HW1 for Machine Learning

Hsiu Hsuan Yeh

30th March 2023

1 Multiple Choice

1.1 (a)

Since the machine learning is a kind of data-driven algorithm and both (b) and (c) don't need to collect the data. They are not suitable. In terms of (d), it's a bit tricky in that we can't find the "actual" look of Zeus, so we can't collect the data and do the machine learning. For (a) we can collect the data and let the machine learn the relations between requests and responses, reacting through the learned relations.

1.2 (d)

$$y_{n(t)} w_t^T x_n \leq 0$$
$$y_{n(t)} w_{t+1}^T x_n > 0$$

let's denote the learning rate to be η

$$y_{n(t)} w_{t+1}^T x_{n(t)} = y_{n(t)} (w_t + y_{n(t)} x_{n(t)} \eta)^T x_{n(t)} = y_{n(t)} w_t^T x_{n(t)} + y_{n(t)}^2 x_{n(t)}^T x_{n(t)} \eta > 0$$
$$\Rightarrow \eta > \frac{-y_{n(t)} w_t^T x_{n(t)}}{y_{n(t)}^2 x_{n(t)}^T x_{n(t)}} = \frac{-y_{n(t)} w_t^T x_{n(t)}}{\|x_{n(t)}\|^2}$$
$$\Rightarrow \eta = \left\lfloor \frac{-y_{n(t)} w_t^T x_{n(t)}}{\|x_{n(t)}\|^2} + 1 \right\rfloor$$

1.3 (c)

$$w_{t+1} \leftarrow w_t + y_{n(t)} z_{n(t)}, z_{n(t)} = \frac{x_{n(t)}}{\|x_{n(t)}\|}$$
$$w_f' = \frac{w_f}{\|w_f\|}, \|w_f'\| = 1$$
$$\rho_z = \min_n \frac{y_n w_f^T z_n}{\|w_f\|} = \min_n y_n w_f'^T z_n$$
$$w_f'^T w_{t+1} \geq w_f'^T w_t + \rho_z, \Rightarrow w_f'^T w_T \geq w_f'^T w_0 + T\rho_z$$

$$\|w_{t+1}^2\| \leq \|w_t\|^2 + \|z_n\|^2 = \|w_t\|^2 + 1, \Rightarrow \|w_T\|^2 \leq \|w_0\|^2 + T$$

$$\frac{w_f'^T w_0 + T\rho_z}{\|w_f'\|\sqrt{\|w_0\|^2 + T}} \leq \frac{w_f'^T w_T}{\|w_f'\|\|w_T\|} = cos\theta \leq 1$$

assume $w_0 = 0$

$$\sqrt{T}\rho_z \leq 1, \Rightarrow T \leq \frac{1}{\rho_z^2}$$

1.4 (b)

$$U = \frac{1}{\rho_z^2} = \frac{1}{(\min_n y_n w_f'^T z_n)^2} = \frac{1}{(\min_n y_n w_f'^T \frac{x_n}{\|x_n\|})^2}$$

$$U_{orig} = \frac{\max_n \|x_n\|^2}{(\min_n y_n w_f'^T x_n)^2}$$

because U_{orig} maximize the numerator and minimize the denominator simultaneously,

$$\Rightarrow U \leq U_{orig}$$

1.5 (c)

- PLA:

$$w_1 = 0 + [-1, 2, -2] = [-1, 2, -2]$$
$$w_2 = [-1, 2, -2] + [1, 1, 1] = [0, 3, -1]$$

wrongly predicted: $(x, y) = ([1, \frac{1}{2}, 2], 1), ([1, \frac{1}{4}, 1], 1)$

- PAM

$$w_1 = 0 + [-1, 2, -2] = [-1, 2, -2]$$
$$w_2 = [-1, 2, -2] + [1, 2, 0] = [0, 4, -2]$$
$$w_3 = [0, 4, -2] + [-1, 1, 0] = [-1, 5, -2]$$
$$w_3 = [-1, 5, -2] + [1, 1, 1] = [0, 6, -1]$$

all the test samples are correctly predicted

The number of test samples are wrongly predicted by PLA but correctly predicted by PAM is 2

1.6 (a)

The ratings of movies is the label and the label is given so it's surpervised learning. Moreover, ratings are within [1, 5] which is continuous so it's a regression task.

