Introduction to Machine Learning

Homework 4

Jan 22, 2016 Nataly Brukhim, Tal Schuster

Theory Questions

1. (a) Assume we wish to do binary classification over the domain $[0,1]$. Let h_1, h_2, h_3 be 3 hypotheses such that their error intervals over $[0,1]$ are $[0,2\epsilon]$, $[\epsilon,3\epsilon]$ for h_1, h_2, respectively, and the error intervals of h_3 are $[0,\epsilon]$ and $[2\epsilon,3\epsilon]$. We have $error(h_i) = 2\epsilon$ for each i, but for each $x \in [0,3\epsilon]$, there are 2 hypotheses that return the wrong label for x, so the majority vote of the hypotheses is wrong over the whole interval $[0,3\epsilon]$, i.e. $error(h) = 3\epsilon$.

   (b) The maximal error of h is obtained by choosing $h_1, ..., h_{2k+1}$ such that their total error is split over the maximum possible disjoint portions of the domain for which the majority is wrong, i.e. by splitting the error to different portions of the domain such that the minimal amount of hypotheses in each such portion is wrong so that h is wrong, which is $k+1$. Specifically, the maximal error of h is bounded by

$$error(h) \leqslant \frac{(2k+1)\epsilon}{k+1} < \frac{(2k+2)\epsilon}{k+1} = 2\epsilon$$

2. (a) Not for submission.

   (b)

$$\Pr_{x\sim D_{t+1}}[h_t(x) \neq y] = \sum_x D_{t+1}(x) I[h_t(x) \neq y] = \sum_x \frac{D_t(x)\cdot e^{-\alpha_t y h_t(x)}}{Z_t} I[h_t(x) \neq y] =$$

$$= \sum_{x:h_t(x)\neq y} \frac{D_t(x)\cdot e^{\alpha_t}}{Z_t} = \frac{e^{\alpha_t}}{Z_t} \sum_{x:h_t(x)\neq y} D_t(x) = \frac{e^{\alpha_t}}{Z_t} \sum_x D_t(x) I[h_t(x) \neq y] =$$

$$= \frac{e^{\alpha_t}}{Z_t} \Pr_{x\sim D_t}[h_t(x) \neq y] = \frac{e^{\alpha_t}}{Z_t}\epsilon_t = \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{Z_t} = \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{2\sqrt{\epsilon_t(1-\epsilon_t)}} = \frac{1}{2}$$

   (c) AdaBoost learnes at each iteration t a hypothesis $h_t \in H$ where H is a class of weak learners, i.e. there exists $\gamma > 0$ such that $error(h_t) \leqslant \frac{1}{2} - \gamma$. Hence, we have

$$\Pr_{x\sim D_{t+1}}[h_{t+1}(x) \neq y] < \frac{1}{2}$$

   Since we have already shown that,

$$\Pr_{x\sim D_{t+1}}[h_t(x) \neq y] = \frac{1}{2}$$

   we conclude that $h_t \neq h_{t+1}$.

3. (a)

$$\bar{K}'_{i,j} = \langle y_i, y_j \rangle = (\phi(x_i) - \frac{1}{m}\sum_{k=1}^m \phi(x_k))^T \cdot (\phi(x_j) - \frac{1}{m}\sum_{l=1}^m \phi(x_l)) =$$

$$\phi(x_i)^T\phi(x_j) - \phi(x_i)^T\frac{1}{m}\sum_{l=1}^m \phi(x_l) - \phi(x_j)\frac{1}{m}\sum_{k=1}^m \phi(x_k)^T + \frac{1}{m^2}\sum_{k=1}^m \phi(x_k)^T \sum_{l=1}^m \phi(x_l) =$$

$$\phi(x_i)^T\phi(x_j) - \frac{1}{m}\sum_{k=1}^m \phi(x_i)^T\phi(x_k) - \frac{1}{m}\sum_{k=1}^m \phi(x_k)^T\phi(x_j) + \frac{1}{m^2}\sum_{k=1}^m\sum_{l=1}^m \phi(x_k)^T\phi(x_l)$$

$$K(x_i,x_j) - \frac{1}{m}\sum_{k=1}^m K(x_k,x_i) - \frac{1}{m}\sum_{k=1}^m K(x_k,x_j) + \frac{1}{m^2}\sum_{k=1}^m\sum_{l=1}^m K(x_k,x_l)$$

