Homework 4
Regularization, Kernel, Perceptron and SVM

CMU 10-601: Machine Learning (Spring 2017)
https://piazza.com/cmu/spring2017/10601
OUT: Feburary 22, 2017
DUE: March 03, 2017 11:59 PM
Authors: Megha Arora, Yongjin Cho, Wei Ma

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the collaboration policy on the website for more information: http://www.cs.cmu.edu/~mgormley/courses/10601-s17/about.html
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/10601-s17/about.html
- **Submitting your work:** Each homework will consist of two parts. For this homework, the first part will be submitted via QnA. The second part of the assignment will be a programming question. For this question, you will submit code via Autolab and answer related questions on QnA as detailed below.
  - **QnA:** We will use an online system called QnA for short answer and multiple choice questions. You can log in with your Andrew ID and password using the button labeled "LOGIN WITH CMU ID". (As a reminder, never enter your Andrew password into any website unless you have first checked that the URL starts with "https://" and the domain name ends in ".cmu.edu" – but in this case it's OK since both conditions are met) A link to the Homework 4 QnA section is provided in Problem 1 below. The deadline displayed on QnA may not correspond to the actual deadline for this homework, since we are allowing late submissions (as discussed in the late submission policy on the course site).
  - **Autolab:** You can access the 10601 course on autolab by going to https://autolab.andrew.cmu.edu/ Using All programming assignments will be graded automatically on Autolab using Octave 3.8.2 and Python 2.7. You may develop your code in your favorite IDE, but please make sure that it runs as expected on Octave 3.8.2 or Python 2.7 before submitting. The code which you write will be executed remotely against a suite of tests, and the results are used to automatically assign you a grade. To make sure your code executes correctly on our servers, you should avoid using libraries which are not present in the basic Octave install. For Python users, you are encouraged to use the `numpy` package. The deadline displayed on Autolab may not correspond to the actual deadline for this homework, since we are allowing late submissions (as discussed in the late submission policy on the course site).

Problem 1: QnA [52 points]

This problem consists of a set of multiple choice and numerical questions posted on the QnA platform at the URL below (also posted in Piazza):

https://qna.cs.cmu.edu/#/pages/view/121

Question 23 to 31 on QnA are based on Problem 2 of this homework and should not be attempted until you have read through Problem 2 below.

Question 32 to 34 on QnA are based on Problem 3 of this homework and should not be attempted until you have read through Problem 3 below.

QnA allows you to submit multiple times for the same question. We will grade the last answer you submit before the homework deadline.

Problem 2: Support Vector Machine (SVM) Experiments [15 points]

For this section, we have provided function `run_svm.py/run_svm.m` both in python and octave, which runs linear SVM classifier and plots the decision boundary. Detailed instructions and documentation for the function is provided in the following link to piazza note : https://piazza.com/class/ixs4v2xr1cz10d?cid=833.

To conduct the experiments, you need to download the handout for Homework 4 from autolab (Options → Download handout) and extract the contents (i.e., by executing `tar xvf hw4handout.tar` at the command line). In the `svm` folder you will find all the codes and data files for the experiments. Make sure to follow the instructions in the given link before conducting experiments.

Note : the SVM objective function for the questions in Problem 2 and in QNA is:

$$\underset{w,\xi_1,\xi_2,\cdots,\xi_m}{\text{argmin}} \left(||w||^2 + C\sum_i \xi_i\right) \text{ such that for all i, } y_i w \cdot x_i \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

where the input is a set of observations $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and ξ_i is slack variable for ith data instance.

Support Vectors and Non-support Vectors : Life is not fair

Based on instructions in https://piazza.com/class/ixs4v2xr1cz10d?cid=833 are done, we will experiment the behavior of SVM (Support Vector Machine) after making changes to the data. In this section, we will be using C (weight of sum of slack variables) value of 4.

1. **[1 point]** Based on the documentation in Appendix section, run SVM classification after loading **X** and **Y** from "hw4data1.mat". Then remove one of the support vectors (20th row of **X** and **Y**) and re-run SVM classifier on the new data. After running SVM on the new data, what changes can be observed? Provide your answer to Question 23 on QnA.
2. **[2 points]** Run SVM after removing 20th row of **X** and **Y** is removed, report the sum of indices of the new support vectors. Provide your answer to the Question 24 on QnA. (Hint : you might find `numpy.delete()` useful)
3. **[1 point]** Remove the first instance (1st row of X and Y), which is not a support vector, from the original data and re-run SVM classifier on that data. What changes can be observed compared to the result of running SVM on the original data? Provide your answer to Question 25 on QnA.

4. **[2 points]** We have experimented how the result of SVM is affected by a removal of a support vector or a non-support vector. What is the reason for similarities/differences between the effects of removing a support vector and of removing a non-support vector? Provide your answer to Question 26 on QnA.

Role of C in SVM

In this section, we will experiment how the value of C (the weight of sum of slack variables in SVM objective function) affects overall behavior of SVM.

