CSCI 5525: Machine Learning (Fall'16)

Homework 2, Due Fri, 10/14/16 11:55 pm

1. **(20 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) Consider the function $K(x, x') = \exp(-(x - x')^2/2016)$ 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 works on the *dual*. The problem is based on the following paper: "LIBSVM: A Library for Support Vector Machines," by Chih-Chung Chang and Chih-Jen Lin. Please read Sections 2.1, 4.1, and 6 of the paper. You can also refer to the discussion on SVM and Lagrange duality in class as needed.

   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 (as a minimization problem) can be written as:

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

   Note that the above is the same as (2) in the above paper.

   Implement Algorithm 1 in Section 4.1.1 of the paper using ideas in Sections 4.1 and 6, and evaluate its performance on the `MNIST-13` dataset. The dataset contains two classes labeled as 1 and 3 in the **first** column of the `csv` file we provided. All other columns are data values. 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 on the entire dataset 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:** Algorithm 1 is a 'decomposition method' which iteratively updates α by updating two components (α_i, α_j) (called working set) 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 working set is 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 (in minimization form) should decrease over iterations to convergence.

Termination condition: Please use a combination of the following two termination conditions: (1) after the current iteration, the improvement on the objective function is smaller than 10^{-4}; or (2) the total number of iterations is greater than 1000. The algorithm terminates if either condition (1) **or** (2) is satisfied.

(ii) **Code:** You will have to submit code for `mySmoSVM(filename, numruns)` (main file). This main file has **input**: (1) a filename (including extension and absolute path) 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 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 can 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.

3. **(50 points)** The goal is to evaluate the performance of optimization algorithms for SVMs which work on the *primal*. The problem is based on the following papers:

(1) "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM," by S. Shalev-Shawtrz, Y. Singer, and N. Srebro. Please read Section 2 of the paper. The algorithm discussed in the paper is a (stochastic) mini-batch (projected) sub-gradient descent method for the SVM non-smooth convex optimization problem. For the experiments, we will assume $b = 0$ for the affine form $f(x) = \mathbf{w}^T\mathbf{x} + b$ (you can see related discussions in Sections 4 and 5).

(2) "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction," by R. Johnson and T. Zhang. Please read Section 2 and Figure 1 of the paper. The algorithm discussed in the paper, called SVRG (stochastic variance reduced gradient), is a variant of stochastic gradient descent (SGD) with variance reduction. We will use SVRG for learning linear SVMs using sub-gradients as in Pegasos and assuming $b = 0$ in the function $f(x) = \mathbf{w}^T\mathbf{x} + b$.

For the homework, you have to do the following:

(1) (25 points) 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 (see below).

(2) (25 points) Implement the SVRG 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 epoch size (see below).

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 % of data), (c) $k = 200$ (10 % of data), (d) $k = 1000$ (50 % of data), and (e) $k = 2000$ (100 % of data). For fixed percent mini-batches, pick the corresponding percentages from each class, e.g., for 10%, pick 10% of samples from each of the two classes.

For each choice of k as above, report the average runtime of Pegasos 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.

SVRG works with a SGD epoch size m for each outer iteration s. When $m = 1$, we get simply do one SGD update in every outer iteration, and when $m = n$, we do n SGD updates in each outer iteration. For the runs, we will consider the following values of m: (a) $m = 1$, (b) $m = 20$ (1 % of data), (c) $m = 200$ (10 % of data), (d) $m = 1000$ (50 % of data), and (e) $m = 2000$ (100 % of data).

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

Termination condition: Please use k_{tot} - the total number of gradient computations - as the termination condition. For this question, set $k_{tot} = 100n$, where n is your data size (number of data points). The algorithm terminates if k_{tot} is reached.

(ii) **Code:** You will have to submit code for (1) `myPegasos(filename, k, numruns)` (main file), and (2) `mySVRG(filename, m, numruns)` (main file). The main files have **input**: (1) a filename (including extension and absolute path) containing the dataset, (2) mini-batch size k for Pegasos and SGD epoch size m for SVRG, and (3) the number of runs, and **output**: (1) the average runtime and standard deviations printed to the terminal (stdout). The functions 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 can 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.

Instructions

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

- **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. `myPegasos` and `mySVRG`: 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.