Homework #4

RELEASE DATE: 10/28/2012

DUE DATE: 11/16/2012, 14:30, in R536

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.

4.1 VC for Composite Hypotheses

(1) (10%) Do Problem 2.19(a) of LFD.

(2) (10%) Do Problem 2.19(b) of LFD.

(3) (10%) Do Problem 2.19(c) of LFD, under the condition of $D > 2e \log_2 D$.

(4) (Bonus 5%) Prove or disprove the result of Problem 2.19(c) of LFD, under the original condition of $D > 2 \log_2 D$.

(*Note: The interesting approach of treating the hypotheses h_k as a feature transform (to $\mathbf{z}$) is an important technique in ML that will be taught later. In the next problem, we will see one practical use of the approach.*)

4.2 Test-Set Linear Regression

The root-mean-square-error (RMSE) of a hypothesis h on a test set $\{(\tilde{\mathbf{x}}_n, \tilde{y}_n)\}_{n=1}^{\tilde{N}}$ is defined as

$$\text{RMSE}(h) = \sqrt{\frac{1}{\tilde{N}} \sum_{n=1}^{\tilde{N}} \left(\tilde{y}_n - h(\tilde{\mathbf{x}}_n)\right)^2}.$$

In the next questions, please consider a case of knowing all the $\tilde{\mathbf{x}}_n$, none of the $\tilde{y}_n$, but allowed to query RMSE(h) for T (different) h.

(1) (10%) If $T = \tilde{N} + 1$, design an algorithm such that you can construct some hypothesis g with RMSE(g) = 0. In other words, you can "cheat" from the RMSEs to obtain $g(\tilde{\mathbf{x}}_n) = \tilde{y}_n$ for every $n = 1, 2, \cdots, \tilde{N}$.

(2) (10%) The algorithm above is slow (and thus "impractical") if $\tilde{N}$ is too large. Let us start designing a smarter way of "cheating." For any given hypothesis h, let

$$\begin{aligned} \mathbf{h} &= (h(\tilde{\mathbf{x}}_1), h(\tilde{\mathbf{x}}_2), \cdots, h(\tilde{\mathbf{x}}_{\tilde{N}})) \\ \tilde{\mathbf{y}} &= (\tilde{y}_1, \tilde{y}_2, \cdots, \tilde{y}_{\tilde{N}}). \end{aligned}$$

Note that you can compute $\mathbf{h}$ but you do not know $\tilde{\mathbf{y}}$. If $T = 2$, design an algorithm to compute $\mathbf{h}^T \tilde{\mathbf{y}}$.

(3) (10%) For any given set of hypotheses $\{h_1, h_2, \cdots, h_K\}$, if $T = K + 1$, use the result in (2) to design an algorithm to solve

$$\min_{w_1, w_2, \cdots, w_K} \text{RMSE}\left(\sum_{k=1}^{K} w_k h_k\right),$$

and obtain the optimal weights $w_1, \cdots, w_K$. In other words, you can use a small $(K + 1)$ queries to obtain the best linear combination of the given set of hypotheses — smart cheating!

4.3 Maximum Likelihood: Sword or Overkill

(1) (10%) Subject to an unknown target function $f\colon \mathcal{X} \to \mathbb{R}$, if each example $(\mathbf{x}, y)$ is generated i.i.d. from the following process:

  (a) generate $\mathbf{x}$ from some $P(\mathbf{x})$
  (b) generate noise ϵ from a Gaussian distribution with zero mean and a fixed (but unknown) variance σ^2
  (c) compute $y = f(\mathbf{x}) + \epsilon$.

For any given hypothesis $h \in \mathcal{H}$ and given data $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$, what is the likelihood that h is the unknown target function used to generate the data?

(2) (10%) According to the process above, argue that linear regression is equivalent to seeking for the maximum likelihood solution for the linear hypothesis set $\mathcal{H}$.

(3) (10%) Subject to an unknown target function $f\colon \mathcal{X} \to \{-1, +1\}$, if each example $(\mathbf{x}, y)$ is generated i.i.d. from the following process:

  (a) generate $\mathbf{x}$ from some $P(\mathbf{x})$
  (b) compute $y = f(\mathbf{x})$
  (c) for some fixed (but unknown) probability $0 < \delta < 0.5$, flip y (assign the value $-y$ to the variable y).

Argue that for any hypothesis set $\mathcal{H}$, minimizing the average 0/1 error in binary classification is equivalent to seeking for the maximum likelihood solution.

(4) (10%) On LFD page 91, there is a footnote "Although the method of maximum likelihood is intuitively plausible, its rigorous justification as an inference tool continues to be discussed in the statistics community." Based on your derivations above, do you think the method of maximum likelihood offers better explanations for the 0/1 and the squared errors (than their straightforward meanings taught in class)? Why or why not?

