HW3 for Machine Learning

Hsiu Hsuan Yeh

27th April 2023

1 Multiple Choice

1.1 1. (b)

$$a(2*\frac{N}{K})\frac{K(K-1)}{2} = a(K-1)N$$

1.2 2. (d)

since collinearity exists between x_2, x_6, x_1, it's impossible to shatter all six inputs.

1.3 3. (c)

$$z_1 = [1,0,0,0,0,0]^T, z_2 = [1,4,0,16,0,0]^T, z_3 = [1,-4,0,16,0,0]^T$$
$$z_4 = [1,0,2,0,0,4]^T z_5 = [1,0,-2,0,0,4]^T$$

- w_1 and w_4 can separate all examples
- w_2 fail to separate z_2, z_3, z_4, z_5
- w_3 fail to separate z_4, z_5

1.4 4. (b)

- $X'_{N*d+1} = X_{N*d+1}\Gamma^T_{d+1*d+1}$
- $w_{lin} = (X^TX)^{-1}X^Ty$

$$\tilde{w} = (X'^TX')^{-1}X'^Ty = (\Gamma X^TX\Gamma^T)^{-1}\Gamma X^Ty = (\Gamma^T)^{-1}(X^TX)^{-1}\Gamma^{-1}\Gamma(X^Ty) = (\Gamma^T)^{-1}w_{lin}$$

- $w_{lin} = \Gamma^T\tilde{w}$

- $E_{in}(w_{lin}) = \frac{1}{N}(w_{lin}^T X^T X w_{lin} - 2w_{lin}^T X^T y + y^T y)$

$$E_{in}(\tilde{w}) = \frac{1}{N}(\tilde{w}^T X'^T X' \tilde{w} - 2\tilde{w}^T X'^T y + y^T y) = \frac{1}{N}(w_{lin}^T \Gamma^{-1}\Gamma X^T X \Gamma^T (\Gamma^T)^{-1} w_{lin} - 2w_{lin}^T \Gamma^{-1}\Gamma X^T y + y^T y)$$

$$E_{in}(\tilde{w}) = \frac{1}{N}(w_{lin}^T X^T X w_{lin} - 2w_{lin}^T X^T y + y^T y) = E_{in}(w_{lin})$$

1.5 5. (b)

- $m_{H_k}(N) = 2N$
- $H = \cup_{k=1}^{d} H_k$

$$m_H(N) \leq d2N, 2^{d_{vc}} \leq m_H(d_{vc}) \leq d2d_{vc}$$

$$d_{vc} \leq 1 + log_2 d_{vc} + log_2 d \leq 1 + \frac{d_{vc}}{2} + log_2 d$$

$$\Rightarrow \frac{d_{vc}}{2} \leq 1 + log_2 d, d_{vc} \leq 2(1 + log_2 d)$$

1.6 6. (c)

by definition, Z_{N*N} is an identity matrix

- $\tilde{w} = (Z^T Z)^{-1} Z^T y = y, \Rightarrow \tilde{w_n} = y_n$
- based on the previous couclusion, $E_{in}(g) = \frac{1}{N}(\tilde{w}^T Z^T Z \tilde{w} - 2\tilde{w}^T Z^T y + y^T y) = \frac{1}{N}(y^T y - 2y^T y + y^T y) = 0$
- $\Phi(X_{N*d}) \neq 2I_{N*N}$, where I is an identity matrix.
- by the definition of the transformation rule: $g(x) = 0$ on those $x \neq x_n$ for any n

1.7 7. (e)

- $E_{aug}(w) = E_{in}(w) + \frac{\pi}{3}||w||_1 = E_{in}(w) + ||w||_1$

$$\nabla E_{aug}(w) = \nabla E_{in}(w) + \begin{bmatrix} sign(w_0) \\ sign(w_1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\nabla E_{in}(w) = \frac{2}{3}(X^T X w - X^T y) = \frac{2}{3}(\begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & -2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix})$$

$$\nabla E_{aug}(w) = \nabla E_{in}(w) + \begin{bmatrix} sign(w_0) \\ sign(w_1) \end{bmatrix} = \begin{bmatrix} 2w_0 + 2w_1 - 2 \\ 2w_0 + \frac{34}{3}w_1 + \frac{4}{3} \end{bmatrix} + \begin{bmatrix} sign(w_0) \\ sign(w_1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \frac{28}{3}w_1 + \frac{10}{3} + (sign(w_1) - sign(w_0)) = 0$$

$$sign(w_1) - sign(w_0) = \begin{cases} 0 & w_0 = \frac{13}{7}, w_1 = \frac{-5}{14} \\ 2 & w_0 = \frac{29}{14}, w_1 = \frac{-4}{7} \\ -2 & w_0 = \frac{9}{14}, w_1 = \frac{-1}{7} \end{cases}$$

