Homework #4

Answer Sheet

DEADLINE: 06/20/2017, 14:00
INSTRUCTOR: Hsuan-Tien Lin

王冠鈞 b03902027

1. The probability that an example $\mathbf{x}_i$ is not sampled from bootstrapping is:

$$P(\mathbf{x}_i \text{ is not sampled}) = \left(\frac{N-1}{N}\right)^{N'}$$

When N is large, it is equivalent to[1]:

$$\lim_{N\to\infty}\left(\frac{N-1}{N}\right)^{N'} = \left(\lim_{N\to\infty}\left(1+\frac{-1}{N}\right)^{N}\right)^{p} = e^{-p}$$

Thus the (expected) number of examples that are not sampled is $N \cdot e^{-p}$.

2. Since $G = \text{Uniform}(\{g_t\})$, for any example, there must be more than a half of trees that will mis-classify it to contribute to an error of G. In this question, there are 3 trees, so an example will contribute to $E_{\text{out}}(G)$ if and only if there are at least 2 trees make mistake on it. Thus, the lowest $E_{\text{out}} = 0$, when all mistakes from g_1, g_2, g_3 are disjointed (and it's possible to perfectly disjoint since $\sum_{i=1}^{3} E_{\text{out}}(g_i) = 0.75 < 1$). That is, define $E_{\text{out}}^k = \{\mathbf{x}_i | [\![y_i \neq g_k(\mathbf{x}_i)]\!]\}, k = 1,2,3$ and $E_{\text{out}}^G = \{\mathbf{x}_i | [\![y_i \neq G(\mathbf{x}_i)]\!]\}$, then for any $i, j (1 \leq i, j \leq 3, i \neq j)$, $E_{\text{out}}^i \cap E_{\text{out}}^j = \emptyset \Leftrightarrow E_{\text{out}}^G = \emptyset \Leftrightarrow E_{\text{out}}(G) = 0$. Oppositely, the maximum $E_{\text{out}}(G)$ occurs when all error examples are in exactly two of the three E_{out}^k. This will lead to the upper bound of $E_{\text{out}}(G) = \frac{1}{2}(0.15 + 0.25 + 0.35) = 0.375$. Combine the results together, we get the range of $E_{\text{out}}(G)$:

$$0 \leq E_{\text{out}}(G) \leq 0.375$$

3. Reusing the definition and the concept in the previous question, if an example $\mathbf{x}_i \in E_{\text{out}}^G$, then there must be at least $\frac{K+1}{2}$ (more than a half) different k such that $\mathbf{x}_i \in E_{\text{out}}^k$. Thus the upper bound of E_{out} is

$$E_{\text{out}}(G) \leq \frac{\sum_{k=1}^{K} e_k}{\frac{K+1}{2}} = \frac{2}{K+1}\sum_{k=1}^{K} e_k$$

, in this case $\forall \mathbf{x}_i \in E_{\text{out}}^G$, there exist exactly $\frac{K+1}{2}$ trees that make mistake on $\mathbf{x}_i$.

4. The new s_n, say it $s_n^{(1)}$, is updated from old s_n, say it $s_n^{(0)}$. According to the algorithm, we

[1] Since $\exp(x) = \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n$

know that $s_n \leftarrow s_n + \alpha_1 g_1(\mathbf{x}_n)$. The new $s_n(s_n^{(1)})$ is derived as the following:

$$\begin{aligned} s_n^{(1)} &= s_n^{(0)} + \alpha_1 g_1(\mathbf{x}_n) \\ &= 0 + \text{OneVarLinearRegression}(\{(g_t(\mathbf{x}_n), y_n - 0)\}) \\ &= \frac{\sum_{n=1}^N g_1(\mathbf{x}_n)(y_n - s_n)}{\sum_{n=1}^N g_1^2(\mathbf{x_n})} \cdot g_1(\mathbf{x}_n) \\ &= \frac{\sum_{n=1}^N 2(y_n - 0)}{\sum_{n=1}^N 4} \cdot 2 \\ &= \frac{\sum_{n=1}^N 2y_n}{4N} \cdot 2 \\ &= \frac{1}{N}\sum_{n=1}^N y_n \end{aligned}$$

