Homework #6

RELEASE DATE: 11/29/2023

RED CORRECTION: 12/16/2023 21:30

DUE DATE: 12/20/2023 (**THREE WEEKS, YEAH!!**), BEFORE 13:00 on GRADESCOPE

QUESTIONS ARE WELCOMED ON DISCORD (INFORMALLY) OR NTU COOL (FORMALLY).

You will use Gradescope to upload your scanned/printed solutions. For problems marked with (*), please follow the guidelines on the course website and upload your source code to Gradescope as well. Any programming language/platform is allowed.

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 or Chinese with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

This homework set comes with 240 points and 20 bonus points. In general, every homework set would come with a full credit of 240 points, with some possible bonus points.

More about Support Vector Machines

1. (20 points) When talking about non-uniform voting in aggregation, we mentioned that $\boldsymbol{\alpha}$ can be viewed as a weight vector learned from any linear algorithm coupled with the following transform:

$$\boldsymbol{\phi}(\mathbf{x}) = \Big(g_1(\mathbf{x}), g_2(\mathbf{x}), \cdots, g_T(\mathbf{x})\Big).$$

When studying kernel methods, we mentioned that the kernel is simply a computational short-cut for the inner product $(\boldsymbol{\phi}(\mathbf{x}))^T(\boldsymbol{\phi}(\mathbf{x}'))$. In this problem, we mix the two topics together using the decision stumps as our $g_t(\mathbf{x})$.

Assume that the input vectors contain only integers between (including) L and R, where $L < R$. Consider the decision stumps $g_{s,i,\theta}(\mathbf{x}) = s \cdot \text{sign}\Big(x_i - \theta\Big)$, where

$$\begin{aligned} i &\in \{1, 2, \cdots, d\}, \\ d &\text{ is the finite dimensionality of the input space,} \\ s &\in \{-1, +1\}, \\ \theta &\in \{\theta_1 = L + 0.5, \theta_2 = L + 1.5, \ldots, \theta_k = R - 0.5\}. \end{aligned}$$

Define $\boldsymbol{\phi}_{ds}(\mathbf{x}) = \Big(g_{+1,1,\theta_1}(\mathbf{x}), g_{+1,1,\theta_2}(\mathbf{x}), \ldots, g_{+1,1,\theta_k}(\mathbf{x}), \ldots, g_{-1,d,\theta_k}(\mathbf{x})\Big)$ as a column vector. What is $K_{ds}(\mathbf{x}, \mathbf{x}') = (\boldsymbol{\phi}_{ds}(\mathbf{x}))^T(\boldsymbol{\phi}_{ds}(\mathbf{x}'))$? Prove your answer.

(Hint: This result shows that aggregation learning with SVMs is possible. Those who are interested in knowing how perceptrons and decision trees can be used instead of decision stumps during kernel construction can read the following early work)

`https://www.csie.ntu.edu.tw/~htlin/paper/doc/infkernel.pdf` .

2. (20 points) For a given valid kernel K, consider a new kernel $\tilde{K}(\mathbf{x}, \mathbf{x}') = uK(\mathbf{x}, \mathbf{x}') + v$ for some $u > 0$. Note that v can be any real value. When $v \geq 0$, it is easy to prove that $\tilde{K}$ is still a valid kernel; when $v < 0$, however, $\tilde{K}$ may not always be a valid kernel. Prove that for the dual of soft-margin support vector machine, using $\tilde{K}$ along with a new $\tilde{C} = \frac{C}{u}$ instead of K with the original C leads to an equivalent g_{SVM} classifier. That is, the optimal $(\boldsymbol{\alpha}, b)$ obtained leads to the same decision boundary.

(Note: This result can be used to shift and scale the kernel function derived in the previous problem to a simpler form (even not as a valid kernel) when coupling it with the soft-margin SVM.)

Blending and Bagging

3. (20 points) Consider an aggregated binary classifier G that is constructed by uniform blending on 17 binary classifiers $\{g_t\}_{t=1}^{17}$. That is,

$$G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{17} g_t(\mathbf{x})\right)$$

Assume that each g_t is of test 0/1 error $E_{\text{out}}(g_t) = e_t > 0$. Define the total error $E = \sum_{t=1}^{17} e_t$. What is the largest value of $\frac{E_{\text{out}}(G)}{E}$? Prove your result.

(Note: The ratio roughly shows how G reduces the total error made by the hypotheses.)

4. (20 points) If bootstrapping is used to sample exactly $\frac{3}{4}N$ examples out of N, what is the probability that an example is *not* sampled when N is very large? List your derivation steps.

(Note: This is just an "easy" exercise of letting you think about the amount of OOB data in bagging/random forest.))

