Introduction to learning and analysis of big data
Exercise 1

Dr. Sivan Sabato

Fall 2017/8

Submission guidelines, **please read and follow carefully**:

- You may submit the exercise in pairs.
- Submit using the submission system.
- The submission should be a zip file named “ex1.zip”.
- The zip file should include **exactly two files in the root - no subdirectories please**.
- The files in the zip file should be:
  1. A file called “answers.pdf” - The answers to the questions, including the graphs.
  2. A file called “perceptron.m” - The Matlab code for the requested function. Note that you can put several auxiliary functions in this file after the definition of the main function. **Make sure that the single file works in Matlab/Octave before you submit it.**

  Anywhere in the exercise where Matlab is mentioned, you can use the free software Octave instead.
- Getting started with Matlab: You can use the guide in this link: `http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf?s_tid=int_tut`. There are plenty of other tutorials, documentation and examples on the web. You can run Matlab on the lab computers. Octave is free to download and use.
- For questions use the course Forum, or if they are not of public interest, send them via the course requests system.
- Grading: Q.1 (`perceptron.m` file): 20 points, Q.2: 27 points, Q.3: 30 points, Q.4: 23 points.

Question 1. Implement a function that runs the **Perceptron** algorithm that we saw in class on a given training sample, and uses it to predict the labels of examples. The function should be implemented in the submitted file called “perceptron.m”. The first line in the file (the signature of the function) should be:

```
function w = perceptron(m, d, Xtrain, Ytrain, maxupdates)
```

The input parameters are:

- `m` - the size of the training sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, an integer $m \geq 1$.
- `d` - the number of features in each example in the training sample, an integer $d \geq 1$. So $\mathcal{X} = \mathbb{R}^d$.
- `Xtrain` - a 2-D matrix of size $m \times d$. Row i in this matrix is a vector with d coordinates that describes example x_i from the training sample.
- `Ytrain` - a column vector of length m (that is, a matrix of size $m \times 1$). The i's number in this vector is the label y_i from the training sample, each y_i is either 1 or -1.
- `maxupdates` - this value limits the maximal number of updates to the vector `w` that the Perceptron algorithm should perform before terminating. If after `maxupdates` updates the stopping condition of the Perceptron as presented in class was not fulfilled yet, your code should stop anyway and return its current w. Of course, it should stop earlier if the stopping condition is fulfilled before that.

The function returns the $d \times 1$ column vector `w`. This is the final value of `w` reached by the algorithm. You may assume that all the input parameters are legal.

Question 2. You will test your Perceptron implementation on the **hand-written digits recognition** learning problem. In this problem, the examples are images of hand-written digits, and the labels indicate which digit is written in the image. The full data set of images, called MNIST, is free on the web. See at the end of the exercise its description and how to work with it.

Answer the following questions in the file "answers.pdf".

(a) Repeat the following runs for two learning tasks: distinguishing between the digits 3 and 5 and distinguishing between the digits 5 and 6. Run your `perceptron` implementation on several MNIST training sample sizes between 1 and 1000 (run on enough values so that your plots are informative). For each sample size, run your implementation, and use the output `w` to calculate the *training error* $\mathrm{err}(h_w, S)$ as well as the *test error* $\mathrm{err}(h_w, T)$, where T is the test sample (see the MNIST explanation on how to get S and T). When you run your algorithm, use `maxupdates` to make sure it always terminates, since the sample might not be separable.

Submit a *single* plot that shows the training error and the test error as a function of the training sample size, for both the $3/5$ and the $5/6$ learning task on the same plot. Don't forget to label the axes and the plot lines. You can use Matlab's `plot` command (or any other plotting software).

(b) Describe the trend in the graph: what is the trend for the training error? what is the trend for the test error? what is the trend for the *difference* between the train and test errors?

(c) Based on what we learned in class, and using the notions of *approximation error*, *estimation error*, and *overfitting*, explain (in words) the reasons for the trends observed in the graph and the differences between the train and test errors as a function of the sample size, and the differences between the results for each of the training tasks ($3/5$ and $5/6$).

Question 3. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a finite set of examples, and let $\mathcal{Y} = \{0, 1\}$. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Suppose that $\mathcal{D}$ has a Bayes-error of zero and that it is c-Lipschitz.

(a) Let $S \sim \mathcal{D}^m$. Prove that for any two pairs $(x_1, y_1), (x_2, y_2) \in S$, if $y_1 \neq y_2$ then $\|x_1 - x_2\| \geq 1/c$.