Therefore, if we assume the calculation of the kernel $K(x_i, x_j)$ takes $O(1)$, then the complexity to compute $\bar{K}'_{i,j}$ is $O(1) + 2O(m) + O(m^2) = O(m^2)$. Note that the value of the last, most expensive, phrase is not determent in i or j so it could be computed only once for $\bar{K}'$.
Hence, the computation of all $\frac{m^2}{2}$ (symmetric) values in $\bar{K}'$ is of complexity $\frac{m^2}{2}O(m)+O(m^2) = O(m^3)$

(b) To show that the u_j is a linear combination of the samples we will first show that, since it is an eigenvector:

$$\lambda_j u_j = \Sigma u_j = \frac{1}{n}\sum_i x_i x_i^T u_j = \frac{1}{n}\sum_i (x_i^T u_j) x_i \tag{1}$$

The second part is from equation 21 in the PCA lecture scribes. We get:

$$u_j = \sum_i \underbrace{\frac{(x_i^T u_j)}{n\lambda_j}}_{\text{scalar}} x_i = \sum_i a_i^{(j)} x_i \tag{2}$$

Therefore, the principal components are linear combinations of the samples. Using equation 1 to project a sample x_i and replacing u_j by equation 2, we get:

$$x_i^T \Sigma u_j = \lambda_j x_i^T u_j \quad \Rightarrow \quad x_i^T \frac{1}{n}\sum_k x_k x_k^T \sum_l a_l^{(j)} x_l = \lambda_j x_i^T \sum_l a_l^{(j)} x_l \tag{3}$$

$$\frac{1}{n}\sum_{l,k} a_l^{(j)} [x_i^T x_k][x_k^T x_l] = \lambda_j \sum_l a_l^{(j)} [x_i^T x_l] \tag{4}$$

Denoting the dot product $[x_i^T x_j]$ as $K_{i,j}$, we get from equation 4:

$$K^2 a^{(j)} = N\lambda_j K a^{(j)} \quad \Rightarrow \quad K a^{(j)} = n\lambda_j a^{(j)} \tag{5}$$

Hence, for each eigenvalue λ_j we can compute a new eigenvalue $n\lambda_j$ for K and compute the eigenvector $a^{(j)}$. We want the eigenvectors to be normalized and therefore:

$$1 = \|u_j\| = u_j u_j^T = \sum_{i,k} a_i^{(j)} a_k^{(j)} [x_i^T x_j] = {a^{(j)}}^T K a^{(j)} = n\lambda_j {a^{(j)}}^T a^{(j)} = 1 \tag{6}$$

Hence:

$$\|a^{(j)}\| = \sqrt{\frac{1}{n\lambda_j}} \tag{7}$$

(c) When receiving a new point x, $\langle u_j, \phi(x)\rangle$ can be calculated by:

$$u_j \phi(x) = \sum_i a_i^{(j)} \phi(x_i)\phi(x) = \sum_i a_i^{(j)} K(x_i, x)$$

The complexity for computing each j product, assuming $K(x_i, x)$ is $O(1)$, is O(m) since a is of size m. Therefore, the computation of all k products is $O(km)$.

4. We would like to find the w with the minimal l_2 norm such that $X^T X w = X^T y$.
Given the SVD of $X = U\Sigma V^T$, we can write the constraint as:

$$V\Sigma^T U^T U\Sigma V^T w = V\Sigma^T \Sigma V^T w = V\Sigma^T U^T y$$

When X is singular, we get multiple solutions for w. Those solutions are a combined from the $w^{\parallel} \in Im(X^T X)$ which is necessary for holding the constraint and $w^{\perp} \in Ker(X^T X)$ that does not affect the solution but can alter the size of the norm of w. Using the pseudo-inverse, we can solve this:

$$w = (V\Sigma^T \Sigma V^T)^+ V\Sigma^T U^T y = V(\Sigma^T \Sigma)^+ V^T V\Sigma^T U^T y$$

$$w = V(\Sigma^T \Sigma)^+ \Sigma^T U^T y = X^+ y$$

This solution is also the one with the minimal $\|w\|$. To show this, let define $P = X^+ X$.
Note that $Pw = X^+ X X^+ y = X^+ y = w$. The middle equation is from the equality:

$$X^+ X X^+ = V\Sigma^+ U^T U\Sigma V^T V\Sigma^+ U^T = V\Sigma^+ \Sigma\Sigma^+ U^T = V\Sigma^+ U^T = X^+$$