Adaptive Boosting and Gradient Boosting

5. (20 points) Consider applying the AdaBoost algorithm on a binary classification data set where 98% of the examples are positive. Because there are so many positive examples, the base algorithm within AdaBoost returns a constant classifier $g_1(\mathbf{x}) = +1$ in the first iteration. Let $u_n^{(2)}$ be the individual example weight of each example in the second iteration. What is

$$\frac{\sum_{n:\, y_n > 0} u_n^{(2)}}{\sum_{n:\, y_n < 0} u_n^{(2)}}?$$

Prove your answer.

(Note: This is designed to help you understand how AdaBoost can deal with "imbalanced" data immediately after the first iteration.)

6. (20 points) For the AdaBoost algorithm introduced in Lectures 208 and 211, let $U_t = \sum_{n=1}^{N} u_n^{(t)}$. That is, $U_1 = 1$ (you are very welcome! ;-)). Assume that $0 < \epsilon_t < \frac{1}{2}$ for each hypothesis g_t. Express $\frac{U_{T+1}}{U_1}$ in terms of $\epsilon_1, \epsilon_2, \ldots, \epsilon_T$. Prove your answer.

(Hint: Consider checking $\frac{U_{t+1}}{U_t}$ first. The result is the backbone of proving that AdaBoost will converge within $O(\log N)$ iterations.)

7. (20 points) For the gradient boosted decision tree, after updating all s_n in iteration t using the steepest η as α_t, prove that $\sum_{n=1}^{N}(y_n - s_n) g_t(\mathbf{x}_n) = 0$.

(Note: This special value may tell us some important physical property on the relationship between the vectors $[y_1 - s_1, y_2 - s_2, \ldots, y_N - s_N]$ and $[g_t(\mathbf{x}_1), g_t(\mathbf{x}_2), \ldots, g_t(\mathbf{x}_N)]$.)

Neural Networks

8. (20 points) For a Neural Network with at least one hidden layer and $\tanh(s)$ as the transformation functions on all neurons (including the output neuron), when all the initial weights $w_{ij}^{(\ell)}$ are set to 0, what gradient components are also 0? Prove your answer.

*(Note: This result tells you that all-*0 *initialization may make it impossible for back-propagation to update some weights.)*

Experiments with Decision Trees and Random Forests

In the following problems, you are asked to implement a preliminary random forest algorithm. You need to implement everything by yourself without using any well-implemented packages.

9. (20 points, *) First, let's implement a simple C&RT algorithm without pruning using the squared error as the impurity measure, as introduced in the class. For the decision stump used in branching, if you are branching with feature i, please sort all the $x_{n,i}$ values to form (at most) $N+1$ segments of equivalent θ, and then pick θ within the median of the segment. If multiple (i, θ) produce the best split, pick the one with the smallest i (and if there is a tie again, pick the one with the smallest θ).

Please run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml23fall/hw6/hw6_train.dat`

and the following file as our test data set for evaluating E_{out}:

`http://www.csie.ntu.edu.tw/~htlin/course/ml23fall/hw6/hw6_test.dat`

The datasets are in LIBSVM format and are processed from

`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/abalone`

What is the $E_{\text{out}}(g)$, where g is the unpruned decision tree returned from your C&RT algorithm and E_{out} is evaluated using the squared error?

10. (20 points, *) Next, we implement the random forest algorithm by coupling bagging (by sampling with replacement) with $N' = 0.5N$ with your unpruned decision tree in the previous problem. You do not need to do random feature selection or expansion. Produce $T = 2000$ trees with bagging. Let $g_1, g_2, \ldots, g_{2000}$ denote the 2000 trees generated. Plot a histogram of $E_{\text{out}}(g_t)$, where E_{out} is evaluated using the squared error.

11. (20 points, *) Let $G(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{x})$ be the random forest formed by the trees above. Plot a scatter plot of $(E_{\text{in}}(g_t), E_{\text{out}}(g_t))$ along with a clear mark of where $(E_{\text{in}}(G), E_{\text{out}}(G))$ is. Describe your findings.

12. (20 points, *) Let $G_t(\mathbf{x}) = \frac{1}{t}\sum_{\tau=1}^{t} g_\tau(\mathbf{x})$ be the random forest formed by the first t trees. Plot the $E_{\text{out}}(g_t)$ as a function of t, and plot $E_{\text{out}}(G_t)$ as a function of t on the same figure. Describe your findings.

Bonus: Crazy XOR

13. (Bonus 20 points) Construct a d-d-1 feed-forward neural network with $\text{sign}(s)$ as the transformation function (such a neural network is also called a Linear Threshold Circuit) to implement $\texttt{XOR}\big((x)_1, (x)_2, \ldots, (x_d)\big)$. Then, prove that it is impossible to implement $\texttt{XOR}\big((x)_1, (x)_2, \ldots, (x_d)\big)$ with any d-$(d-1)$-1 feed-forward neural network with $\text{sign}(s)$ as the transformation function.