4.4 Derivation of Multiclass Logistic Regression

In this problem, we will guide you through deriving the multiclass logistic regression (MLR) algorithm. For a K-class classification problem, we will denote the output space $\mathcal{Y} = \{1, 2, \cdots, K\}$. The hypotheses considered by MLR are indexed by a list of weight vectors $(\mathbf{w}_1, \cdots, \mathbf{w}_K)$, each weight vector of length $d + 1$. The list represents a probability distribution

$$P(y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{i=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})}.$$

MLR then seeks for the maximum likelihood solution over all such hypotheses.

(1) (10%) For $K = 2$, argue that MLR is equivalent to the (binary) logistic regression algorithm taught in class. More specifically, derive the equivalence between the $(\mathbf{w}_1, \mathbf{w}_2)$ used by MLR and the $\mathbf{w}$ used by logistic regression.

(2) (15%) For general K, derive an $E_{\text{in}}(\mathbf{w}_1, \cdots, \mathbf{w}_K)$ like (3.9) of LFD by minimizing the negative log likelihood.

(3) (15%) For the E_{in} derived above, write down its gradient ∇E_{in}.

(*Based on the derivation above, you can now perform multiclass probability estimation using the linear model, and use the estimate for classification or other more sophisticated tasks.*)

4.5 Polynomial Regression (*)

(1) (20%) Perform polynomial regression by coupling $\mathbf{\Phi}_Q$ with linear regression. You can either use the naïve $\mathbf{\Phi}_Q$ explained in Chapter 3 of the LFD, or the Legendre polynomial discussed in Chapter 4 of LFD. If the resulting $\mathrm{Z}^T\mathrm{Z}$ is not invertible, please try using the pseudo inverse $\mathrm{Z}^\dagger$. (*Hint: The definition of the pseudo inverse can be found in Homework 0.2(4a). We suggest you to implement the linear regression part with pseudo inverse in the first place to handle both the invertible and the non-invertible cases smoothly.*)

Run the algorithm with $Q = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$, on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/data/hw4_5_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/data/hw4_5_test.dat`

Assume that g_Q is the hypothesis returned by the algorithm. Use the squared error to evaluate $E_{\text{in}}(g_Q)$ and $E_{\text{out}}(g_Q)$, and plot them as a function of Q. Are the curves similar to the blue curves in Figure 2.3 of LFD? Briefly state your findings.

4.6 Gradient Descent for Logistic Regression (*)

(1) (20%) Implement the fixed learning rate stochastic gradient descent algorithm for logistic regression. Assume that

$$g_1^{(t)}(\mathbf{x}) = \text{sign}\left(\mathbf{w}(t)^T\mathbf{x}\right),$$

where $\mathbf{w}(t)$ are generated from the stochastic gradient descent algorithm. Run the algorithm with $\eta = 0.001$ and $T = 2000$ on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/data/hw4_6_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml12fall/data/hw4_6_test.dat`

Plot $E_{\text{in}}\left(g_1^{(t)}\right)$ and $E_{\text{out}}\left(g_1^{(t)}\right)$ as a function of t and briefly state your findings.

(2) (20%) Implement the fixed learning rate gradient descent algorithm for logistic regression. Assume that

$$g_2^{(t)}(\mathbf{x}) = \text{sign}\left(\mathbf{w}(t)^T\mathbf{x}\right),$$

where $\mathbf{w}(t)$ are generated from the gradient descent algorithm. Run the algorithm with $\eta = 0.001$ and $T = 2000$ on the same sets above. Plot $E_{\text{in}}\left(g_2^{(t)}\right)$ and $E_{\text{out}}\left(g_2^{(t)}\right)$ as a function of t, compare it to your plot for $g_1^{(t)}$, and briefly state your findings.

4.7 Smart Cheating Revisited

In Problem 4.2, we discussed the "slow cheating" and "smart cheating" techniques. :-)

(1) (Bonus 5%) Generally, ML competitions do not show E_{test} (say, RMSE) of the whole test set on the scoreboard during the competition. Instead, only the performance with respect to a fixed subset (say, 30%) is revealed as the feedback, and the other subset is hidden and used for determining the final winner. Argue that such a setup prevents "slow cheating."

(2) (Bonus 5%) Assume that the fixed subset above is randomly selected in the first place, argue that the setup does not fully prevent "smart cheating." In particular, why did smart cheating still work in the past KDD Cups?

(*Note: I want to take this opportunity to state my perspective about the "smart cheating" technique. In ML competitions, the efforts to improve the models are mostly similar to "test-set iterative learning/tuning" — observe the performance on the test set as the feedback for fine-tuning the models. The test-set linear regression approach, on the other hand, is doing "test-set analytic learning/tuning." If you recall our discussions about iterative (PLA) versus analytic (linear regression) learning in the class, you may agree that no sin is truly involved in doing the "smart cheating." The big concern, IMHO, is not the "cheating", but the highly impractical nature of the technique.*)