5. The value α_t can be represented as a closed-form formula:

$$\alpha_t = \eta^* = \frac{\sum_{n=1}^N g_t(\mathbf{x}_n)(y_n - s_n^{(t-1)})}{\sum_{n=1}^N g_t^2(\mathbf{x}_n)}$$

, where $s_n^{(t-1)}$ is the value of s_n in the $(t-1)$-th iteration. We can arrange it into the following:

$$\begin{aligned} & \alpha_t \cdot \sum_{n=1}^N g_t^2(\mathbf{x}_n) = \sum_{n=1}^N g_t(\mathbf{x}_n) y_n - \sum_{n=1}^N g_t(\mathbf{x}_n) s_n^{(t-1)} \\ \Rightarrow \quad & \sum_{n=1}^N g_t(\mathbf{x}_n) \cdot (s_n^{(t-1)} + \alpha_t g_t(\mathbf{x}_n)) = \sum_{n=1}^N g_t(\mathbf{x}_n) y_n \\ \overset{\because s_n^{(t)} = s_n^{(t-1)} + \alpha_t g_t(\mathbf{x}_n)}{\Rightarrow} \quad & \sum_{n=1}^N g_t(\mathbf{x}_n) \cdot s_n^{(t)} = \sum_{n=1}^N g_t(\mathbf{x}_n) y_n \end{aligned}$$

And we directly get $\sum_{n=1}^N s_n g_t(\mathbf{x}_n) = \sum_{n=1}^N y_n g_t(\mathbf{x}_n)$ from above.

6. *Proof.* In the first step in the 1st iteration ($s_n = 0$), we have found a g_1 such that $\frac{1}{N}\sum_{n=1}^N (y_n - g_1(\mathbf{x}_n))^2$ (squared error of linear regression) is the smallest. In the second step, the algorithm will solve the following optimization problem:

$$\min_{\alpha_1} \quad \frac{1}{N}\sum_{n=1}^N ((y_n - 0) - \alpha_1 g_1(\mathbf{x}_n))^2$$

Note that this optimization problem has the same form of the previous step, and we have known that g_1 is the optimal function for the first step. In other words, we can use the result from the first step to directly solve this problem, and the optimal condition is:

$$g_1(\mathbf{x}_n) = \alpha_1 g_1(\mathbf{x}_n)$$

The condition holds only when $\alpha_1 = 1$. Thus the statement holds.

□

7. *Proof.* From question 6, we get $\alpha_1 = 1$. Then the new $s_n = s_n^{(1)} = s_n^{(0)} + 1 \cdot g_1(\mathbf{x}_n) = g_1(\mathbf{x}_n)$. Since g_1 is derived from linear regression, after recalling from the Lecture 9 of ML Foundations[2], $g_1(\mathbf{x}_n) = (X^\dagger \mathbf{y})^T \mathbf{x}_n$, where the symbols are defined in the linear regression lecture.

Now in the 2nd iteration, the algorithm will find g_2, which is the optimal solution of LinearRegression($\{(\mathbf{x}_n, y_n - g_1(\mathbf{x}_n)\}$). According to the linear algorithm in MLF Lecture 9, the solution can be derived below:

$$\begin{aligned}
g_2(\mathbf{x}_n) = (\mathbf{w}_{\mathsf{LIN}}^{(2)})^T \mathbf{x}_n &= \left(X^\dagger(\mathbf{y} - g_1(\mathbf{x}))\right)^T \mathbf{x}_n \\
&= \left(X^\dagger(\mathbf{y} - ((X^\dagger \mathbf{y})^T X^T)^T)\right)^T \mathbf{x}_n \\
&= \left(X^\dagger(\mathbf{y} - XX^\dagger \mathbf{y})\right)^T \mathbf{x}_n \\
&= \left((X^TX)^{-1}X^T(\mathbf{y} - X(X^TX)^{-1}X^T\mathbf{y})\right)^T \mathbf{x}_n \\
&= \left((X^TX)^{-1}X^T\mathbf{y} - (X^TX)^{-1}(X^TX)(X^TX)^{-1}X^T\mathbf{y}\right)^T \mathbf{x}_n \\
&= \left((X^TX)^{-1}X^T\mathbf{y} - (X^TX)^{-1}X^T\mathbf{y}\right)^T \mathbf{x}_n \\
&= \mathbf{0}^T \cdot \mathbf{x}_n \\
&= 0
\end{aligned}$$

Thus the statement holds.

□

8. The `OR` operation means that any of $x_1, x_2, \cdots, x_d$ is TRUE returns TRUE, i.e. only when all x_i are FALSE will make $\sum_{i=0}^{d} w_i x_i < 0$. To achieve that, we can set $w_0 = d - 1$ and $w_i = +1 (i \in 1, 2, \cdots, d)$, (i.e. $(w_0, w_1, w_2, \cdots, w_d) = (d-1, +1, +1, \cdots, +1)$) such that only when $x_1 = x_2 = \cdots = x_d = -1$ will make $\sum_{i=0}^{d} w_i x_i = -1 < 0$.

9. The `XOR` operation has the property: $\texttt{XOR}(x_1, x_2, x_3) = \texttt{XOR}(\texttt{XOR}(x_1, x_2), x_3)$. In this question, there are 5 inputs, and the `XOR` operation will return TRUE when there are exactly odd number of +1, and will return FALSE otherwise. The least number to implement a 5-input `XOR`, in my opinion is $D = 5$. To implement such a 5-5-1 NNet, we can define the meaning the nodes in the middle layer of the neuron net: *the i-th (except the 0-th, which is always +1) node in the middle layer denotes that whether there are at least* $(6 - i)$ *inputs with value* $= +1$. With such definition, the weight of the first layer can be defined:

$$\begin{aligned}
w_{i1}^{(1)} &= (-5, +1, +1, +1, +1, +1) \\
w_{i2}^{(1)} &= (-4, +1, +1, +1, +1, +1) \\
w_{i3}^{(1)} &= (-3, +1, +1, +1, +1, +1) \\
w_{i4}^{(1)} &= (-2, +1, +1, +1, +1, +1) \\
w_{i5}^{(1)} &= (-1, +1, +1, +1, +1, +1)
\end{aligned}$$

, and the second layer:

$$w_{i1}^{(2)} = (-1, +1, -1, +1, -1, +1)$$

[2] `http://www.csie.ntu.edu.tw/~htlin/mooc/doc/09_handout.pdf#12`, the linear regression algorithm

If there are odd (even) number of positive inputs, then the second layer will make it become positive and return TRUE (FALSE).

10. We already know:

$$\begin{aligned} \frac{\partial e_n}{\partial w_{ij}^{(l)}} &= \delta_j^{(l)} \cdot \left(x_i^{(l-1)}\right) \\ \text{where } \delta_j^l &= -2\left(y_n - s_1^{(L)}\right) && \text{if } l = L \\ \delta_j^l &= \sum_k \left(\delta_k^{(l+1)}\right)\left(w_{jk}^{(l+1)}\right)\left(\tanh\left(s_j^{(l)}\right)\right) && \text{if } 1 \le l < L \end{aligned}$$

And from this problem, all initial weights w_{ij}^l are 0, which will make the term $\sum_k \left(\delta_k^{(l+1)}\right)\left(w_{jk}^{(l+1)}\right)\left(\tanh\left(s_j^{(l)}\right)\right) = \sum_k \left(\delta_k^{(l+1)}\right)(0)\left(\tanh\left(s_j^{(l)}\right)\right) = 0$. That is, for $1 \le l < L, \delta_j^{(l)} = 0 \Rightarrow \frac{\partial e_n}{\partial w_{ij}^{(l)}} = 0 \cdot \left(x_i^{(l-1)}\right) = 0$.

