hw6-Solution

Graded

1 Day, 23 Hours Late

Student

Chiang Yi Jie

Total Points

201 / 260 pts

Question 1

Problem 1

15 / 20 pts

+ **20 pts** Correct answer with clear explanation

✔ + **15 pts** answer with minor mistake or lack of detail

+ **10 pts** answer with major mistake or lack of most detail

+ **5 pts** answer almost incoorect

+ **0 pts** wrong or no Answer

1 You need to explain this step more

Question 2

Problem 2

10 / 20 pts

+ **20 pts** Answer with a correct and clear proof for the optimal (α, b) (by proving the optimal solution of maximum or minimum)

+ **15 pts** Answer with a correct proof process, but containing errors or lacking detail

✔ + **10 pts** Answer without proof for the optimal (α, b)

+ **0 pts** Answer incorrect or missing

Question 3

Problem 3

15 / 20 pts

Upperbound

✔ **+ 15 pts** Tightest upperbound

+ 10 pts Tightest upperbound with flaws in proof or unclear description

+ 5 pts Reasonable effort

+ 0 pts Wrong upperbound or no upperbound

Construction of maximum $\frac{E_{out}(G)}{E}$ or the proof of maximality

+ 5 pts Correct construction

✔ **+ 0 pts** Wrong construction or no construction

Question 4

Problem 4

20 / 20 pts

✔ **+ 20 pts** Correct

+ 10 pts Find $\left(\frac{N-1}{N}\right)^{\frac{3}{4}N}$

+ 0 pts Incorrect

Question 5

Problem 5

16 / 20 pts

✔ **+ 4 pts** Write out Ɛ1.

✔ **+ 4 pts** Correct u^(2)_n for gn=1

✔ **+ 4 pts** Correct u^(2)_n for gn=-1

✔ **+ 4 pts** Correct process on calculate u^(2)_n

✔ **+ 4 pts** Correct answer.

+ 0 pts No correctness.

✔ **− 2 pts** Partially incorrect or not clear.

✔ **− 2 pts** Partially incorrect or not clear.

Question 6

Problem 6

20 / 20 pts

✔ **+ 10 pts** Correctly write out U_(t+1) by u_n^(t).

+ 5 pts Partially correctly write out U_(t+1) by u_n^(t).

✔ **+ 5 pts** Correctly simplify U_(t+1).

✔ **+ 5 pts** Correct conclusion.

+ 0 pts No solution, wrong page selected, wrong answer.

− 3 pts Unclear process.

− 3 pts Unclear process.

Question 7

Problem 7

20 / 20 pts

✔ **+ 8 pts** Write correct form of η.

+ 4 pts Partially correct form of η.

✔ **+ 6 pts** Write out relation between s^(t)_n and s^(t-1)_n.

✔ **+ 6 pts** Correct conclusion.

+ 0 pts No correctness.

− 3 pts Unclear process or partially wrong.

+ 0 pts Unclear process.

Question 8

Problem 8

5 / 20 pts

+ 20 pts Correct. $\frac{\partial e}{\partial w_{0j}^{(L)}} \neq 0$ when $\frac{\partial e}{\partial x_j^{(L)}} \neq 0$ for $1 \leq j \leq d^{(L)}$ The rest are zero.

+ 15 pts Minor mistake

+ 10 pts Mistake

✔ **+ 5 pts** Reasonable effort

+ 0 pts Wrong or blank

Question 9

Problem 9

20 / 20 pts

✔ **+ 20 pts** Correct answer within reasonable range.

+ 10 pts An answer out of reasonable range.

+ 0 pts Wrong or blank.

− 2 pts Wrong selected page.

Question 10

Problem 10

20 / 20 pts

✔ **+ 20 pts** The histogram is on the right trend.

+ 10 pts The histogram is on a weird trend.

− 5 pts Missing numbers or numbers are out of reasonable range.

+ 0 pts Wrong or blank.

− 2 pts Wrong selected page.

Question 11

Problem 11

20 / 20 pts

✔ **+ 20 pts** Correct answer

Partially correct.

+ 5 pts Reasonable g_t scatter trend.

+ 5 pts Reasonable and correct distribution range.

