Machine Learning HW1

B11901174 傅啟恩

1. One of the applications for self-supervised learning is "Speech-to-Speech Translation", which use speech waveforms as input data (even without written form) and output as waveforms in another language. It utilizes SSL to train upstream model so that it can deal with the downstream tasks.

   To be specific, the upstream model is trained through self-supervised learning, which means it leverage large amounts of unlabeled speech data to train models, and it can be fine-tuned on downstream tasks related to speech translation. For example, it can be further trained to perform tasks like speech recognition, machine translation, and speech synthesis.

2. Yes, I agree with the answer. Reinforcement learning can be used to deal with the shortest path in a maze problem by choosing a person/player as the agent, and the maze as the unchanged environment and properly setting rewards such as the fewer returning times, the higher the rewards, or the shortest the path, the higher the rewards.

   For example, we can define a set of actions that the agent can take, such as moving up, down, left, or right. The agent starts at a specific position in the maze and explores the environment by taking different actions.

   By using reinforcement learning algorithms, the agent can learn through trial and error to navigate the maze and optimize its actions to maximize the cumulative rewards. Over time, the agent can develop a policy that enables it to consistently find the shortest path to the exit.

3. It seems like ChatGPT does not disapprove any possibilities in this problem. I think the off-the-shelf algorithm, which is proved to be the fastest or most optimal, cannot be improved by machine learning. However, for problems that have not been proven yet, such as sorting problems, algorithms with fewer instructions than human benchmarks can be discovered using AlphaDev.

   According to this article, AlphaDev has the potential to find algorithms that outperform human benchmarks and provide more efficient solutions. By utilizing AlphaDev and other similar machine learning techniques, we can push the boundaries of what is considered possible in algorithm design. With the continuous advancements in machine learning, we can uncover innovative solutions that were previously unimaginable.

So, while the off-the-shelf algorithms may serve their purpose in many cases, I think AlphaDev and similar tools can have advancements in various domains, including sorting problems and beyond.

4.

4. Suppose $\hat{w}_{PLA} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$.

If "mistake" $\Leftrightarrow$ sign $(w_t^T x_{n(t)}) \neq y_{n(t)}$,

$\hat{w}_{t+1} = \hat{w}_t + y_{n(t)} x_{n(t)} : \left(\begin{bmatrix} 1 \\ \vdots \\ \vdots \end{bmatrix} \right)$ $x_0 = 1$

and $-\text{threshold} = w_0 = 0$

$\Rightarrow$ with $T+$ mistakes as $y_{n(t)} = +1$,
$T-$ " " $= -1$

$w_0 = 1 \times (T+) \times (+1) + 1 \times (T-) \times (-1) = \underline{(T+) - (T-)}$ #

5. 5. As we proved in the course, the upperbound of the total steps to W_{PLA} (T) is bounded by $\dfrac{(\max_n \|x_n\|^2)}{(\min_n y_n \frac{w_f^T}{\|w_f\|} x_n)^2}$, and $X = \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{(d+1)\times 1}$ $\}d$

and composed of $\{0,1\}$, which means $X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$ has at most $m \times 1$s

because in email x it contains at most m distinct words

$\Rightarrow \max_n \|x_n\|^2 \le (m + 1)$ ($x_0 = 1$) ①

$f(n) = \text{sign}(\overset{\text{spam-like}}{z_+(x)} - \overset{\text{not spam-like}}{z_-(x)} - 0.5)$

In each step we want to check $\text{sign}(w_t^T x_t) \overset{?}{=} y_t$,

To approximate f, $\boxed{w_f^T = [-0.5, \pm 1, \pm 1, \pm 1 \cdots \pm 1]_{1\times(d+1)}}$

$+1$: spam-like, -1 : not spam-like

$\Rightarrow \left(\min_n y_n \frac{w_f^T}{\|w_f\|} x_n\right)^2$ happens if (w_f: final perfect w)
predict correctly

1.

$$z_+(x) = z_-(x)\ ,\ \mathrm{sign}(\overset{-0.5}{w_f^T x_n}) = y_n = -1$$

2.

$$z_+(x) = z_-(x) + 1\ ,\ \mathrm{sign}(\overset{+0.5}{w_f^T x_n}) = y_n = +1$$

And $\left(\min_n y_n w_f^T x_n\right)^2 = (0.5)^2 = 0.25$

$$\|w_f\| = \sqrt{(0.5)^2 + d \times 1^2} = \sqrt{0.25 + d}$$

$$\Rightarrow \left(\min_n y_n \frac{w_f^T}{\|w_f\|} x_n\right)^2 = \frac{0.25}{0.25 + d} \quad \text{—— ②}$$

By ①, ②, $(\text{the upperbound of } T) = \frac{R^2}{\rho^2}$

$$= \frac{\max_n \|x_n\|^2}{\left(\min_n y_n \frac{w_f^T}{\|w_f\|} x_n\right)^2} = \frac{(m+1)}{\frac{0.25}{0.25+d}} = (4d+1)(m+1)$$

#

$\Rightarrow$ $(4d+1)(m+1)$ bounds the number of mistakes that PLA can make spam classification.

