CSE 6740B HW3

Che-Ting,Meng

October 27th,2022

1 Linear Regression

1.1 (a)

With the given normal Eq.1 and least square mean $\hat{\theta}$, we derive:

$$\begin{aligned}
\hat{\theta} &= (X^T X)^{-1} X^T Y \\
&= (X^T X)^{-1} X^T (\theta^T X + \epsilon) \\
&= \theta^T + (X^T X)^{-1} X^T \epsilon \\
E[\hat{\theta}] &= E[\theta] + E[(X^T X)^{-1} X^T \epsilon] \\
&= E[\theta] + (X^T X)^{-1} X^T E[\epsilon] \\
&\because E[\theta] = \theta^T, E[\epsilon] = 0, \\
&= \theta
\end{aligned}$$

1.2 (b)

$$
\begin{aligned}
Var[\hat{\theta}] &= Var[(X^T X)^{-1} X^T Y] \\
&= (X^T X)^{-2} Var[X^T Y] \\
&= (X^T X)^{-2} Var[X^T(\theta^T X + \epsilon)] \\
&\because (X^T \theta^T X) constant, \\
&= (X^T X)^{-2} Var[X^T \epsilon] \\
&\because (X^T) constant, \\
&= \frac{(X^T X}{(X^T X)^2} Var[\epsilon] \\
&\because Var[\epsilon] = \sigma^2 \\
&= \frac{\sigma^2}{X^T X}
\end{aligned}
$$

1.3 (c)

Yes, I agree. Because under independence white noise assumption given, the conditional probability

$$p(y|x,\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^i - \theta^T x^i)^2}{2\sigma^2}\right),$$

and the maximum log likelihood derives the maximum $\hat{\theta} = \frac{1}{m}\Sigma(y^i - \theta^T x^i)^2$, which is the same with the LMS derived maximum $\hat{\theta}$. This means that by applying the closed form solution of LMS where $\hat{\theta} = (X^T X)^{-1} X^T Y$, we are assuming Gaussian distribution in $\hat{\theta}$.

1.4 (d)

1.4.1 1.

False, because in linear regression with LMS, the loss function is convex. Therefore, gradient descend will always converge to a global minimum.

1.4.2 2.

True, because at global minimum, the gradient would be zero. Therefore, the iteration would not update the parameter θ.

1.4.3 3.

False, because though a larger learning rate amplifies gradient and might speed up the process, it is not always the case. A learning rate too large can cause the gradient to be too large and take too big a step every time that it actually increases the L(θ) and misses the point of convergence.

2 Ridge Regression

2.1 (a)

Assuming m data points, posterior $P(\theta|y,X) = \frac{P(y|\theta,X)P(\theta)}{P(y)} = \frac{P(y|\theta,X)P(\theta)}{\int P(y|\theta,X)P(\theta)d\theta}$, after implementing Gaussian equation and MLE, we get:

$$log(P(\theta|y,X) = -C - \frac{(y-X\theta)^T(y-X\theta)}{2\sigma^2} - \frac{\theta^T\theta}{2\tau^2},$$
(C denotes all terms without θ)
$$\therefore \hat{\theta} = (XX^T + \frac{\sigma^2}{\tau^2}I)^{-1}Xy$$

When $\lambda = \frac{\sigma^2}{\tau^2}$, $\hat{\theta}$ would be equivalent to that in ridge regression.
With Gaussian assumption, the covariance satisfies $\Sigma^{-1} = \frac{1}{\sigma^2}(X^TX + \frac{\sigma^2}{\tau^2}I)$, this gives $\hat{\theta} = \frac{1}{\sigma^2}\Sigma X^T y$ which equates to the mean of $P(y|\theta,X)$.

2.2 (b)

Denote that $\theta^{n_i k_i} = (X_k X_k^T + \lambda_n I)X_k y$,$(\lambda_n \in \Lambda)$
calcuate MSE_n: $argmin_n L(\theta^{n_i k_i})_n = \frac{1}{mk}\Sigma_k\Sigma_m(y^i - \theta^{n_i k_i}x_k^i)^2$, $(k \in [1,2,...,K])$.