1.7 (d)

Human labeler'goal is to compare and decide which one is better. The main purpose of this step is to train the model to have general understanding of the text. Further tuning will be conduct in the down-stream task. Best assoicated learning problem is the self-surperivsed learning.

1.8 (c)

In this case, there are two kinds of label +1, -1. If we plot the data, we can observe that it's linare separable. Therefore, the minimum out sample error is 0. However, if all the data's label in the training set are same kind either +1, -1, there might be the case that the out sample error is 1 in that the algorithm fail to identity.

1.9 (d)

The distribution of data is unifomly within [+1, -1]*[+1, -1]. The sample space is a square with side length 2. Therefore, each data point's probability ensity is $\frac{1}{4}$

On one hand, the out sample error for hopothesis h_1 occurs when the samples lie in the area within the unit circle but outside the circel with radius 0.5.

$$E_{out}(h_1) = \frac{\pi 1^2 - \pi 0.25^2}{2 * 2} = \frac{3\pi}{16}$$

On the other hand, the out sample error for hopothesis h_2 occurs when the samples lie in the area within the circlue with radius 0.5 but outside the rhombus with side length $\sqrt{0.5}$

$$E_{out}(h_2) = \frac{\pi 0.25^2 - 4 * \frac{1}{2} * 0.5 * 0.5}{2 * 2} = \frac{\pi - 2}{16}$$

1.10 (b)

Following the previous problem, samples that make the in sample error equal to 0 for both hypothesis lie in the area within the rhombus or between the unit circle and the square. Since we sample 4 times the probabiltiy should be the probability of one sample to the power of 4.

$$(\frac{(4 * \frac{1}{2} * 0.5 * 0.5) + (4 - \pi 1^2)}{4})^4 = (\frac{\frac{9}{2} - \pi}{4})^4 \approx 0.0133$$

1.11 (d)

First denote the random variable $X_i = \mathbb{1}(\text{the darts in the circle})$

$$\mathbb{P}(X_i = 1) = \frac{\pi}{4}, \mathbb{E}(\frac{\Sigma_{i=1}^n X_i}{N}) = \frac{\pi}{4}$$
$$\mathbb{P}(|\frac{\Sigma_{i=1}^n X_i}{N} - \frac{\pi}{4}| \leq \frac{10^{-2}}{4}) > 0.99$$
$$\mathbb{P}(|\frac{\Sigma_{i=1}^n X_i}{N} - \frac{\pi}{4}| < \frac{10^{-2}}{4}) \leq 0.99$$
$$2\exp(-2(\frac{10^{-2}}{4})^2 N) = 0.01, \Rightarrow N = \frac{-8}{10^{-4}} \log\frac{0.01}{2} \approx 423866$$

1.12 (d)

$$\mathbb{P}(b_m \text{ is } \epsilon\text{-optimal}) = \mathbb{P}((|\frac{c_1}{N} - p_1| \leq \frac{\epsilon}{2}) \cap (|\frac{c_2}{N} - p_2| \leq \frac{\epsilon}{2}) \cap ... \cap (|\frac{c_M}{N} - p_M| \leq \frac{\epsilon}{2}))$$
$$= 1 - \mathbb{P}((|\frac{c_1}{N} - p_1| > \frac{\epsilon}{2}) \cup (|\frac{c_2}{N} - p_2| > \frac{\epsilon}{2}) \cup ... \cup (|\frac{c_M}{N} - p_M| > \frac{\epsilon}{2}))$$
$$> 1 - 2M\exp\frac{-\epsilon^2 N}{2}$$
$$\Rightarrow \delta = 2M\exp\frac{-\epsilon^2 N}{2}, N = \frac{2}{\epsilon^2}\ln\frac{2M}{\delta}$$