The next to last equation is true because Σ^+ is in the image space of $\Sigma^+\Sigma$
Note also that $P^T = P$ and denote $\hat{w}$ as some solution for w that holds the constraint. so: $x\hat{w} = y$.
We will show that the $\|\hat{w}| \geqslant \|w\|$ and so w is the optimal solution. First, we will notice that:

$$w^T(\hat{w} - w) = (Pw)^T(\hat{w} - w) = w^T P(\hat{w} - w) = w^T(X^+ X\hat{w} - w) = w^T(X^+ y - w) = w^T(w - w) = 0$$

We will look at the squared norm of $\hat{w}$ and get that:

$$\|\hat{w}\|^2 = \|w + (\hat{w} - w)\|^2 = \|w\|^2 + 2w^T(\hat{w} - w) + \|\hat{w} - w\|^2$$

$$= \|w\|^2 + \|\hat{w} - w\|^2 \geqslant \|w\|^2$$

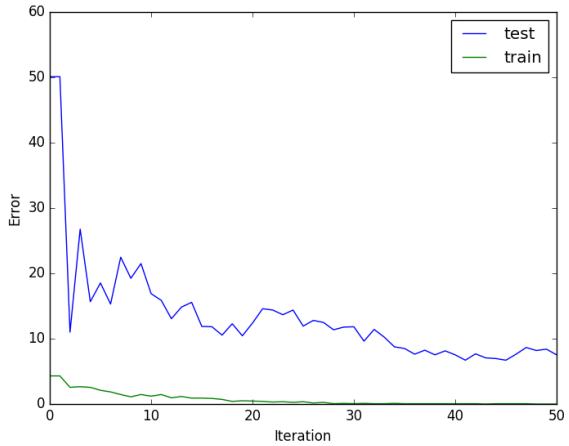
Hence, every other solution of w have greater or equal norm and so, using the pseudo-inverse leaves us with only $w^{\parallel} \in Im(X^T X)$ and the optimal solution is:

$$w = X^+ y$$

Programming Assignment

5. The code for the following questions can be found in:
"/a/home/cc/students/cs/talschuster/new_hw/ex4/q5/run.py"

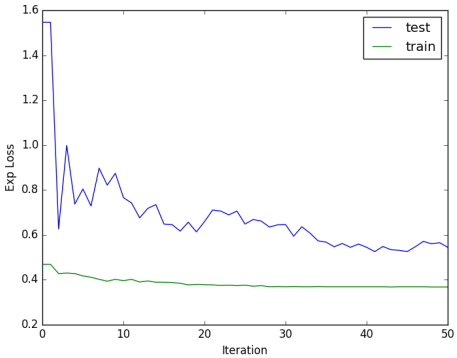
   (a) Errors per iteration:



The error graph is a bit noisy since, with time, more weight is given to the wrong classified samples and eventually it succeed to reach zero training error. However, the generalization of this model is not so good since it seem to overfit the training data very fast while the test error remains high. This might be a cause of a small training set.
Some more intuition for the training could be gained by observing the pixel locations of the weak learners that where used with iterations. The images show from left to right the positions of the weak learners for every 5 iterations. Darker dots represent greater coefficient for that learner.

One could notice that, like on previous our previous 0/8 classification exercises, the middle pixels have are most important for the classification while the surroundings could add some valuable information.

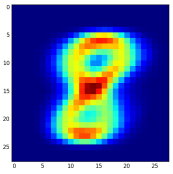
   (b) Exponential loss per iteration:



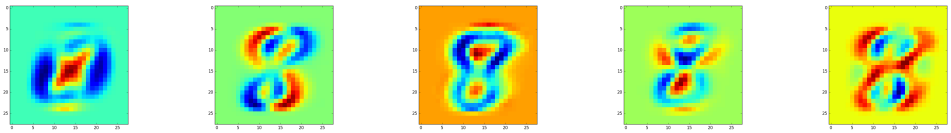
One could notice that the graph behaves similar to the errors graph which is coherent with what we saw in class.

6. The code for the following questions can be found in:
"/a/home/cc/students/cs/natalybr/ML/ex4/**X**.py", for **X**=a/b/c/d/e.

