

CS 7641 CSE/ISYE 6740 Homework 3

Deadline: 11/07 Mon, 11:55pm

- Submit your answers as an electronic copy on T-square.
- No unapproved extension of deadline is allowed. Zero credit will be assigned for late submissions. Email request for late submission may not be replied.
- For typed answers with LaTeX (recommended) or word processors, extra credits will be given. If you handwrite, try to be clear as much as possible. No credit may be given to unreadable handwriting.
- Explicitly mention your collaborators if any.
- Recommended reading: PRML¹ Section 3.1, 3.2

1 Linear Regression [30 pts]

In class, we derived a closed form solution (normal equation) for linear regression problem: $\hat{\theta} = (X^T X)^{-1} X^T Y$. A probabilistic interpretation of linear regression tells us that we are relying on an assumption that each data point is actually sampled from a linear hyperplane, with some noise. The noise follows a zero-mean Gaussian distribution with constant variance. Specifically,

$$Y^i = \theta^T X^i + \epsilon^i \quad (1)$$

where $\epsilon^i \sim \mathcal{N}(0, \sigma^2 I)$, $\theta \in \mathbb{R}^d$, and $\{X^i, Y^i\}$ is the i -th data point. In other words, we are assuming that each every point is independent to each other and that every data point has same variance.

(a) Using the normal equation, and the model (Eqn. 1), derive the expectation $\mathbb{E}[\hat{\theta}]$. Note that here X is fixed, and only Y is random, i.e. “fixed design” as in statistics. [6 pts]

Solution:

Combine the normal equation and equation 1, we can obtain:

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

According to the linear property, $E(A + BZ) = A + BE(Z)$, besides, X is fixed and only Y is random. $E(\epsilon) = 0$ and $\theta \in \mathbb{R}^d$. Thus we can obtain:

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

¹Christopher M. Bishop, Pattern Recognition and Machine Learning, 2006, Springer.

(b) Similarly, derive the variance $\text{Var}[\hat{\theta}]$. [6 pts]

Solution: According to linear property, $\text{Var}(A + BZ) = \text{Var}(BZ) = B\text{Var}(z)B^T$

$$\begin{aligned}\text{Var}(\hat{\theta}) &= \text{Var}((X^T X)^{-1} X^T Y) \\ &= (X^T X)^{-1} X^T \text{Var}(X\theta + \epsilon) ((X^T X)^{-1} X^T)^T \\ &= (X^T X)^{-1} X^T \text{Var}(\epsilon) ((X^T X)^{-1} X^T)^T \\ &= (X^T X)^{-1} X^T (X^T X)^{-1} X^T \sigma^2 I \\ &= (X^T X)^{-1} X^T X ((X^T X)^{-1})^T \sigma^2 I \\ &= \sigma^2 ((X^T X)^{-1})^T \\ &= \sigma^2 (X^T X)^{-1}\end{aligned}$$

(c) Under the white noise assumption above, someone claims that $\hat{\theta}$ follows Gaussian distribution with mean and variance in (a) and (b), respectively. Do you agree with this claim? Why or why not? [8 pts]

Answer:

I agree with this claim. Since $\hat{\theta} = (X^T X)^{-1} X^T (X\theta + \epsilon)$ and X is fixed, θ is constant. Thus, $\hat{\theta}$ is a linear combination of θ and ϵ . Because $\epsilon^i \sim \mathcal{N}(0, \sigma^2 I)$, then $\hat{\theta}$ follows Gaussian distribution with mean and variance in (a) and (b).

(d) **Weighted linear regression**

Suppose we keep the independence assumption but remove the same variance assumption. In other words, data points would be still sampled independently, but now they may have different variance σ_i . Thus, the covariance matrix of Y would be still diagonal, but with different values:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}. \quad (2)$$

Derive the estimator $\hat{\theta}$ (similar to the normal equations) for this problem using matrix-vector notations with Σ . [10 pts]

Solution:

According to equation (1), since $\epsilon^i \sim \mathcal{N}(0, \sigma_i^2 I)$, then $Y^i \sim \mathcal{N}(0, \sigma_i^2 I)$. Thus, we can interpret the LMS from the probabilistic view.