2 Coding

2.1 (b)

```
features, label, N = read_data()
avg_error = mean([PLA(features, label, M=N/2, seed=i)[3] for i =1:1000])
print("average of the in sample error: $avg_error")
```

```
average of the in sample error: 0.02003515625
```

2.2 (a)

```
avg_error = mean([PLA(features, label, M=4N, seed=i)[3] for i =1:1000])
print("average of the in sample error: $avg_error")
```

```
average of the in sample error: 0.0001875
```

2.3 (d)

```
med_iter = median([PLA(features, label, M=4N, seed=i)[2] for i =1:1000])
print("median of the iterations: $med_iter")
```

```
median of the iterations: 453.0
```

2.4 (e)

```
med_w_0 = median([PLA(features, label, M=4N, seed=i)[1][1] for i =1:1000])
print("median of the w_0: $med_w_0")
```

```
median of the w_0: 35.0
```

2.5 (d)

```
features, label, N = read_data(scale=2)
med_iter = median([PLA(features, label, M=4N, seed=i)[2] for i =1:1000])
print("median of the iterations: $med_iter")
```

```
median of the iterations: 453.0
```

2.6 (d)

```
x_0 = 0
features, label, N = read_data(x_0=x_0)
med_iter = median([PLA(features, label, M=4N, seed=i)[2] for i =1:1000])
print("median of the iterations: $med_iter")
```

```
median of the iterations: 445.5
```

2.7 (e)

```
x_0 = -1
features, label, N = read_data(x_0=x_0)
med_w_0x_0 = median([x_0*(PLA(features, label, M=4N, seed=i)[1][1]) for i =1:1000])

print("median of the w_0x_0: $(med_w_0x_0)")
```

```
median of the w_0x_0: 35.0
```

2.8 (c)

```
x_0 = 0.1126
features, label, N = read_data(x_0=x_0)
med_w_0x_0 = median([x_0*(PLA(features, label, M=4N, seed=i)[1][1]) for i =1:1000])

print("median of the w_0x_0: $(med_w_0x_0)")
```

```
median of the w_0x_0: 0.43107784000000001
```

3 Code Reference

```julia
import DelimitedFiles: readdlm
import Random: seed!
import Distributions: sample
import Statistics: median, mean
import LinearAlgebra: normalize

normalize_row(a) = vcat([transpose(normalize(i, 2)) for i=eachrow(a)]...)

function read_data(path="../hw1_train.txt"; x_0=1, scale=1., normalize=false)
    data = readdlm(path, '\t', Float64, '\n')
    features = hcat(x_0*ones(size(data, 1)), data[:, begin:end-1]) ./ scale
    normalize && (features = normalize_row(features))
    label = data[:, end]

    return features, label, size(features, 1)
end

check_sign(a) = a == 0. ? (return 1.) : (return sign(a))

finderror(s::Symbol, args...) = finderror(Val{s}(), args...)
finderror(::Val{:single}, features, w, label) = findfirst(
    x->x!=0.,
    check_sign.(features*w) .- label
)
finderror(::Val{:multiple}, features, w, label) = findall(
    x->x!=0.,
    check_sign.(features*w) .- label
)

function PLA(features, label; M, seed=20230525)
    seed!(seed)
    obs, w, iter, m, M = 1:size(features, 1),
                         zeros(size(features, 2)),
                         0,
                         0,
                         ceil(Int, M)
    while !isnothing(m)
        idx = rand(finderror(:multiple, features, w, label))
        w += label[idx]*features[idx, :]
        iter += 1

        idxes = sample(obs, M; replace=true)
        m = finderror(:single, features[idxes, :], w, label[idxes])
    end
    error = length(finderror(:multiple, features, w, label)) / length(obs)

    return w, iter, error
end
```