Homework #5

RELEASE DATE: 11/16/2012

DUE DATE: 11/29/2012, BEFORE THE END OF CLASS

QUESTIONS ABOUT HOMEWORK MATERIALS ARE WELCOMED ON THE FORUM.

Unless granted by the instructor in advance, you must turn in a printed/written copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

5.1 Decision Stumps Revisited, Theoretically

(1) (10%) Do Exercise 3.15(a) of LFD.

(2) (10%) Do Exercise 3.15(b) of LFD.

5.2 More on Weight Decay

(1) (10%) Do Problem 4.6(a) of LFD.

(2) (10%) Do Problem 4.8 of LFD.

5.3 Regularization and Virtual Examples

Note that this is a good opportunity for you to learn about Tikhonov regularization, a simple yet more generalized form than the one taught in class.

(1) (10%) Do Problem 4.9(a) of LFD.

(2) (10%) Do Problem 4.9(b) of LFD.

5.4 Leave-one-out Error for Linear Regression

(1) (10%) Do Problem 4.26(a) of LFD.

(2) (10%) Do Problem 4.26(b) of LFD.

(3) (10%) Do Problem 4.26(c) of LFD.

(4) (10%) Do Problem 4.26(d) of LFD.

(5) (10%) Do Problem 4.26(e) of LFD.

5.5 Be a Learning Expert

(1) (10%) Do Problem 5.3(a) and 5.3(b) of LFD, using 1% instead of 0.1% as your performance tolerance.

5.6 Experiments with Coordinate Descent for Perceptron Learning (*)

In class, we mention that learning can be achieved with the following steps:

(1) Connect E_{out} to E_{in} (VC bound or other theoretical bounds).

(2) Define an error function E directly from E_{in} or as something related to E_{in}.

(3) Minimize E with tools in optimization.

For instance, the pocket algorithm follows the VC bound, assumes an error function E that is exactly E_{in} in classification and minimizes E_{in} (the NP-hard problem) with an special iterative solver. One representative iterative solver that we have introduced is gradient descent (and stochastic gradient descent).

Recall that in the gradient descent solver, we update the weight vector by

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot (-\nabla E(\mathbf{w}(t)))$$

with a fixed small $\eta > 0$. In this problem, we explore the possibility of using a more general update formula. That is

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t \cdot \mathbf{v}(t)$$

for some update direction $\mathbf{v}(t) \in \mathbb{R}^{d+1}$ and step size $\eta_t \in \mathbb{R}$. We will consider a greedy search algorithm as follows. In the t-th iteration, the goal of the algorithm is to choose $\mathbf{v}(t)$ and η_t such that

$$E(\mathbf{w}(t+1))$$

is minimized.

(1) (10%) Consider the simplest case of using some $\mathbf{v}(t) = \mathbf{v}$ that satisfies $v_i = 1$ for a specific i and $v_i = 0$ otherwise. In other words, $\mathbf{w}(t+1)$ and $\mathbf{w}(t)$ differs only in the i-th component (direction). Consider $E = E_{\text{in}}$ with the classification (0/1) error. When given $\mathbf{w}(t)$ and $\mathbf{v}(t)$, **ASSUME THAT all $(\mathbf{x}_n)_i$ are non-zero**, derive an efficient algorithm that finds the step η_t along the direction $\mathbf{v}(t)$ which minimizes E. That is, solve the following optimization problem.

$$\begin{aligned} \min_{\eta_t \in \mathbb{R}} \quad & \frac{1}{N} \sum_{n=1}^{N} [\![y_n \neq \text{sign}\big(\mathbf{w}(t+1)^T \mathbf{x}_n\big)]\!] \\ \text{subject to} \quad & \mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t \mathbf{v}(t). \end{aligned}$$

Hint: decision stump, Decision stump, Decision Stump, DECISION STUMP.
Hmm, did we provide enough hint?

(2) (5%) If some of the feature components $(\mathbf{x}_n)_i$ are zero, how does your algorithm above change?

(3) (5%) If the $\mathbf{v}(t)$ given is an arbitrary vector rather than the specific one above, how does your algorithm above change?