(b) Let $\mathcal{B}$ be a set of balls of radius $1/(3c)$ that *cover* the space of points $\mathcal{X}$ (so that every point from $\mathcal{X}$ is in at least one ball in $\mathcal{B}$). Let $S \sim \mathcal{D}^m$. Suppose that for every ball $B \in \mathcal{B}$, there is some pair $(x, y) \in S$ which satisfies $x \in B$. Prove that under this assumption, and the other assumptions on $\mathcal{D}$ given above, $\mathrm{err}(f_S^{nn}, \mathcal{D}) = 0$, where f_S^{nn} is the 1-nearest-neighbor function defined in class.

Question 4. Let $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{1, 2, 3, 4\}$. Consider the following hypothesis class:

$$\mathcal{H} := \{h_{w_1,w_2} \mid w_1, w_2 \in \mathbb{R}^d\},$$

where

$$h_{w_1,w_2}(x) := \begin{cases} 1 & \text{sign}(\langle x, w_1 \rangle) = 1 \text{ and sign}(\langle x, w_2 \rangle) = 1 \\ 2 & \text{sign}(\langle x, w_1 \rangle) = 1 \text{ and sign}(\langle x, w_2 \rangle) = -1 \\ 3 & \text{sign}(\langle x, w_1 \rangle) = -1 \text{ and sign}(\langle x, w_2 \rangle) = 1 \\ 4 & \text{sign}(\langle x, w_1 \rangle) = -1 \text{ and sign}(\langle x, w_2 \rangle) = -1 \end{cases}$$

Let $S = (x_1, y_1), \ldots, (x_m, y_m)$ be a sample with pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Give a pseudo-code for an algorithm that gets S as input, and outputs w_1, w_2 such that $\mathrm{err}(h_{w_1,w_2}, S) = 0$, if such w_1, w_2 exist. If they do not exist, the algorithm outputs "no solution". The algorithm should use as a black-box a Linear Program solver of the following form: `solution` $=$ `linprogsolve`(u, v, A), where u, v, A are as defined for linear programs in class. `solution` is a solution to the linear program if one exists, and "null" if one does not. Prove the correctness of the algorithm.

The MNIST dataset

The MNIST dataset includes 70,000 images with the digits 0-9.
Each image in MNIST has 28 by 28 pixels, and each pixel has a value, indicating how dark it is. Each example is described by a vector listing the $28 \cdot 28 = 784$ pixel values, so we have $\mathcal{X} = \mathbb{R}^{784}$, every example is described by a 748-coordinate vector.

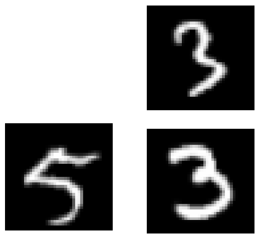

Figure 1: Some examples of images of digits from the data set MNIST

The images in MNIST are split to training images and test images. The test images are used to estimate the success of the learning algorithm: In addition to the training sample S as we saw in class, we have an additional **test sample**, T which also consists of pairs of labeled examples.

The learning algorithm gets S, decides on $\hat{h}_S$, and then predicts the labels of the test images in T using $\hat{h}_S$. The error of the prediction rule $\hat{h}_S$ on the test images is $\mathrm{err}(\hat{h}_S, T) = \frac{1}{m}\sum_{(x,y)\in T}\mathbb{I}[\hat{h}_S(x) \neq y]$. It is a good estimate of the error of $\hat{h}_S$ on the distribution $\mathrm{err}(\hat{h}_S, \mathcal{D})$.
You will need the following files, which can be found in the course website `http://www.cs.bgu.ac.il/~inabd181/Assignments`: `mnist_all.mat` and `gensmallm.m`
To load all the MNIST data to Matlab, run the following command (if the file `mnist_all.mat` is not in your Matlab path or current directory, add the file path to the name of the file):

```
>> load('mnist_all.mat');
```

After this command you will have the variables:
`train0,train1,...,train9,test0,test1,...,test9` in your Matlab workspace.
For this exercise, we will use only two digits at a time, so that the Perceptron algorithm tries to learn to distinguish between them. We will set the label of one digit to -1 and the other to 1.
To generate a training sample of size `m` with images only of two of the digits, you can use the provided function `gensmallm`, which is used as follows:

```
>> [Xtrain,Ytrain,Xtest,Ytest] = gensmallm(trainA, trainB, testA, testB, m);
```

The function `gensmallm` selects random subsets of each training matrix from the training data and mixes them together in a random order, to generate the training sample `Xtrain,Ytrain`. It also mixes the test images together and generates the test sample `Xtest,Ytest`. It labels the examples in `trainA` as -1 and those in `trainB` as 1, and similarly for the test set.