+ 5 pts Reasonable relation between G and g_t in the graph.

+ 5 pts Reasonable and correct findings.

+ 0 pts Wrong answer or blank.

Question 12

Problem 12

20 / 20 pts

✔ **+ 20 pts** Correct answer

Partially correct.

- **+ 5 pts** Correct $E_{out}(g_t)$ plot.
- **+ 5 pts** Correct $E_{out}(G_t)$ trend.
- **+ 5 pts** Correct distribution range of $E_{out}(G_t)$.
- **+ 5 pts** Reasonable and correct findings.

- **+ 0 pts** Wrong answer or blank.
- **− 2 pts** Generally correct, but without training the full 2000 trees.
- **− 2 pts** Wrong selected page.

Question 13

Problem 13

0 / 20 pts

- **− 0 pts** Totally correct, no logical flaw.
- **− 5 pts** A minor logical flaw exists.
- **− 6 pts** Without implementing XOR() with d-d-1 FFN in detail.
- **− 10 pts** Without implementing or implementing XOR() with d-d-1 FFN wronly.
- **− 10 pts** Do not prove or prove d-(d-1)-1 FFN is impossible for implementation wronly.

✔ **− 20 pts** Wrong answer

No questions assigned to the following page.

Question assigned to the following page: 1

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:

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

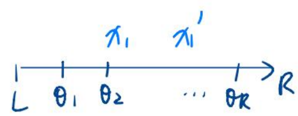
When studying kernel methods, we mentioned that the kernel is simply a computational short-cut for the inner product $(\boldsymbol{\phi}(\mathbf{x}))^T(\boldsymbol{\phi}(\mathbf{x}'))$. In this problem, we mix the two topics together using the decision stumps as our $g_t(\mathbf{x})$.

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

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

Define $\boldsymbol{\phi}_{ds}(\mathbf{x}) = \left(g_{+1,1,\theta_1}(\mathbf{x}), g_{+1,1,\theta_2}(\mathbf{x}), \ldots, g_{+1,1,\theta_k}(\mathbf{x}), \ldots, g_{-1,d,\theta_k}(\mathbf{x})\right)$. 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 this early work



$k = R - L$

$\phi(x) . (g_1(x) . g_2(x) . g_3(x) \cdots g_T(x))$

$g_{s,i,\theta}(x) = s \cdot \text{sign}(x_i - \theta)$

$K_{ds}(x, x') = \phi_{ds}(x)^T \phi_{ds}(x')$

$= \sum_{i=1}^{d} \sum_{n=1}^{k} \underline{g_{+1,i,n}(x) g_{+1,i,n}(x)} + i$

$g_{s,i,\theta_n}(x) g_{s,i\theta_n}(x') = [s \cdot \text{sign}(x_i - \theta_n)] \times [s \cdot \text{sign}(x_i' - \theta_n)]$

$= \text{sign}(x_i - \theta_n) \cdot \text{sign}(x_i' - \theta_n)$

$= 2 \cdot \sum_{i=1}^{d} \sum_{n=1}^{k} \text{sign}(x_i - \theta_n) \text{sign}(x_i' - \theta_n)$

$= 2 \sum_{i=1}^{d} \left((R-L) - 2(x_i - x_i') \right) = 2d(R-L) - 4 \sum_{i=1}^{d} (x_i - x_i')$ #

Question assigned to the following page: 2

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.)

$\tilde{C} = \frac{C}{u}$. $\tilde{K} = uk(x, x') + v$

$b = y_s - \sum_{SV} \alpha_n y_n K(x_n, x_s)$, $0 < \alpha_n < C$

$\sum_{n=1}^{N} \alpha_n y_n = 0$

$\tilde{K} = \min_{\alpha} \frac{1}{2} \sum_{n=1}^{N} \sum_{n=1}^{N} \alpha_n \alpha_m y_n y_m \left(uk(x_m, x_n) + v \right) - \sum_{n=1}^{N} \alpha_n$, subject to $0 \leq \alpha_n < \frac{C}{u}$

$g_{svm}(x) = \text{sign}\left(\sum_{SV} \frac{\alpha_n}{u} y_n \tilde{K}(x_n, x) + b \right)$

$= \text{sign}\left(\sum_{SV} \frac{\alpha_n}{u} y_n \left(u \times K(x_n, x) + v + b \right) \right)$

$= \text{sign}\left(\sum_{SV} \alpha_n y_n K(x_n, x) + y_s - \sum_{SV} \alpha_n y_n K(x_n, x_s) \right)$

$= \text{sign}\left(\sum_{SV} \alpha_n y_n K(x_n, x) + b \right) \rightarrow$ eqvivelant g_{svm}.