5. **[1 point]** Run SVM classifier on $\mathbf{X}$ and $\mathbf{Y}$ loaded from `hw4data2.mat` with three different values of $C = \{10, 20, 50\}$. What changes can be observed as we increase C? Report your answer to the Question 27 on QnA.
6. **[2 points]** What could be the reason for your observation? Report your answer to the Question 28 on QnA.

Slack variables

In this section, you will be asked questions regarding support vectors and their slack variables. Run SVM on `hw4data2.mat` with $C = 10$.

7. **[2 points]** Select the correct statements about the slack variables values for each of the red points in the right subplot? Provide your answer to Question 29 on QNA.
8. **[2 points]** Select the correct statements about the slack variables values for each of the blue points in the right subplot? Provide your answer to Question 30 on QNA.
9. **[2 points]** Select the correct statements about the slack variables values for each of the black points in the right subplot? Provide your answer to Question 31 on QNA.

Problem 3: Implementing Perceptron and Kernel Perceptron [40 points]

For this question, you will implement Perceptron and Kernel Perceptron classifiers. You are given part of the MNIST dataset[1], the dataset contains handwritten digits from 0 to 9. In this question you will implement the perceptron and kernel perceptron algorithm to distinguish between handwritten digits 4 and 7 in part of MNIST dataset. Each row of the data matrix represent a 32×32 figure, the values of features is the intensity of pixels. The data is standardized and added with one biased term of value 1 to allow the decision boundary offset from origin.

We also associate each figure with a label y such that

$$y = \begin{cases} 1 & \text{if the image is digit 4} \\ -1 & \text{if the image is digit 7} \end{cases}$$

Programming Instructions

You will implement some functions for training Perceptron and Kernel Perceptron classifiers for this question. You will submit your code online through the CMU autolab system, which will execute it remotely against a suite of tests. Your grade will be automatically determined from the testing results. Since you get immediate

[1] http://yann.lecun.com/exdb/mnist/

feedback after submitting your code and you are allowed to submit as many different versions as you like (without any penalty), it is easy for you to check your code as you go.

To get started, you can log into the autolab website (https://autolab.andrew.cmu.edu). From there you should see 10-601 in your list of courses. Download the handout for Homework 4 (Options → Download handout) and extract the contents (i.e., by executing `tar xvf hw4handout.tar` at the command line). In the `perceptron_starter_codes` folder you will find three folders. The `data` folder contains the data files for this problem. The `python` folder contains a `perceptron.py` file which contains empty function templates for each of the functions you are asked to implement. Similarly, the `octave` folder contains separate `.m` files for each of the functions that you are asked to implement.

To finish each programming part of this problem, open the `perceptron.py` or the function-specific `.m` template files and complete the function(s) defined inside. When you are ready to submit your solutions, you will create a new tar archive of the files you are submitting the `hw4` directory. Please create the tar archive exactly as detailed below.

If you are submitting Python code:

```
tar cvf hw4_handin.tar perceptron.py
```

If you are submitting Octave code:

```
tar cvf hw4_handin.tar perceptron_predict.m perceptron_train.m RBF_kernel.m
kernel_perceptron_predict.m kernel_perceptron_train.m
```

If you are working in Octave and are missing **any** of the function-specific `.m` files in your tar archive, you will receive zero points.

We have provided all of the data for this assignment as `csv` files in the `data` folder in your handout. You can load the data using the `numpy.genfromtext` function in Python or the `csvread` and `csv2cell` (`io` package) functions in Octave. We have provided a `hw4_script` file for each language to help get you started. You should build on the `hw4_script` to test out the functions we are asking you to implement and turn in, but you will not submit the script itself. After loading the data, you should have 4 variables: `XTrain`, `yTrain`, `XTest` and `yTest`.

- `XTrain` is a $n \times d$ dimensional matrix describing the n figures used for training your (kernel) perceptron. The entry `XTrain(i,j)` is the darkness of the pixel j in i^{th} figure.
- `yTrain` is a $n \times 1$ dimensional vector containing the class labels for the training figures. `yTrain(i)` is 1 if the i^{th} document belongs to digit 4 and -1 if it belongs to digit 7.
- `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.

The script also contains the codes to visualize the image, you can check the actual figure in `ind`th data row by changing the `ind` parameter. You can test your implementations and conduct the experiments by extending the given script. It is also possible to write the script yourself. The script is not required to submit.

Grading process on autolab may take minutes, especially before the homework due. So we recommend you thoroughly test your implementation locally before you submit codes, which will also save your time.

For all the implementations and experiments below, you SHOULD NOT change the order of the data rows in `XTrain` and `XTest` datasets, perceptron algorithm is sensitive to the order the data feeds. Simply iterate the data from first row to the last row, DO NOT permutate the data.

