Introduction to Machine Learning **Fall Semester**

Homework 4: Jan 8th, 2017

Due: Jan 22th, 2017 (See the submission guidelines in the course web site)

Theory Questions

1. **Majority of hypotheses.** Let $h_1, \ldots, h_{2k+1} \in \mathcal{H}$ be hypotheses, and let h be their majority: $h(\mathbf{x}) = majority(h_1(\mathbf{x}), \ldots, h_{2k+1}(\mathbf{x}))$.
   (a) Show that taking a majority may result in worse error. For $2k+1 = 3$, show an example of 3 hypotheses where $error(h_i) = 2\epsilon$ for each i, but $error(h) = 3\epsilon$.
   (b) Show that we can still bound the deterioration in accuracy. Assume that $error(h_i) \leq \epsilon$, for $i = 1, \ldots, 2k+1$. Show that $error(h) \leq 2\epsilon$.
2. **AdaBoost.** Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^d$ and $y_1, \ldots, y_m \in \{-1, 1\}$ its labels. We run the AdaBoost algorithm, and we are in iteration t. Assume that $\epsilon_t > 0$.
   (a) (Do not submit.) Recall that $\epsilon_t e^{\alpha_t} = \sqrt{\epsilon_t(1-\epsilon_t)} = (1-\epsilon_t)e^{-\alpha_t}$. Recall also that the normalization factor is $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.
   (b) Show that the error of the current hypothesis relative to the new hypothesis is exactly 1/2, that is:
   $$\Pr_{\mathbf{x}\sim D_{t+1}}[h_t(\mathbf{x}) \neq y] = \frac{1}{2}$$
   (c) Show that AdaBoost will not pick the same hypothesis twice consecutively; that is $h_{t+1} \neq h_t$.
3. **Kernel PCA.** In the PCA algorithm, we are given a sample $\mathbf{x}_i \in \mathbb{R}^d$, for $i = 1, \ldots, m$. We would like to extend it to Kernel PCA, as follows. We are given a mapping function $\phi : \mathbb{R}^d \to H$, where H is a space to which points are mapped. We would like to perform PCA on the mapped points, $\phi(\mathbf{x}_i)$.

   Recall that in PCA, we require the sample to be mean-centered,
   $$\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i = 0$$
   using a preprocessing stage in which we subtract the mean of each coordinate from it. Since now we operate in H, we will require instead that
   $$\frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i) = 0.$$
   But now, note that we cannot explicitly subtract in H.
   (a) In the Kernel PCA algorithm, we will use the matrix $\bar{K}$ defined as
   $$\bar{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j).$$

where as usual, $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$. Denote $\mathbf{y}_1, \ldots, \mathbf{y}_m \in H$ the mean-centered sample in H, that is:

$$y_i = \phi(\mathbf{x}_i) - \frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{x}_i).$$

Note that we cannot compute this directly, but it is well defined mathematically. Define $\bar{K}'$ as the matrix

$$\bar{K}'_{i,j} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle .$$

That is, the element $\bar{K}'_{i,j}$ is be equal to the application of the original kernel on the i, j-th points, after the sample has been mean-centered in the space H. Show how $\bar{K}'$ can be calculated - of course, you cannot use $\phi(\mathbf{x}_i)$ directly. What is the complexity of this action?

(b) Assume that the sample was appropriately centered, e.g. with the procedure you described in (a). We would like to apply PCA on the mapping $\phi(\mathbf{x}_i)$. Denote by $\mathbf{u}_1, \ldots, \mathbf{u}_k$ the first k principal components in H, corresponding to the sample $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)$. Show that $\mathbf{u}_j$ (for $j = 1, \ldots, k$) is a linear combination of $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)$. How can its coefficients be calculated?

(c) Since H can be infinite-dimensional or with a high dimension, we will not look for the principal components themselves, but instead will be satisfied with the ability to perform a dot product of each principal component with the mapping $\phi(\mathbf{x})$ of a new point, $\mathbf{x}$. More explicitly, let $\mathbf{x}$ be a new point. Show how we can calculate

$$\langle \mathbf{u}_j, \phi(\mathbf{x}) \rangle$$

for $j = 1, \ldots, k$. What is the complexity of the solution?