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

By independence assumption, likelihood is:

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

Hence the log-likelihood is:

$$\log L(\theta) = \sum_{i=1}^n \log \frac{1}{\sqrt{2\pi}\sigma_i} - \sum_{i=1}^n \frac{(y^i - \theta^T x^i)^2}{2\sigma_i^2}$$

Since the first instance is constant, and the latter part is close to the LMS form and we can maximize log-likelihood by minimizing $\sum_{i=1}^n \frac{(y^i - \theta^T x^i)^2}{2\sigma_i^2}$. Let $T(\theta) = \sum_{i=1}^n \frac{(y^i - \theta^T x^i)^2}{2\sigma_i^2}$, then set gradient to 0 and find parameter.

$$\begin{aligned} \frac{\partial T(\theta)}{\partial \theta} &= - \sum_{i=1}^n \frac{(y^i - \theta^T x^i)}{\sigma_i^2} x^i = 0 \\ \Leftrightarrow - \sum_{i=1}^n \frac{y^i x^i}{\sigma_i^2} + \sum_{i=1}^n \frac{x^i x^{iT} \theta}{\sigma_i^2} &= 0 \end{aligned}$$

Define $X = (x^1, x^2 \dots x^n)$, $Y = (y^1, y^2 \dots y^n)^T$, gradient becomes:

$$\begin{aligned} -X^T \Sigma^{-1} Y + X^T \Sigma^{-1} X \theta &= 0 \\ \Rightarrow \hat{\theta} &= (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} Y \end{aligned}$$

2 Ridge Regression [15 pts]

For linear regression, it is often assumed that $y = \theta^T \mathbf{x} + \epsilon$ where $\theta, \mathbf{x} \in \mathbb{R}^m$ by absorbing the constant term, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. Given n i.i.d samples $(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^n, y^n)$, we define $\mathbf{y} = (y^1, \dots, y^n)^T$ and $X = (\mathbf{x}^1, \dots, \mathbf{x}^n)^T$. Thus, we have $\mathbf{y} \sim \mathcal{N}(X\theta, \sigma^2 I)$. Show that the ridge regression estimate is the mean of the posterior distribution under a Gaussian prior $\theta \sim \mathcal{N}(0, \tau^2 I)$. Find the explicit relation between the regularization parameter λ in the ridge regression estimate of the parameter θ , and the variances σ^2, τ^2 .

Solution:

According to the Bayes rule, we can obtain:

Prior is $p(\theta) = \mathcal{N}(\theta|\tau^2)$, likelihood is $p(y|\theta) = \mathcal{N}(y|\sigma^2)$, normalization constant is $p(y) = \sum' \mathcal{N}(\theta|\tau^2) \mathcal{N}(y|\sigma^2)$, let $C = p(y)$.

$$\begin{aligned} \text{Posterior : } p(\theta|y) &= \frac{p(y|\theta)p(\theta)}{p(y)} \\ &= \frac{\mathcal{N}(y|\sigma^2) \mathcal{N}(\theta|\tau^2)}{\sum' \mathcal{N}(\theta|\tau^2) \mathcal{N}(y|\sigma^2)} \\ &= \frac{1}{C} \mathcal{N}(y|X\theta, \sigma^2 I) \mathcal{N}(\theta|0, \tau^2) \end{aligned}$$

$$\begin{aligned}
p(\theta|y) &= \frac{1}{C} \frac{1}{(\sqrt{2\pi}\sigma^2)^n} \exp\left(-\frac{(y - X\theta)^T(y - X\theta)}{2\sigma^2}\right) \frac{1}{(\sqrt{2\pi}\tau^2)^n} \exp\left(-\frac{\theta^T\theta}{2\tau^2}\right) \\
&= \frac{1}{C(\sqrt{2\pi}\sigma^2)^n(\sqrt{2\pi}\tau^2)^n} \exp\left(-\frac{(y - X\theta)^T(y - X\theta)}{2\sigma^2} - \frac{\theta^T\theta}{2\tau^2}\right)
\end{aligned}$$