Separate the training set to K dividends,then calculate the average MSE of all K dividends with a given λ_i. Finally, choose the λ_i with the smallest corresponding average MSE as the optimal hyper-parameter.

3 Bayes Classifier

3.1 Joint Bayes vs Naive Bayes Classifier

3.1.1 (a. Joint Bayes Classifier)

(i)
$P(K=1|a=1\cap b=1\cap c=0) = \frac{P(a=1\cap b=1\cap c=0|K=1)P(K=1)}{P(a=1\cap b=1\cap c=0)} = \frac{\frac{1}{5}*\frac{5}{9}}{\frac{2}{9}} = \frac{1}{2}$
(ii)
$P(K=0|a=1\cap b=1) = \frac{P(a=1\cap b=1|K=0)P(K=0)}{P(a=1\cap b=1)} = \frac{\frac{1}{4}*\frac{4}{9}}{\frac{1}{3}} = \frac{1}{3}$

3.1.2 (b. Naive Bayes Classifier)

(i)
$P(K=1|a=1\cap b=1\cap c=0)$
$= P(a=1|K=1)P(b=1|K=1)P(c=0|K=1)P(K=1)$
$= \frac{4}{5}*\frac{2}{5}*\frac{2}{5}*\frac{5}{9}$
$= \frac{16}{225}$

(ii)
$P(K=0|a=1\cap b=1)$
$= P(a=1|K=0)P(b=1|K=0)P(K=0)$
$= \frac{1}{4}*\frac{3}{4}*\frac{4}{9}$
$= \frac{1}{12}$

3.2 *Bayes Classifier with Gaussian Class Conditional Distribution

$log(p(x|y=k)) = -\frac{n}{2}log(2\pi) - \frac{1}{2}\Sigma_k^{-1} - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)$

$\therefore f(x) = sign[-\frac{1}{2}(\Sigma_1^{-1}-\Sigma_2^{-1}) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) + log(P(y=1) - log(P(y=-1))]$

$= sign[-\frac{1}{2}(\Sigma_1^{-1}-\Sigma_2^{-1}) - \frac{1}{2}[x^T\Sigma_1^{-1}x - x^T\Sigma_1^{-1}\mu_1 - \mu_1^T\Sigma_1^{-1}x + \mu_1^T\Sigma_1^{-1}\mu_1] + \frac{1}{2}[x^T\Sigma_2^{-1}x - x^T\Sigma_2^{-1}\mu_2 - \mu_2^T\Sigma_2^{-1}x + \mu_2^T\Sigma_2^{-1}\mu_2] + log(P(y=1)) - log(P(y=-1))]$

$\because$ covariance matrix is symmetric, $x^T\Sigma_k^{-1}\mu_k = x\Sigma_k^{-1}\mu_k^T$

$= sign[-\frac{1}{2}(\Sigma_1^{-1} - \Sigma_2^{-1}) + x^T\Sigma_1^{-1}\mu_1 - \frac{1}{2}[x^T\Sigma_1^{-1}x + \mu_1^T\Sigma_1^{-1}\mu_1] + -x^T\Sigma_2^{-1}\mu_2 + \frac{1}{2}[x^T\Sigma_2^{-1}x + \mu_2^T\Sigma_2^{-1}\mu_2] + log(P(y=1)) - log(P(y=-1))]$

3.2.1 (a)

Under Gaussian with non-identical covariance matrix, the boundary would be a log-based curve.

3.2.2 (b)

If $\Sigma_1^{-1} = \Sigma_2^{-1}$, which means that covariance is class independent, then f(x) can be transformed to a linear expression. To explain geometrically on a 2-D plane, it would be two different center points(mean) having the same contour lines, making the decision boundary that satisfies equal distance to these contour lines of two classes equivalent. Hence, it would be a linear boundary.

