Introduction to learning and analysis of big data
Exercise 2

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 "ex2.zip".
- The zip file should include **exactly three 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. Files called "softsvm.m" and "softsvmrbf.m" - The Matlab code for the requested functions. 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.
- For questions use the course Forum, or if they are not of public interest, send them via the course requests system.

Question 1. (15pts) 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 $m \times d$. 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 from the training sample. You can assume that each label in `Ytrain` is either -1 or 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 2. (a) (5 pts) Run your soft SVM implementation on a sample of size 50 with examples only of digits 3 and 5. Try the following values of λ: $\lambda = 10^n$, for $n \in \{-10, -9, \ldots, 0, 1, \ldots, 8\}$. Repeat the experiment with a sample of size 1000. Submit a plot of the test errors as a function of lambda with a line for each sample size. Plot λ on a log scale.

(b) (10pts) 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:

- The optimal value for λ in each of the sample sizes: which should be larger and why?
- The trend in the test error as a function of λ: should it be decreasing/increasing/something else? Why?

Question 3. (15pts) Implement the soft-margin kernel SVM routine described in class, using MATLAB's `quadprog` command. 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, m, d, Xtrain, Ytrain)
```

The input parameters are:

- `lambda` - the parameter λ of the soft SVM algorithm.
- `sigma` - the bandwidth parameter σ of the RBF kernel.
- `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 $m \times d$. 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 from the training sample. You can assume that each label is either -1 or 1.

The function returns the variable `alpha`. This is the vector of coefficients found by the algorithm, $\alpha \in \mathbb{R}^m$, a column vector of length `d`.

Question 4. For this question, use the data file `ex2data.mat`, which contains data points $x_i \in \mathbb{R}^2$ and labels $y_i \in \{-1, 1\}$. There 1000 training examples and 100 test points.

(a) (5 pts) Run your RBF soft SVM code on the given data. Perform 5-fold cross-validation to tune λ and σ. Try the values $\lambda \in \{0.01, 0.1, 1\}$ and $\sigma \in \{0.05, 1, 2\}$ — a total of 9 parameter pairs to try. Report the 9 average validation error values for each of the pairs (λ, σ) as well as the optimal pair (i.e., the one that achieves the lowest validation error) and its performance on the test set.

In addition, run your soft SVM code on the given data. Perform 5-fold cross-validation to tune λ, using the same values as above. Report the average validation error of this approach as well.

(b) (7 pts) 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 lower validation error, compared to the soft SVM approach.

(c) (5 pts) Set $\lambda = 0.01$ and consider $\sigma \in \{0.05, 1, 2\}$. For these values, run the soft-margin RBF SVM and plot the decision boundary 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 where they fall under the classifier's prediction. You can use the `heatmap` routine for visualizing this. Submit the three plots.

(d) (8 pts) Explain the difference between the three plots submitted above, using what we learned in class regarding the formula of the separator generated by an RBF kernel, and the effect of σ.

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

(a) (5 pts) Prove that the following function *cannot be* a kernel function for any feature mapping ψ:

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

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

(b) (5 pts) Prove that the following function *cannot be* a kernel function for any feature mapping ψ:

$$K(x, x') := (x(1) + x(2))(x'(3) + x'(4)).$$

(c) (10 pts) Recall that the Gaussian kernel is

$$K(x, x') := e^{-\frac{\|x-x'\|^2}{2\sigma}},$$

Prove that the Gaussian kernel is the correct kernel for the coordinate mapping which we gave in class: a coordinate is defined for each finite sequence z with values in $\{1, \ldots, d\}$, and the value of coordinate z in the mapping of x is:

$$\psi(x)(z) := \frac{1}{\sqrt{n!}} e^{-\frac{\|x\|^2}{2\sigma}} \cdot \prod_{i=1}^{n} x(z(i))/\sigma.$$

Question 6. (10 pts) **Hoeffding's bound.** Consider a learning algorithm with k numerical parameters. Suppose that we use a validation set of size n to decide which value to use for each of the parameters. For each numerical parameter we have r possible values, and we try each combination of values, by running the learning algorithm with this value combination on the training set, and calculating the validation error of the resulting classifier. Finally, we select the classifier that got the smallest validation error.

Let $\delta, \epsilon \in (0, 1)$. What should the size of the validation set n be, to guarantee that with a probability of at least $1 - \delta$ over the choice of the validation set, the error of the classifier that we selected is at most ϵ more than the error of the best classifier in the set of classifiers that we tested? Your answer should depend on δ, ϵ, k, r. Prove your claim using Hoeffding's bound.