HTML_2023_HW2

tags: `Personal`

```
Discuss with anonymous and TAs.
```

Theory of Generalization

P1

Define a perceptron $h(x) = \text{sign}(w^T x) \text{ s.t. } w_1(x_1 - 11.26) + w_2(x2 - 62.11) = -w_0, \forall x \in \mathbb{R}^2$ Consider a two-dimensional plane filled with numerous points. Visualize a line that gradually rotates around the fixed lucky point $(11.26, 62.11)$. Since the line passes over each point on the plane, the classification of that point changes, thereby creating a new state. After the line completes a full rotation around the lucky point, it crosses over a total of $2N$ points, which creates $2N$ distinct states.

As a result, we should choose $[d]$ as our solution.

P2

Consider the hypothesis set $h_m(x) = \text{sign}(w_m^T x), m = 1, 2, \cdots, 1126$ to be 1126 binary classifiers. In order to shatter all possible outputs for any combination of n different points, we need 2^n kinds of different results. Since we have hypothesis set $h_m(x) = \text{sign}(w_m^T x), m = 1, 2, \cdots, 1126$, there can exist at most 1126 outputs. Thus, we need to find the largest n such that $2^n = m_H(d) \leq 1126$. Solving for n, we get $n = \log_2(1126)$.

As a result, we should choose $[a]$ as our solution.

P3

[a]

By sorting the possible result, a key limitation of the problem is that the sequence $\{\text{o,x,o,x,o}\}$ can never occur when the values of the wrong sign $\{\text{x}\}$ are sorted in increasing order, which means $d_{vc} = 4$.

[b]

By sorting the possible result, a key limitation of the problem is that the sequence $\{1, -1, 1, -1, 1\}$ can never occur since the polynomial of degree 3 only pass $\sum_{i=0}^{3} w_i x^i = y = 0$ for three times, which means $d_{vc} = 4$.

[c]

For any $n \in \mathbb{N}$, consider the set of points $\{x_1, x_2, \ldots, x_n\}$ with arbitrary labels $\{y_1, y_2, \ldots, y_n\} \in \{-1, +1\}^n$. Then let

$$x_j = 1126^{-j}$$
$$w = \pi(1 + \sum_{i=1}^{n} 1126^i y_i'), \text{ where } y_i' = \frac{1 - y_i}{2}$$

For any $j \in [1, n]$ we have

$$h(x_j) = \sin\left(1126^{-j} \times \pi(1 + \sum_{i=1}^{n} 1126^{i} y_i^{'}\right) \qquad \Rightarrow \sin\left(1126^{-j} \times \pi(1 + \sum_{i=1}^{n} \frac{1126^{i}(1-y_i)}{2}\right))$$

$$= \sin\left(\pi(1126^{-j} + (\sum_{i=1}^{n} 1126^{i-j} y_j^{'}))\right) \qquad \Rightarrow \sin\left(\pi(1126^{-j} + (\sum_{i=1}^{n} \frac{1126^{i-j}(1-y_i)}{2}))\right)$$

$$\Rightarrow \sin\left(\pi(1126^{-j} + \sum_{i:i<j, y_i=-1} \frac{1126^{i-j}(1-y_i)}{2}\right.$$

$$\Rightarrow \sin\left(\pi(1126^{-j} + \frac{(1-y_i)}{2} + \sum_{i:i<j, yi=-1} 2 \times\right.$$

$$= \sin\left(\pi(1126^{-j} + (\sum_{i=1}^{j-1} 1126^{i-j} y_i^{'}) + y_j^{'} + (\sum_{i=1}^{n-j} 1126^{i} y_i^{'}))\right) \qquad \Rightarrow \sin\left(\pi(1126^{-j} + \frac{(1-y_i)}{2} + \sum_{i:i<j, yi=-1} 112\right.$$

For any $y_i = 1$ and any $i < j$, the last term can be dropped from the sum since it only contributes multiples of 2π, which causes no change in value. By using the fact that $\sin(\pi + x) = -\sin(x)$, it is trivial that the summation in the last term and the second term is always less than 1. Since $y_i^{'}$ the remaining term, that is

$$\sin\left(\pi(1126^{-j} + (\sum_{i=1}^{j-1} 1126^{i-j} y_i^{'}) + y_j^{'})\right) = \sin\left(\pi(\sum_{i=1}^{j-1} 1126^{-i} y_i^{'} + 1126^{-j} + y_j^{'})\right)$$

