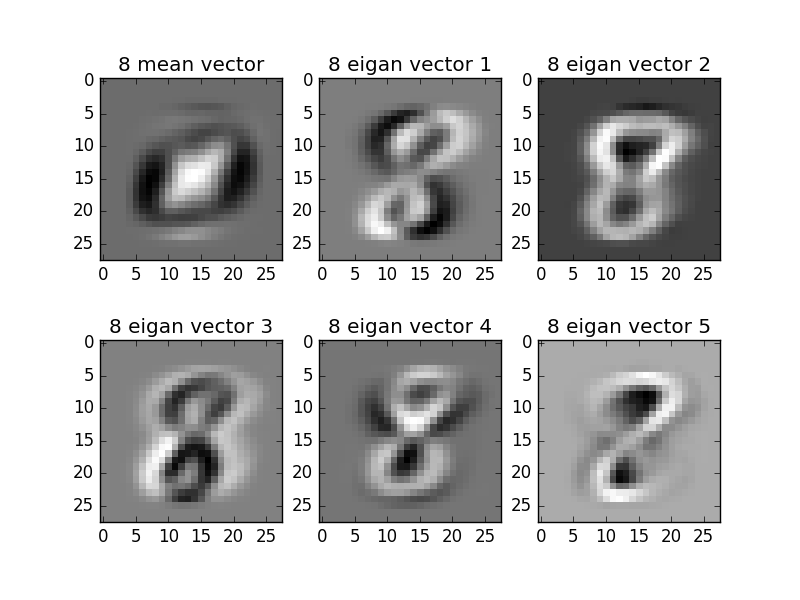
1. Plot of the given loss function for the train and the test data with respect to the number of iterations :



As expected, the loss function at the training data deteriorates quite similarly to the training (exponentially). The test loss function is the surprising part: It doesn't seem to diminish at all, even though the test error does diminish. A possible explanation is that while minimizing the loss function is the same as taking the best weak learners at AdaBoost, we have no promise on the correlation between the two methods beyond the minimal solution.

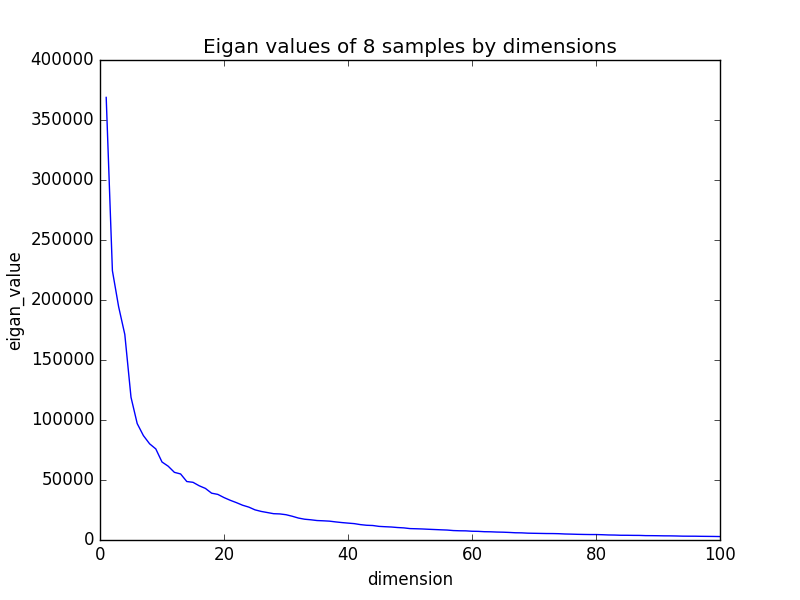
**Question 6:**

1. '8' Eigen vectors:

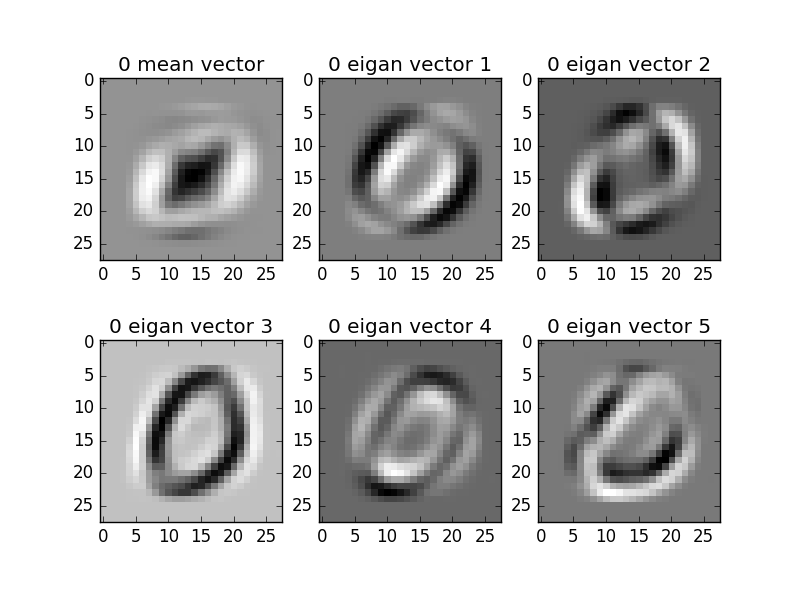


Overall, the digit '8' is quite clear at the first 5 eigenvectors. In addition, notice the contrast between them. This can be explained by the fact some of the digits are in black and part are in white.

'8' eigenvalues:

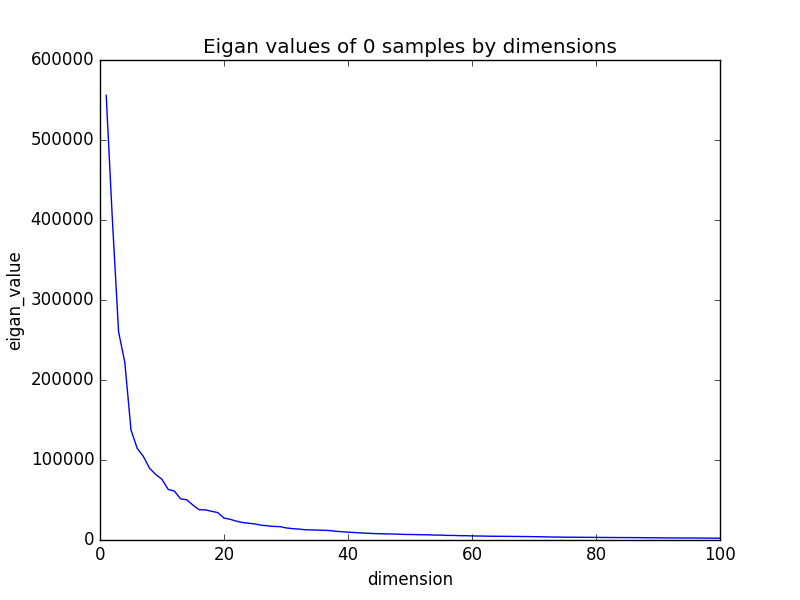


1. '0' eigenvectors:

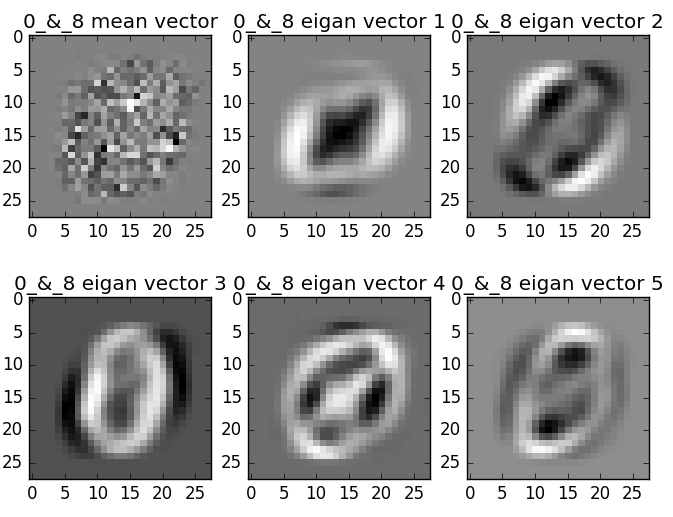


Overall, the digit '0' is quite clear at the first 5 eigenvectors. In addition, notice the contrast between them. This can be explained by the fact that some of the digits are in black and part are in white.

