Homework 3

Chujie Chen
cchen641@gatech.edu

November 13, 2019

1 Linear Regression

(a)

Since

$$\begin{aligned}\hat{\theta} &= (X^TX)^{-1}X^TY \\ &= (X^TX)^{-1}X^T(\theta^TX+\epsilon) \\ &= \theta^T + (X^TX)^{-1}X^T\epsilon\end{aligned}$$

We can get

$$E(\hat{\theta}) = \theta^T + (X^TX)^{-1}X^TE(\epsilon) = \theta^T$$

(b)

$$\begin{aligned}Var[\hat{\theta}] &= E[(\hat{\theta} - \bar{\hat{\theta}}^2] \\ &= E[\hat{\theta}^2] - E[\hat{\theta}]^2 \\ &= E[\theta^T\theta + (X^TX)^{-1}X^T\epsilon^2X(XX^T)^{-1}] - E[\hat{\theta}]^2 \\ &= E[(XX^T)^{-1}\epsilon^2] \\ &= (XX^T)^{-1}E(\epsilon^2) \\ &= (XX^T)^{-1}\sigma^2\end{aligned}$$

(c)

Yes. For

$$\hat{\theta} = (X^TX)^{-1}X^TY$$

Y is random and ϵ follows $\mathcal{N}(0, \sigma^2 I)$. So $\hat{\theta}$ is also a random variable. So $\hat{\theta}$ follows a Gaussian distribution $\sim \mathcal{N}(\theta^T, (XX^T)^{-1}\sigma^2)$.

(d)

For

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(y^{(i)} - \theta^Tx^{(i)})^2}{2\sigma_i^2}\right)$$

We can have the likelihood function

$$L(\theta) = \Pi_{i=1}^n p(y^{(i)}|x^{(i)};\theta)$$

The log-likelihood function

$$\mathcal{L} = const - \sum_i \frac{(y^{(i)} - \theta^Tx^{(i)})^2}{2\sigma_i^2} = const. - \frac{1}{2}(Y - \theta^TX)^T\Sigma^{-1}(Y - \theta^TX)$$

Take derivative we can have

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta} &= -\frac{1}{2}[-X^T\Sigma^{-1}(Y-\theta^T X)+(Y-\theta^T X)^T\Sigma^{-1}(-X)] \\ &= X^T\Sigma^{-1}Y - X^T\Sigma^{-1}X\theta \quad \hat{=} \quad 0\end{aligned}$$

We can get

$$\hat{\theta} = (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1}Y$$

2 Ridge Regression

References: Ref1 and Ref2. Given what we have, we can get the posteriror distribution

$$p(\theta|y,X) = \frac{1}{Z}p(y|\theta,X)p(\theta)$$

where $Z = \int p(y|\theta,X)p(\theta)d\theta$ is the normalization constant that does not dependent on θ. If we write the log-posterior distribution explicitly, we have

$$\begin{aligned}logp(\theta|y,X) &= log(\frac{1}{Z}\frac{1}{\sqrt{(2\pi)^n}\sigma^n}exp[-\frac{1}{2\sigma^2}(y-X\theta)^T(y-X\theta)]\frac{1}{\sqrt{(2\pi)^n}\tau}exp[-\frac{\theta^T\theta}{2\tau^2}]) \\ &= const. - \frac{1}{2\sigma^2}(y-X\theta)^T(y-X\theta) - \frac{\theta^T\theta}{2\tau^2}\end{aligned}$$

Then, we can get the first derivative and set it to be zero:

$$\frac{dlogp(\theta|y,X)}{d\theta} = \frac{1}{\sigma^2}X^T(y-X\theta) - \frac{\theta}{\tau^2} \quad \hat{=} \quad 0$$

We will get

$$\hat{\theta} = (X^TX + \frac{\sigma^2}{\tau^2}I)^{-1}X^Ty$$

We can see this is what we have for ridge regression estimate with $\frac{\sigma^2}{\tau^2} = \lambda$. More specifically, when σ^2 is small (we convince ourselves that the θ is close to zero), then λ is large (large θ is penalized). When σ^2 is small (the observed y are not noisy), we focus on fitting the data (small λ). In a similar way, we can show that the posterior distribution is a Gaussian. Let's say it has mean μ and covariance Σ.