Log-likelihood:

$$\log p(\theta|y) = -\frac{(y - X\theta)^T(y - X\theta)}{2\sigma^2} - \frac{\theta^T\theta}{2\tau^2} + \text{constant}$$

Take the derivative of the log-likelihood, we have:

$$\begin{aligned}
\frac{\partial \log p(\theta|y)}{\partial \theta} &= \frac{2X^T(y - X\theta)}{2\sigma^2} - \frac{2\theta}{2\tau^2} = 0 \\
&= \frac{X^T(y - X\theta)}{\sigma^2} - \frac{\theta}{\tau^2} = 0 \\
&\Rightarrow \hat{\theta} = (X^T X + \frac{\sigma^2}{\tau^2} I)^{-1} X^T y
\end{aligned}$$

So if we compared it to the ridge regression estimate $(X^T X + \lambda I)^{-1} X^T y$, then let $\lambda = \frac{\sigma^2}{\tau^2}$.

Since prior and likelihood both are Gaussian distribution, thus posterior also follows Gaussian distribution. Then we can define posterior as $p(\theta|y) \sim \mathcal{N}(\mu, \Sigma)$. Then the exponent part of Gaussian equation is:

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

Second order of θ :

$$\theta^T \Sigma^{-1} \theta = \frac{1}{\sigma^2} \theta^T X^T X \theta + \frac{1}{\tau^2} \theta^T \theta = \frac{1}{\sigma^2} \theta^T (X^T X + \frac{\sigma^2}{\tau^2} I) \theta \quad (3)$$

$$\Rightarrow \Sigma^{-1} = \frac{1}{\sigma^2} (X^T X + \frac{\sigma^2}{\tau^2} I) \quad (4)$$

First order of θ with μ :

$$\theta^T \Sigma^{-1} \mu = \frac{1}{\sigma^2} \theta^T X^T y \quad (5)$$

$$\Rightarrow \mu = (X^T X + \frac{\sigma^2}{\tau^2} I) X^T y \quad (6)$$

Therefore the mean of the posterior distribution, μ is equal to the ridge regression estimate. And the explicit relation between regularization parameter λ in the ridge regression estimate and the σ^2, τ^2 is $\lambda = \frac{\sigma^2}{\tau^2}$.

3 Bayes Classifier

3.1 Bayes Classifier With General Loss Function

In class, we talked about the popular 0-1 loss function in which $L(a, b) = 1$ for $a \neq b$ and 0 otherwise, which means all wrong predictions cause equal loss. Yet, in many other cases including cancer detection, the asymmetric loss is often preferred (misdiagnosing cancer as no-cancer is much worse). In this problem, we assume

to have such an asymmetric loss function where $L(a, a) = L(b, b) = 0$ and $L(a, b) = p, L(b, a) = q, p \neq q$. Write down the the Bayes classifier $f : X \rightarrow Y$ for binary classification $Y \in \{-1, +1\}$. Simplify the classification rule as much as you can. [20pts]

Solution:

According to the problem, we can obtain:

$$L(Y_1, Y_2) = \begin{cases} 0 & \text{if } Y_1 = Y_2 = 1 \text{ or } Y_1 = Y_2 = -1 \\ p & \text{if } Y_1 = 1, Y_2 = -1 \\ q & \text{if } Y_1 = -1, Y_2 = 1 \end{cases} \quad (7)$$