- th contradiction exists when $sign(w_1) - sign(w_0) = 0, 2$
- $||w||_1 = |w_0| + |w_1| = |\frac{9}{14}| + |\frac{-1}{7}| = \frac{11}{14}$

$$E_{in}(w) = \frac{1}{3}(Xw - y)^T(Xw - y) = \frac{1}{3}(\begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} \frac{9}{14} \\ \frac{-1}{7} \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix})^T(\begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} \frac{9}{14} \\ \frac{-1}{7} \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}) = \frac{105}{196}$$

$$\Rightarrow E_{aug}(w) = E_{in}(w) + ||w||_1 = \frac{259}{196} \approx 1.32$$

1.8 8. (b)

- $E_{aug}(w) = E_{in}(w) + \frac{\lambda}{2}||w||_2^2$
- find λ such that $w_0 + w_1 = 4$

$$\nabla E_{aug}(w) = \nabla E_{in}(w) + \lambda \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\nabla E_{in}(w) = \frac{2}{2}(X^T X w - X^T y) = \begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \end{bmatrix} = \begin{bmatrix} 2w_0 - 8 \\ 8w_1 - 20 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 2w_0 - 8 + \lambda w_0 \\ 8w_1 - 20 + \lambda w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\Rightarrow \begin{cases} (2+\lambda)w_0 = 8 \\ (8+\lambda)w_1 = 20 \end{cases} \Rightarrow \begin{cases} w_0 = \frac{8}{2+\lambda} \\ w_1 = \frac{20}{8+\lambda} \end{cases}$$

$$\frac{8}{2+\lambda} + \frac{20}{8+\lambda} = 4, \lambda^2 + 3\lambda - 10 = 0, \lambda = -5, 2$$

since $\lambda > 0, \lambda = 2$

1.9 9. (b)

- $\nabla E_{aug}(w) = \nabla E_{in}(w) + \frac{2\lambda}{N} w$

$$w_t - \eta \nabla E_{aug}(w_t) = w_t - \eta(\nabla E_{in}(w_t) + \frac{2\lambda}{N} w_t) = (1 - \frac{2\eta\lambda}{N})w_t - \eta \nabla E_{in}(w_t)$$

1.10 10. (b)

$$||Xw - y||^2 + ||\tilde{X}w - \tilde{y}||^2 = ||Xw - y||^2 + \lambda||w||^2, \Rightarrow \begin{cases} \tilde{X} = \sqrt{\lambda} I_{d+1} \\ \tilde{y} = 0 \end{cases}$$

1.11 11. (c)

$$\mathbb{E}(X_h^T X_h) = \mathbb{E}(\Sigma_{i=1}^N x_i x_i^T + \Sigma_{i=1}^N \tilde{x}_i \tilde{x}_i^T) = \Sigma_{i=1}^N x_i x_i^T + \Sigma_{i=1}^N \mathbb{E}(\tilde{x}_i \tilde{x}_i^T) = X^T X + \Sigma_{i=1}^N \mathbb{E}((x_i+\epsilon_i)(x_i+\epsilon_i)^T)$$
$$= X^T X + \Sigma_{i=1}^N \mathbb{E}(x_i x_i^T + x_i \epsilon_i^T + \epsilon_i^T x_i + \epsilon_i \epsilon_i^T) = X^T X + \Sigma_{i=1}^N x_i x_i^T$$
$$+\Sigma_{i=1}^N x_i \mathbb{E}(\epsilon_i)^T + \Sigma_{i=1}^N \mathbb{E}(\epsilon_i) x_i^T + \Sigma_{i=1}^N \mathbb{E}((\epsilon_i - 0)(\epsilon_i - 0)^T)$$
$$\Rightarrow \mathbb{E}(X_h^T X_h) = 2X^T X + \Sigma_{i=1}^N \frac{r^2}{3} I_{d+1} = 2X^T X + \frac{N}{3} r^2 I_{d+1}$$

1.12 12. (b)

- $y^* = \frac{(\Sigma_{n=1}^N y_n) + K}{N + 2K}$, $N y^* = \Sigma_{n=1}^N y_n + (1 - 2y^*)K$

$$\frac{\partial}{\partial y}(\frac{1}{N}\Sigma_{n=1}^N (y - y_n)^2 + \frac{\lambda}{N}\Omega(y)) = \frac{1}{N}\Sigma_{n=1}^N 2(y - y_n) + \frac{\lambda}{N}\Omega'(y) = 2y - \frac{2}{N}\Sigma_{n=1}^N y_n + \frac{\lambda}{N}\Omega'(y) = 0$$
$$2y = \frac{2}{N}\Sigma_{n=1}^N y_n - \frac{\lambda}{N}\Omega'(y), Ny = \Sigma_{n=1}^N y_n - \frac{\lambda}{2}\Omega'(y)$$
$$\Rightarrow -\frac{\lambda}{2}\Omega'(y) = (1 - 2y)K, \Omega(y) = \frac{2K}{\lambda}(y - 0.5)^2$$

