Roi Herzig ID: 300360310 EMAIL:roeiherzig@mail.tau.ac.il

Moshe Raboh ID:300611878 EMAIL:shikorab@gmail.com

**Introduction To Machine Learning – EX 4**

1. Directory: " /specific/a/home/cc/students/csguests/roeiherzig/ML/EX4"See main function in file "q5.py".

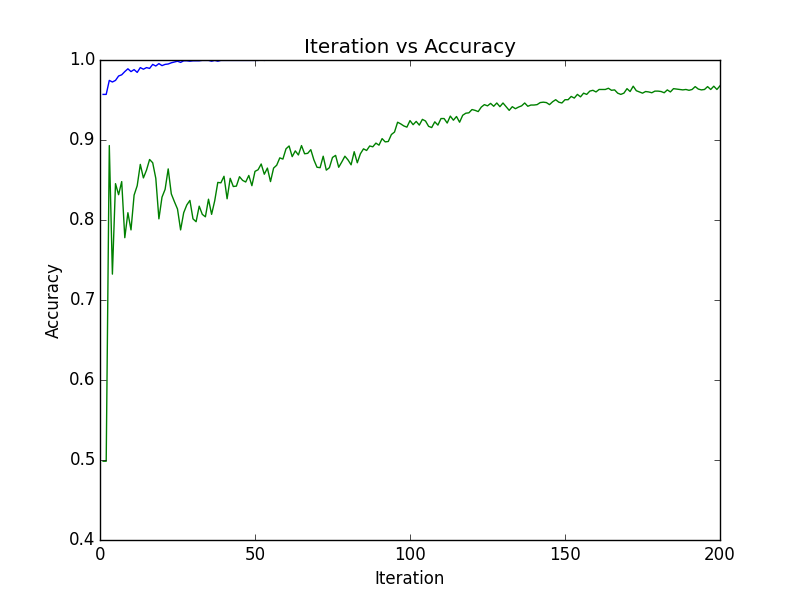
We implemented Ada boost algorithm:  
The algorithm run with T=200 and the training dataset supplied with the EX.  
Algorithm:

* D 🡨uniform distribution
* For t=0..T:
  + Find best hypothesis relative to distribution D  
    Best hypothesis include the following params:
    - pixel index
    - theta
    - type – the value of prediction in case bigger than the threshold
    - accuracy relative to D

best\_hypo 🡨 FindBestHypo(training\_data\_set, D)

* + SaveBestHypoParams(t, best\_hypo)  
    including calculate alpha
  + UpdateDistribution(D, training\_data\_set, labels, best\_hypo)

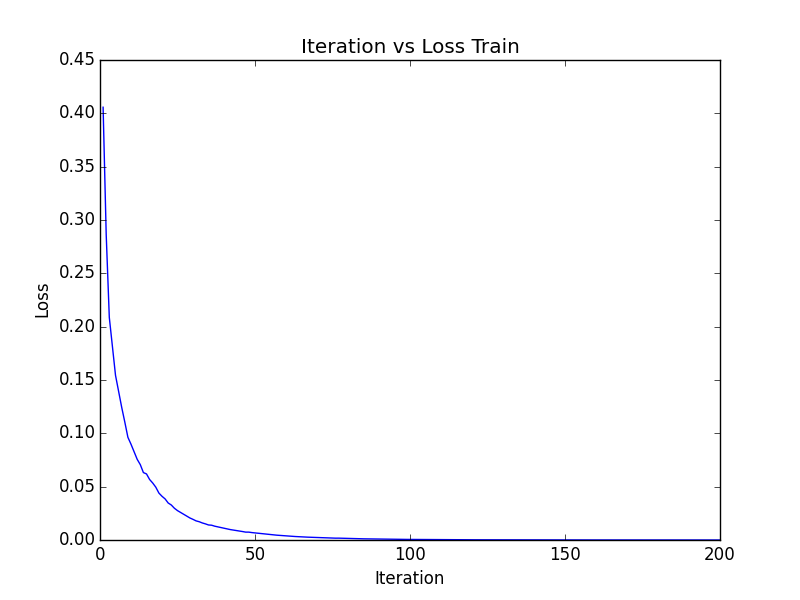
**A:**Directory: " /specific/a/home/cc/students/csguests/roeiherzig/ML/EX4"See function "part\_a" in file "q5.py"  
Create Image “q5\_part\_a.png” – training and test accuracy vs T



We calculated the accuracy in the training data set and test data as a function of T. **I**n the image we can that the training accuracy increases exponentially as proved in class.  
In addition the test accuracy continue to improves after the training accuracy is 1 due to the fact that we are using additional information extracted from data to differentiate and increase the margin.

**B:**

Directory: " /specific/a/home/cc/students/csguests/roeiherzig/ML/EX4"See function "part\_b" in file "q5.py"  
Create Image “q5\_part\_b\_train.png”, “q5\_part\_b\_test.png” – training and test loss vs T.  
We calculated the loss in the training data set and test data as a function of T.  
In the images below we can see that the training loss drops exponentially similarly to error we saw in section A and demonstrate the fact that Adaboost tries to minimize the empirical loss function.  
The test loss, however, might emphasize the test errors, as T increase, more hypotheses will agree on some error prediction, the sum of this error prediction will increase as T increase and therefore the relevant elements in the loss function might exponentially increase.







**A + B + C:**

Directory: " /specific/a/home/cc/students/csguests/roeiherzig/ML/EX4"See function "part\_a\_ and\_b\_and\_c " in file "q6.py"  
The function gets the training data set and the required labels (0, 8 or both)  
We create the following plots:

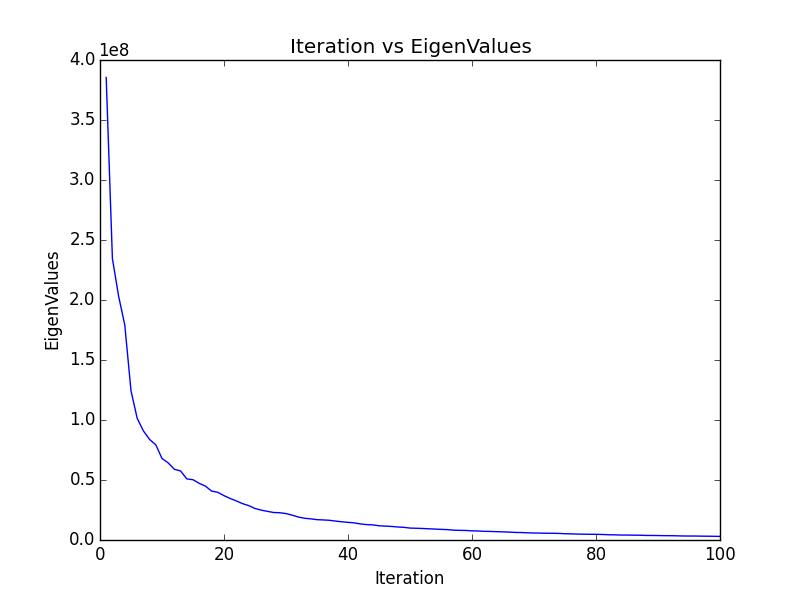
* Images for the first 5 eigenvectors -   
  named “q6\_part\_<part>\_i<index>\_label\_<label>.png”
* Plot of of the 100 eigenvalues –   
  named “q6\_part\_<part>\_label\_<label>.png”
* Mean image -   
  named “q6\_part\_<part>\_mean\_image.png”

The images can be found below.