can be upper and lower bound as follows

$$\pi(\sum_{i=1}^{j-1} 1126^{-i} y_i^{'}) + 1126^{-j} + y_j^{'} \leq \pi(\sum_{i=1}^{j-1} 1126^{-j} + y_j^{'}) \leq \pi(1 + y_j^{'})$$
$$\pi(\sum_{i=1}^{j-1} 1126^{-i} y_i^{'}) + 1126^{-j} + y_j^{'} > \pi y_j^{'}$$

$$\therefore \begin{cases} h(x_j) = 1 = y_j, & \text{where } 0 < wx_j < \pi \quad \text{if } y_j = 1 \\ h(x_j) = -1 = y_j, & \text{where } \pi < wx_j < 2\pi \quad \text{if } y_j = -1 \end{cases}$$

As a result, $h(x_j) = y_j$ for all j, and the set $\{1126^{-1}, \ldots, 1126^{n}\}$ can be shattered for any value for n, which means $d_{vc} = \infty$.

[d]

It is easy to see that a set of 4 points with coordinate $(0,0), (1,-1), (2,-2), (3,-3)$ can be shattered by such triangle. However, it is not possible to completely shatter a set of 5 points using triangles. Consider this situation as a convex shape, and labeling all points positively except for the one within the interior of the convex hull is not possible because a triangle convex shape is a degenerate case where no points are in the interior of the convex hull. This is trivial that $d_{vc} = 4$.

[e]

It's trivial that the set of 3 points with coordinates $(1,0), (0,1), (-1,0)$ can be shattered by axis-aligned squares while there is no set of 4 points that can be fully shattered.
Support that P_{top} e the highest point, P_{bottom} the lowest, P_{left} the leftmost, and P_{right} the rightmost. Assuming without loss of generality that the difference d_{bt} of y-coordinates between P_{bottom} and P_{top} is greater than the difference d_{lf} of x-coordinates between P_{left} and P_{right}. In such a scenario, it is not possible to label P_{bottom} and P_{top} positively while labeling P_{left} and P_{right} negatively. Therefore $d_{vc} = 3$.

As a result, we should choose $[e]$ as our solution.

Ref: VC Dimension and Model Complexity (https://www.cs.cmu.edu/~epxing/Class/10701/slides/lecture16-VC.pdf)
Ref: 10-701 Machine Learning Fall 2011: Homework 2 Solutions (https://www.cs.cmu.edu/~epxing/Class/10701-11f/HW/HW2_solution.pdf)
Ref: Foundations of Machine Learning (https://cs.nyu.edu/~mohri/ml16/sol2.pdf)

Ref: Mehryar Mohri_Foundations of Machine Learning 2014 (https://cs.nyu.edu/~mohri/ml14/hw2.pdf)
Ref: Advanced Machine Learning: Problem Set II Solutions (https://cse.iitkgp.ac.in/~pabitra/course/aml/hw2_19.pdf)
Ref: Question about VC-dimension [closed] (https://math.stackexchange.com/questions/3940837/question-about-vc-dimension)
Ref: CS683 Scribe Notes (https://www.cs.cornell.edu/courses/cs683/2008sp/lecture%20notes/683notes_0428.pdf)

P4

It is possible to demonstrate $\mathcal{H}$ in two-dimensional case. From lecture slide `class0316.pdf` Page 40, we can simplify $\mathcal{H}$ to be a many positive intervals question. For one positive interval, we have 2 free parameters, and $d_{vc} = 2$, for two positive intervals, we have 4 free parameters, and $d_{vc} = 4$, therefore, for M positive intervals, we have $2M$ free parameters, which are $a_1, b_1, a_2, b_2, \ldots\ldots, a_M, b_M$, and it is trivial that $d_{vc} = 2M$. For more rigorous proof, I will prove it below.
To claim that $d_{vc}(\mathcal{H}) \geq 2M$, consider a set $\{x_1, x_2, \ldots, x_{2M}\}$, where $x_i = e_i$ if $i \in [M]$, and $x_i = -e_{i-M}$ if $i > M$, then it is trivial that all x can be shattered. Besides, let $\{y_1, y_2, \ldots, y_{2M}\} \in \{-1, 1\}^{2M}$, and let A be a set of one-hot vectors in $\mathbb{R}^d$ and their negations.

$$A = \{l_i | 1 \leq i \leq M\} \cup \{-l_i | 1 \leq i \leq M\}$$
$$\text{if } l_i^M = (0, \ldots, 0, 1, 0, \ldots, 0) \text{ where only } i\text{th dimension of } l_i = 1$$