4. **Linear regression with dependent variables.** Recall that in linear regression, we are interested in solving the following problem:

$$\arg\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|^2$$

where X is a $m \times d$ matrix of the variables, $\mathbf{y}$ is a column vector of size m, and $\mathbf{w}$ is a column vector of size d of coefficients. If there are dependent variables, i.e., the columns of X are linearly dependent, there are infinite possible solutions that achieve this minimum (convince yourself why). One sensible criterion to choose one among all possible solutions, is to prefer a solution with a minimal ℓ_2 norm. That is, we search for $\mathbf{w}$ that optimizes the following:

$$\begin{aligned} \arg\min_{\mathbf{w}} \quad & \|\mathbf{w}\| \\ \text{s.t.} \quad & X^T X \mathbf{w} = X^T \mathbf{y} \end{aligned}$$

Given the SVD of $X = U\Sigma V^T$, solve this optimization problem.

Programming Assignment

In this exercise, we will study the performance of the AdaBoost and PCA on the MNIST dataset, with which you are by now *extremely* familiar. We divide the data into *training set* and *test set*. Under

`~schweiger/courses/ML2016-7/hw4.py`

(in the file system accessible from `nova`) you will find the code to load the training and test sets. It is recommended to use `numpy` and `scipy` where possible. We will investigate how well we can classify a digit to 0 or 8. We will mean-center each coordinate of our data points (this is done in the script above).

5. **AdaBoost.** Implement the AdaBoost algorithm. The class of weak learners we will use is the class of hypothesis of the form

$$h(\mathbf{x}) = \begin{cases} 1 & x_{i,j} \leq \theta \\ -1 & x_{i,j} > \theta \end{cases}, \quad h(\mathbf{x}) = \begin{cases} -1 & x_{i,j} \leq \theta \\ 1 & x_{i,j} > \theta \end{cases}$$

That is, comparing a single pixel to a threshold. At each iteration, AdaBoost will select the best weak learner. Note that the labels are $\{-1, 1\}$.

   (a) Run AdaBoost for $T \geq 50$ iterations. Show plots for the training error and the test error of the classifier implied at each iteration t, $sign(\sum_{j=1}^{t} \alpha_j h_j(\mathbf{x}))$. Explain the behaviour of the errors.

   (b) We have seen in the recitation that AdaBoost minimizes the average exponential loss

$$\ell = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{T} \alpha_j h_j(\mathbf{x}_i)}.$$

   Show plots of ℓ as a function of T, for the training and the test sets. Explain the behaviour of the errors.

6. **PCA.** Principal component analysis (PCA) provides a way of creating an optimal low-dimensional representation of a dataset. Write a function to perform PCA on a set of images. Input the dimension k of the PCA, and output the set of eigenvectors and their corresponding eigenvalues. You are not allowed to use Python's PCA-related functions, but can use eigendecomposition or SVD functions. Do all PCA analysis on the training set.

   (a) Perform PCA on the training set corresponding to images of the digit 8 (positive label). Plot the mean image and then the first 5 eigenvectors (as images). Plot the eigenvalues (in decreasing order) as a function of dimension (for the first 100 dimensions). Can you give an interpretation for some of the first few eigenvectors?

   (b) Perform the same analysis for images of the digit 0 (negative label).

   (c) Now, perform the same analysis, for images of the digits 0 and 8 jointly. Compare and contrast what you find in these plots to the ones you created in (a) and (b). Is there a difference in the magnitude of the eigenvalues, and why?

   (d) On the full dataset, show a 2d scatterplot showing the projections of the images on the first two principal axes. That is, each image is a point; the x axis of the points is its projection on the first principal axis; the y axis is the projection on the second principal axis. Color positive (8) and negative (0) samples in red and blue. Explain the result you see.

   (e) Select 4 images (2 positive and 2 negative), or more if you like. Reconstruct each image as the sum of its projections on the first k principal axes, using $k = 10, 30, 50$. Discuss the results.