6.

6.

$$w_0 = 0 \rightarrow \boxed{x_n = \begin{pmatrix} 1 \\ \vdots \\ x_n^{orig} \end{pmatrix}, \quad x_n' = \begin{pmatrix} -1 \\ \vdots \\ x_n^{orig} \end{pmatrix}} \begin{matrix} \rightarrow w_{PLA} \\ \rightarrow w_{PLA}' \end{matrix}$$

with same $n(t)$, $t = 0, 1, 2 \cdots$,

Each updates, $\quad y_{n(t)} \underline{w_t^T x_{n(t)}}$

$$w_1 = w_0 + y_{n(0)} \begin{pmatrix} 1 \\ x_{n(0)}^{orig} \end{pmatrix}, \quad w_1' = w_0 + y_{n(0)} \begin{pmatrix} -1 \\ x_{n(0)}^{orig} \end{pmatrix}$$

$$w_2 = w_1 + y_{n(1)} \begin{bmatrix} 1 \\ x_{n(1)}^{orig} \end{bmatrix} \quad w_2' = w_1' + y_{n(1)} \begin{bmatrix} -1 \\ x_{n(1)}^{orig} \end{bmatrix}$$

$$= \begin{bmatrix} y_{n(0)} + y_{n(1)} \\ y_{n(0)} x_{n(0)}^{orig} + y_{n(1)} x_{n(1)}^{orig} \end{bmatrix} \quad = \begin{bmatrix} -y_{n(0)} - y_{n(1)} \\ y_{n(0)} x_{n(0)}^{orig} + y_{n(1)} x_{n(1)}^{orig} \end{bmatrix}$$

$$\vdots \qquad\qquad \vdots$$

$$w_{PLA} = w_T + y_{n(T)} \begin{bmatrix} 1 \\ x_{n(T)}^{orig} \end{bmatrix} \quad w_{PLA}' = w_T' + y_{n(T)} \begin{pmatrix} -1 \\ x_{n(T)}^{orig} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_i^T y_{n(i)} \\ \sum_i^T y_{n(i)} x_{n(i)}^{orig} \end{pmatrix} \quad = \begin{pmatrix} -\sum_i^T y_{n(i)} \\ \sum_i^T y_{n(i)} x_{n(i)}^{orig} \end{pmatrix}$$

And for every t, $y_{n(t)} w_t^T x_{n(t)} = y_{n(t)} w_t'^T x_n'(t)$!

$\Rightarrow$ $\underline{w_{PLA} \text{ is not necessarily equal to } w_{PLA}' \text{ unless } \sum_i^T y_{n(i)} = 0}$ #

But without considering $w_{PLA}[0]$, $w_{PLA}'[0]$, They will both return the same $\text{sign}(w_{PLA} x_n)$ and $\text{sign}(w_{PLA}' x_n')$, which means $\text{sign}\left(\left[\sum_i^T y_{n(i)} \quad \sum_i^T y_{n(i)} x_{n(i)}^{orig} \right] \begin{bmatrix} 1 \\ x_{n(T)}^{orig} \end{bmatrix} \right)$

$$= \text{sign}\left(\left[-\sum_i y_{n(i)} \quad \sum_i y_{n(i)} x_n'(T)^{orig} \right] \begin{bmatrix} -1 \\ x_n'(T)^{orig} \end{bmatrix} \right)$$ #

7.