3.2.3 (c)

If the covariance matrix is an identity matrix, then the covariance between classes are zero, meaning each class is independent. Then the decision boundary would presumably be a horizontal or vertical line on a 2-D plane.

4 Basics of Optimization

4.1 (a)

$L(\lambda x + (1-\lambda)y) = log[1 + exp(-(\lambda x + (1-\lambda)y))]$
$L(\lambda x) + L((1-\lambda)y) = log[(1+exp(-\lambda x))(1+exp(-(1-\lambda)y))] = log[1 + exp(-\lambda x) + exp(-(1-\lambda)y) + exp(-\lambda x)exp((1-\lambda)y)]$
$\because$ exponential function is convex: $\exp(-\lambda x) + exp(-(1-\lambda)y) > exp[-(\lambda x + (1-\lambda)y]$, and that $exp(-\lambda x)exp((1-\lambda)y) > 0$,
$\therefore L(\lambda x) + L((1-\lambda)y) > L(\lambda x + (1-\lambda)y)$, logistic loss is a convex function.

4.2 (b)

For concave function, it has $f(\alpha_i x_i + (1-\alpha_i)x_{i+1})) \geqslant \alpha_i f(x_i) + (1-\alpha_i)f(x_{i+1})$

With $f(\Sigma_n \alpha_i x_i) = f((1-\alpha_n)\frac{\Sigma_{n-1}\alpha_i x_i}{1-\alpha_n} + \alpha_n x_n) \geqslant (1-\alpha_n)f(\frac{\Sigma_{n-1}\alpha_i x_i}{1-\alpha_n}) + \alpha_n f(x_n)$,

where $f(\frac{\Sigma_{n-1}\alpha_i x_i}{1-\alpha_n})$ can be seen as another $f(\Sigma_n \alpha_i' x_i)$ with $\Sigma\alpha_i' = 1$.

By doing the first step again, we would have:
$f(\Sigma_n \alpha_i x_i) \geqslant (1-\alpha_n) f((1-\frac{\alpha_{n-1}}{1-\alpha_{n-1}})\frac{\Sigma_{n-2}\alpha_i x_i}{1-\frac{\alpha_{n-1}}{1-\alpha_{n-1}}} + \frac{\alpha_{n-1}x_{n-1}}{1-\alpha_n}]) + \alpha_n x_n = (1-\alpha_n)(1-\frac{\alpha_{n-1}}{1-\alpha_{n-1}}) f(\frac{\Sigma_{n-2}\alpha_i x_i}{\frac{\alpha_{n-1}}{1-\alpha_{n-1}}}) + \alpha_{n-1}x_{n-1} + \alpha_n f(x_n)$,
and by doing this iteratively, it would eventually expand into $\Sigma_n \alpha_i f(x_i)$
hence proving $f(\Sigma_n \alpha_i x_i) \geqslant \Sigma_n \alpha_i f(x_i)$

5 Logistic Regression

5.1 (a)

Assuming Y ∈ (0,1), then $P[Y=0|X=x] = 1 - P[Y=1|X=x]$.
$\therefore ln(\frac{P[Y=1|X=x]}{P[Y=0|X=x]}) = ln(\frac{\frac{exp(w_0+w^T x)}{1+exp(w_0+w^T x)}}{\frac{1}{1+exp(w_0+w^T x)}}) = ln(exp(w_0+w^T x)) = w_0 + w^T x$

5.2 (b)

$$
\begin{aligned}
L(\theta) &= log \sqcap_m \sqcap_C \frac{exp(\theta_c^T x^i)}{\Sigma_C exp(\theta_{c'}^T x^i)} \\
&= \Sigma_m \Sigma_C log \frac{exp(\theta_c^T x^i)}{\Sigma_C exp(\theta_{c'}^T x^i)} \\
&= \Sigma_m \Sigma_C [\theta_c^T x^i - \sqcap_C \theta_{c'}^T x^i] \\
l(\theta) &= -L(\theta) \\
\frac{\partial l}{\partial \theta_c} &= -\Sigma_m I(y_i = c)[x^i - \sqcap_C x^i]
\end{aligned}
$$