Given any $B \subseteq A$, we can choose

$$\begin{array}{lll} \text{if both } l_i \text{ and } -l_i \in B & a_i = -2 & b_i = 2 \\ \text{if } l_i \in B, -l_i \notin B & a_i = 0 & b_i = 2 \\ \text{if } l_i \notin B, -l_i \in B & a_i = -2 & b_i = 0 \\ \text{otherwise} & a_i = 0 & b_i = 0 \end{array}$$

To claim that $d_{vc}(\mathcal{H}) \leq 2M$, consider any $A \subseteq \mathbb{R}^d$ where $|A| > 2d$, take subset $B \subseteq A$ where for every $1 \leq i \leq M$, choose points ll_i and rr_i with maximum and minimum values respectivelt in the ith coordinate. For every $\mathcal{H}$ that includes all members of B, we have $A \subseteq \mathcal{H}$, thus, $\mathcal{H}$ cannot cut B from A, which means $\mathcal{H}$ can not shatter A.
By shifting all parameters to be larger than 0, then we prove the statements.
Moreover, for the proof of a set C whose size is at least $2M + 1$, we can prove it by using pigeonhole principle. If there exists an element $x \in C$ s.t. $\forall j \in [M]$, there exists $x' \in C \text{ with } x'_j \leq x_j$, and $x'' \in C \text{ with } x''_j \leq x_j$. With collinearity, labeling in which x is negative, and the rest of the elementsin C are positive can not be obtianed, which means $\mathcal{H}$ cannot shatter C with size at least $2M + 1$.

As a result, we should choose $[b]$ as our solution.
Ref: CS 485/685 Course Notes P. 14 (https://www.richardwu.ca/notes/cs485-notes.pdf)
Ref: Understanding Machine Learning Solution Manual P. 13 (https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/MLbookSol.pdf)
Ref: Understanding Machine Learning: From Theory to Algorithms P.79 (https://mcube.lab.nycu.edu.tw/~cfung/docs/books/shalev_shwartz2014ML.pdf)

P5

It's trivial that the following is necessary conditions for $d_{vc} \leq d$

- some set of d distinct inputs is shattered by $\mathcal{H}$
- some set of $d + 1$ distinct inputs is not shattered by $\mathcal{H}$
- any set of $d + 1$ distinct inputs is not shattered by $\mathcal{H}$

Since d_{vc} is the maximum that $m_H(N) = 2^N$, and $m_H(N) = 2^N$ is the most number of dichotomies of N inputs. Thus, if we cannot find 2^{d+1} dichotomies on some and any $d + 1$ inputs, $m_H(d + 1) < 2^{d+1}$ and hence, $d_{vc} < d + 1$, that is $d_{vc} \leq d$. On the other hand, if we can find 2^d

dichotomies on some d inputs, then $m_H(d) = 2^d$, that is $d_{vc} \geq d$. Although the statement "any set of d distinct inputs is shattered by $\mathcal{H}$" meets the criteria while it is not a necessary conditions.

As a result, we should choose $[c]$ as our solution.
Ref: Learning from data](https://work.caltech.edu/telecourse.html (https://work.caltech.edu/telecourse.html))

Linear Models

P6

In order to find optimal w, we can find w_{LIN} such that $\nabla E_{in}(w_{LIN}) = 0$

$$
\begin{aligned}
E_{in}(w) &= \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(w_n x_n - y_n)^2 \\
\nabla E_{in}(w) &= \frac{1}{N} \cdot \frac{\partial}{\partial w}(\sum_{n=1}^{N}(w^T x^T x w - 2w^T x^T y + y^T y)) \\
&= \frac{1}{N}\sum_{n=1}^{N}(2x^T x w - 2x^T y) = \frac{2}{N}\sum_{n=1}^{N}(x^T x w - x^T y) \\
&= 0 \\
0 &= \frac{2}{N}\sum_{n=1}^{N}(x^T x w - x^T y) \\
w_{LIN} &= \sum_{n=1}^{N}((x^T x)^{-1} x^T y) = \sum_{n=1}^{N}(\frac{x^T y}{x^T x}) \\
&= \frac{\sum_{n=1}^{N} y_n x_n}{\sum_{n=1}^{N} x_n^2}
\end{aligned}
$$

As a result, we should choose $[b]$ as our solution.