Now that we know when $1 \le l < L$, the all the gradient components $\frac{\partial e_n}{\partial w_{ij}^{(l)}} = 0$. We now examine what will lead to when $l = L$. We know that $\frac{\partial e_n}{\partial w_{ij}^{(L)}} = -2\left(y_n - s_1^{(L)}\right) \cdot \left(x_i^{(L-1)}\right)$, which has no weights that will directly lead it to 0. However, note that the term $\left(x_i^{(L-1)}\right)$ here, which denotes the hyperbolic tangent of the score to the neurons in the previous layer, and since

$$\text{for } i \neq 0, x_i^{(L-1)} = \tanh(s_i^{(L-1)}) = \tanh(\sum_{j=0}^{d^{(L-2)}} w_{ji}^{(L-1)} x_j^{(L-2)}) = \tanh(0) = 0$$

, which are also 0. However, note that when $i = 0$, $x_i^{(L-1)} = +1$ (according to the neural network model), and $\frac{\partial e_n}{\partial w_{01}^{(L)}} = -2\left(y_n - s_1^{(L)}\right)$, which is not necessarily 0 (depend on $[\![y_n = s_1^{(L)}]\!]$). For all the others, $\frac{\partial e_n}{\partial w_{ij}^{(l)}} = 0$.

11. *Proof.* In the first step (stochastic), it just alter the input $\mathbf{x}$, and is nothing to do with the proof here.

In the second step (forward), it will compute all $x_j^{(1)} = \tanh\left(\sum_{i=1}^{d^{(0)}} w_{ij}^{(1)} x_i^{(0)}\right)$. Since all $w_{ij}^{(1)}$ now is 1, it's apparent that $x_1^{(1)} = x_2^{(1)} = \cdots = x_{d^{(1)}}^{(1)} \neq x_0^{(1)} = +1$.

In the third step (backward), it will perform backpropagation and compute $\delta_j^{(l)}$. it will first compute $\delta_1^{(2)} = -2\left(y_n - \sum_{i=1}^{d^{(1)}} w_{i1}^{(2)} x_i^{(1)}\right)$. Then it will compute $\delta_i^{(1)} = \left(\delta_1^{(2)}\right)\left(w_{i1}^{(2)}\right)\left(\tanh'\left(s_i^{(1)}\right)\right)$. We know that $\delta_1^{(2)}$ is fixed here, as shown above, and all $w_{i1}^{(2)} = 1$, and we further notice that since $s_i^{(1)} = \sum_{i=1}^{d^{(0)}} w_{ij}^{(1)} x_i^{(0)}$, with the same reason of the previous step, we get $s_1^{(1)} = \cdots = s_{d^{(1)}}^{(1)}$ and thus $\delta_1^{(1)} = \delta_2^{(1)} = \cdots = \delta_{d^{(1)}}^{(1)}$.