Implementation

Recall the perceptron algorithm we learned in class for $\mathbf{x}_i \in R^d$ and $y_i \in \{-1, 1\}$

- `Initialize the weight vector` $\mathbf{w} \in R^d$

- `Iterate num_epoch times:`
  - `For` $i = 1 \dots n$`:`
    - Compute $\hat{y}_i = \phi(\mathbf{w}^T \mathbf{x}_i)$
    - `If:` $\hat{y}_i \neq y_i$`; Then:` update $\mathbf{w}$

where

$$\varphi(z) = \begin{cases} -1 & \texttt{if } z \leq 0 \\ 1 & \texttt{otherwise} \end{cases}.$$

1. **[4 points]** Complete the function `y = perceptron_predict(w, x)` which computes the $\hat{y}_i$. The function takes the weight vector $\mathbf{w}$ and data features $\mathbf{x}$, the output should be the predicted label for $\mathbf{x}$. (Note when $\varphi(w^T x_i) = 0$, you need to output $\hat{y}_i = -1$)
2. **[6 points]** Complete the function `w = perceptron_train(w0, XTrain, yTrain, num_epoch)` which implements the perceptron training process. The function takes the initial weight vector $\mathbf{w}_0$, training dataset `XTrain, yTrain` and goes through the training data `num_epoch` times, the output should be the trained weight vector.

Now you can start implementing the kernel perceptron algorithm, recall the kernel perceptron algorithm we learned in class for $x_i \in R^d$ and $y_i \in \{-1, 1\}$

- `Initialize the counting vector` $\alpha \in R^n$
- `Iterate num_epoch times:`
  - `For` $i = 1 \dots n$`:`
    - $\hat{y}_i = \varphi(\sum_{r=1}^{n} \alpha_r y_r K_{i,r})$
    - `If:` $\hat{y}_i \neq y_i$`; Then:` $\alpha_i = \alpha_i + 1$

where

$$\varphi(z) = \begin{cases} -1 & \texttt{if } z \leq 0 \\ 1 & \texttt{otherwise} \end{cases}.$$

and

$$K_{i,r} = K(\mathbf{x}_i, \mathbf{x}_r)$$

Note that in the code we use a instead of α.

3. **[6 points]** Complete the function `K = RBF_kernel(X1, X2, sigma)` which computes a kernel matrix of the Radial basis function (RBF) kernel[2].

$$K(X_{1i}, X_{2j}) = \exp\big(-\frac{||X_{1i} - X_{2j}||_2^2}{2\sigma^2}\big) \tag{1}$$

   where X_{1i} is i^{th} row of $X_1 \in R^{n \times d}$ and X_{2j} is j^{th} row of $X_2 \in R^{m \times d}$.

   The parameter `sigma` is the parameter σ in RBF kernel. The function takes two feature matrices $X_1 \in R^{n \times d}$ and $X_2 \in R^{m \times d}$, the output should be $K \in R^{n \times m}$. In the following kernel perceptron implementations, you will use the RBF function as the kernel function.

4. **[6 points]** Complete the function `y = kernel_perceptron_predict(a, XTrain, yTrain, x, sigma)` which computes the $\hat{y}_i$. The function takes `a` as the counting vector, `XTrain, yTrain` as training data, `x` as the data we want to prediction the label, `sigma` is the parameter σ in RBF kernel, the output should be the predicted label for `x`.

[2] https://en.wikipedia.org/wiki/Radial_basis_function_kernel

5. **[4+4 points]** Complete the function `a = kernel_perceptron_train(a0, XTrain, yTrain, num_epoch, sigma)` which implements the kernel perceptron training process. The function takes the initial counting vector `a0`, training dataset `XTrain, yTrain` and goes through the training data `num_epoch` times, `sigma` is the parameter σ in RBF kernel, the output should be the trained counting vector. The score for this question is computed based on both correctness and efficiency of your codes, you may need to optimize the runtime efficiency of your codes to get full credits. (Hint: Try to think of the disadvantage of using `y = kernel_perceptron_predict(a, XTrain, yTrain, x, sigma)` function when `num_epoch` is large; or you can try to optimiza `RBF_kernel` function).

Experiments

In each experiments, you should set the initial weight vector w or counting vector α to be all zeros for each train. You should not use any "warm start" techniques for training each model.

6. **[2 points]** What is the error rate of the trained linear perceptron on the XTest dataset after iterating the XTrain dataset 10 times? Please submit your answer to question 32 on QnA.

   The error rate is evaluated as the following equation:

$$\text{error rate} = \frac{\text{number of misclassfied data points}}{\text{number of total data points}} \tag{2}$$

7. **[6 points]** Conduct six experiments on kernel perceptron, with $\sigma = 0.01, 0.1, 1, 10, 100, 1000$ for the RBF kernel function. Train the kernel perceptron by going through the data twice in each experiment, which σ achieve the lowest test error (error rate on `XTest` dataset)? Please submit your answer to question 33 on QnA.

8. **[2 points]** Why does different σ influence the test error? Please answer question 34 on QnA.