P7

We have

$$
\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n
$$

[a]

$$\begin{aligned}
P(x) &= \frac{e^{-\lambda}\lambda^x}{x!} \Rightarrow P(x) = \prod_{n=1}^{N} f_\theta(x_n) \\
L(x|\lambda) &= \prod_{n=1}^{N} \frac{e^{-\lambda}\lambda^{x_n}}{x_n!} \\
\ln(L(x|\lambda)) &= ln(\prod_{n=1}^{N} \frac{e^{-\lambda}\lambda^{x_n}}{x_n!}) = \sum_{n=1}^{N} \ln(\frac{e^{-\lambda}\lambda^{x_n}}{x_n!}) \\
&= \sum_{n=1}^{N} (\ln(e^{-\lambda}) + \ln(\lambda^{x_n}) - \ln(x_n!)) \\
&= \sum_{n=1}^{N} (-\lambda + x_n \ln(\lambda) - \ln(x_n!)) \\
&= -n\lambda + \ln(\lambda) \sum_{n=1}^{N} (x_n) - \underbrace{\sum_{n=1}^{N} (\ln(x_n!))}_{\text{constant}} \\
\frac{\partial \ln(L(x|\lambda))}{\partial \lambda} &= \frac{\partial}{\partial \lambda}(-n\lambda + \ln(\lambda) \sum_{n=1}^{N} (x_n) - \sum_{n=1}^{N} (\ln(x_n!))) \\
&= -n + \frac{1}{\lambda} \sum_{n=1}^{N} x_n = 0 \\
\lambda^* &= \frac{\sum_{n=1}^{N} x_n}{n} = \bar{x}
\end{aligned}$$

[b]

$$\begin{aligned}
p(x) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2} \\
L(x|\mu) &= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2} \\
&= \frac{1}{\sqrt{(2\pi)^N}} e^{-\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2} \\
\ln(L(x|\mu)) &= \ln(\frac{1}{\sqrt{(2\pi)^N}} e^{-\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2}) \\
&= -\frac{N}{2}(\ln(2\pi)) - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2 \\
\frac{\partial \ln(L(x|\mu))}{\partial \mu} &= \frac{\partial}{\partial \mu}(-\frac{N}{2}(\ln(2\pi)) - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2) \\
&= \sum_{n=1}^{N}(x_n-\mu) = \sum_{n=1}^{N} x_n - n\mu = 0 \\
\mu^* &= \frac{1}{n}\sum_{n=1}^{N} x_n = \bar{x}
\end{aligned}$$

[c]

$$
\begin{aligned}
p(x) &= \frac{1}{2}e^{-|x-\mu|} \\
L(x|\mu) &= \prod_{n=1}^{N}\frac{1}{2}e^{-|x-\mu|} \\
&= \frac{1}{2^N}e^{-\sum_{n=1}^{N}|x_n-\mu|} \\
\ln(L(x|\mu)) &= \ln(\frac{1}{2^N}e^{-\sum_{n=1}^{N}|x_n-\mu|}) \\
&= -N\ln(2) - \sum_{n=1}^{N}|x_n-\mu| \\
\frac{\partial \ln(L(x|\mu))}{\partial \mu} &= \frac{\partial}{\partial \mu}(-N\ln(2) - \sum_{n=1}^{N}|x_n-\mu|) \\
&= \sum_{n=1}^{N}|x_n-\mu| \\
\because \frac{\partial |x|}{\partial x} &= \frac{\partial \sqrt{x^2}}{\partial x} = x(x^2)^{-\frac{1}{2}} = \frac{x}{|x|} = \operatorname{sign}(x) \\
\therefore \frac{\partial \ln(L(x|\mu))}{\partial \mu} &= \sum_{n=1}^{N}\operatorname{sign}(x_n-\mu) = 0
\end{aligned}
$$

Since N can be odd or even, we have two cases.
If N is odd, $\mu^* = \operatorname{median}(x_1, x_2, \ldots, x_N)$, and there are $\frac{N-1}{2}$ cases $x_n < \mu$ and $\frac{N-1}{2}$ cases $x_n > \mu$
If N is even, we can’t simply choose one x_n which will satisfy the condition above. while we can shoose either $x_{\frac{N}{2}}$ or $x_{\frac{N+1}{2}}$.
As a result, $\mu^* = \operatorname{median}(x_1, x_2, \ldots, x_N)$

[d]

$$
\begin{aligned}
P(x) &= (1-\theta)^{x-1}\theta \\
L(x|\theta) &= \prod_{n=1}^{N}(1-\theta)^{x-1}\theta \\
&= (1-\theta)^{\sum_{n=1}^{N}x_n - N}\theta^N \\
\ln(L(x|\theta)) &= \ln((1-\theta)^{\sum_{n=1}^{N}x_n - N}\theta^N) \\
&= (\sum_{n=1}^{N}x_n - N)\ln(1-\theta) + N\ln\theta \\
\frac{\partial \ln(L(x|\theta))}{\partial \theta} &= -\frac{(\sum_{n=1}^{N}x_n - N)}{(1-\theta)} + \frac{N}{\theta} = 0 \\
\theta^* &= \frac{N}{\sum_{n=1}^{N}x_n} = \frac{1}{\bar{x}}
\end{aligned}
$$

As a result, we should choose $[c]$ as our solution.