Question assigned to the following page: 3

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.)

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

for $y=+1$, there must be more g_t that gives wrong prediction than g_t that have right prediction

at least 9 g_t classify as -1 to $E_{out}(G)=1$

$E = \sum_{t=1}^{17} E_{out}(g_t) \geq 9$, $\max\left(\frac{E_{out}(G)}{E}\right) = \frac{1}{9}$ #

Question assigned to the following page: 4

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.))

$$\lim_{N\to\infty} \left(1-\frac{1}{N}\right)^{\frac{3}{4}N} = \lim_{N\to\infty} \left(\frac{N-1}{N}\right)^{\frac{3}{4}N}$$

$$= \lim_{N\to\infty} \frac{1}{\left(\frac{N}{N-1}\right)^{\frac{3}{4}N}}$$

$$= \lim_{N\to\infty} \frac{1}{\left(1+\frac{1}{N-1}\right)^{\frac{3}{4}N}} \qquad \lim_{N\to\infty} \left(1+\frac{1}{N}\right)^N = e.$$

$$= \lim_{N\to\infty} \left(\frac{1}{\left(1+\frac{1}{N-1}\right)^{\frac{3}{4}N}}\right) \quad , \quad \lim_{N\to\infty} \left(\frac{1}{1+\frac{1}{N}}\right)^N = \frac{1}{e}$$

$$\parallel$$

as $N\to\infty$
$N-1\to\infty$ $\qquad \left(\frac{1}{e}\right)^{\frac{3}{4}}$ #

Question assigned to the following page: 5

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.)

prediction :

If all correct → $y_n > 0$

If all wrong → $y_n < 0$

→ $y_n > 0$, weighted correct rate 2%

$$\frac{\sum_n : y_n>0\ u_n^{(2)}}{\sum_n : y_n<0\ u_n^{(2)}} = \frac{\sum_n y>0\ u_n^{(1)} \times 0.02}{\sum_n y<0\ u_n^{(1)} \times 0.98} = 1 \ \#$$

↘ $y_n < 0$, weighted incorrect rate 98%

Question assigned to the following page: 6

6. (20 points) For the AdaBoost algorithm introduced in Lecture 212, 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.)

$$U_t \varepsilon_t = \sum u_n^{(t)} [\![y_n \neq g_t(x_n)]\!]$$

$$U_t (1-\varepsilon_t) = \sum u_n^{(t)} [\![y_n = g_t(x_n)]\!]$$

$$U_{t+1} = U_t \cdot \varepsilon_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} + (U_t - U_t \varepsilon_t) \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}$$

$$= U_t \left(\varepsilon_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} + (U_t - U_t \varepsilon_t) \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}\right.$$

$$= U_t \left(\varepsilon_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} + (1-\varepsilon_t) \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \right) = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}\, U_t$$

since $U_{t+1} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}\, U_t$

$$U_{t+2} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}\, U_{t+1} = \left(2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right)^2 U_{t+1}$$

$$U_{t+3} = \left(2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right)^3 U_t$$

$$\vdots$$

$$U_{t+T} = \left(\left(2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right)^T \right) U_t$$

$t=1$, $\dfrac{U_{T+1}}{U_1} = 2^T \prod_{t=1} \sqrt{\varepsilon_t(1-\varepsilon_t)}$ #

Question assigned to the following page: 7

7. (20 points) For the gradient boosted decision tree, after updating all s_n in iteration t using the steepest η as α_t, what is the value of $\sum_{n=1}^{N} s_n g_t(\mathbf{x}_n)$? Prove your answer.