7. Start from $w_0 = 0$

① $\|w_T\|^2 \le \max_n \left\| \frac{x_n}{\|x_n\|} \right\|^2 \times T$

② $w_f^T w_{t+1} = w_f^T \left(w_t + y_{n(t)} \frac{x_{n(t)}}{\|x_{n(t)}\|}\right)$

$\geq w_f^T w_t + \min_n y_n w_f^T z_n$

$w_f^T w_T \geq T \cdot \|w_f\| \rho_z$

① + ②

$1 \geq \frac{w_f^T}{\|w_f\|} \frac{w_T}{\|w_T\|} \geq T \rho_z \frac{1}{\sqrt{T}} = \sqrt{T} \rho_z$

$\sqrt{T} \rho_z \le 1 \;,\; T \le \frac{1}{\rho_z^2}$ #

8. τ-separable

$\Leftrightarrow$ exist perfect w_f such that $\underline{y_n w_f^T x_n > \tau}$

$y_{n(t)} w_f^T x_{n(t)} > \underbrace{\min_n y_n w_f^T x_{n(t)}}_{=\rho} > \tau$

① $w_f^T w_{t+1} = w_f^T (w_t + y_{n(t)} x_{n(t)}) > w_f^T w_t + \tau$

② $\|w_{t+1}\|^2 \le \|w_t\|^2 + 2\tau + \| y_{n(t)} x_{n(t)} \|^2$

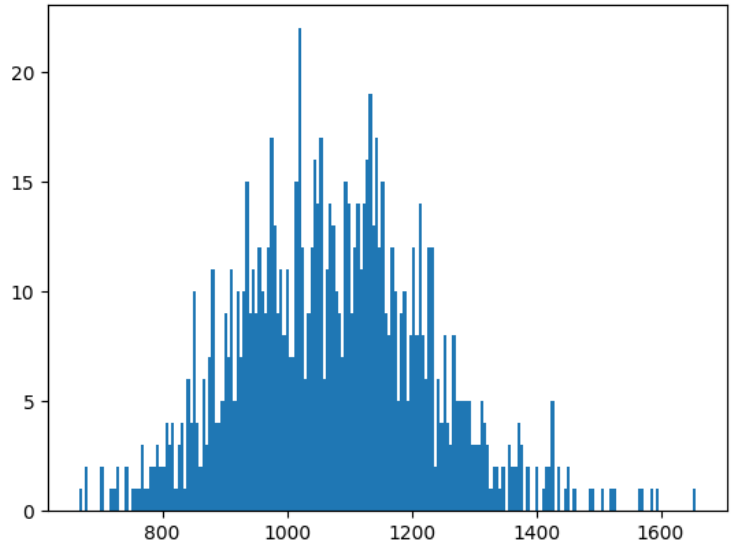
$\le \|w_t\|^2 + \underbrace{2\tau + \max_n \|y_n x_n\|^2}$ $= R^2$

$\Rightarrow$ like PLA, PAM has upper bound for updates time and "lines" are more and more aligned with w_f

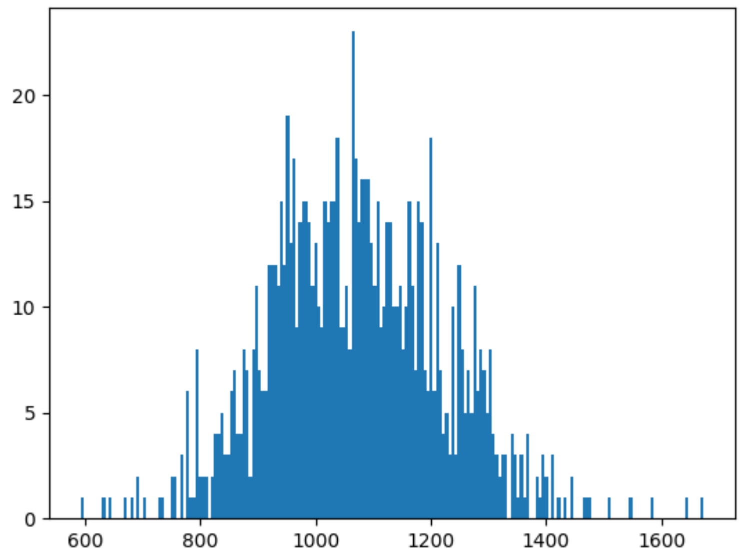
$\Rightarrow$ PAM will halt in finite number $T \le \frac{R^2}{\tau^2}$ #
of steps

\# $1 \geq \frac{w_f^T w_T}{\|w_f\| \|w_T\|} \geq \frac{T \cdot \tau}{\sqrt{T} \cdot R}$