Ref: MLE for a Poisson Distribution (Step-by-Step) (https://www.statology.org/mle-poisson-distribution/)
Ref: Chapter 8.3. Maximum Likelihood Estimation (https://mathweb.ucsd.edu/~gptesler/283/slides/283_mle_19-handout.pdf)
Ref: Normal distribution - Maximum Likelihood Estimation (https://www.statlect.com/fundamentals-of-statistics/normal-distribution-maximum-likelihood)
Ref: Topic 15: Maximum Likelihood Estimation∗ (https://www.math.arizona.edu/~jwatkins/o-mle.pdf)
Ref: Maximum Likelihood Estimation of Gaussian Parameters (http://jrmeyer.github.io/machinelearning/2017/08/18/mle.html)
Ref: Finding the maximum likelihood estimator (https://math.stackexchange.com/questions/240496/finding-the-maximum-likelihood-estimator)
Ref: HOMEWORK 7 SOLUTIONS (https://web.stanford.edu/class/archive/stats/stats200/stats200.1172/Solutions07.pdf)

Ref: Examples of Parameter Estimation based on Maximum Likelihood (MLE): the exponential distribution and the geometric distribution (https://www.projectrhea.org/rhea/index.php/MLE_Examples:_Exponential_and_Geometric_Distributions_Old_Kiwi)

P8

$$
\begin{aligned}
\bar{h}(x) &= \frac{1 + w^T x + |w^T x|}{2 + 2|w^T x|} \\
\theta(s) &= \frac{1 + s + |s|}{2 + 2|s|} \\
L(\bar{h}(y_n x_n)) &= \prod_{n=1}^{N} \frac{1 + y_n w^T x_n + |y_n w^T x_n|}{2 + 2|y_n w^T x_n|} \\
\ln(L(\bar{h}(y_n x_n))) &= \ln(\prod_{n=1}^{N} \frac{1 + y_n w^T x_n + |y_n w^T x_n|}{2 + 2|y_n w^T x_n|}) \\
&= \frac{1}{N} \sum_{n=1}^{N} - \ln \theta(y_n w^T x_n) \\
&= \underbrace{\frac{1}{N} \sum_{n=1}^{N} -(\ln(1 + y_n w^T x_n + |y_n w^T x_n|) - \ln(2 + 2|y_n w^T x_n|)}_{E_{in}(w)} \\
\frac{\partial \ln(L(\bar{h}(y_n x_n)))}{\partial w} &= \frac{\partial}{\partial w}(\frac{1}{N} \sum_{n=1}^{N} -(\ln(1 + y_n w^T x_n + |y_n w^T x_n|) - \ln(2 + 2|y_n w^T x_n|) \\
&= -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n x_n}{(1 + y_n w^T x_n + |y_n w^T x_n|)(1 + |y_n w^T x_n|)}
\end{aligned}
$$

As a result, we should choose $[a]$ as our solution.

Beyond Gradient Descent

P9

$$
\begin{aligned}
E_{in}(w) &= \frac{1}{N}||xw - y||^2 = \frac{1}{N}(w^T x^T x w - 2 w^T x^T y + y^T y) \\
\nabla E_{in}(w) &= \frac{2}{N}(x^T x w - x^T y) \\
\nabla^2 E_{in}(w) &= \frac{2}{N} x^T x
\end{aligned}
$$

As a result, we should choose $[b]$ as our solution.

P10

$$
\begin{aligned}
E_{in}(w) &= \frac{1}{N}||xw - y||^2 = \frac{1}{N}(w^T x^T x w - 2 w^T x^T y + y^T y) \\
\nabla E_{in}(w) &= \frac{2}{N}(x^T x w - x^T y) = 0 \\
w_{LIN} &= \underbrace{(x^T x)^{-1} x^T}_{\text{pseudo-inverse } x^\dagger} y = x^\dagger y
\end{aligned}
$$

Therefore, when applying Newton method on linear regression, we can simply calculate the pseudo-inverse, then calculate the result w_{LIN}, which means it only causes 1 iteration.