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

$$E = \min_{\eta} \frac{1}{N} \sum_{n=1}^{N} \left((y_n - s_n) - \eta g_t(x_n) \right)^2$$

$$\frac{\partial E}{\partial \eta} = \frac{1}{N} \sum_{n=1}^{N} 2 \left((y_n - s_n) - \eta g_t(x_n) \right) \left(-g_t(x_n) \right) = 0$$

$$\Rightarrow \sum_{n=1}^{N} (y_n - s_n) g_t(x_n) - \eta \sum_{n=1}^{n} (g_t(x_n))^2 = 0 \quad , \quad s_n \leftarrow s_n + \eta g_t(x_n)$$

$$\sum_{n=1}^{N} (y_n - s_n) g_t(x_n) = \sum_{n=1}^{N} (y_n - s_n - \eta g_t(x_n))(g_t(x_n))$$

$$= \sum_{n=1}^{N} (y_n - s_n) g_t(x_n) - \eta \sum_{n=1}^{N} (g_t(x_n))^2 = 0 \ \#$$

Question assigned to the following page: 8

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.)

$$x_j^{(1)} = \tanh\left(\Sigma_i W_{ij}^{(1)} x_i^{(0)}\right) = \tanh(0) = 0$$

$$x_k^{(2)} = \tanh\left(\Sigma_j W_{jk}^{(2)} x_j^{(1)}\right) = \tanh(0) = 0$$

$$\vdots$$

$$g_{NN}(x) = \left(\cdots \tanh\left(\Sigma_j W_{jk}^{(2)} \tanh\left(\Sigma_i W_{ij}^{(1)} \cdot x_i^{(0)}\right)\right) = 0\right.$$

The output will be be 0.

$$\frac{\partial e_n}{\partial w_{ij}^{(\ell)}} = \frac{\partial e_n}{\partial s_j^{(\ell)}} \cdot \frac{\partial s_j^{(\ell)}}{\partial w_{ij}^{(\ell)}} = s_j^{(\ell)} \cdot \underline{w_{ij}^{(\ell-1)}} = 0 \quad , \quad w_{ij}^{(\ell-1)} = 0$$

$$w_{ij}^{(\ell)} \leftarrow w_{ij}^{(\ell)} - \eta\, x_i^{(\ell-1)} \delta_j^{(\ell)} \quad , \quad x_i^{(\ell-1)} = 0$$

$$\Rightarrow w_{ij}^{(\ell)} \leftarrow w_{ij}^{(\ell)} \quad \Rightarrow \text{weight always } 0 \ \#$$

Question assigned to the following page: 9

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

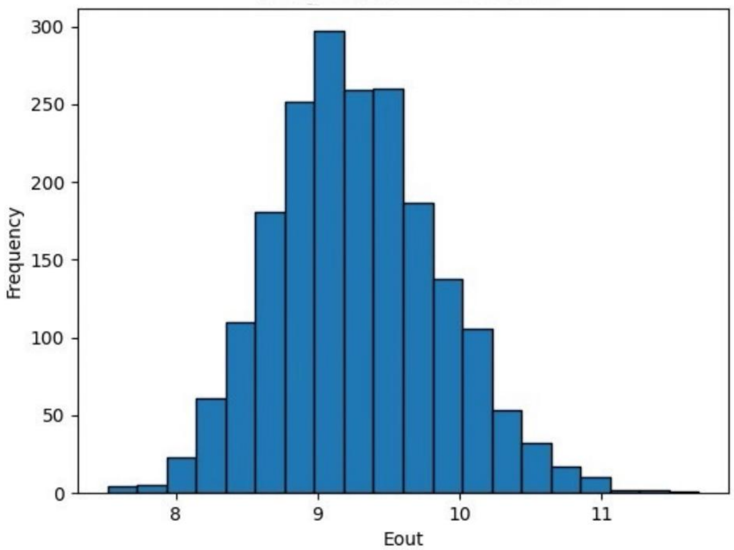
`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/abalone`

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?

```
Eout(g): 8.79324462640737
```