We can only look at the exponential term EXP, from above, we have

$$EXP = -\frac{1}{2\sigma^2}(y-X\theta)^T(y-X\theta) - \frac{\theta^T\theta}{2\tau^2}$$

While for a Gaussian we define, we have

$$EXP = -\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu) = -\frac{1}{2}(\theta^T\Sigma^{-1}\theta - 2\theta^T\Sigma^{-1}\mu + \mu^T\Sigma^{-1}\mu)$$

We can compare the second order θ terms from above two equations and get

$$\theta^T\Sigma^{-1}\theta = \frac{1}{\sigma^2}\mu^TX^TX\mu + \frac{1}{\tau^2}\mu^T\mu = \frac{1}{\sigma^2}\mu^T(X^TX + \frac{\sigma^2}{\tau^2}I)\mu$$

Thus,

$$\Sigma^{-1} = \frac{1}{\sigma^2}[X^TX + \frac{\sigma^2}{\tau^2}I]$$

Then we can get μ by comparing the first order terms:

$$\mu = (X^TX + \frac{\sigma^2}{\tau^2}I)^{-1}X^Ty$$

So we can see that it is truly the $\hat{\theta}$.

3 Bayes Classifier

3.1 Bayes Classifier With General Loss Function

References: Ref3 and Ref4. If we set a_i as the action (predicted label) and c_j as the true class (intrinsic label). We can have Bayes error function

$$R(a_i|x) = \sum_j L(a_i, c_j)p(c_j|x)$$

Then, our classifier function be

$$\hat{a}_i = \underset{a_i}{\operatorname{argmin}} R(a_i|x)$$

That is to say

$$R(y = +1|x) = L(a, a)p(y = +1|x) + L(a, b)p(y = -1|x) = L(a, b)p(y = -1|x)$$

$$R(y = -1|x) = L(b, a)p(y = +1|x) + L(b, b)p(y = -1|x) = L(b, a)p(y = +1|x)$$

Thus, if we compare above two, if

$$p(y = -1|x)L(a, b) < p(y = +1|x)L(b, a)$$

We take action +1. That is to say, the classifier function is

$$a(x) = sign[\frac{p(x|y = +1)p(y = +1)L(1, -1)}{p(x|y = -1)p(y = -1)L(-1, 1)}]$$

3.2 Gaussian Class Conditional distribution

(a)

From above analysis, we need $p(x|y_i)$ and $p(y_i)$. For class conditional distribution

$$p(x|y_i) = \frac{\alpha}{\sqrt{|\Sigma_y|}} \exp[-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)]$$

where α is a normalization based on the data dimension. Also

$$p(Y = y) = \frac{1}{N}\sum_{i=1}^{N} H(y^i = y) \quad \hat{=} \quad \frac{\hat{n}_y}{N}$$

where H is the Heaviside step function. Then we can rewrite what we had above.

$$\begin{aligned} a(x) &= sign[log\frac{p(x|y = +1)p(y = +1)L(1, -1)}{p(x|y = -1)p(y = -1)L(-1, 1)}] \\ &= sign[-\frac{1}{2}(x - \mu_{+1})^T\Sigma_{+1}^{-1}(x - \mu_{+1}) + \frac{1}{2}(x - \mu_{-1})^T\Sigma_{-1}^{-1}(x - \mu_{-1}) - \frac{1}{2}log\frac{det(\Sigma_+1)}{det(\Sigma_{-1})} + log\frac{n_{+1}}{n_{-1}} + log\frac{L(1, -1)}{L(-1, 1)}] \end{aligned}$$

Since there is a quadratic term, the boundary is a curved surface in high dimension space.

(b)

We let $\Sigma_{+1} = \Sigma_{-1} = \Sigma$

$$a(x) = sign[(\mu_{+1} - \mu_{-1})^T\Sigma^{-1}X - \frac{1}{2}\mu_{+1}^T\Sigma^{-1}\mu_{+1} + \frac{1}{2}\mu_{-1}^T\Sigma^{-1}\mu_{-1} + log\frac{n_{+1}}{n_{-1}} + log\frac{L(1, -1)}{L(-1, 1)}]$$

which is propotional to X, so it is a hyper-plane in high dimension space.

(c)

Similarly, but we let $\Sigma = I$, then

$$a(x) = sign[(\mu_{+1} - \mu_{-1})^T X - \frac{1}{2}\mu_{+1}^T \mu_{+1} + \frac{1}{2}\mu_{-1}^T \mu_{-1} + log\frac{n_{+1}}{n_{-1}} + log\frac{L(1,-1)}{L(-1,1)}]$$

As described in b, it is still a hyper-plane in high dimension space of which the direction is determined by $(\mu_{+1} - \mu_{-1})^T$

4 Logistic Regression

(a)

$$\begin{aligned} ln(\frac{P(Y=1|X=x)}{P(Y=0|X=x)} &= ln(\exp{(w_0 + w^T x)}) \\ &= w_0 + w^T x \end{aligned}$$

(b)

We can get the derivatives

$$\begin{aligned} \frac{dL(z)}{dz} &= \frac{-\exp{(-z)}}{1 + exp(-z)} \\ \frac{d^2 L}{dz^2} &= \frac{exp(-z)}{[1 + exp(-z)]^2} > 0 \end{aligned}$$

So it is a convex function.