As a result, we should choose $[a]$ as our solution.

Decision Stumps

P11

The growth function of decision stumps is $m_H(N) = 2N$, hence, $m_H(2N) = 4N$. Then we have

$$\mathbb{P}\left[\exists h \in \mathcal{H} \text{ s.t. } |E_{in}(h) - E_{out}(h)| > \epsilon\right] \leq 4 \cdot m_H(2N) \cdot \exp(-\frac{1}{8}\epsilon^2 N)$$

$$\begin{aligned}
\delta &\leq 4 \cdot m_H(2N) \cdot \exp(-\frac{1}{8}\epsilon^2 N) \\
\delta &\leq 4 \cdot 4N \cdot \exp(-\frac{1}{8}\epsilon^2 N) \\
\epsilon &\leq \sqrt{\frac{8}{N}\ln(\frac{16N}{\delta})}
\end{aligned}$$

$$\begin{array}{lll}
[\text{a}] \Rightarrow N = 100, & \epsilon = 0.774428 & > 0.05 \\
[\text{b}] \Rightarrow N = 1000, & \epsilon = 0.0958634 & > 0.05 \\
[\text{c}] \Rightarrow N = 10000, & \epsilon = 0.0114284 & > 0.05 \\
[\text{d}] \Rightarrow N = 100000, & \epsilon = 0.00132705 & < 0.05 \\
[\text{e}] \Rightarrow N = 1000000, & \epsilon = 0.000151125 & < 0.05
\end{array}$$

As a result, we should choose $[d]$ as our solution.

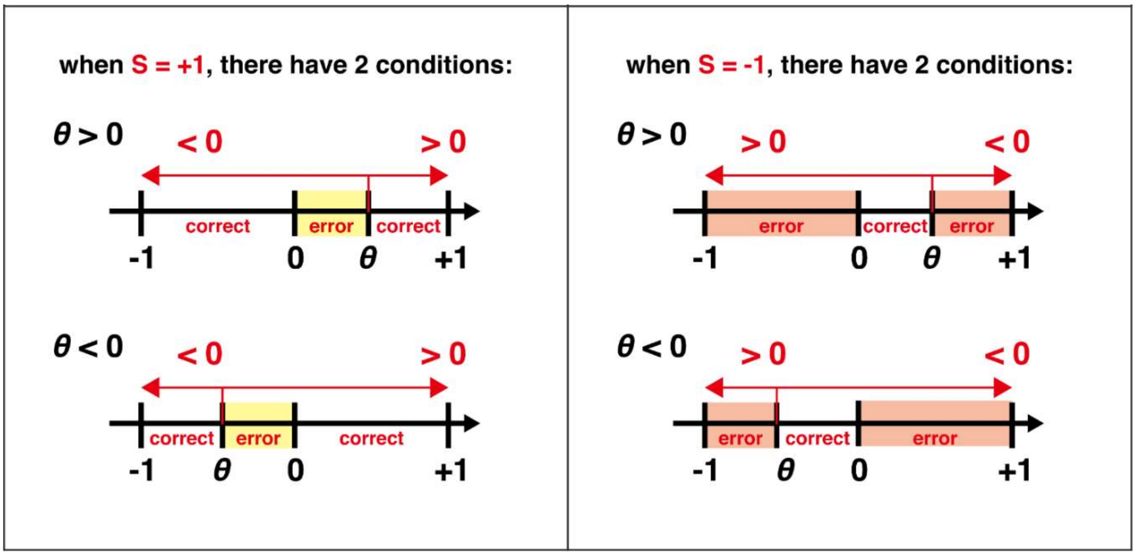
P12

There exists a probability τ to flip the output, which causes the noise. On the other hand, there exists a probability $1 - \tau$ without changing the output. Therefore, $E_{out}(h, \tau)$ will equal to [(noise)x(classified correct in origin)+(no noise)x(classified incorrect in origin)].

$$\begin{aligned}
E_{out}(h, \tau) &= (1 - \tau)E_{out}(h, 0) + \tau(1 - E_{out}(h, 0)) \\
&= E_{out}(h, 0) - \tau E_{out}(h, 0) + \tau - \tau E_{out}(h, 0) \\
&= E_{out}(h, 0)(1 - 2\tau) + \tau
\end{aligned}$$

Since $E_{out}(h, 0)$ means there are no noise and $h(+1, \theta)$ means the direction indicator $s = +1$, we can only consider the relationship between x and θ in the absence of any noise.