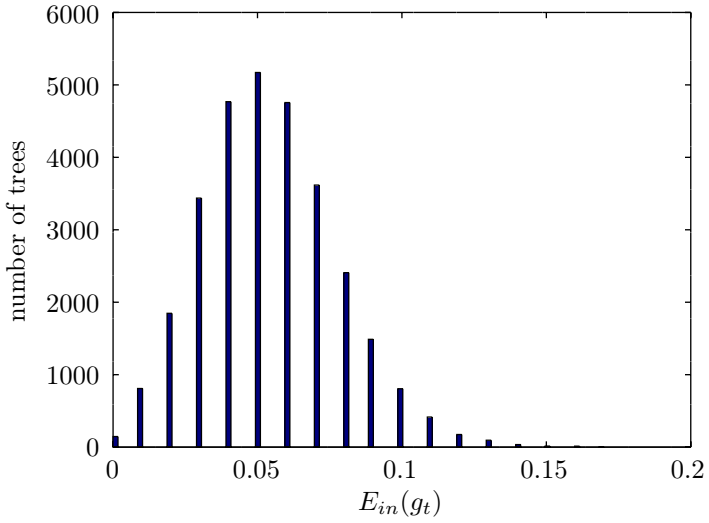
In the fourth step (gradient descent), it will perform $w_{ij}^{(1)} \leftarrow w_{ij}^{(1)} - \eta x_i^{(0)} \delta_j^{(1)}$. For $0 \le i \le d^{(0)}$ and for $1 \le j < d^{(1)}$, we have initially $w_{ij}^{(1)} = w_{i(j+1)}^{(1)} = 1$ and from the previous step we

have $\delta_j^{(1)} = \delta_{(j+1)}^{(1)}$. Thus after gradient descent, we have the new $w_{ij}^{(1)} = 1 - \eta x_i^{(0)} \delta_j^{(1)} = 1 - \eta x_i^{(0)} \delta_{(j+1)}^{(1)} = w_{i(j+1)}^{(1)}$. Thus the statement holds.

□

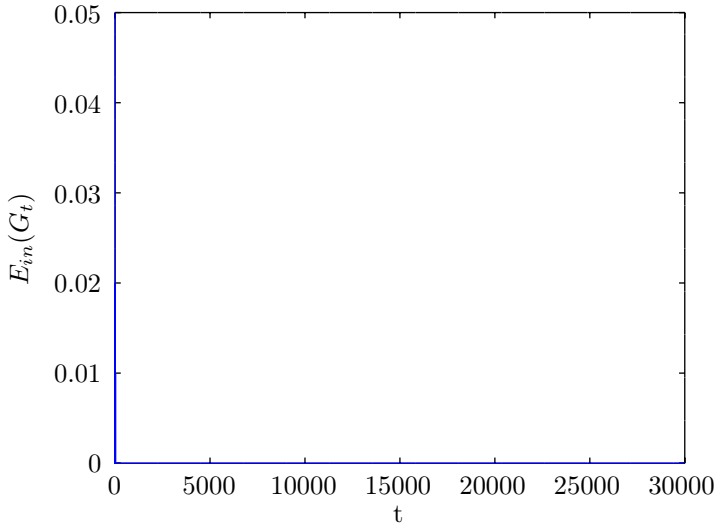
12. Here is the histogram, where the x axis is $E_{\text{in}}(g_t)$, and the y axis denotes the number of trees that lead to such E_{in}.