The posterior probability of a test point:

$$P(Y = i|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

For $i = 1$, then

$$L_1 = P(Y_1 = 1|X)L(Y_1 = 1, Y_2 = 1) + P(Y_1 = -1|X)L(Y_1 = -1, Y_2 = 1) = P(Y_1 = -1|X)p$$

For $i = -1$, then

$$L_{-1} = P(Y_1 = 1|X)L(Y_1 = 1, Y_2 = -1) + P(Y_1 = -1|X)L(Y_1 = -1, Y_2 = -1) = P(Y_1 = 1|X)q$$

To find a Bayes classifier rule for $f : X \rightarrow Y$ for binary classification $Y \in \{-1, +1\}$, then the Bayes decision rule:

$$\frac{L_1(X)}{L_{-1}(X)} = \frac{P(Y_1 = -1|X)p}{P(Y_1 = 1|X)q} = \frac{P(Y_1 = -1)P(X|Y_1 = -1)p}{P(Y_1 = 1)P(X|Y_1 = 1)q}$$

If test point belongs to 1, then $\frac{L_1(X)}{L_{-1}(X)} < 1$, if test point belongs to -1, then $\frac{L_1(X)}{L_{-1}(X)} > 1$.

Therefore,

$$f(X) = \begin{cases} 1 & \text{if } P(Y_1 = -1|X)p > P(Y_1 = 1|X)q \\ -1 & \text{if } P(Y_1 = -1|X)p < P(Y_1 = 1|X)q \end{cases} \quad (8)$$

3.2 Gaussian Class Conditional distribution

(a) Suppose the class conditional distribution is a Gaussian. Based on the general loss function in problem 3.1, write the Bayes classifier as $f(X) = \text{sign}(h(X))$ and simplify h as much as possible. What is the geometric shape of the decision boundary? [10 pts]

(b) Repeat (a) but assume the two Gaussians have identical covariance matrices. What is the geometric shape of the decision boundary? [10 pts]

(c) Repeat (a) but assume now that the two Gaussians have covariance matrix which is equal to the identity matrix. What is the geometric shape of the decision boundary? [10 pts]

(a)Solution:

According to the problem stated, the class conditional distribution is a Gaussian. Then we can obtain:

$$P(X|Y = i) = P(X|\mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp(-(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)/2)$$

Thus from problem 3.1, we can obtain:

$$\begin{aligned}
g(X) &= \frac{P(Y_1 = 1)P(X|Y_1 = 1)p}{P(Y_1 = -1)P(X|Y_1 = -1)q} \\
&= \frac{P(Y_1 = 1)P(X|\mu_1, \Sigma_1)p}{P(Y_1 = -1)P(X|\mu_{-1}, \Sigma_{-1})q} \\
&= \frac{P(Y_1 = 1)}{P(Y_1 = -1)} \frac{\Sigma_{-1}^{-1/2}}{\Sigma_1^{-1/2}} \exp\left(\frac{-(X - \mu_1)^T \Sigma_1^{-1} (X - \mu_1) + (X - \mu_{-1})^T \Sigma_{-1}^{-1} (X - \mu_{-1})}{2}\right)
\end{aligned}$$

Since if $h(X) > 0$, then $f(X) = \text{sign}(h(X)) = 1$, if $h(X) < 0$, then $f(X) = -1$.

Thus we can let $h(X) = \log(g(X))$:

When $g(X) > 1$, then $h(X) > 0$, $f(X) = 1$, if $g(X) < 1$, then $h(X) < 0$, $f(X) = -1$

Geometric decision boundary:

$$\begin{aligned}
h(X) &= \log(g(X)) = \log\left(\frac{P(Y_1 = 1)}{P(Y_1 = -1)} \frac{\Sigma_{-1}^{-1/2}}{\Sigma_1^{-1/2}}\right) - \frac{1}{2}((X - \mu_1)^T \Sigma_1^{-1} (X - \mu_1) - (X - \mu_{-1})^T \Sigma_{-1}^{-1} (X - \mu_{-1})) = 0 \\
&= \log\left(\frac{P(Y_1 = 1)}{P(Y_1 = -1)} \frac{\Sigma_{-1}^{-1/2}}{\Sigma_1^{-1/2}}\right) - \frac{1}{2}[X^T (\Sigma_1^{-1} - \Sigma_{-1}^{-1})X - (\mu_1^T \Sigma_1^{-1} - \mu_{-1}^T \Sigma_{-1}^{-1})X - X^T (\mu_1 \Sigma_1^{-1} - \mu_{-1} \Sigma_{-1}^{-1}) \\
&\quad - \mu_1^t \Sigma_1^{-1} \mu_1 + \mu_{-1}^t \Sigma_{-1}^{-1} \mu_{-1}]
\end{aligned}$$