- For $s = +1$
  - $x > \theta \Rightarrow h_{s,\theta}(x) = +1$
  - $x \leq \theta \Rightarrow h_{s,\theta}(x) = -1$
- For $s = -1$
  - $x > \theta \Rightarrow h_{s,\theta}(x) = -1$
  - $x \leq \theta \Rightarrow h_{s,\theta}(x) = +1$



$$\begin{aligned} E_{out}(h_{+1}, 0) &= \mathbb{P}[y = f(x) = +1, h_{s,\theta}(x) = -1] + \mathbb{P}[y = f(x) = -1, h_{s,\theta}(x) = +1] \\ &= \frac{1}{2} \cdot \frac{\theta}{0.5 - (-0.5)} + \frac{1}{2} \cdot \frac{|\theta|}{0.5 - (-0.5)} \text{ where } \theta \in [-\infty, 0.5] \\ &= \min(|\theta|, 0.5) \end{aligned}$$

Therefore, we have

$$\begin{aligned} E_{out}(h, \tau) &= E_{out}(h, 0)(1 - 2\tau) + \tau \\ &= \min(|\theta|, 0.5)(1 - 2\tau) + \tau \end{aligned}$$

As a result, we should choose $[d]$ as our solution.

P13

```
import math
import numpy as np
from tqdm import trange

def decision_stump(args, x, y):
    # h(x) = s * sign (x - theta)
    s = 1; theta = 0
    theta_list = np.array([float('-inf')] + [((x[i]+x[i+1])/2) for i in range(0, x.sh
    error = float('inf')
    for theta_hypothesis in theta_list:
        y_pos = np.where(x > theta_hypothesis, 1, -1)
        y_neg = np.where(x <= theta_hypothesis, 1, -1)
        error_pos = sum(y_pos != y)
        error_neg = sum(y_neg != y)
        if error_pos > error_neg:
            if error_neg < error:
                error = error_neg
                theta = theta_hypothesis
                s = -1
        else:
            if error_pos < error:
                error = error_pos
                theta = theta_hypothesis
                s = 1

    if theta == float('-inf'): theta = -0.5
    return s, theta, float(error/x.shape[0])

def generate_data(args):
    # np.random.seed(args['seed'])  # fixed seed may causes error
    x = np.sort(np.random.uniform(args['range_neg'], args['range_pos'], args['size'])
    y = np.sign(x)
    y[y == 0] = -1
    probability = np.random.uniform(0, 1, args['size'])
    noise_ratio = 1 - args['tau']
    y[probability > noise_ratio] *= -1
    return x, y

def main(args):
    total_E_in = 0; total_E_out = 0

    for i in trange(args['repeat_time']):
        x, y = generate_data(args)
        s, theta, E_in = decision_stump(args, x, y)
        E_out = min(math.fabs(theta), args['range_pos'])*(1-2*args['tau'])+args['tau'
        total_E_in += E_in
        total_E_out += E_out

    print("Average total_E_in:", total_E_in/args['repeat_time'])
    print("Average total_E_out:", total_E_out/args['repeat_time'])
    print("mean of E_out(g,tau)-E_in(g):", (total_E_out-total_E_in)/args['repeat_time

if __name__ == '__main__':
    args = {
        'range_neg': -0.5,
        'range_pos': 0.5,
        'repeat_time': 100000,
        'seed': 1126,
        'size': 2,
        'tau': 0
    }

    main(args)
```

```
Average total_E_in: 0.0
Average total_E_out: 0.2922032839476481
mean of E_out(g,tau)-E_in(g):  0.2922032839476481
```

As a result, we should choose $[b]$ as our solution.

P14

We use the same code above, then update the parameter of args.

```
if __name__ == '__main__':
    args = {
        'range_neg': -0.5,
        'range_pos': 0.5,
        'repeat_time': 100000,
        'seed': 1126,
        'size': 128,
        'tau': 0
    }

    main(args)

```

```
Average total_E_in: 0.0
Average total_E_out: 0.0038555886544969927
mean of E_out(g,tau)-E_in(g):  0.0038555886544969927
```

As a result, we should choose $[b]$ as our solution.

P15

We use the same code above, then update the parameter of args.

```
if __name__ == '__main__':
    args = {
        'range_neg': -0.5,
        'range_pos': 0.5,
        'repeat_time': 100000,
        'seed': 1126,
        'size': 2,
        'tau': 0.2
        }

    main(args)
```

```
Average total_E_in: 0.0
Average total_E_out: 0.3913878886120975
mean of E_out(g,tau)-E_in(g):  0.3913878886120975
```

As a result, we should choose $[c]$ as our solution.

