Homework #5

TA email: ml2011ta@csie.ntu.edu.tw

RELEASE DATE: 11/21/2011

DUE DATE: 12/5/2011, BEFORE THE END OF CLASS

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 and predictions 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 Data-Dependent Transforms

(1) (10%) Consider Problem 3.21(a) of LFD and E_{in} evaluated with the 0/1 error. Argue that there exists some $\tilde{\mathbf{w}}$ in the $\mathcal{Z}$ space such that $E_{\text{in}} = 0$. In other words, the transformed data set is always linearly separable in the $\mathcal{Z}$ space.

(2) (5%) Do Problem 3.21(a) of LFD.

(3) (10%) Consider Problem 3.21(b) of LFD and E_{in} evaluated with the 0/1 error. Argue that there exists some $\tilde{\mathbf{w}}$ in the $\mathcal{Z}$ space such that $E_{\text{in}} = 0$. In other words, the transformed data set is always linearly separable in the $\mathcal{Z}$ space. *(Note: You can use the fact that if $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ are all different, the matrix $\mathrm{A} = [a_{mn}]$ with $a_{mn} = \phi_m(\mathbf{x}_n)$ with the Gaussian ϕ_m is always invertible.)*

(4) (5%) Do Problem 3.21(b) of LFD.

5.2 Gradient and Newton Directions

Do Problem 3.18 of LFD using

$$E(u, v) = e^{2u} + e^{v} + e^{4uv} + u^2 - uv + 2v^2 - 4u + v.$$

(1) (5%) Do Problem 3.18(a) of LFD.

(2) (5%) Do Problem 3.18(b) of LFD.

(3) (5%) Do Problem 3.18(c) of LFD.

(4) (5%) Do Problem 3.18(d) of LFD.

(5) (5%) Do Problem 3.18(e) of LFD.

5.3 Model Overfit

Do Exercise 4.3 of LFD, assuming that the model complexity is always below the target complexity. This is a brain-storming problem that requires **reasoning** rather than just answers. Thus, you are strongly encouraged to discuss about this problem on the book forum.

(1) (5%) Do Exercise 4.3(a) of LFD.

(2) (5%) Do Exercise 4.3(b) of LFD.

(3) (5%) Do Exercise 4.3(c) of LFD.

(4) (5%) Do Exercise 4.3(d) of LFD.

5.4 Regularization and Virtual Examples

In Tikhonov regularization (see Exercise 4.5), when using the augmented error, the regularized weights are given by

$$\mathbf{w}_{\text{reg}} = (\mathrm{Z}^T\mathrm{Z} + \lambda\Gamma^T\Gamma)^{-1}\mathrm{Z}^T\mathbf{y}.$$

The Tikhonov regularizer Γ is a $k \times (\tilde{d}+1)$ matrix, each row corresponding to a $\tilde{d}+1$ dimensional vector. On the other hand, each row of Z corresponds to a $\tilde{d}+1$ dimensional vector. In this problem, we see how the two matrices are related. For each row of Γ, construct a *virtual example* $(\mathbf{z}_{N+i}, y_{N+i})$ for $i = 1, \dots k$, where $\mathbf{z}_{N+i}$ is the vector obtained from the i-th row of Γ multiplied by $\sqrt{\lambda}$ and $y_{N+i} = 0$.

(1) (5%) Write down the virtual examples that are added to the original data set when using the common squared regularizer in Equation (4.5) of LFD.

(2) (10%) Prove that solving Tikhonov-regularized linear regression is equivalent to applying the plain-vanilla linear regression on $\{(\mathbf{z}_n, y_n)\}_{n=1}^{N+k}$.

You can check Problem 4.8 of LFD (which unfortunately contains some bugs) for some explanations of the meaning of this problem.

5.5 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.

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

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

(2) (5%) If some of $(\mathbf{x}_n)_i$ 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) (15%) 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/ml11fall/data/hw5_train.dat`

Then, use the returned perceptron to predict the label of each example within the following test set:

`http://www.csie.ntu.edu.tw/~htlin/course/ml11fall/data/hw5_test.dat`

Submit your predictions to the designated place (to be announced on the course website)—*note that you can only submit **once**.*

(5) (15%) 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/ml11fall/data/hw5_train.dat`

Then, use the returned perceptron to predict the label of each example within the following test set:

`http://www.csie.ntu.edu.tw/~htlin/course/ml11fall/data/hw5_test.dat`

Submit your predictions to the designated place (to be announced on the course website)—*note that you can only submit **once**.*

(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/ml11fall/data/hw5_train.dat`

Compare the E_{in} you get for pocket, cyclic coordinate descent and random coordinate descent. Briefly state your findings.

5.6 Polynomial Regression and Overfitting (*)

Hint/Warning: Time-consuming! Please start running the experiments as early as possible.
Do a variant of Exercise 4.2 of LFD with the following changes:

- Change $\mathcal{H}_2$ to $\mathcal{H}_1$ (i.e., the original linear regression).
- Change $\mathcal{H}_{10}$ to $\mathcal{H}_8$ (eighth order polynomials).

You may find some hints in Problem 4.3 of LFD.

(1) (10%) First, read the definition of Legendre polynomials in Problem 4.2 of LFD. Then, plot and list the formula of L_5 to L_8. You'll actually find L_0 to L_5 listed on LFD page 4-11.

(2) (10%) Then, explain how you compute $E_{\text{out}}(g)$ for any polynomial function $g(x)$ under the setting of Exercise 4.2 of LFD. (*Hint: There is an analytic solution. You get partial credit if you compute E_{out} through E_{test} and explain your procedure clearly.*)

(3) (20%) Plot your results with figures like those in Figure 4.2 of LFD.

(4) (10%) Compare your figures with Figure 4.2 of LFD and briefly state your findings.

5.7 Hints and Virtual Examples

(1) (Bonus 10%) Continue from Homework 5.4. Assume that we have some known hints $\mathbf{w}_{\text{hint}}$ about the rough value of $\mathbf{w}$ and hence want to use

$$\lambda \|\mathbf{w} - \mathbf{w}_{\text{hint}}\|^2$$

as the regularizer instead of the squared $\lambda \mathbf{w}^T \mathbf{w}$. What virtual examples should we equivalently add to the original data set?