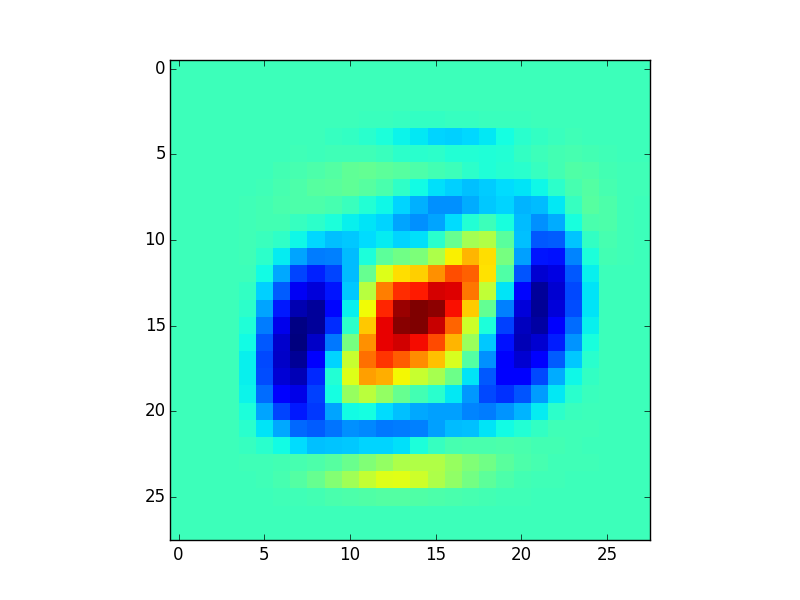
In section A, we analyzed only 8 labels, we can see that eigenvalues decrease very rapidly, which means that most of the information of the images can be represented by a small amount of eigen vectors.  
The most significant eigenvectors are some part of “8” shapes which allow to create the required 8 with a linear combination of them.  
Section B shows that this analysis fits to label 0 as well.  
In section C, which is the combination of label 8 and 0, will generally have larger variance.  
We have bigger magnitude of eigenvalues, which means that the combination of both of the images has larger variance in the direction of the first eigenvectors.

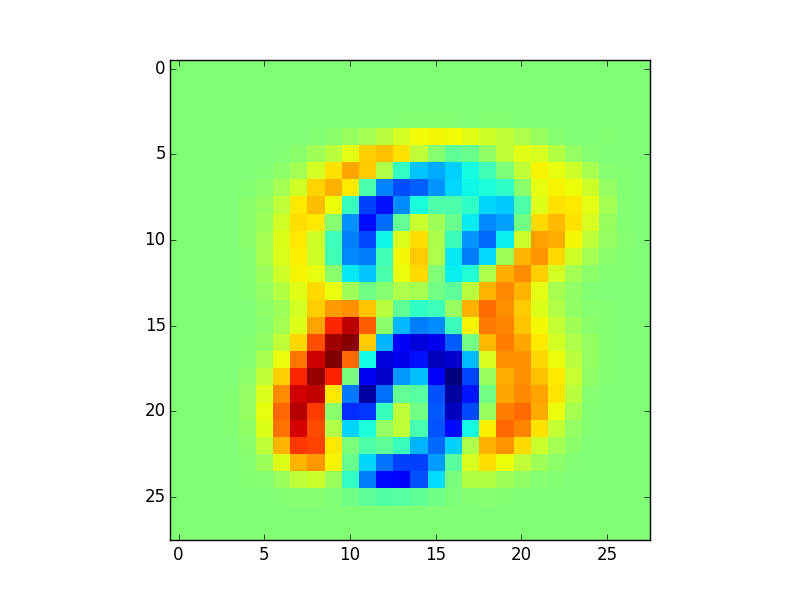
In addition, we can see that mean image in section C is very close to 0 (since we mean centered the data set at the beginning of the question) and therefore the mean images of 8 and 0 separately are complimentary of each other (there sum of both should be 0 as in Section C) and shows the mean differences between an 8 and a 0.

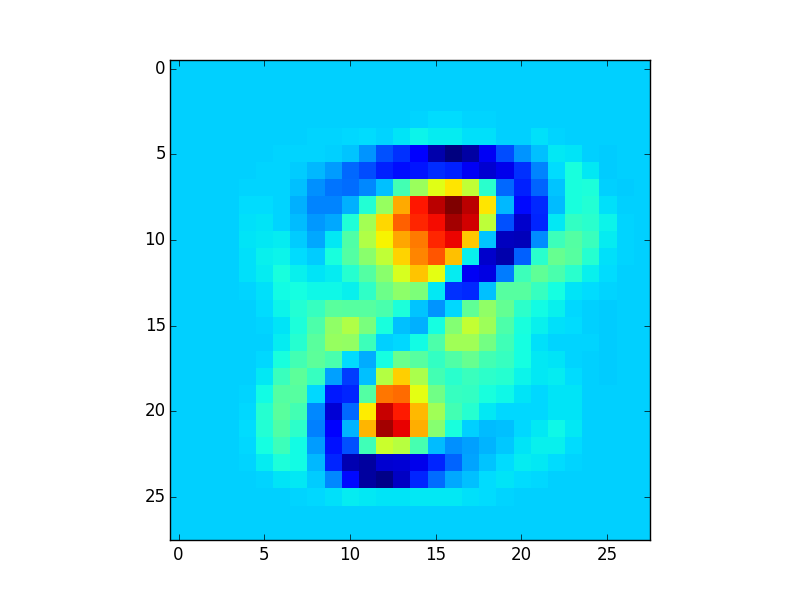
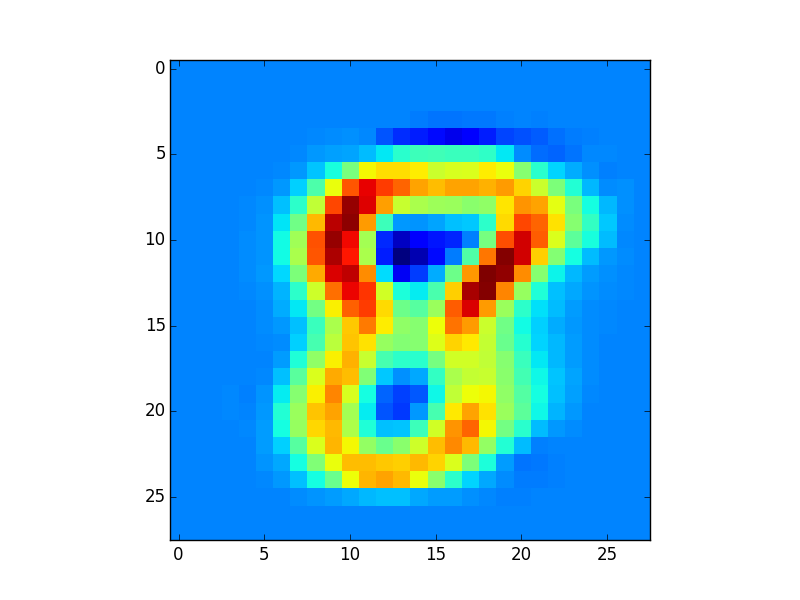
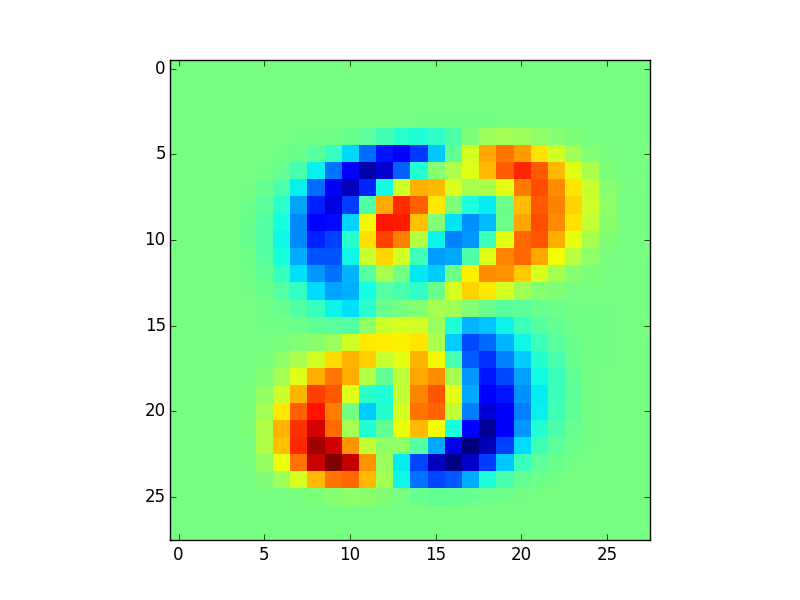
Part A: label – 8  
List of eigenvalues:



Mean Image:



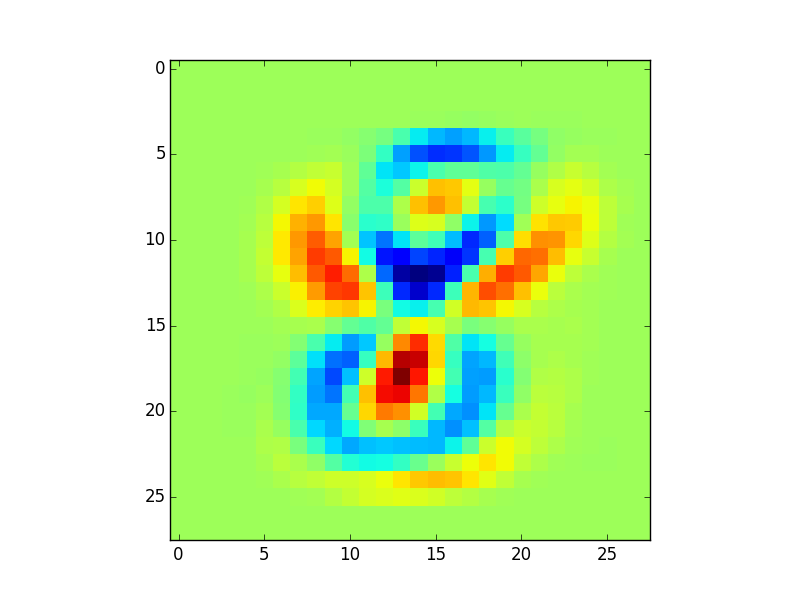
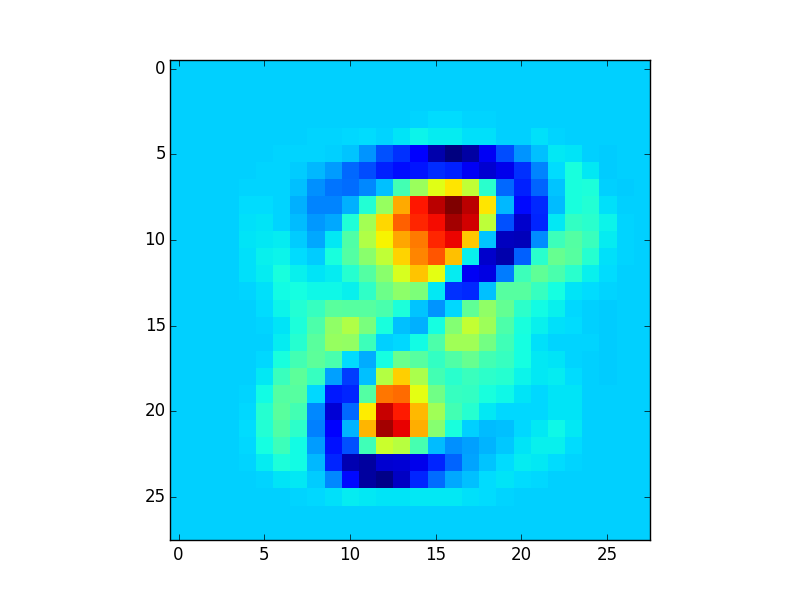
Eigen-vectors:



Part C: label – 0 and 8

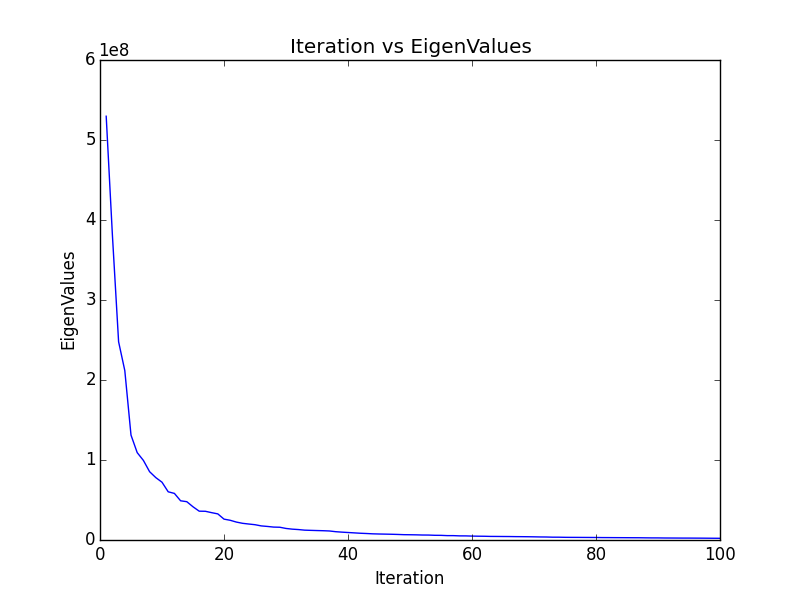
Mean Image:

Eigen-vectors:

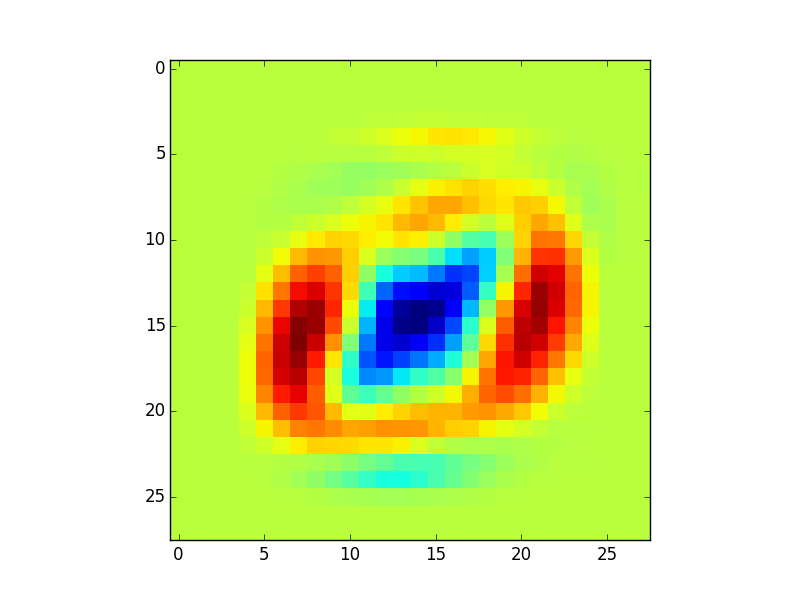


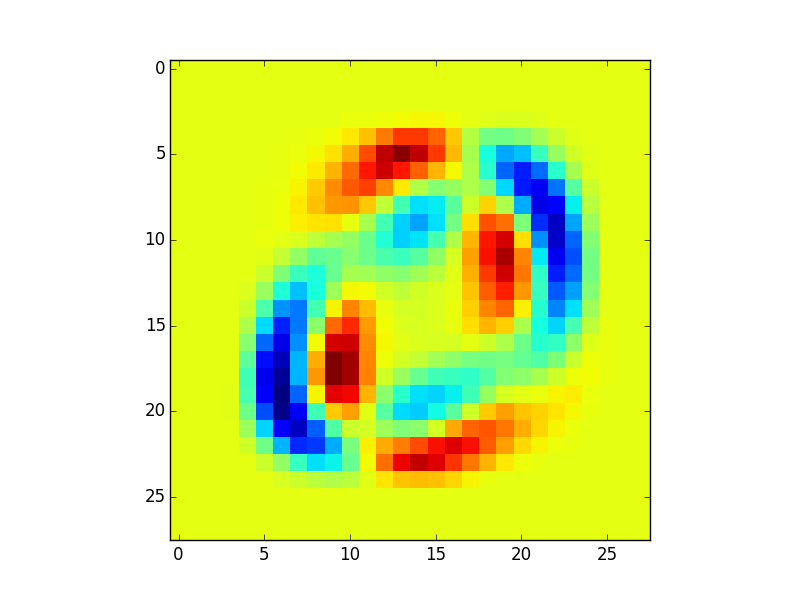
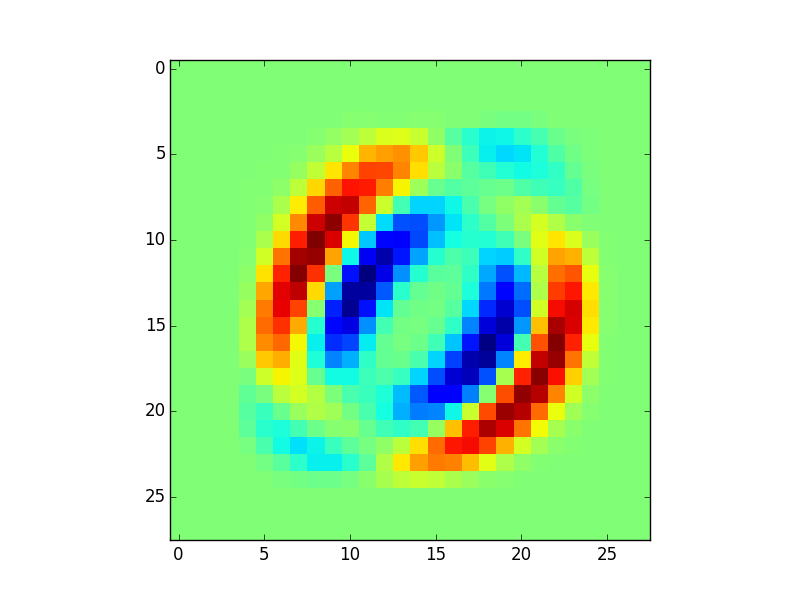
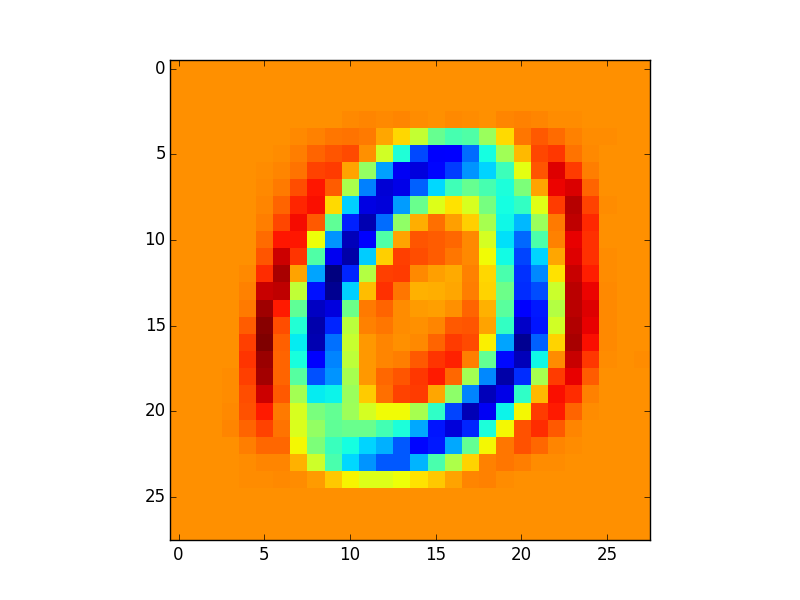
Part B: label – 0

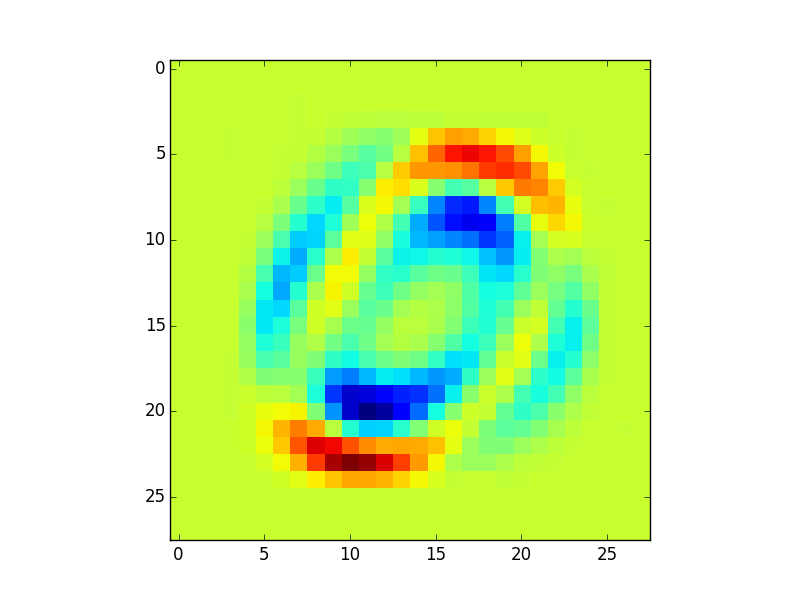
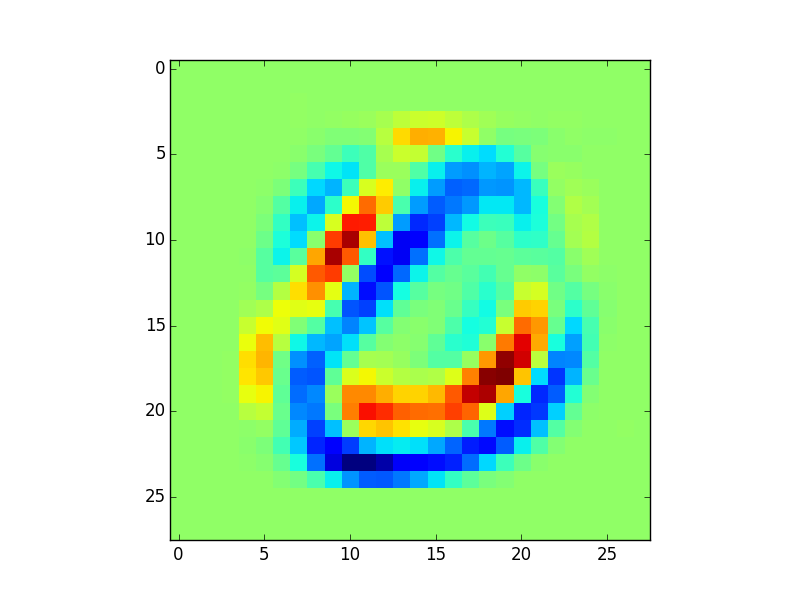
List of eigenvalues:



Mean image:

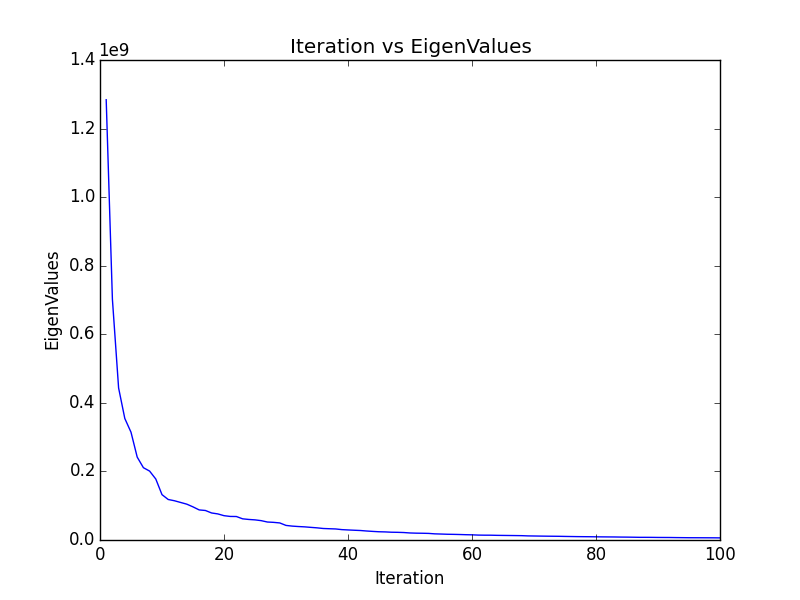
  
Eigen-vectors:



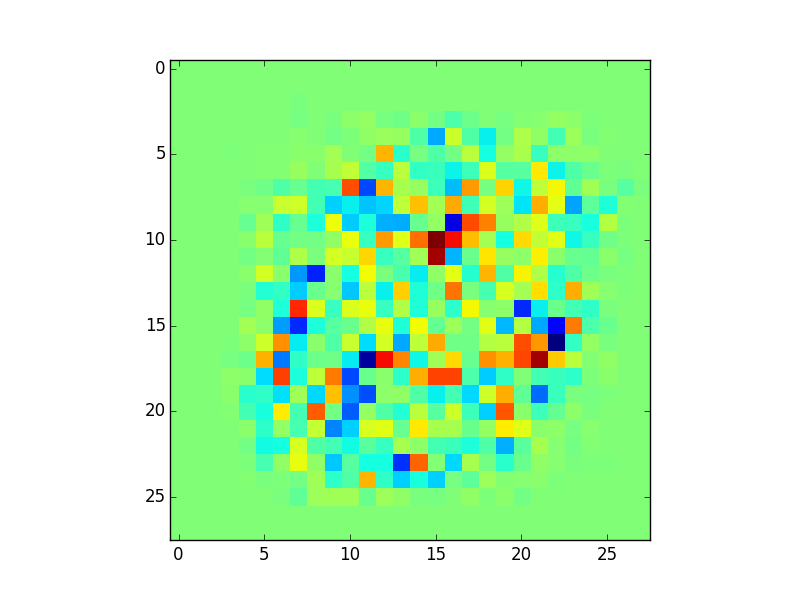
Part D:

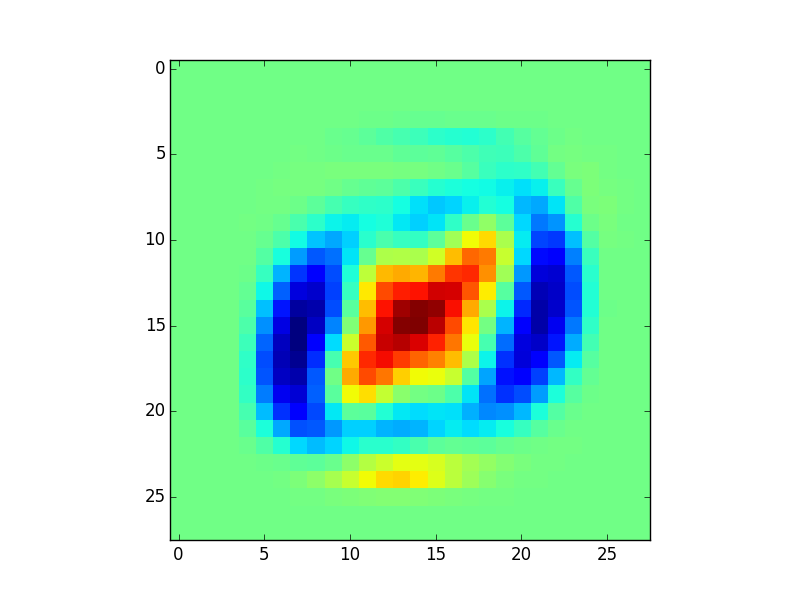
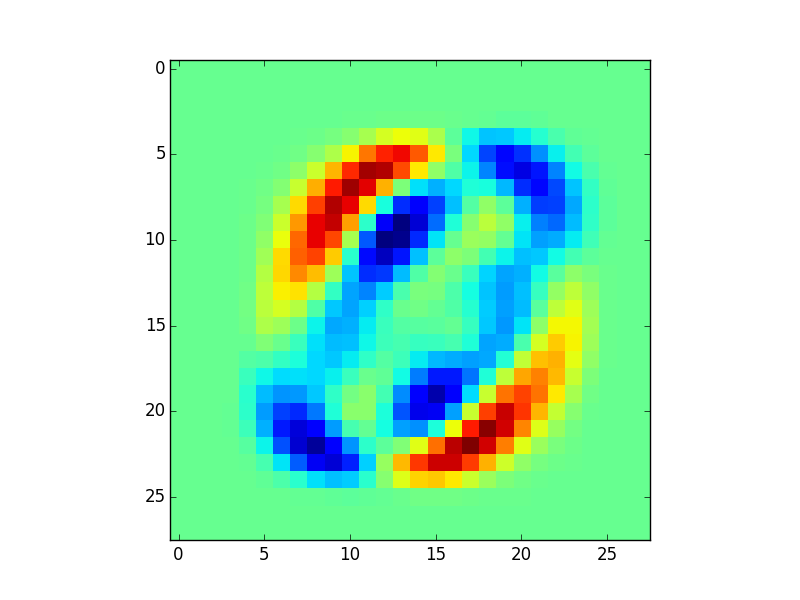
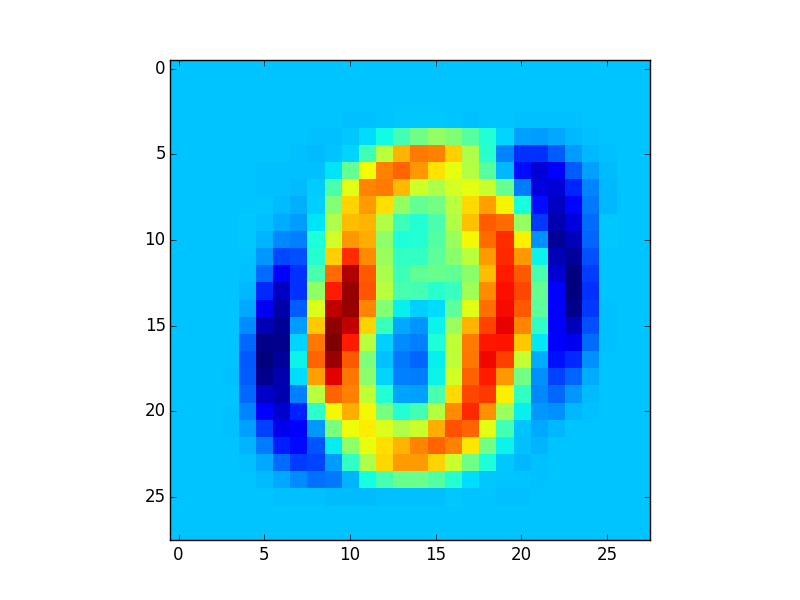
Part C: label – 0&8

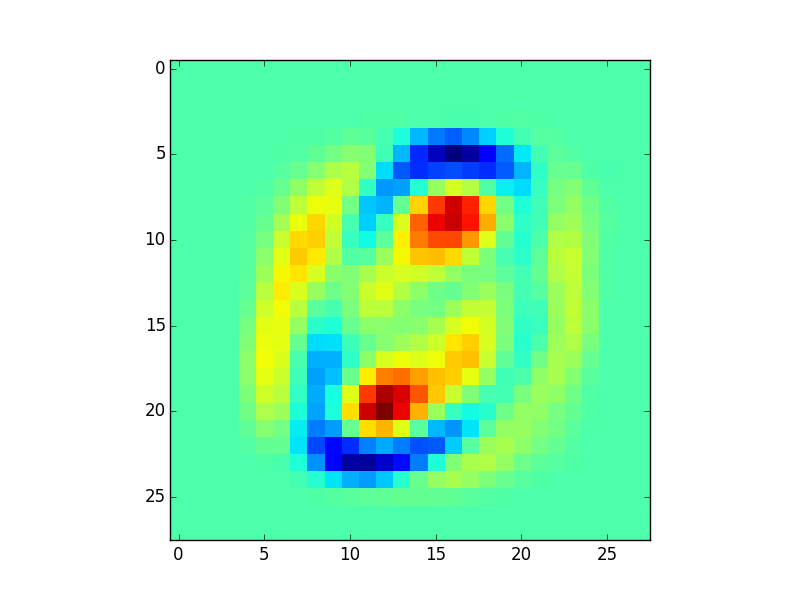
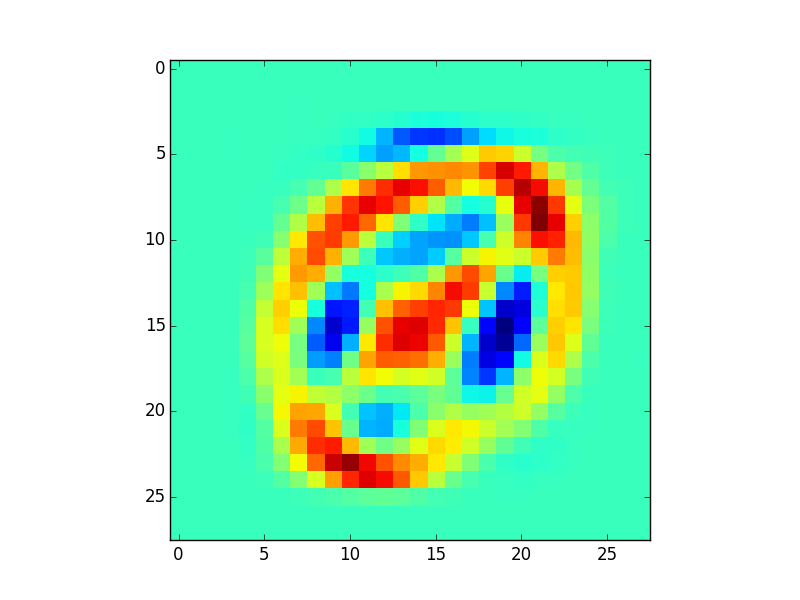
List of eigenvalues:



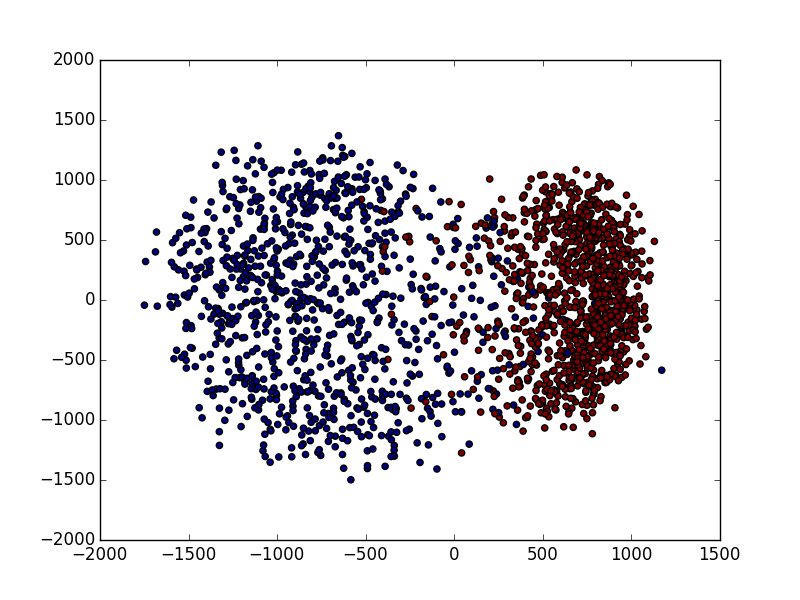
Mean image:

  
Eigen-vectors:





D:  
Directory: " /specific/a/home/cc/students/csguests/roeiherzig/ML/EX4"See function "part\_d " in file "q6.py"  
Create image “q6\_part\_d.png” which is a 2d scatterplot showing the projections of the images on the first two principal axes.  
Those principal axes hold the most significant information when trying to classify 8’s and 0’s. The scatter plot below shows that with this principal axis alone will allow us to create a decent classifier.



E:

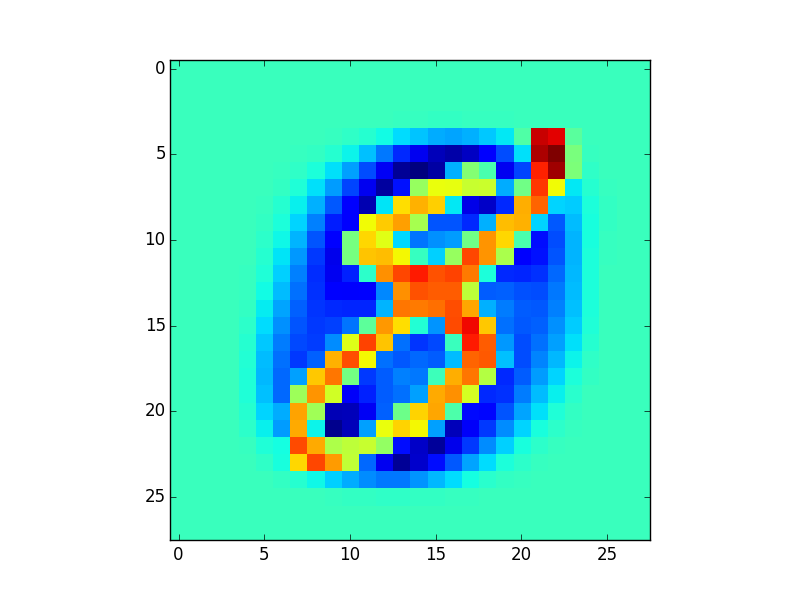
Directory: " /specific/a/home/cc/students/csguests/roeiherzig/ML/EX4"See function "part\_e " in file "q6.py"

Create the following images (per image index and k):

* “q6\_part\_e\_original\_<index>\_k\_<k\_val>.png” the original image before the encoding-decoding process.
* “q6\_part\_e\_decode\_<index>\_k\_<k\_val>.png” the image after the encoding-decoding process.

The function gets four images (two positives and two negatives) and encodes and decodes it using PCA with dimensions k (10, 30, and 50).  
The images below demonstrate that with small number of principal axes allows us to preserve most of the information of the original pictures.  
Those images are the original image vs the images recovered (process of encoder-decoder to some lower dimensional space K). K = 10 tends to lose some of the information, but bigger K seems to preserve most of the information.

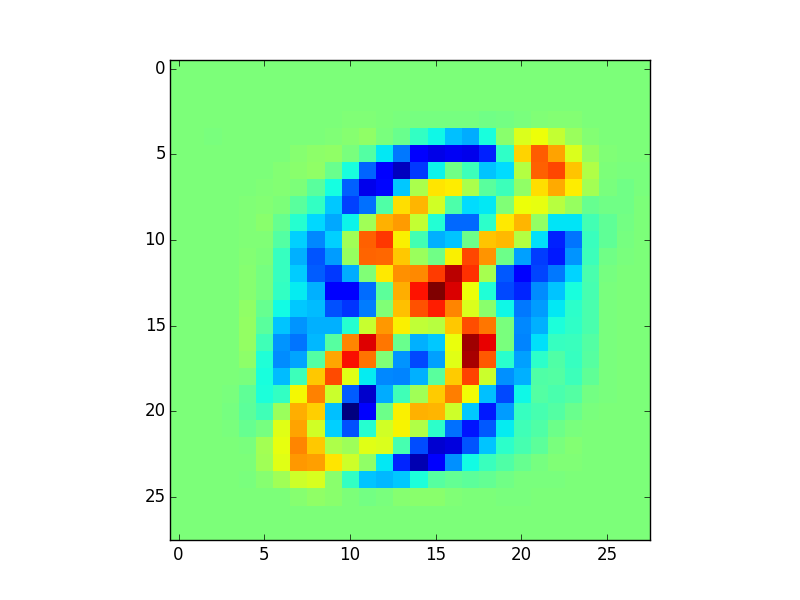
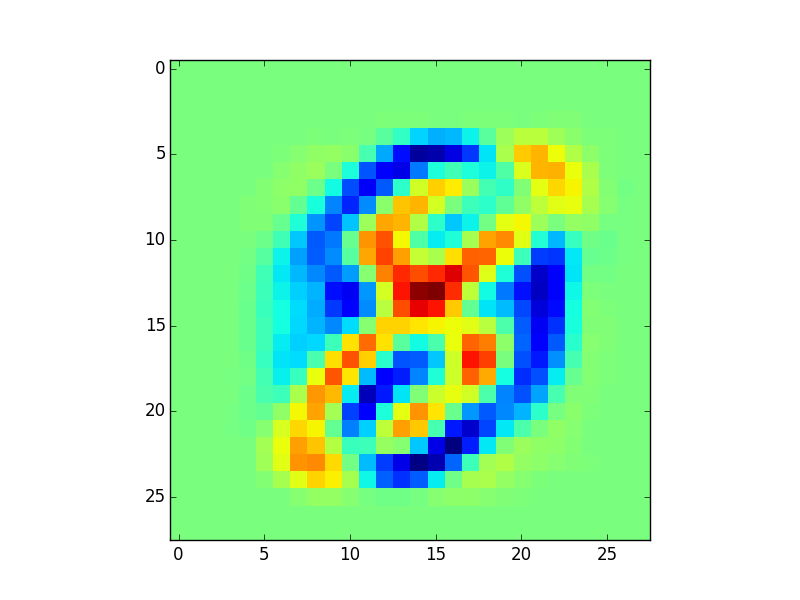
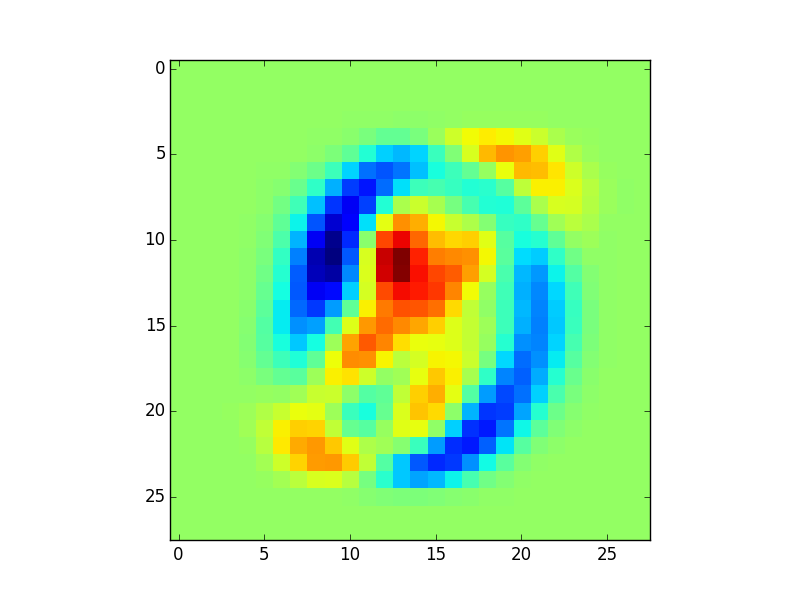
Original