(4) (10%) The coordinate descent algorithm for optimization is as follows:

(a) initialize a $(d+1)$-dimensional vector $\mathbf{w}(1)$, say, $\mathbf{w}(1) \longleftarrow (0, 0, \ldots, 0)$.

(b) for $t = 1, 2, \ldots, T$

- choose some direction $\mathbf{v}(t)$
- update

$$\mathbf{w}(t+1) \longleftarrow \mathbf{w}(t) + \eta_t \mathbf{v}(t).$$

by minimizing the error function over η_t.

One special case of the coordinate descent algorithm is to choose

$$\begin{aligned}
\mathbf{v}(1) &= (1,0,0,\cdots,0)\\
\mathbf{v}(2) &= (0,1,0,\cdots,0)\\
&\cdots\\
\mathbf{v}(d+1) &= (0,0,0,\cdots,1)\\
\mathbf{v}(d+2) &= (1,0,0,\cdots,0)\\
&\cdots
\end{aligned}$$

The special case is called *cyclic coordinate descent*. Implement the cyclic coordinate descent algorithm, use $T = 10000$, and run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_train.dat`

Then, use the $\mathbf{w}(t)$ to predict the label of each example within the following test set:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_test.dat`

Plot the E_{in} and E_{out} you get as a function of t, and briefly state your findings.

(5) (10%) Another special case of the coordinate descent algorithm is to choose $\mathbf{v}$ randomly. Consider picking each component of $\mathbf{v}$ by a standard Gaussian distribution (mean 0 and variance 1). The special case is called *random (Gaussian) coordinate descent*. Implement the random coordinate descent algorithm, use $T = 10000$, and run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_train.dat`

Then, use the $\mathbf{w}(t)$ to predict the label of each example within the following test set:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_test.dat`

Plot the E_{in} and E_{out} you get as a function of t, and briefly state your findings.

(6) (10%) Run the pocket algorithm you have implemented with $T = 10000$ on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_train.dat`

Plot the E_{in} you get for pocket, cyclic coordinate descent and random coordinate descent as functions of t, and briefly state your findings.

5.7 Experiments with Iterative Least Squares for Regularized Logistic Regression (*)

Next, we derive an algorithm for minimizing the augmented error for regularized logistic regression:

$$E_{\text{aug}}(\mathbf{w}) = \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{N}\sum_{n=1}^{N}\log\Big(1+\exp(-y_n\mathbf{w}^T\mathbf{x}_n)\Big).$$

(1) (10%) Write down the second-order Taylor expansion of the augmented error function above in the neighborhood of some $\mathbf{w}(t)$.

It can be proved that the Hessian matrix of $E_{\text{aug}}(\mathbf{w})$ above is always strictly positive definite and hence invertible. We will leave the full proof to the students next year. (*Yeah!! :-)*)

(2) (10%) Recall that we explained update rule of the gradient descent algorithm as the "optimal" step when using the first-order Taylor expansion to approximate the error function. When the Hessian matrix at some $\mathbf{w}(t)$ is strictly positive definite, derive an update rule that corresponds to the "optimal" step when using the second-order Taylor expansion to approximate the augmented error function. (*Hint: Homework 0*)

(3) (10%) Replacing the update rule in gradient descent with the update rule you derive above leads to a new algorithm, usually called Newton algorithm for optimization. It is important to keep Newton algorithm in your toolbox. Implement Newton algorithm for regularized logistic regression, use $T = 10000$ and $\lambda = 0.000001$, and run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_train.dat`

Then, use the $\mathbf{w}(t)$ to predict the label of each example within the following test set:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/hw5/hw5_test.dat`

Plot the E_{in} and E_{out}, **calculated by the 0/1 error**, as functions of t, and briefly state your findings.

(*When Newton algorithm is coupled with the logistic regression error function, the resulting algorithm looks much like repeatedly doing the least-square linear regression with varying inputs/outputs, and hence is often called the "Iterative Least Squares" algorithm for Logistic Regression.*)

5.8 Be a Learning Expert with Bonus

Please use 1% instead of 0.1% as your performance tolerance in the problems.

(1) (Bonus 5%) Do Problem 5.3(c) of LFD.

(2) (Bonus 5%) Do Problem 5.3(d) of LFD.