Since $\Sigma_1^{-1} \neq \Sigma_{-1}^{-1}$, then obviously the boundary equation has a quadratic part of X , thus the shape of the decision boundary would be a n-dimensional quadratic surface.

(b)Solution:

If two Gaussians have identical covariance matrices, then $\Sigma_1 = \Sigma_{-1}$, the second order part of X is eliminated.

$$h(X) = \text{constant} + \frac{1}{2}[(\mu_1^T - \mu_{-1}^T)\Sigma_1^{-1}X + X^T(\mu_1 - \mu_{-1})\Sigma_1^{-1} + \mu_1^t \Sigma_1^{-1} \mu_1 - \mu_{-1}^t \Sigma_1^{-1} \mu_{-1}]$$

Therefore the shape of the decision boundary would be a n-dimensional plane.

(c)Solution:

If two Gaussians have identical covariance matrix which is equal to the identity matrix, then $\Sigma_1 = \Sigma_{-1} = I$, then:

$$\begin{aligned}
h(X) &= \text{constant} + \frac{1}{2}[(\mu_1^T - \mu_{-1}^T)X + X^T(\mu_1 - \mu_{-1})] \\
&= \text{constant} + [(\mu_1^T - \mu_{-1}^T)X]
\end{aligned}$$

Therefore the shape of the decision boundary would be a n-dimensional space plane which is orthogonal to $\mu_1^T - \mu_{-1}^T$.

4 Logistic Regression

Logistic regression is named after the log-odds of success (the logit of the probability) defined as below:

$$\ln \left(\frac{P[Y = 1|X = x]}{P[Y = 0|X = x]} \right)$$

where

$$P[Y = 1|X = x] = \frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)}$$

(a) Show that log-odds of success is a linear function of X . [6 pts]

Solution:

Since

$$P[Y = 1|X = x] = \frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)}$$

then

$$P[Y = 0|X = x] = 1 - P[Y = 1|X = x] = 1 - \frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)} = \frac{1}{1 + \exp(w_0 + w^T x)}$$

Plug these two equations into the log-odds of success function:

$$\ln \left(\frac{P[Y = 1|X = x]}{P[Y = 0|X = x]} \right) = \ln \left(\frac{\frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)}}{\frac{1}{1 + \exp(w_0 + w^T x)}} \right) = \ln (\exp(w_0 + w^T x)) = w_0 + w^T x$$

Therefore, log-odds of success is a linear function of X .

(b) Show that the logistic loss $L(z) = \log(1 + \exp(-z))$ is a convex function. [9 pts]

Solution:

Second derivative of logistic loss function $L(z)$:

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

Since the second derivative of logistic loss function is always greater than 0, for all z , then logistic function is a convex function.

5 Programming: Recommendation System [40 pts]

Personalized recommendation systems are used in a wide variety of applications such as electronic commerce, social networks, web search, and more. Machine learning techniques play a key role to extract individual preference over items. In this assignment, we explore this popular business application of machine learning, by implementing a simple matrix-factorization-based recommender using gradient descent.

Suppose you are an employee in Netflix. You are given a set of ratings (from one star to five stars) from users on many movies they have seen. Using this information, your job is implementing a personalized rating predictor for a given user on unseen movies. That is, a rating predictor can be seen as a function $f : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$, where \mathcal{U} and \mathcal{I} are the set of users and items, respectively. Typically the range of this function is restricted to between 1 and 5 (stars), which is the the allowed range of the input.