(a) The plot of the mean image (of the unscaled data) is,

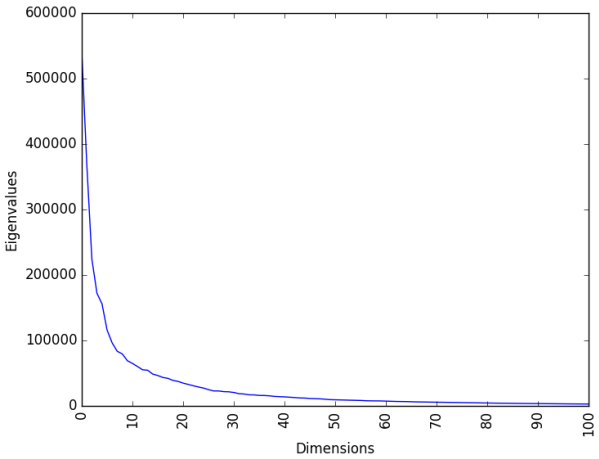


The following plots are the first 5 eigenvectors,



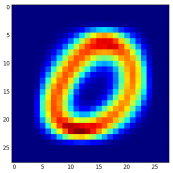
1 2 3 4 5

The following is a plot of the eigenvalues as a function of dimension,

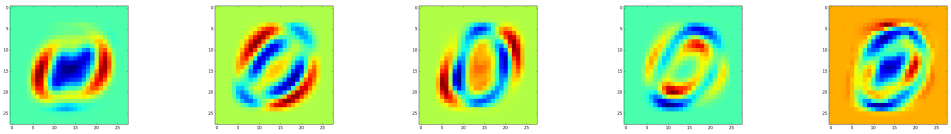


As we have shown in class, the j-th eigenvector, that corresponds with the j-largest eigenvalue, has the j-largest empirical variance obtained by projecting the data on the j-th eigenvector. This means that the first few eigenvectors capture most of the empirical variance of the data, which is demonstrated by their images.

(b) The plot of the mean image (of the unscaled data) is,

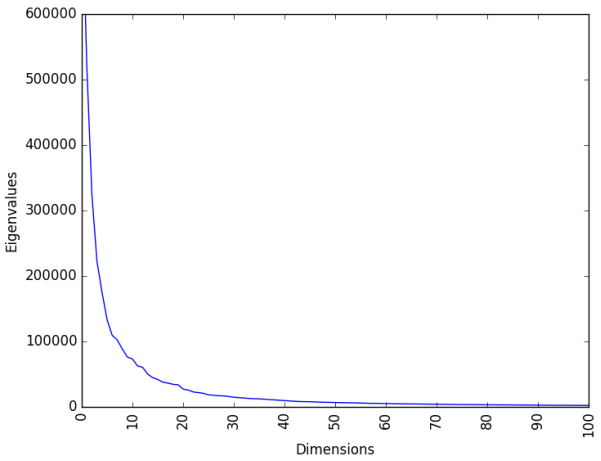


The following plots are the first 5 eigenvectors,

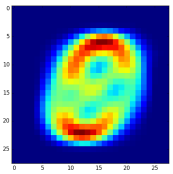


1 2 3 4 5

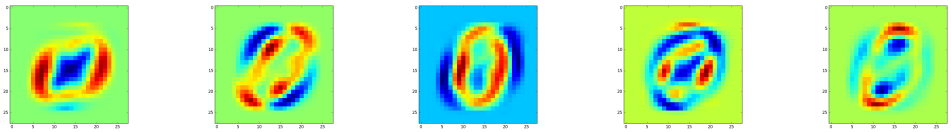
The following is a plot of the eigenvalues as a function of dimension,