'0' eigenvalues:

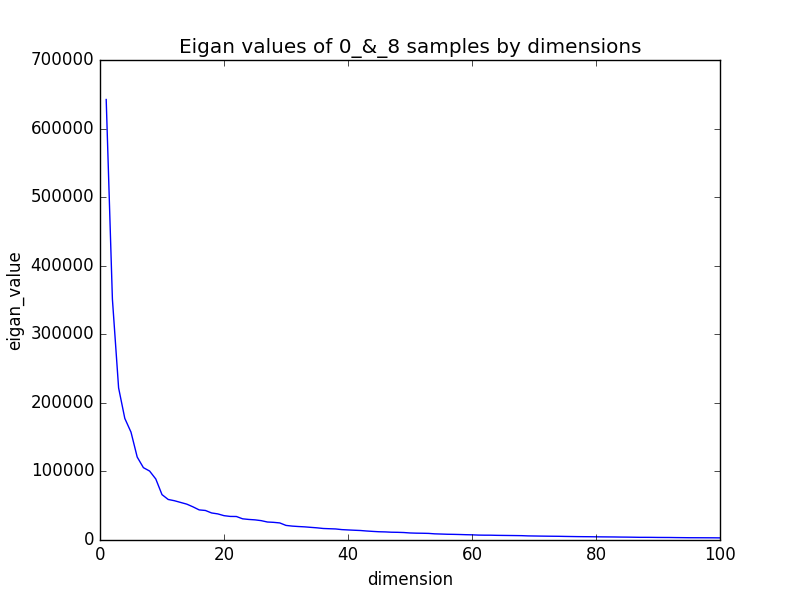


1. Both '0' & '8' eigenvectors:



Notice that now we can see '8' & '0' in each vector. Because there are no labels, each vector tries to "capture" both digits. the differences in the contrast are same as before.

'0' & '8' eigenvalues:



Notice that the magnitude of the eigenvalues is bigger. The reason for it is clear by the following explanation:

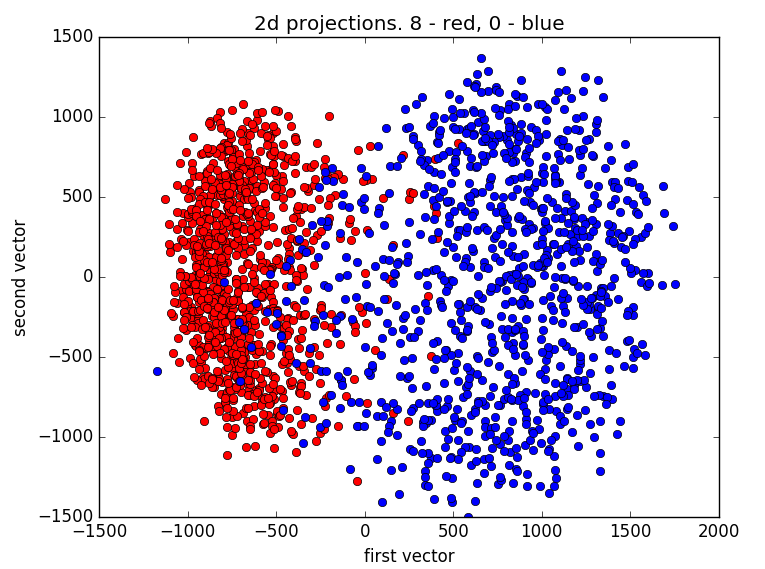
1. With both digits PCA the pixels obviously vary more and thus have bigger variance.

2. These eigenvalues come from the covariance matrix of the pixels, and its diagonal is the pixels’ variance.

3. The trace of every matrix is the sum of its eigenvalues, so the sum of them must be bigger.

4. The covariance matrix is PSD so the eigenvalues are non-negative so we expect them to be bigger.

1. '0' & '8' projections on the first 2 eigenvectors:



To explain this, we need to look closely at the vectors from section c, and recall that black has low values and white has high values.

We see that the first vector "captured" the main feature of '8' (in contrast to '0') which is the middle. The '0' digits covered most of the white pixels in the first vector and not the middle black ones. the '8' digits did cover the black ones and there for are mostly negative. In addition, the Projections on the second vector have the highest variance, in the orthogonal direction of the first projection (like we discussed in recitation).

1. We reconstructed 4 images as instructed. At all of them, we can clearly see the improvement of the reconstruction as k is bigger, which is compatible with the theory.