Now, let's think about the data representation. Suppose we have m users and n items, and a rating given by a user on a movie. We can represent this information as a form of matrix, namely rating matrix M . Suppose rows of M represent users, while columns do movies. Then, the size of matrix will be $m \times n$. Each cell of the matrix may contain a rating on a movie by a user. In $M_{15,47}$, for example, it may contain a rating on the item 47 by user 15. If he gave 4 stars, $M_{15,47} = 4$. However, as it is almost impossible for everyone to watch large portion of movies in the market, this rating matrix should be very sparse in nature. Typically, only 1% of the cells in the rating matrix are observed in average. All other 99% are missing values, which means the corresponding user did not see (or just did not provide the rating for) the corresponding movie. Our goal with the rating predictor is estimating those missing values, reflecting the user's preference learned from available ratings.

Our approach for this problem is matrix factorization. Specifically, we assume that the rating matrix M is a low-rank matrix. Intuitively, this reflects our assumption that there is only a small number of factors (e.g. genre, director, main actor/actress, released year, etc.) that determine like or dislike. Let's define r as the number of factors. Then, we learn a user profile $U \in \mathbb{R}^{m \times r}$ and an item profile $V \in \mathbb{R}^{n \times r}$. (Recall that m and n are the number of users and items, respectively.) We want to approximate a rating by an inner product of two length r vectors, one representing user profile and the other item profile. Mathematically, a rating by user u on movie i is approximated by

$$M_{u,i} \approx \sum_{k=1}^r U_{u,k} V_{i,k}. \quad (9)$$

We want to fit each element of U and V by minimizing squared reconstruction error over all training data points. That is, the objective function we minimize is given by

$$E(U, V) = \sum_{(u,i) \in M} (M_{u,i} - U_u^T V_i)^2 = \sum_{(u,i) \in M} (M_{u,i} - \sum_{k=1}^r U_{u,k} V_{i,k})^2 \quad (10)$$

where U_u is the u th row of U and V_i is the i th row of V . We observe that this looks very similar to the linear regression. Recall that we minimize in linear regression:

$$E(\theta) = \sum_{i=1}^m (Y^i - \theta^T x^i)^2 = \sum_{i=1}^m (Y^i - \sum_{k=1}^r \theta_k x_k^i)^2 \quad (11)$$

where m is the number of training data points. Let's compare (10) and (11). $M_{u,i}$ in (10) corresponds to Y^i in (11), in that both are the observed labels. $U_u^T V_i$ in (10) corresponds to $\theta^T x^i$ in (11), in that both are our estimation with our model. The only difference is that both U and V are the parameters to be learned in (10), while only θ is learned in (11). This is where we personalize our estimation: with linear regression, we apply the same θ to any input x^i , but with matrix factorization, a different profile U_u are applied depending on who is the user u .

As U and V are interrelated in (10), there is no closed form solution, unlike linear regression case. Thus, we need to use gradient descent:

$$U_{v,k} \leftarrow U_{v,k} - \mu \frac{\partial E(U, V)}{\partial U_{v,k}}, \quad V_{j,k} \leftarrow V_{j,k} - \mu \frac{\partial E(U, V)}{\partial V_{j,k}}, \quad (12)$$

where μ is a hyper-parameter deciding the update rate. It would be straightforward to take partial derivatives of $E(U, V)$ in (10) with respect to each element $U_{v,k}$ and $V_{j,k}$. Then, we update each element of U and V using the gradient descent formula in (12).

