CSCI 5525: Machine Learning (Fall'15)
Homework 2, Due Fri, 10/23/15 11:55 pm

1. **(40 points)** Recall that a function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a valid kernel function if it is symmetric and positive semi-definite function. For the current problem, we assume that the domain $\mathcal{X} = \mathbb{R}$.

   (a) (10 points) Let $K_1, \ldots, K_m$ be a set of valid kernel functions. Show that for any $w_j \geq 0, j = 1, \ldots, m$, the function $K = \sum_{j=1}^{m} w_j K_j$ is a valid kernel function.

   (b) (10 points) Let K_1, K_2 be two valid kernel functions. Show that $K = K_1 \odot K_2$ is a valid kernel function. Here, $\odot$ denotes the Hadamard product so that $K(x_i, x_j) = K_1(x_1, x_j)K_2(x_i, x_j)$ for all x_i, x_j.

   (c) (10 points) Consider the function $K(x, x') = (xx' + 1)^{2015}$ where $x, x' \in \mathbb{R}$. Show that K is a valid kernel function.

   (d) (10 points) Consider the function $K(x, x') = \exp(-(x - x')^2/2)$ where $x, x' \in \mathbb{R}$. Show that K is a valid kernel function.

2. **(30 points)** The goal is to evaluate the performance of an optimization algorithm for SVMs which work on the *dual*. The problem is based on the following paper: "Fast Training of Support vector Machines using Sequential Minimal Optimization," by J. C. Platt. Please read sections 12.2, 12.2.1, 12.2.2 of the paper. The beginning of Section 12.1 (before 12.1.1) can be a good review of material we have seen in class. The idea discussed in the paper is known as SMO (sequential minimal optimization).

   Recall that the dual problem for a non-separable linear SVM is a quadratic program on $\alpha \in \mathbb{R}^n$, where n is the number of data points and α_i is the Lagrange multiplier corresponding to point i. The quadratic program has a box-constraint on each α_i, i.e., $0 \leq \alpha_i \leq C$ and a linear constraint $\sum_{i=1}^{n} y_i \alpha_i = 0$ where $y_i \in \{-1, +1\}$. The dual QP can be written as:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \mathbf{x}_i^T \mathbf{x}_j \alpha_i \alpha_j \\ & 0 \leq \alpha_i \leq C \ , \quad \forall i \\ & \sum_{i=1}^{n} y_i \alpha_i = 0 \ . \end{aligned} \tag{1}$$

   Implement the SMO algorithm and evaluate its performance on the `MNIST-13` dataset. The evaluation will be based on the optimization performance, and not classification accuracy. So, we will use the entire dataset for training (optimization). For the runtime results, run the code 5 times and report the average runtime and standard deviation, to account for any system level fluctuations (file i/o, other processes, etc.).

   You will have to submit (i) **summary of methods and results** report, and (ii) **code** for each algorithm:

(i) **Summary of methods and results:** SMO iteratively updates α by updating two components (α_i, α_j) in each iteration while keeping the remaining $(n-2)$ components fixed. Clearly describe the sub-problem on (α_i, α_j), how the sub-problem can be solved efficiently, and how the sub-problems are selected in each iteration. Report the average runtime over 5 runs on the entire dataset, along with the standard deviation. In a figure, plot the value of dual objective function with increasing number of iterations for each run, i.e., five plots overlayed on the same figure. The dual objective should increase over iterations to convergence.

(ii) **Code:** You will have to submit code for `mysmosvm(filename, numruns)` (main file). This main file has **input**: (1) a filename containing the dataset and (2) the number of runs, and **output**: (1) the average runtime and standard deviations printed to the terminal (stdout). The function *must* take the inputs in this order and display the output via the terminal. The filename will correspond to a plain text file for a dataset, with each line corresponding to a data point: the first entry will be the label and the rest of the entries will be feature values of the data point. The data for the plots should be written in a file `tmp` in the same directory, but (i) we will not be using this data for doing plots, and (ii) you do not have to explain how you did the plots from this data.

You can submit additional files/functions (as needed) which will be used by the main file. Put comments in your code so that one can follow the key parts and steps in your code. Please do not include input file in your submission folder.

