CS 395T	Scalable Machine Learning	Fall 2014

Homework 3

Lecturer: Inderjit Dhillon — Date Due: Sep 29, 2014

Keywords: *SVM, Coordinate Descent, OpenMP*

Note:

1. In this assignment we use the following two datasets: RCV1 and Covtype. These two datasets can be downloaded from the Assignments tab in Canvas (called `hw3-data.zip`).
2. You will run your code and collect timing results on TACC machines. See `https://www.tacc.utexas.edu/user-services/user-guides/stampede-user-guide` to know about using the Stampede cluster.
3. Submit your code through Canvas. Submit a hard copy of your answers in class.

Problem 1: Dual Coordinate Descent for Linear SVM

Given training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, each $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$, consider the following variant of SVM problem (without the bias term):

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^d} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^n \xi_i^2 \equiv g(\boldsymbol{w}) \\ \text{subject to} \quad & y_i\boldsymbol{w}^T\boldsymbol{x}_i \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, \ldots, n. \end{aligned} \tag{1}$$

Note that in the above formulation we consider the squared hinge ξ_i^2 instead of ξ_i. The dual problem has the following form:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{\alpha}^T(Q + \frac{1}{2C}I)\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \equiv f(\boldsymbol{\alpha}), \\ \text{subject to} \quad & 0 \leq \alpha_i, i = 1, \ldots, n, \end{aligned} \tag{2}$$

where $Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$. The optimal primal variable $\boldsymbol{w}^*$ and optimal dual variable $\boldsymbol{\alpha}^*$ satisfy

$$\boldsymbol{w}^* = \sum_{i=1}^n y_i \alpha_i^* \boldsymbol{x}_i.$$

Write a C/C++ program which implements the coordinate descent algorithm for solving the SVM dual problem (Algorithm 1), and parallelize it using OpenMP.

(a) (5 pt) Show that (2) is the dual problem of (1).

(b) (2 pt) Applying coordinate descent to (2) requires solving the following single variable sub-problem.

$$\delta^* = \arg\min_{\delta} f(\boldsymbol{\alpha} + \delta\boldsymbol{e}_i). \tag{3}$$

Derive the closed form solution to (3). What is the time complexity for directly computing the closed form solution?

Algorithm 1 Co-ordinate Descent for Dual SVM

- Input: Data $\{y_i, \boldsymbol{x}_i : i = 1, \ldots, n\}$ and C
- Initialize $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ to zero
- For iter $= 1, \ldots,$ (Outer iterations)
  - Choose a random permutation π
  - For $s = 1, \ldots, n$ (Inner iterations)
    * $i = \pi(s)$
    * Solve single variable problem for α_i:

$$\delta^* = \arg\min_{\delta} f(\boldsymbol{\alpha} + \delta \boldsymbol{e}_i).$$

    * Perform updates:
      $\alpha_i = \alpha_i + \delta^*$
      maintain $\boldsymbol{w}$
  - Output timing, objective function value, and test accuracy

(c) (3 pt) Function value evaluation $f(\boldsymbol{\alpha})$ involves a term $\boldsymbol{\alpha}^T Q \boldsymbol{\alpha}$, which requires $O(n^2)$ time as Q is a dense matrix. Now assume that a d dimensional variable $\boldsymbol{w} = \sum_{i=1}^n y_i \alpha_i \boldsymbol{x}_i$ is available. Give an alternative formulation of $f(\boldsymbol{\alpha})$ using both $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ such that the function value evaluation $f(\boldsymbol{\alpha})$ costs only $O(n + d)$.

(d) (2 pt) How to use the assumption in part (c) (*i.e.*, $\boldsymbol{w} = \sum_{i=1}^n y_i \alpha_i \boldsymbol{x}_i$) to reduce the time complexity for computing the closed form solution for δ^* to $O(d_i)$, where d_i is the number of non-zeros of $\boldsymbol{x}_i$?

(e) (1 pt) How would you maintain $\boldsymbol{w}$ after the α_i update to ensure the validity of the assumption $\boldsymbol{w} = \sum_{i=1}^n y_i \alpha_i \boldsymbol{x}_i$? The time complexity for the maintenance should also be $O(d_i)$.

(f) (18 pt) Implement the dual co-ordinate descent for solving (2) in C/C++ and parallelize the inner iterations using OpenMP. The pseudo code is given in Algorithm 1. At the end of each outer iteration, print the following quantities:

  (a) The wall time for co-ordinate descent updates (excluding preprocessing time and time for computing the measurements below).
  (b) Compute and output the dual objective function value (using the current $\boldsymbol{\alpha}, \boldsymbol{w}$).
  (c) Compute and output the primal objective function value using $\boldsymbol{w}$.
  (d) Compute and output the prediction accuracy using current $\boldsymbol{w}$.
  (e) Compute and output $\| \sum_{i=1}^n y_i \alpha_i \boldsymbol{x}_i - \boldsymbol{w} \|_2^2$. This quantity should be zero if $\boldsymbol{w}$ is correctly maintained. To ensure the value is zero in the multi-threaded environment, you might need to use **# pragma omp atomic**.

  Report the 20-iteration running time, primal objective function, and prediction accuracy for both RCV1 and Covtype datasets with $1, 8, 16$ threads using $C = 0.1$. What is the speedup you observe?

(g) (4 pt) Remove the atomic flag (or locks, critical sections) in your code for correctly maintaining $\boldsymbol{w} = \sum_{i=1}^n y_i \alpha_i \boldsymbol{x}_i$.

For each dataset, and for the locking and non-locking cases, plot the wall time versus relative error ($(g(\boldsymbol{w}) - g(\boldsymbol{w}^*))/g(\boldsymbol{w}^*)$), where $g(\cdot)$ is the primal SVM objective function (1). In each of the four plots, include curves for $1, 8, 16$ threads using $C = 0.1$. Plot the relative error in log-scale. Comment on your observations.

Note that you can use the sequential version optimal solution as $\boldsymbol{w}^*$.

Below are some guidelines for your implementation.

- Fix the number of outer iterations to 20 and initialize $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ with the zero vector.
- Write a Makefile to compile your code using `icc`. The name of the output binary file should be `cd-svm`.
- Usage of the binary file: `./cd-svm C nr_threads traindata testdata`
- Datasets:
  - `hw3-data.zip` contains the text-format version of the three datasets used in HW3
  - For each dataset, there is a testing file (ended with .t) and a training file (ended with .tr). Each line of the files contain a data point, started with the label (+1 or -1) and followed by a sorted list of $<$ feature $>$:$<$ value $>$ pairs. See SVMLight format described in `http://svmlight.joachims.org/` for detail.