6 Programming report

6.1 (a)

$\frac{\partial E(U,V)}{\partial U_{v,k}} = \Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k V_{i,k} + \Sigma_k 2U_{u,k}V_{i,k}^2)$
$\therefore$ Update: $U_{v,k} = U_{v,k} - \mu * \Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k V_{i,k} + \Sigma_k 2U_{u,k}V_{i,k}^2)$

$\frac{\partial E(U,V)}{\partial V_{v,k}} = \Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k U_{u,k} + \Sigma_k 2V_{i,k}U_{u,k}^2)$
$\therefore$ Update: $V_{v,k} = V_{v,k} - \mu * \Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k U_{u,k} + \Sigma_k 2V_{i,k}U_{u,k}^2)$

6.2 (b)

$\frac{\partial E(U,V)}{\partial U_{v,k}} = \Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k V_{i,k} + \Sigma_k 2U_{u,k}V_{i,k}^2) + 2\lambda\Sigma_{u,k}U_{u,k}$
$\therefore$ Update: $U_{v,k} = U_{v,k} - \mu * [\Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k V_{i,k} + \Sigma_k 2U_{u,k}V_{i,k}^2) + 2\lambda\Sigma_{u,k}U_{u,k}]$

$\frac{\partial E(U,V)}{\partial V_{v,k}} = \Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k U_{u,k} + \Sigma_k 2V_{i,k}U_{u,k}^2) + 2\lambda\Sigma_{i,k}V_{i,k}$
$\therefore$ Update: $V_{v,k} = V_{v,k} - \mu * [\Sigma_{(u,i)\in M}(-2M_{u,i}\Sigma_k U_{u,k} + \Sigma_k 2V_{i,k}U_{u,k}^2) + 2\lambda\Sigma_{i,k}V_{i,k}]$

6.3 (c) Report

6.3.1 Results

[training-set accuracy, test-set accuracy, runtime]
1. max iter=1000, learning rate=0.0000001, λ=0.05
SVD-noReg-1 3.6133, 3.6029, 5.23
SVD-withReg-1 3.6149, 3.6048, 4.86
SVD-noReg-3 3.5473, 3.5355, 13.64
SVD-withReg-3 3.5460, 3.5344, 13.65
SVD-noReg-5 3.5400, 3.5290, 15.27
SVD-withReg-5 3.5405, 3.5296, 15.25
SVD-noReg-15 3.5370, 3.5261, 16.10
SVD-withReg-15 3.5378, 3.5267, 16.06

2. max iter=1000, learning rate=0.0000001, λ=0.01
SVD-noReg-1 3.6097, 3.6002, 5.69
SVD-withReg-1 3.6275, 3.6147, 6.74
SVD-noReg-3 3.5479, 3.5376, 31.10
SVD-withReg-3 3.5493, 3.5378, 28.40
SVD-noReg-5 3.5418, 3.5308, 34.79
SVD-withReg-5 3.5420, 3.5311, 34.16
SVD-noReg-15 3.5376, 3.5269, 36.71
SVD-withReg-15 3.5364, 3.5254, 36.55

6.3.2 Choosing parameters

Choosing μ:
A common for learning rate would be 0.001, however, the gradient seems to be positive in the beginning and could easily go over the value range of float in python. Therefore, I chose a significantly smaller learning rate of 0.0000001 in case the code crashes. Then I increased the number of iteration to 1000 to make sure of convergence. Both final gradients in every run is smaller than |0.1|.
Choosing λ:
A common λ for L2 normalization would be 0.01, but since there are k*(943+1682) parameters, I assumed that there could be severe over-fitting. However, at least up to k=15 (bound by computing power), no significant differences in train accuracy and test accuracy were found.