(a) Derive the update formula in (12) by solving the partial derivatives. [10 pts]

Solution:

Since

$$\begin{aligned}\frac{\partial E(U, V)}{\partial U_{v,k}} &= \frac{\partial (M_{v,j} - \sum_{k=1}^r U_{v,k} V_{j,k})^2}{\partial U_{v,k}} \\ &= -2V_{j,k}(M_{v,j} - \sum_{k=1}^r U_{v,k} V_{j,k}) \\ \frac{\partial E(U, V)}{\partial V_{j,k}} &= \frac{\partial (M_{v,j} - \sum_{k=1}^r U_{v,k} V_{j,k})^2}{\partial V_{j,k}} \\ &= -2U_{v,k}(M_{v,j} - \sum_{k=1}^r U_{v,k} V_{j,k})\end{aligned}$$

Thus formula in (6) are updated to:

$$\begin{aligned}U_{v,k} &\leftarrow U_{v,k} - \mu \frac{\partial E(U, V)}{\partial U_{v,k}} \\ &= U_{v,k} + 2\mu V_{j,k}(M_{v,j} - \sum_{k=1}^r U_{v,k} V_{j,k}) \\ V_{j,k} &\leftarrow V_{j,k} - \mu \frac{\partial E(U, V)}{\partial V_{j,k}} \\ &= V_{j,k} + 2\mu U_{v,k}(M_{v,j} - \sum_{k=1}^r U_{v,k} V_{j,k})\end{aligned}$$

(b) To avoid overfitting, we usually add regularization terms, which penalize for large values in U and V . Redo part (a) using the regularized objective function below. [5 pts]

$$E(U, V) = \sum_{(u,i) \in M} (M_{u,i} - \sum_{k=1}^r U_{u,k} V_{i,k})^2 + \lambda \sum_{u,k} U_{u,k}^2 + \lambda \sum_{i,k} V_{i,k}^2$$

(λ is a hyper-parameter controlling the degree of penalization.)

Solution:

We can just add the derivative of the addition parts to the previous derivative results.

$$\begin{aligned}\frac{\partial E(U, V)}{\partial U_{v,k}} &= -2V_{j,k}(M_{v,j} - \sum_{l=1}^r U_{v,l} V_{j,l}) + 2\lambda U_{v,k} \\ \frac{\partial E(U, V)}{\partial V_{j,k}} &= -2U_{v,k}(M_{v,j} - \sum_{l=1}^r U_{v,l} V_{j,l}) + 2\lambda V_{j,k}\end{aligned}$$

Thus the new formulas in (6) are:

$$\begin{aligned}U_{v,k} &\leftarrow U_{v,k} + 2\mu V_{j,k}(M_{v,j} - \sum_{l=1}^r U_{v,l} V_{j,l}) - 2\lambda\mu U_{v,k} \\ V_{j,k} &\leftarrow V_{j,k} + 2\mu U_{v,k}(M_{v,j} - \sum_{l=1}^r U_{v,l} V_{j,l}) - 2\lambda\mu V_{j,k}\end{aligned}$$

(c) **Implement `myRecommender.m` by filling the gradient descent part.**

You are given a skeleton code `myRecommender.m`. Using the training data `rateMatrix`, you will implement your own recommendation system of rank `lowRank`. The only file you need to edit and submit is `myRecommender.m`. In the gradient descent part, repeat your update formula in (b), observing the average reconstruction error between your estimation and ground truth in training set. You need to set a stopping criteria, based on this reconstruction error as well as the maximum number of iterations. You should play with several different values for μ and λ to make sure that your final prediction is accurate.

Formatting information is here:

Input

- **rateMatrix**: training data set. Each row represents a user, while each column an item. Observed values are one of $\{1, 2, 3, 4, 5\}$, and missing values are 0.
- **lowRank**: the number of factors (dimension) of your model. With higher values, you would expect more accurate prediction.

Output

- **U**: the user profile matrix of dimension user count \times low rank.
- **V**: the item profile matrix of dimension item count \times low rank.

Evaluation [15 pts]

Upload your `myRecommender.m` implementation file. (Do not copy and paste your code in your report. Be sure to upload your `myRecommender.m` file.)

To test your code, try to run `homework3.m`. You may have noticed that the code prints both training and test error, in RMSE (Root Mean Squared Error), defined as follows:

$$\sum_{(u,i) \in M} (M_{u,i} - f(u,i))^2$$

where $f(u,i)$ is your estimation, and the summation is over the training set or testing set, respectively. For the grading, we will use another set-aside testing set, which is not released to you. If you observe your test error is less than 1.00 without cheating (that is, training on the test set), you may expect to see the similar performance on the unseen test set as well.