Result of Question 12

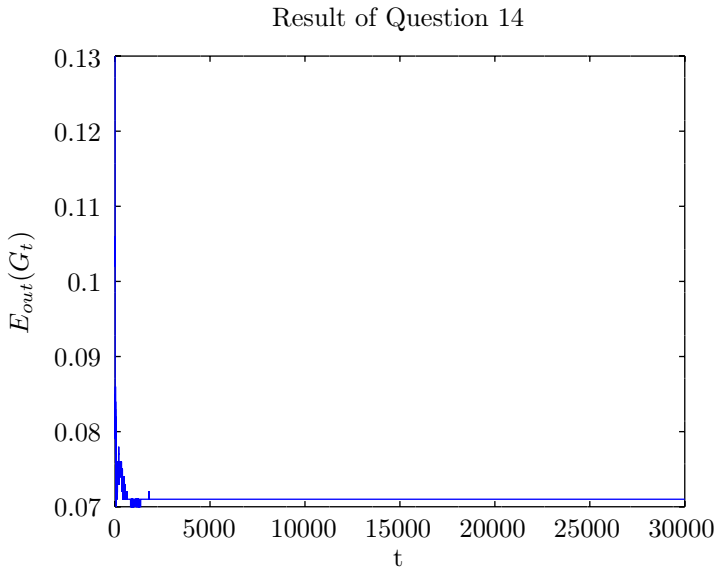


13. The relation between t and $E_{\text{in}}(G_t)$ is shown below.

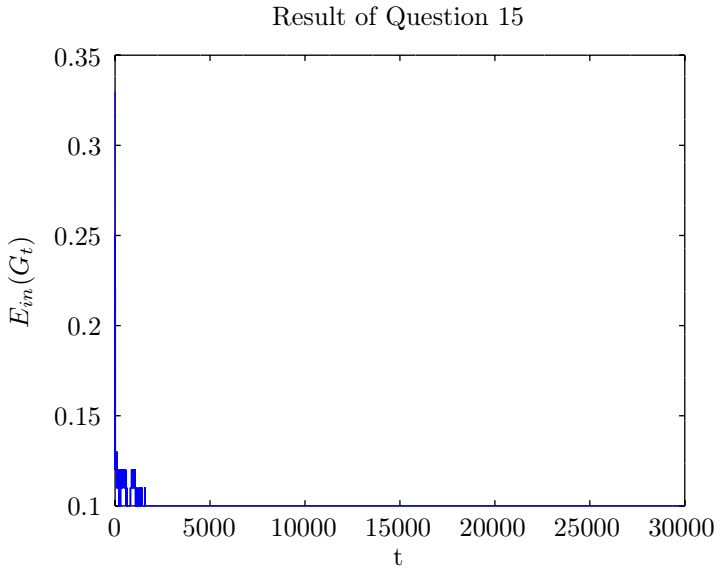
Result of Question 13



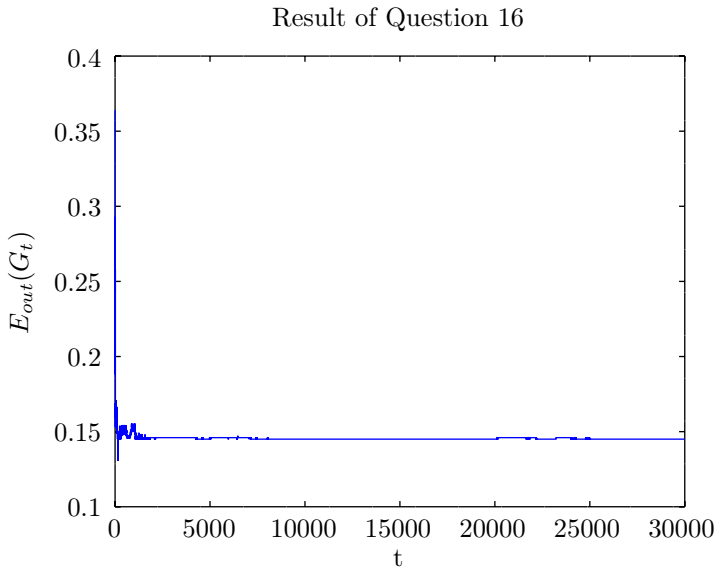
14. The relation between t and $E_{\text{out}}(G_t)$ is shown below. As we can see, when t is large, both E_{in} and E_{out} will end up being constant. The difference is that the terminal E_{in} is zero, while the terminal E_{out} is 0.071.



15. The relation between t and $E_{\text{in}}(G_t)$ is shown below.



16. The relation between t and $E_{\text{out}}(G_t)$ is shown below. Similar with Questions 13-14, both E_{in} and E_{out} will approach a constant when t is large: E_{in} will end up with 0.1, while E_{out} will end up with 0.145 (small difference may occur).


Result of Question 16
0.4
0.35
0.3
0.25
0.2
0.15
0.1
$E_{out}(G_t)$
0
5000
10000
15000
20000
25000
30000
t