2 Coding

2.1 13. (c)

```
train_x, train_y = read_data("../train.txt")

@printf(
    "mean squared error: %.5f",
    mean_squared_error(
        train_x,
        train_y,
        LS_estimator(train_x, train_y)
    )
)
```

```
mean squared error: 0.79223
```

2.2 14. (d)

```
@printf(
    "averaged mean squared error: %.5f",
    mean([
        mean_squared_error(
            train_x,
            train_y,
            regSGD_estimator(train_x, train_y; seed=i)
        )
        for i=1:1000
    ])
)
```

```
averaged mean squared error: 0.82329
```

2.3 15. (c)

```
@printf(
    "averaged cross entropy error: %.5f",
    mean([
        cross_entropy_error(
            train_x,
            train_y,
            logitSGD_estimator(train_x, train_y; seed=i)
        )
        for i=1:1000
    ])
)
```

```
averaged cross entropy error: 0.65725
```

2.4 16. (a)

```
w_0 = LS_estimator(train_x, train_y)
@printf(
    "averaged cross entropy error: %.5f",
    mean([
        cross_entropy_error(
            train_x,
            train_y,
            logitSGD_estimator(train_x, train_y; seed=i, w_0=w_0)
        )
        for i=1:1000
    ])
)
```

```
averaged cross entropy error: 0.60527
```

2.5 17. (a)

```
test_x, test_y = read_data("../test.txt")

experiment = zeros(1000)
for i = eachindex(experiment)
    w = logitSGD_estimator(train_x, train_y; seed=i, w_0=w_0)
    experiment[i] = abs(binary_error(train_x,train_y,w)-binary_error(test_x,test_y,w))
end
@printf "averaged difference of train/test binary error: %.5f" mean(experiment)
```

```
averaged difference of train/test binary error: 0.03158
```

2.6 18. (b)

```
@printf(
    "difference of train/test binary error: %.5f",
    abs(binary_error(train_x, train_y, w_0) - binary_error(test_x, test_y, w_0))
)
```

```
difference of train/test binary error: 0.04000
```

2.7 19. (c)

```
train_x_, test_x_ = polynomial_transform.([train_x, test_x]; Q=2)
w =  LS_estimator(train_x_, train_y)

@printf(
    "difference of train/test binary error: %.5f",
    abs(binary_error(train_x_, train_y, w) - binary_error(test_x_, test_y, w))
)
```

```
difference of train/test binary error: 0.08250
```

2.8 20. (d)

```
train_x_, test_x_ = polynomial_transform.([train_x, test_x]; Q=8)
w =  LS_estimator(train_x_, train_y)

@printf(
    "difference of train/test binary error: %.5f",
    abs(binary_error(train_x_, train_y, w) - binary_error(test_x_, test_y, w))
)
```

```
difference of train/test binary error: 0.41500
```

3 Code Reference

```julia
using Printf

import DelimitedFiles: readdlm
using Distributions, Random

function read_data(path)
    data = readdlm(path, '\t', Float64, '\n')
    features = hcat(ones(size(data, 1)), data[:, begin:end-1])
    label = data[:, end]

    return features, label
end

mean_squared_error(X, y, w) = mean((X*w-y).^2)
cross_entropy_error(X, y, w) = -mean(log.(logistic.(y .* X*w)))

check_sign(a) = a == 0. ? 1. : sign(a)
binary_error(X, y, w) = mean(check_sign.(X*w) .!= y)

LS_estimator(X, y) = inv(transpose(X)*X) * (transpose(X)*y)

function SGD_estimator(X, y, sg; it=800, η=0.001, seed=20230420, w_0=nothing)
    Random.seed!(seed)
    w = isa(w_0, Nothing) ? zeros(size(X, 2)) : w_0
    idx = sample(1:size(X, 1), it, replace=true)
    for i = 1:it
        X_i, y_i = X[idx[i], :], y[idx[i]]
        w -= η * sg(X_i, y_i, w)
    end

    return w
end

reg_stochastic_gradient(X_i, y_i, w) = 2 * (X_i*transpose(X_i)*w - X_i*y_i)
regSGD_estimator(X, y; kwargs...) = SGD_estimator(X, y, reg_stochastic_gradient;
kwargs...)

logistic(x) = 1 / (1+exp(-x))
logit_stochastic_gradient(X_i, y_i, w) = logistic(-y_i*transpose(w)*X_i) * (-y_i*X_i)
logitSGD_estimator(X, y; kwargs...) = SGD_estimator(X, y, logit_stochastic_gradient;
kwargs...)

function polynomial_transform(X; Q)
    d = size(X, 2)-1
    X_ = Matrix{Float64}(undef, size(X, 1), Q*d+1)
    X_[:, begin:d+1] = X; X = X[:, begin+1:end]

    beg = d + 2
    for q = 2:Q
        en = beg + d - 1
        X_[:, beg:en] = X .^ q
        beg = en + 1
    end

    return X_
end
```