Note that we provide `homework3.m` just to help you evaluate your code easily. You are not expected to alter or submit this to us. In other words, we will not use this file when we grade your submission. The only file we take care of is `myRecommender3.m`.

Grading criteria:

- Your code should output U and V as specified. The dimension should match to the specification.
- We will test your output on another test dataset, which was not provided to you. The test RMSE on this dataset should be at least 1.05 to get at least partial credit.
- We will measure elapsed time for learning. If your implementation takes longer than 3 minutes for rank 5, you should definitely try to make your code faster or adjust parameters. Any code running more than 5 minutes is not eligible for credit.
- Your code should not crash. Any crashing code will be not credited.

Report [10 pts]

In your report, show the performance (RMSE) both on your training set and test set, with varied `lowRank`. (The default is set to 1, 3, and 5, but you may want to vary it further.) Discuss what you observe with varied low rank. Also, briefly discuss how you decided your hyper-parameters (μ, λ).

Note

- Do not print anything in your code (e.g, iteration 1 : err=2.4382) in your final submission.
- Do not alter input and output format of the skeleton file. (E.g, adding a new parameter without specifying its default value) Please make sure that you returned all necessary outputs according to the given skeleton.
- Please do not use additional file. This task is simple enough that you can fit in just one file.
- Submit your code with the best parameters you found. We will grade without modifying your code. (Applying cross-validation to find best parameters is fine, though you do not required to do.)
- Please be sure that your program finishes within a fixed number of iterations. Always think of a case where your stopping criteria is not satisfied forever. This can happen anytime depending on the data, not because your code is incorrect. For this, we recommend setting a maximum number of iteration in addition to other stopping criteria.

Grand Prize

Similar to the Netflix competition held in 2006 to 2009, the student who achieves the lowest RMSE on the secret test set will earn the Grand Prize. We will give extra 10 bonus points to the winner, and share the student's code to everyone. (Note that the winner should satisfy all other grading criteria perfectly, including answer sanity check and timing requirement. Otherwise, the next student will be considered as the winner.)

Typing Bonus

As usual, we will give 5 bonus points for typed submissions. Note that **all** questions should be typed to get this credit.

Report:

The following table shows the final output of myRecommender3.m file.

lowRank	Training RMSE	Testing RMSE
1	0.9163	0.9469
3	0.8604	0.9366
5	0.8313	0.9397
7	0.8177	0.9404
9	0.7964	0.9360

• Observation:

(1)According to the final output, we can notice that the root mean square error of training set is decreasing when the number of lowRank is increasing.

(2)However, root mean square error of testing set didn't decrease along with the increasing of number of lowRank. The RMSE is approximately between 0.936 and 0.947 which is different to training RMSE.

- **How to decide hyper-parameter(μ, λ)**

First of all, here are my parameter configuration: `maxIter=500,learningRate=0.0002,regularizer=0.02` and `threshold=0.001`. (If $E(U, V)$ is small enough, then the algorithm will stop iterating.)

(1)With regard to learning rate μ , we can have a lot of optional values, however, on the one hand, if we choose a larger step value, it might miss the optimal result, on the other hand, if we choose a very small step, it will take a long time to converge. Thus, on the premise of satisfying the time limitation, I tried to find the smaller learning rate so that we won't miss the optimal value. For example, if we choose $\mu = 0.001$, we might get invalid results like `NaN` or `Inf`. As long as I choose a smaller μ , this issue can be solved.

(2)After we roughly found a learning rate, we can choose different regularizer parameters λ so that the RMSE is decreasing.

(3)Since the number of iteration won't affect the accuracy trend of the algorithm. We can use a small number of iteration at first to save time, and if we find relatively better parameters. Then we can increase the number of iteration so that we can get more accurate results under the limitation of time.