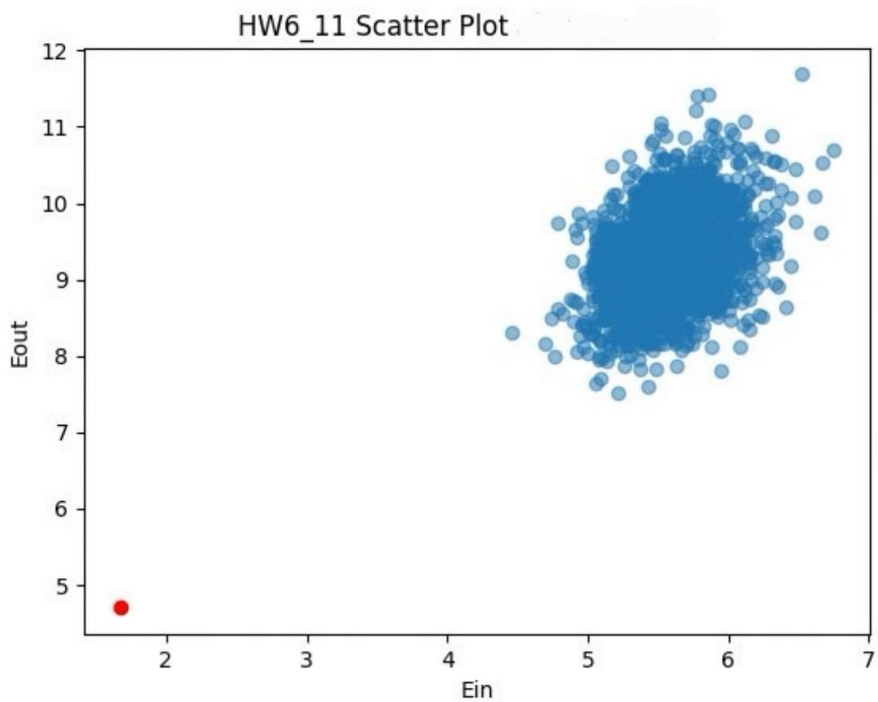
Question assigned to the following page: 10

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.



Question assigned to the following page: 11

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.

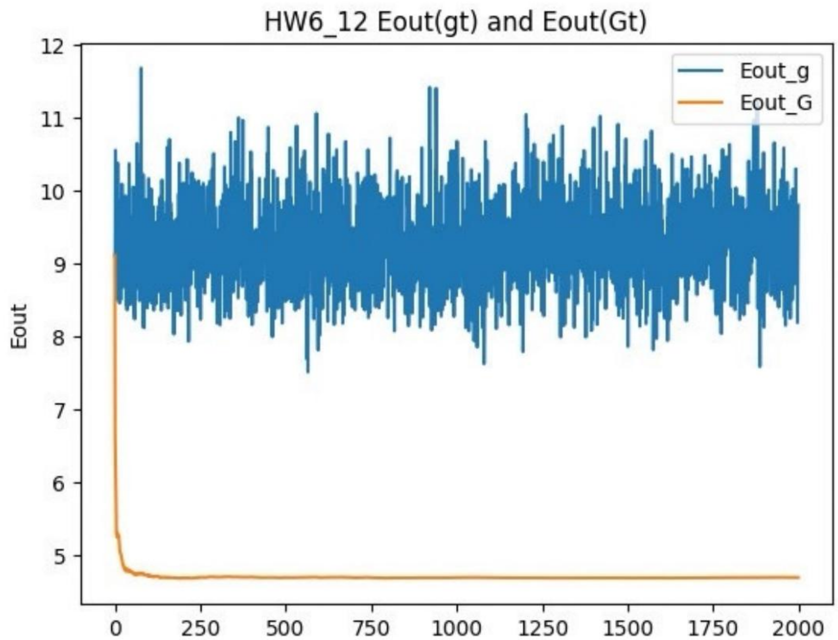


Through this plot, we can clearly find out that Ein(G) >> Ein(g), indicating that forest really reduce training data error . In concern, we should make sure that this algorithm may overfit.

However, Eout(G) is extremely smaller than Eout(g), which means that it will also act better on test data.

Question assigned to the following page: 12

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.



In the beginning, G act similiarly as g, they both make errors in same situation.
However,after adding more g into forest, Gout becoming more stable and produce less errors on testing data.
As we can see, as time go longer, the decay on errors slower than beginning.
At t=2000, stable errors probably stop at 2~3, comparing beginning error~10, verified that forest improve the whole algorithm.