Exercise 2

Prof. Kontorovich and Dr. Sabato

Fall 2016/7

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 "ex2.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 "softsvm.m" - The Matlab/Octave 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 Octave instead**.
- Question 1: 20 points; Question 2 (the implementation): 30 points; Question 3: 25 points. Question 4: 25 points.
- For questions use the course Forum, or email `inabd17@gmail.com`.

Question 1. For a given training sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, consider the following **modified version** of the soft-SVM optimization problem:

$$\lambda \|w\|^2 + \sum_{i=1}^{m} [\ell^h(w, (x_i, y_i))]^2,$$

where $\ell^h(w, (x_i, y_i)) = \max\{0, 1 - y_i \langle w, x_i \rangle\}$ is the *hinge loss* defined in the lecture.

Express this optimization problem as a quadratic program in standard form, of the type MATLAB can solve, using the following steps:

(a) Write a quadratic minimization problem with constraints that is equivalent to the problem above, using auxiliary variables similar to the ξ_i in the soft-SVM implementation.

(b) Write what H,u,A,v in the definition of a Quadratic Program should be set to so as to solve the minimization problem you wrote.

Question 2. Implement the soft-SVM algorithm that we learned in class in MATLAB. Submit the solution as a file called "softsvm.m". The first line in the file (the signature of the function) should be:

```
function w = softsvm(lambda, m, d, Xtrain, Ytrain)
```

The input parameters are:

- `lambda` - the parameter λ of the soft SVM algorithm.
- 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, and integer $d \geq 1$. So $\mathcal{X} = \mathbb{R}^d$.
- `Xtrain` - a 2-D matrix of size $m \times d$ (note: first number is the number of rows, second is the number of columns). 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. You can assume that each label is either -1 or 1. **Important: The labels in the input to soft SVM should be -1 or 1, and not 0 or 1 (just like in Exercise 1).**

The function returns the variable `w`. This is the linear predictor $w \in \mathbb{R}^d$, a column vector of length `d`.

- You may assume all the input parameters are legal.
- Use Matlab's `quadprog` function to implement the algorithm. Don't forget to look it up to make sure you are using all the parameters correctly.
- To save memory, it is recommended to use a sparse matrix representation. Look up the Matlab functions `sparse, full, eye, speye`.

Question 3. Test your soft SVM implementation on the **hand-written digits recognition** learning problem MNIST. Use the `mnist_all.mat` data file provided in Exercise 1 under `http://www.cs.bgu.ac.il/~inabd171/Assignments`, and generate random samples from it using the function "gensmallm" as in Exercise 1. **Important: The labels in the MNIST data are 0 or 1. The soft SVM function requires the labels to be -1 or 1. You should convert the MNIST labels before using the soft SVM function.**

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

(a) Run the soft SVM implementation on a sample of size 50 with examples only of digits 3 and 5. Generate a sample of this size using `gensmallm`, and then run soft SVM on this sample with the following values of λ: $\lambda = 10^n$, for $n \in \{-10, -9, \ldots, 0, 1, \ldots, 8\}$. For each choice λ, calculate the error on the test sample. For this you need to first apply the linear predictor to the test sample, and then compare the resulting labels to the true test labels.

Submit a plot of the test error as a function of lambda (plot the λ on a log scale!).

(b) Repeat the experiment in (a), this time with a sample size of 1000. Submit a new plot, in which the test error as a function of λ is given for the two different sample sizes (a line for each sample size).

(c) Compare the two lines you got: what is similar and what is different, in terms of the form, trend and height of the line? Explain the findings in words, based on what we learned in class.

(d) Take one of the classifiers that got a small test error out of the tests you ran for the plots above, and draw it as an image with 28×28 pixels: each pixel should represent the value of the coordinate of w that matches this pixel. For instance, positive values can be blue (darker blue for a larger value), and negative values can be orange (darker orange is a negative value with a larger absolute value). Useful matlab functions for this task are `reshape` and `HeatMap`. Provide the image in the answer file.

(e) Explain why the image of w looks the way it does.

Question 4. In this exercise, we will see that without margin assumptions, the Perceptron algorithm might run for a long time, exponential in the dimension d. We define a special sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, where $m = d$, $x_i \in \mathbb{R}^d$, and $y_i \in \{-1, 1\}$, where $y_i := (-1)^{i+1}$, and $x_i \in \mathbb{R}^d$ is defined by:

$$x_i(j) = \begin{cases} (-1)^i, & j \leq i \\ (-1)^{i+1}, & j = i \\ 0, & j > i. \end{cases}$$

In the following steps, you will prove that for the sample above, the Perceptron performs a number of updates at least exponential in d:

(a) Prove that in any iteration t of the Perceptron on the given sample S, and for any $i \leq d$, $|w^{(t+1)}(i)| \leq t$.
Hint: use induction on the Perceptron update $w^{(t+1)} \leftarrow w^{(t)} + y_i x_i$.

(b) Let $w^{(T)}$ be the separator that the Perceptron outputs. Prove that for every coordinate i, $|w^{(T)}(i)| \geq 2^{i-1}$. Hint: prove this by induction on i, using the fact that the separator defined by $w^{(T)}$ labels correctly all the examples in the given sample S.

(c) Conclude from the two previous items that the number of updates of the Perceptron until it stops and outputs $w^{(T)}$ is exponential in d.

To help with intuition, you may want to implement the Perceptron algorithm defined in class (no need to submit the code), and check its behavior on the sample S for $d = 1, 2, \ldots, 10$.