Sample input/output in MATLAB:

```
 >>mysmosvm("D:\MNIST-13", 5)
Avg runtime for 5 runs:  5.9 sec.
Std runtime for 5 runs:  1.2 sec.
Plot data exported to ./tmp.txt
```

Sample input/output test for Python:

```
 $python mysmosvm.py /export/MNIST-13 3
Avg runtime for 3 runs:  6.2 sec.
Std runtime for 3 runs:  1.8 sec.
Plot data exported to ./tmp.txt
```

3. **(30 points)** The goal is to evaluate the performance of an optimization algorithm for SVMs which work on the *primal*. The problem is based on the following paper: "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM," by S. Shalev-Shawtrz, Y. Singer, and N. Srebro. Please read section 2 of the paper. Section 1 gives a good review of ideas for SVM optimization (till 2007, when the paper was written). The idea discussed in the paper is known as Pegasos, a mini-batch (projected) sub-gradient descent method for the SVM non-smooth convex optimization problem.

Implement the Pegasos algorithm and evaluate its performance on the `MNIST-13` dataset. The evaluation will be based on the optimization performance, and not classification accuracy. So, we will use the entire dataset for training (optimization). For the runtime results, run the code 5 times and report the average runtime and standard deviation. The runs will be repeated with different choices of mini-batch size.

You will have to submit (i) **summary of methods and results** report, and (ii) **code** for each algorithm:

(i) **Summary of methods and results:** Pegasos works with a mini-batch A_t of size k in iteration t. When $k = 1$, we get (projected) stochastic gradient descent, and when $k = n$, we get (projected) gradient descent, since the entire dataset is considered in each iteration. For the runs, we will consider the following values of k: (a) $k = 1$, (b) $k = 20$ (1 %), (c) $k = 100$ (5 %), (d) $k = 200$ (10 %), and (e) $k = n$, the full data. For fixed percent mini-batches, pick the corresponding percentages from each class, e.g., for 5%, pick 5% of samples from each of the two classes.

For each choice of k as above, report the average runtime over 5 runs on the entire dataset, along with the standard deviation. For each choice of k, plot the primal objective function value for each run with increasing number of iterations. Thus, there will be a figure with 5 plots (different runs) overlayed for each choice of k, and a separate figure for each k.

(ii) **Code:** You will have to submit code for `mysgdsvm(filename, k, numruns)` (main file). This main file has **input**: (1) a filename containing the dataset, (2) minibatch size, and (3) the number of runs, and **output**: (1) the average runtime and standard deviations printed to the terminal (stdout). The function *must* take the inputs in this order and display the output via the terminal. The filename will correspond to a plain text file for a dataset, with each line corresponding to a data point: the first entry will be the label and the rest of the entries will be feature values of the data point. The data for the plots should be written in a file `tmp` in the same directory, but (i) we will not be using this data for doing plots, and (ii) you do not have to explain how you did the plots from this data.

You can submit additional files/functions (as needed) which will be used by the main file. Put comments in your code so that one can follow the key parts and steps in your code. Please do not include input file in your submission folder.

Sample input/output in MATLAB:

```
 >>mysgdsvm("D:\MNIST-13", 15, 5)
Avg runtime for 5 runs with minibatch size of 15:  5.9 sec.
Std runtime for 5 runs with minibatch size of 15:  1.2 sec.
Plot data exported to ./tmp.txt
```

Sample input/output test for Python:

```
 $python mysgdsvm.py /export/MNIST-13 10 3
Avg runtime for 3 runs with minibatch size of 10:  6.2 sec.
Std runtime for 3 runs with minibatch size of 10:  1.8 sec.
Plot data exported to ./tmp.txt
```

Instructions

Follow the rules strictly. If we cannot run your functions, you get 0 points.

- **Convergence:** Lets θ be the variable being optimized - so $\theta = \alpha$ for Q2 and $\theta = w$ for Q3. Consider the union of two stopping criteria as the convergence measure: (i) a stopping criterion of $||\theta_t - \theta_{t-1}||_2 \leq 0.1$ (ii) the maximum number of iterations to be $T_{\max} = 10,000$ So, if (i) is met for $t < T_{\max}$, the code will exit, otherwise it runs till $T_{\max}$ iterations.

- **Things to submit**

  1. hw2.pdf: The report that contains the solutions to Problems 1,2, and 3, including the summary of methods and results.
  2. `mysmosvm`: Code for Question 2.
  3. `mysgdsvm`: Code for Question 3.
  4. README.txt: README file that contains your name, student ID, email, instructions on how to your run program, any assumptions you're making, and any other necessary details.
  5. Any other files, except the data, which are necessary for your program.