5 Programming: Recommendation System

(a)

To avoid confusion raised by parameters, we can get derivatives first:

$$\begin{aligned} \frac{\partial E(U,V)}{\partial U_{p,q}} &= -2 \sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k}) V_{i,q} \\ \frac{\partial E(U,V)}{\partial V_{p,q}} &= -2 \sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k}) U_{i,q} \end{aligned}$$

Then our update formula become:

$$U_{v,k} \leftarrow U_{v,k} + 2\mu \sum_{(u,i)\in M} (M_{u,i} - \sum_{q=1}^{r} U_{u,q} V_{i,q}) V_{i,k}$$

$$V_{j,k} \leftarrow V_{j,k} + 2\mu \sum_{(u,i)\in M} (M_{u,i} - \sum_{q=1}^{r} U_{u,q} V_{i,q}) U_{u,k}$$

(b)

Similarly, we can get derivatives:

$$\begin{aligned} \frac{\partial E(U,V)}{\partial U_{p,q}} &= -2 \sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k}) V_{i,q} + 2\lambda U_{p,q} \\ \frac{\partial E(U,V)}{\partial V_{p,q}} &= -2 \sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k}) U_{i,q} + 2\lambda V_{p,q} \end{aligned}$$

So the update formula become:

$$U_{v,k} \leftarrow U_{v,k} + 2\mu \sum_{(u,i)\in M} (M_{u,i} - \sum_{q=1}^{r} U_{u,q}V_{i,q})V_{i,k} - 2\mu\lambda U_{v,k}$$

$$V_{j,k} \leftarrow V_{j,k} + 2\mu \sum_{(u,i)\in M} (M_{u,i} - \sum_{q=1}^{r} U_{u,q}V_{i,q})U_{u,k} - 2\mu\lambda V_{j,k}$$

(c) Report

The performance on varied lowRank are below:

Rank	1			3			5		
RMSE_train	0.9254	0.9253	0.9251	0.8648	0.8655	0.8663	0.8221	0.8269	0.8225
RMSE_test	0.9535	0.9546	0.9523	0.9313	0.9430	0.9333	0.9493	0.9513	0.9450
Runtime (s)	4.32	4.48	4.16	12.40	12.70	12.32	14.78	14.58	16.65
RMSE_train_avg	0.9253			0.8655			0.8238		
RMSE_test_avg	0.9335			0.9359			0.9485		
Avg. runtime (s)	4.32			12.47			15.34		

Observation:

1. RMSE_train and RMSE_test:

The higher the rank (more factors) is, the better our model describe the train dataset, while it doesn't necessarily mean that we can predict test dataset better. In other words, for higher rank, we might have a better chance to fit the data better in the meantime it is also more likely to overfit the train set.

2. Runtime:

Clearly, lower the rank is, shorter the runtime is. That is because even the gradient descent converges fast, for higher rank, it takes longer time to converge.

Hyper-paramters:

1.μ - learning rate:

With a higher learning rate, we can cover more ground each step while we are take the risk of overshooting the lowest point. With a smaller learning rate, we can get more precise result (closer to local minimum in most cases), but it is time-consuming. To decide the μ I am using, I compare the iterations and runtime required for the objective function getting converged. By defining the loop:

```
threshold = 10^(-4) * numberOfNonzeroElementsInMatrix;
while(changeOfObjectiveFunction > threshold && iteration < maxIter)
    UpdateObjectiveFunction;
    iteration++;
end
```

With my learning rate, rank 1 would stop after $\sim$ 300 iterations, rank 3 would stop after $\sim$ 900 iterations and rank 5 would stop after about $\sim$ 1200 with acceptable runtimes. That's how I chose my learning rate. In other words, I chose this value to make the program run in an affordable time and produce well converged results as well. Increasing it would make the result not precise enough but fast, decreasing it would make the runtime longer.

2.λ - regularizer:

Small regularizer would make the model fits the train set better but might induce over-fitting. Larger regularizer would mitigate over-fitting but would weaken the capability of prediction.

Practically, when $RMSE_{train}$ is way lower than $RMSE_{test}$, that is a sign of over-fitting, I will enlarge the regularizer. When both RMSE got compromised badly, I will choose smaller λ.