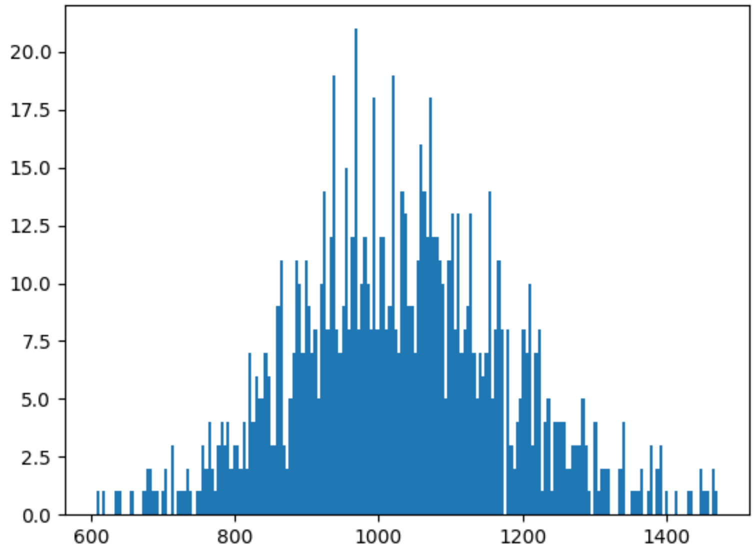
9. (Original)
   Median number of updates: 1073.5 (updates)



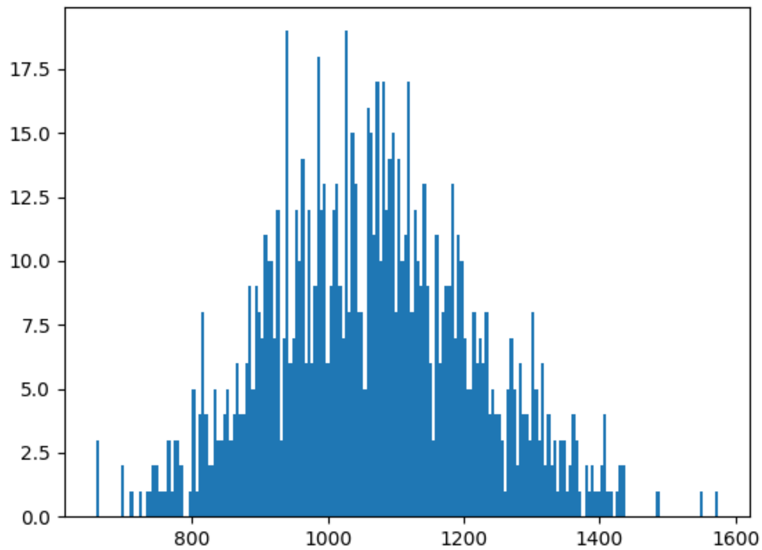
10. (Scaling by 11.26)
    Median number of updates: 1067.0 (updates), compared to problem 9, the number of updates decreases a little bit or even the same, which means we need similar steps to get w_{PLA}.



11. (Set x0 = 11.26)
   Median number of updates: 1024 (updates), compared to problem 9, the number of updates decreases about 5%, which means we need less steps to get w_{PLA}.



12. (Keep correcting the same example until it is perfectly classified)
   Median number of updates: 1064.5 (updates), compared to problem 9, the number of updates decreases a little bit or even the same, which means we need similar steps to get w_{PLA}.



13.

13. Give one example that normalization actually speed up PLA:

$x_1 = (1, 50)$ $y_1 = -1$
$x_2 = (1, 100)$ $y_2 = +1$ and $w_0 = (0, 0)$

Go through PLA: (By code)

and choosing randomly

It needs on average 13825 updates to get w_{PLA}

$w_{PLA} = (-302, 100)$

While after normalization, and given $x_0 = 1$

$$\begin{array}{ll} & \quad\ \ \hat{x_0} \qquad\quad\ \ x_1 \qquad\quad\ \ x_2 \\ \hat{x_1} = & (0.0199,\ 0.0199,\ 0.9996) \end{array}$$

$\hat{x_2} = (0.0099, 0.0099, 0.9999)$

It takes on average 2 updates to get

$w_{PLA} = (-0.0099, 0.0003)$

⇒ In this case normalization seems speed up PLA

While for another case

$x_1 = (1, 1)$ $y_1 = -1$ $w_0 = 0$
$x_2 = (1, -1)$ $y_2 = +1$

By running code (randomly choose x)

It takes 2 updates to get $W_{PLA} = [0, -2]$

after normalization

$\hat{x}_0$

$\hat{x}_1 = [0.577, 0.577, 0.577]$

$\hat{x}_2 = [0.577, 0.577, -0.577]$

It also on averge takes 2 steps to get W_{PLA}

$W_{PLA} = [0, -1.1547]$

$\Rightarrow$ normalization cannot speed up more on simple data

In conclusion, normalization can speed up in some case, especially for complicated data