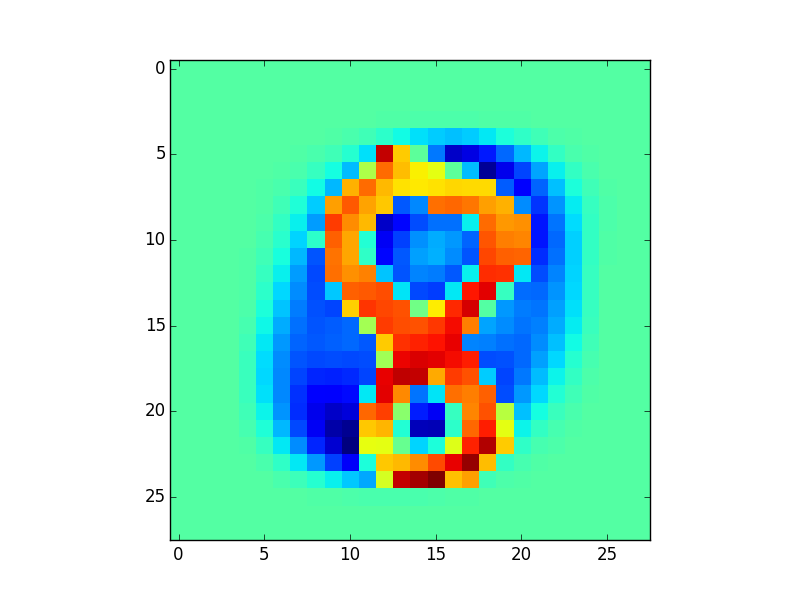
K=50

K=30

K=10



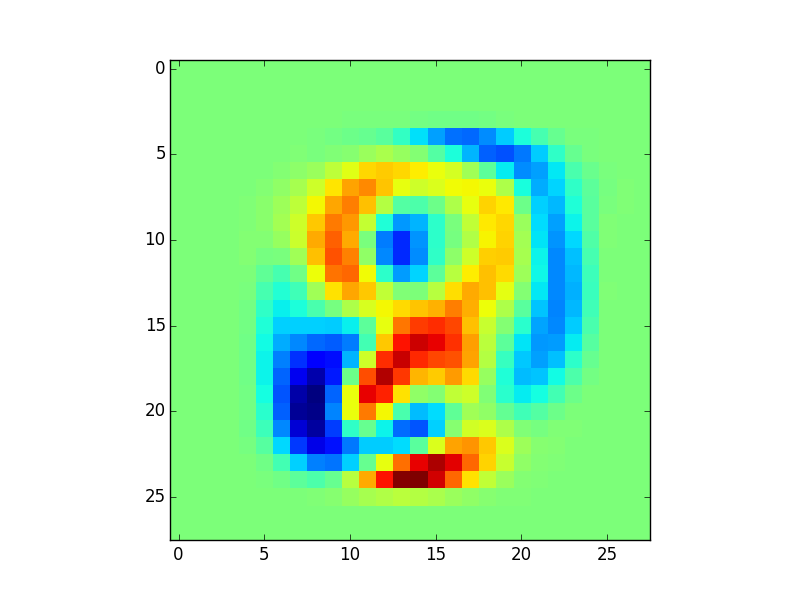
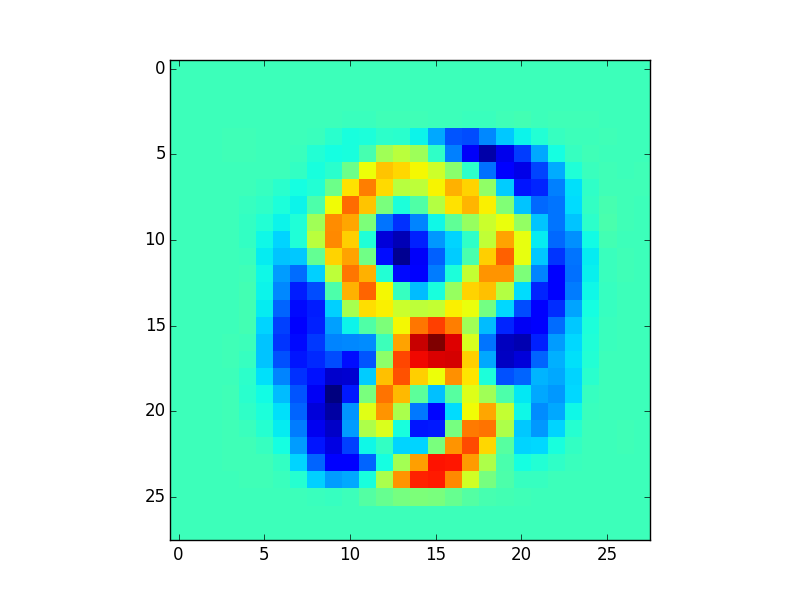
Original

