Introduction to learning and analysis of big data
Exercise 2

Dr. Sivan Sabato

Fall 2020/1

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

- The exercise is submitted in pairs.
- Submit using the submission system.
- The submission should be a zip file named "ex1.zip".
- The zip file should include **only the following files in the root - no subdirectories please**.
  1. A file called "answers.pdf" - The answers to the questions, including the graphs.
  2. Matlab files (with extension ".m") - The Matlab code for the requested functions questions below.

  Anywhere in the exercise where Matlab is mentioned, you can use the free software Octave instead.
- For questions use the course Forum, or if they are not of public interest, send them via the course requests system.
- This exercise is worth 50% of the course's overall exercise grade.
- Grading: Q1: 10, Q2: 10, Q3:10, Q4:15, Q5:12, Q6: 12, Q7:8, Q8: 15, Q9: 8.

Question 1. 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, Xtrain, Ytrain)
```

The input parameters are:

- `lambda` - the parameter λ of the soft SVM algorithm.
- `Xtrain` - a 2-D matrix of size $m \times d$, where m is the sample size and d is the dimension of the examples. 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. The i's number in this vector is the label $y_i \in \{-1, 1\}$ from the training sample.

The function returns the linear predictor `w` which is a column vector in $\mathbb{R}^d$.

- You may assume all the input parameters are legal.
- Use Matlab's `quadprog` function to implement the algorithm. Look it up to make sure you are using all the parameters correctly.

Question 2. In this question you will run your soft SVM implementation on data from the image data set CIFAR-10 (You can see some image examples here: `http://www.cs.toronto.edu/~kriz/cifar.html`). For this task, we took a subset of this data set that includes only images of cars and trucks, and the goal of the predictor is to distinguish between these two types of vehicles. Each example represents an image of size 32×32. You can load the data set, which is already divided to train and test, from the file `Ex2q2_data.mat` on the course website.

Run two experiments on this data set. In the first experiment, use a sample size of 100. To generate this small sample, draw it randomly from the provided training sample. Repeat the "small sample" experiment 10 times, and when you report the results, average over these 10 experiments, and plot also error bars which show the maximum and minimum values you got over all experiments. Run your soft-SVM implementation with each of the following values of λ: $\lambda = 10^n$, for $n \in \{5, \ldots, 15\}$.

In the second experiment, use a sample size of 1000, which you should also draw randomly from the training set. Run your soft-SVM implementation with each of the following values of λ: $\lambda = 10^n$, for $n \in \{5, 10, 15\}$. To make the running time feasible, you are **not required** to repeat this experiment 10 times and only required to run with 3 values of λ.

(a) Submit a plot of the training error and test error of the small sample size results as a function of λ (plot λ on a logarithmic scale), with a line for the train and test errors. Each line should show an average of the 10 experiments, and error bars which show the maximum and minimum values you got over all experiments. Add to the plot the points describing the training error and test error of the large sample size, but don't draw lines between the points in this case: Only show each point individually, since you only tested three values of λ.

(b) Based on what we learned in class, what would you expect the results to look like? Do the results you got match your expectations? In your answer address the following issues:

- Which sample size should get a smaller training error? What about test error? Do the results match your expectations?
- What should be the trend in the training error as a function of λ (decreasing/increasing/something else)? Why? Do the results (for the small sample size) match your expectations?
- What should be the trend in the test error as a function of λ (decreasing/increasing/something else)? Why? Do the results (for the small sample size) match your expectations?

Question 3. Implement the soft-margin kernel SVM routine described in class, using MATLAB's `quadprog` command. The algorithm should use the Gaussian (RBF) kernel. The function should be implemented in the submitted file called "softsvmrbf.m". The first line in the file (the signature of the function) should be:

```
function alpha = softsvmrbf(lambda, sigma, Xtrain, Ytrain)
```

The input parameters are:

- `lambda` - the parameter λ of the soft SVM algorithm.

- `sigma` - the bandwidth parameter σ of the RBF kernel.
- `Xtrain` - a 2-D matrix of size $m \times d$, where m is the sample size and d is the dimension of the examples. Row i gives example $x_i \in \mathbb{R}^d$ from the training sample.
- `Ytrain` - a column vector of length m . The i's number in this vector is the label $y_i \in \{-1, 1\}$ from the training sample.

The function returns the column vector `alpha` $\in \mathbb{R}^m$ which describes the coefficients found by the algorithm.

Question 4. For this question, use the data file `ex2data.mat` provided on the course website, which contains data points in the domain $\mathcal{X} = \mathbb{R}^2$ and labels in $\{-1, 1\}$, split into a training set and test set.

(a) We would like to use soft SVM to learn a predictor for this problem. Plot the points in the training set in $\mathbb{R}^2$, and color them by their label. Explain why it may be a better idea use kernel soft SVM and not linear soft SVM.

(b) Run your RBF soft SVM code on the training set. Perform 5-fold cross-validation to tune λ and σ. Try the values $\lambda \in \{1, 10, 100\}$ and $\sigma \in \{0.01, 0.5, 1\}$ — a total of 9 parameter pairs to try. Report the 9 average validation error values for each of the pairs (λ, σ). Report which pair was selected by the cross validation, rerun the training using this pair on the entire training set, and report the test error of the resulting classifier.

Repeat the procedure above using the linear (non-kernel) soft SVM code on the given training set. In this case, you only need to choose λ since there is no σ parameter.

(c) Which approach (RBF or soft SVM) achieved a better validation error? Is it what you expected? Give one reason why, in general, the RBF SVM might give a better validation error, and one reason why it might give a worse validation error, compared to the linear soft SVM approach.

(d) Set $\lambda = 100$ and consider $\sigma \in \{0.01, 0.5, 1\}$. For these values, run the soft-margin RBF SVM on the training set, and plot the resulting predictor in $\mathbb{R}^2$ as follows: Define a fixed region (roughly the region in which the training data resides), divide it into a fine grid, and color the grid points red or blue, depending on the label predicted by the classifier for each point. You can use the `heatmap` routine to plot this. On each plot, show also the training set points, colored by their label. Submit the three plots.

(e) Explain the difference between the three plots submitted above. Base your answer on what we learned in class regarding the effect of σ on the formula of the separator generated by an RBF kernel.

Question 5. **Kernel functions.** Consider a space of examples $\mathcal{X} = \mathbb{R}^d$. Let $x, x' \in \mathcal{X}$.

(a) Prove that there **does not exist** a feature mapping ψ such that the following function is its kernel function:

$$K(x, x') := -x(1)x'(1).$$

Hint: consider the case of $x = x'$.

(b) Prove that there **does not exist** a feature mapping ψ such that the following function is its kernel function:

$$K(x, x') := x(2)x'(1).$$

(c) Consider the following optimization objective:

$$\text{Minimize}_{w\in\mathbb{R}^d}\lambda\|w\|^4+\sum_{i=1}^{m}\exp^{-y_i|\langle w,x_i\rangle|}$$

Use the representer theorem that we showed in class to prove that there exists a solution $w\in\mathbb{R}^d$ to this minimization problem which can be formulated as a linear combination of $x_1,\ldots,x_m$.

(d) Prove that the following function is a kernel function, by showing a ψ that matches it:

$$K(x,x')=x(1)x'(1)+x(2)x'(1)+x(1)x'(2)+x(2)x'(2).$$

Question 6. Let $\mathcal{X}\in\mathbb{R}$, $\mathcal{Y}\in\{0,1\}$. Given a text string c which describes a mathematical condition, define $h_c:\mathcal{X}\to\mathcal{Y}$ as follows: $h_c(x):=\mathbb{I}[x \text{ satisfies } c]$. For instance, if c is the condition "x is a natural number", then $h_c(x)=\mathbb{I}[x\in\mathbb{N}]$. For any $n\in\mathbb{N}$, define the hypothesis class $\mathcal{H}_n\subseteq\mathcal{Y}^{\mathcal{X}}$ as follows:

$$\mathcal{H}_n:=\{h_c\mid c \text{ is a condition which can be described using at most } n \text{ characters}\}$$

For instance, the condition c given above is in $\mathcal{H}_n$ for all $n\geq 21$. The text of the conditions can include only ASCII characters (there are 128 ASCII characters).

Suppose that the examples x are temperature values, and the labels y indicate whether a doctor gives medicine to a person with this temperature. Suppose that we know that the rule that the doctors use to decide whether to give the medicine has at most 10 characters and that the doctors follow this rule exactly. We get a random independent sample from the distribution $\mathcal{D}$ over $\mathcal{X}\times\mathcal{Y}$ over temperatures and doctor decisions. We would like to run an ERM algorithm with the hypothesis class $\mathcal{H}_{10}$ to find a condition with an error of at most 0.1. We have unlimited computational resources.

(a) Explain why running the ERM algorithm with the hypothesis class $\mathcal{H}_n$ for $n<10$ or for $n>10$ may be a worse idea than running it with $\mathcal{H}_{10}$.

(b) Use PAC bounds to calculate an upper bound on the size of a sample size that would be sufficient to find, with a probability at least 0.99, a condition with an error at most 0.1 on the distribution, using an ERM algorithm $\mathcal{H}_{10}$. Give a formal description of the guarantee that you are providing.

(c) Suppose we run ERM with $\mathcal{H}_n$ on the training sample for some $n<10$. Calculate an upper bound on the size of a sample size that would be sufficient to find, with a probability at least 0.99, a condition with an *excess* error at most 0.1 on the distribution, using an ERM algorithm with $\mathcal{H}_n$. Give a formal description of the guarantee that you are providing.

Question 7. 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 class.

Express this optimization problem as a quadratic program in standard form, by providing H, u, A and v. Hint: Use auxiliary variables. Prove that the solution to your program is also a solution to the optimization problem above.

Question 8. 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:

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

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 (at least) exponential in d.

Question 9. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. Consider a Gradient Descent algorithm that attempts to minimize the following objective:

$$\text{Minimize}_{w \in \mathbb{R}^d} \lambda \|w\| + \sum_{i=1}^{m} (\langle w, x_i \rangle - y_i)^2.$$

(a) Suppose that Gradient Descent is run on S with a step size η. Calculate the formula for $w^{(t+1)}$ as a function of $w^{(t)}$ and η. Explain the steps of your derivation.

(b) Now, suppose we run Stochastic Gradient Descent on the same objective. What is the formula for $w^{(t+1)}$ in this case?

A note on "problem is not convex" from Matlab: Matlab expects the matrix H in a quadratic program to be positive definite, that is: to have only non-negative eigenvalues (though sometimes it can tolerate zero eigenvalues as well).
However, due to numerical inaccuracies in calculations, when some of the eigenvalues are very close to zero (though still positive), Matlab might think they are negative (e.g. a tiny amount smaller than zero). You can check this by running "eig(H)" and see what Matlab thinks are the eigenvalues. If they are very close to zero, either positive or negative, this explains why you are getting this error.
To avoid this issue, if you have it, the solution is to add to the diagonal of the matrix a **small** positive value, let's call it ϵ. If you add epsilon to the diagonal, all the eigenvalues grow by epsilon, so if one of the eigenvalues was too close to zero for Matlab to work with, it will now be a little larger so that Matlab is not confused thinking it's negative. However, adding anything to the matrix might change the result, and if you add a large number to the diagonal it might change the result too much. So the best strategy is to add the smallest value that works, so that on the one hand Matlab doesn't get confused, and on the other hand the results don't change significantly.