P16

We use the same code above, then update the parameter of args.

```
if __name__ == '__main__':
    args = {
        'range_neg': -0.5,
        'range_pos': 0.5,
        'repeat_time': 100000,
        'seed': 1126,
        'size': 128,
        'tau': 0.2
    }

    main(args)
```

```
Average total_E_in: 0.195386171875
Average total_E_out: 0.20937300431054243
mean of E_out(g,tau)-E_in(g):  0.01398683243554242
```

As a result, we should choose $[b]$ as our solution.

P17

```
import numpy as np

def decision_stump(args, x, y):
    # h(x) = s * sign (x - theta)
    s = 1; theta = 0
    theta_list = np.array([float('-inf')] + [((x[i]+x[i+1])/2) for i in range(0, x.sh
    error = float('inf')
    for theta_hypothesis in theta_list:
        y_pos = np.where(x > theta_hypothesis, 1, -1)
        y_neg = np.where(x <= theta_hypothesis, 1, -1)
        error_pos = sum(y_pos != y)
        error_neg = sum(y_neg != y)
        if error_pos > error_neg:
            if error_neg < error:
                error = error_neg
                theta = theta_hypothesis
                s = -1
        else:
            if error_pos < error:
                error = error_pos
                theta = theta_hypothesis
                s = 1

    if theta == float('-inf'): theta = -0.5
    return s, theta, float(error/x.shape[0])

def decision_stump_multi(args, x, y):
    s = np.zeros((args['size'],))
    theta = np.zeros((args['size'],))
    error = np.zeros((args['size'],))

    for i in range(args['size']):
        s[i], theta[i], error[i] = decision_stump(args, x[:, i], y)
    dimension = np.argmin(error)
    if args['type'] == 0: dimension = np.argmax(error)

    return dimension, s[dimension], theta[dimension], error[dimension]
```

```
def predict(dimension, s, theta, x_test):
    predict = s * np.sign(x_test[:, dimension]-theta)
    predict[predict == 0] = -1
    return predict

def read_file(filename):
    data = np.loadtxt(filename, dtype=float)
    x = data[:,:-1]
    y = data[:,-1]
    return x, y

def main(args):
    x_train, y_train = read_file(args['filename_train'])
    x_test, y_test = read_file(args['filename_test'])

    # change args parameter
    args['size'] = x_train.shape[1]

    dimension_best, s_best, theta_best, E_in_best = decision_stump_multi(args, x_trai
    args['type'] = 0
    dimension_worst, s_worst, theta_worst, E_in_worst = decision_stump_multi(args, x_

    # predict
    y_pred_best = predict(dimension_best, s_best, theta_best, x_test)
    y_pred_worst = predict(dimension_worst, s_worst, theta_worst, x_test)

    # E_out
    E_out_best = np.sum(y_pred_best != y_test)/len(y_test)
    E_out_worst = np.sum(y_pred_worst != y_test)/len(y_test)

    print("E_in_best: ", E_in_best)
    print("E_out_best: ", E_out_best)
    print("delta E_in: ", E_in_worst-E_in_best)
    print("delta E_out: ", E_out_worst-E_out_best)

if __name__ == '__main__':
    args = {
        'filename_test': "hw2_test.dat",
        'filename_train': "hw2_train.dat",
        'size': 1126,
        'type': 1,  # 1: best, 0: worst
        'tau': 0
    }

    main(args)
```

```
E_in_best:  0.026041666666666668
E_out_best:  0.078125
delta E_in:  0.3020833333333333
delta E_out:  0.34375
```

As a result, we should choose $[c]$ as our solution.

P18

Since the result is written in the last question at the same time, therefore, we use the same code above without updating anything.

```
E_in_best:  0.026041666666666668
E_out_best:  0.078125
delta E_in:  0.3020833333333333
delta E_out:  0.34375
```

As a result, we should choose $[e]$ as our solution.

P19

Since the result is written in the last question at the same time, therefore, we use the same code above without updating anything.

```
E_in_best:  0.026041666666666668
E_out_best:  0.078125
delta E_in:  0.3020833333333333
delta E_out:  0.34375
```

As a result, we should choose $[d]$ as our solution.

P20

Since the result is written in the last question at the same time, therefore, we use the same code above without updating anything.

```
E_in_best:  0.026041666666666668
E_out_best:  0.078125
delta E_in:  0.3020833333333333
delta E_out:  0.34375
```

As a result, we should choose $[b]$ as our solution.