(c) The plot of the mean image (of the unscaled data) is,

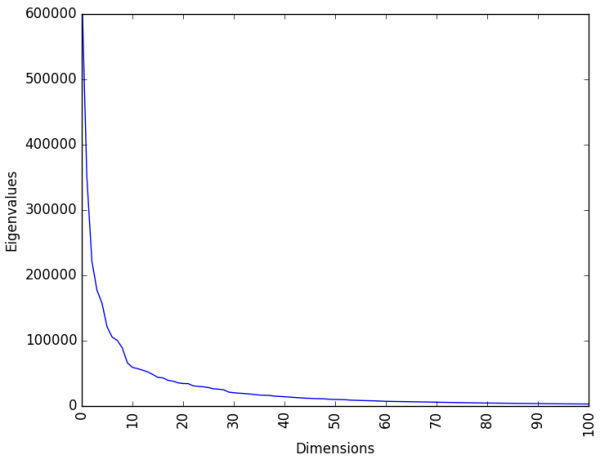


The following plots are the first 5 eigenvectors,



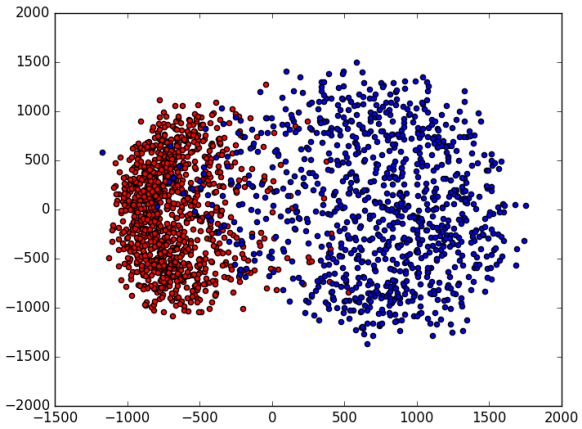
1 2 3 4 5

The following is a plot of the eigenvalues as a function of dimension,



There is a small difference in the magnitude of the eigenvalues compared to the previous PCA analysis we performed. We can see that the 0 digit had larger empirical variance in the data, which corresponds to larger eigenvalues than those of the digit 8. Hence, the eigenvalues obtained from the joint dataset are smaller than those obtained by the 0 digit data, though they are larger than those obtained by the 8 digit data.

(d) The following 2D scatterplot shows the projections of the images on the first two principal axes,

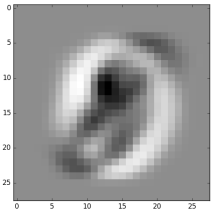


It can be clearly seen in the scatterplot that the red points which correspond to 8-digit images are grouped in a cluster that is almost separate from the blue points which corresponds to the 0-digit images that form another cluster.

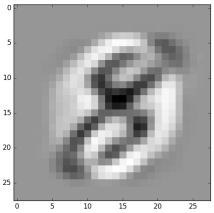
We deduce that the first two principal axes might be enough to distinguish between the digits, meaning that we could have applied a dimensionality reduction to 2 dimensions for this classification task (depending on the required accuracy).

The blue cluster is less dense then the red cluster which indicates that the first two principal components are able to capture more variance within 0-digit images rather than 8-digit images.

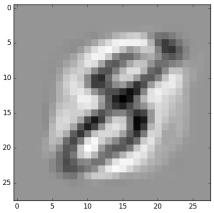
(e) The images in the following page are 2 specific positive images from the data followed by 2 specific negative images, all reconstructed from their projections on the first k principal axes, using $k = 10, 30, 50$. The images demonstrate that the reconstructed images are more clear and resembles an actual digit when using more principal axes, i.e. lower reconstruction error.



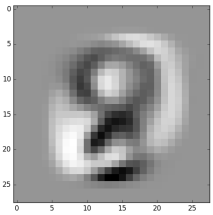
k=10



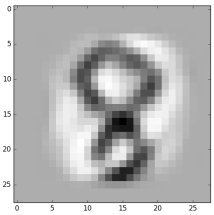
k=30



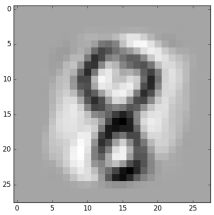
k=50



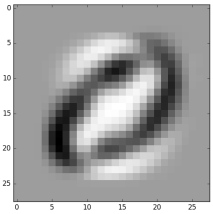
k=10



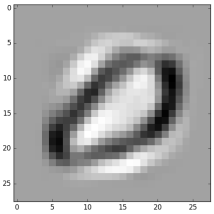
k=30



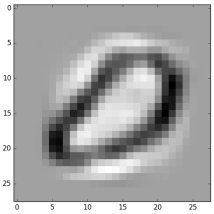
k=50



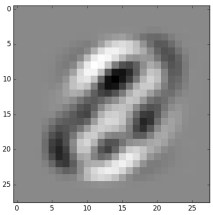
k=10



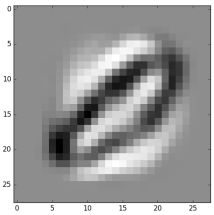
k=30



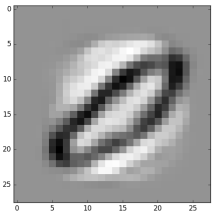
k=50



k=10



k=30



k=50