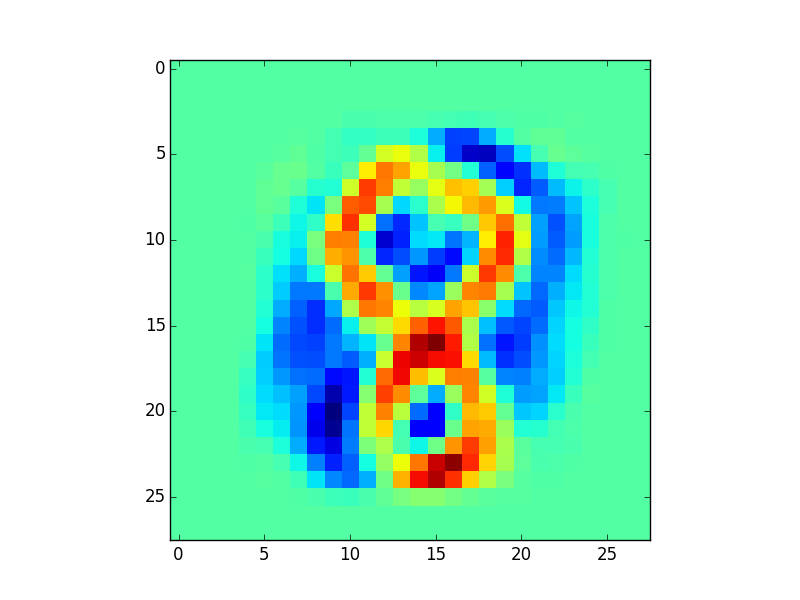


K=50

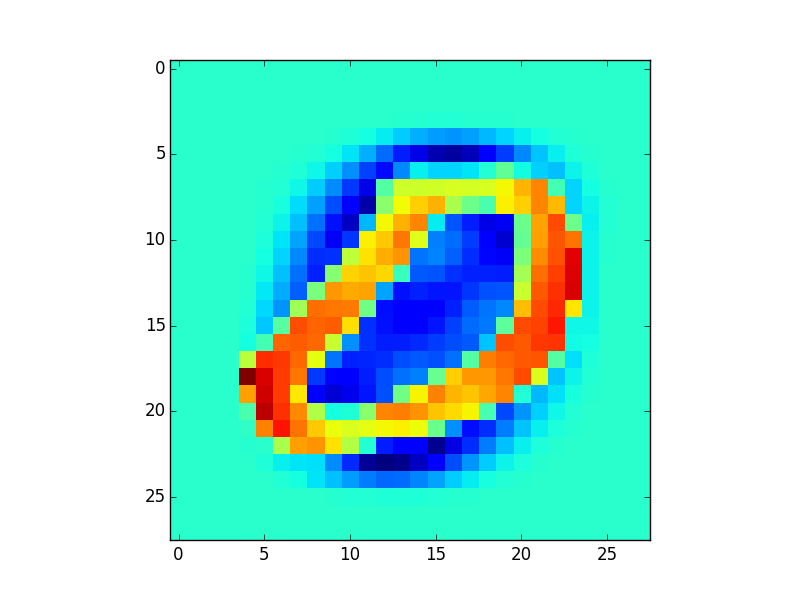
K=30

K=10





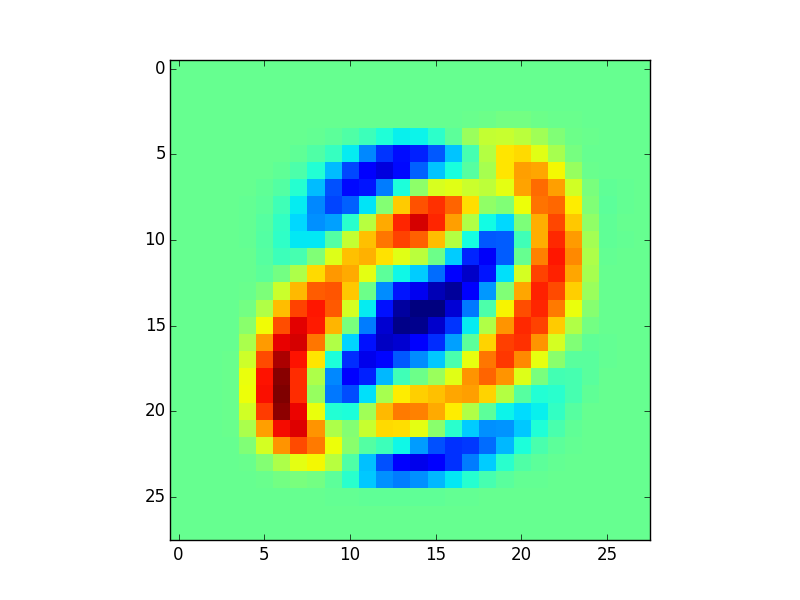
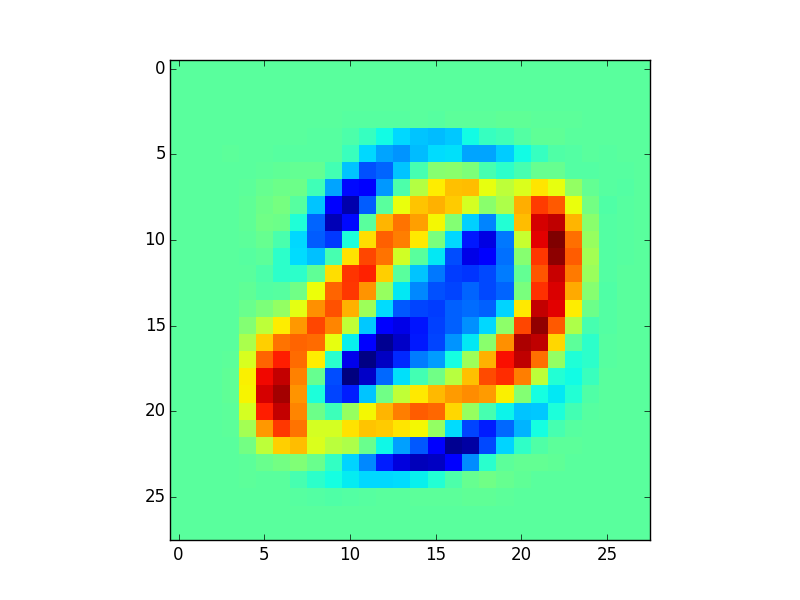
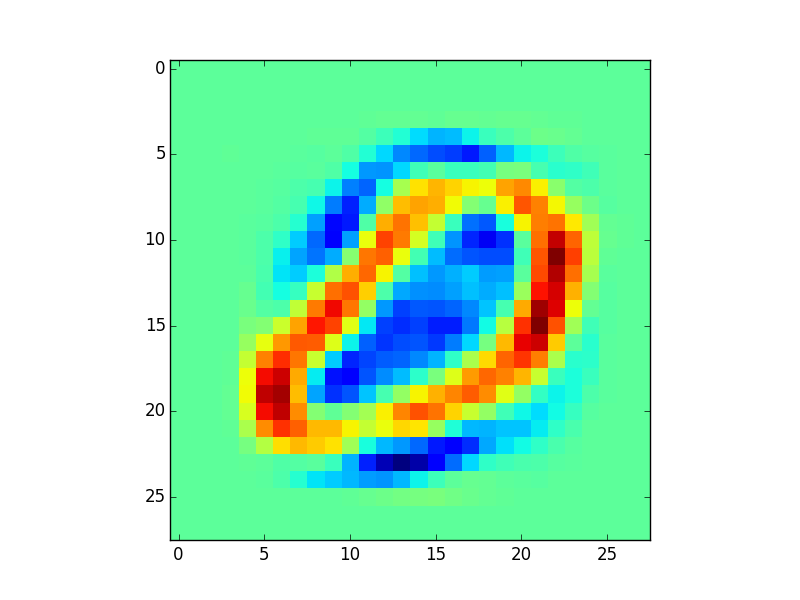
Original



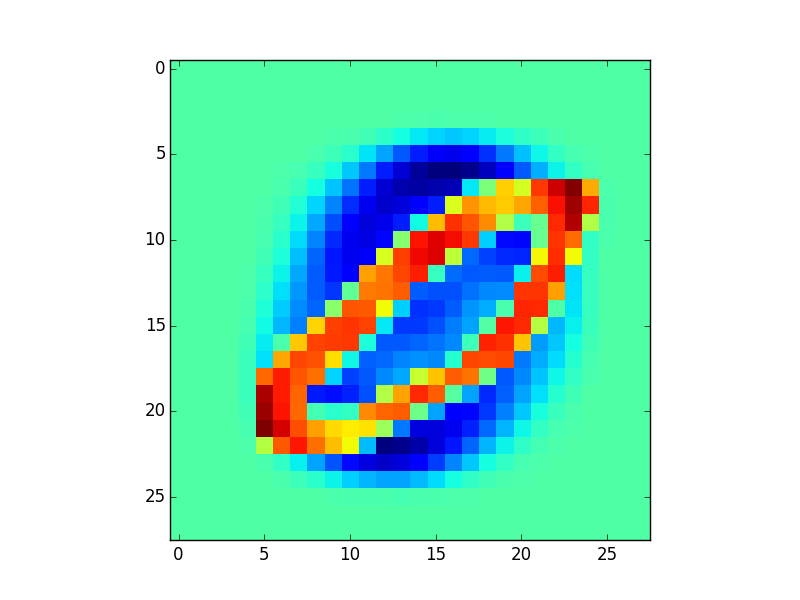
K=50

K=30

K=10



Original



K=50

K=30

K=10

