CS 7641 CSE/ISYE 6740 Homework 3

Le Song

Deadline: 11/13 Tue, 11:59pm

- 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: PRML[1] 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 white noise. The noise follows a zero-mean Gaussian distribution with constant variance. Mathematically,

$$Y^i = \theta^T X^i + \epsilon^i \tag{1}$$

where $\epsilon^i \sim \mathcal{N}(0, \sigma^2 I)$. In other words, we are assuming that each data point is independent to each other (no covariance) and that each data point has same variance.

(a) Using the normal equation, derive the expectation $\mathbb{E}[\hat{\theta}]$. [6 pts]

$$Y = \theta^T X + \epsilon \tag{2}$$

where $\epsilon^i \sim \mathcal{N}(0, \sigma^2 I)$
This equation basically tells that Y is a combination of Linear function in X and Normal Distributed error with 0 mean and constant variance.
Thus we can write this as

$$Y/X \sim \mathcal{N}(\theta^T X, \sigma^2 I) \tag{3}$$

In other words, linear regression assumes that given X, Y is normally distributed with mean that is linearly increasing in X and constant variance. In particular, no assumption is made on the distribution P(X). Thus Y also follows Normal Distribution.

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

Now , we know that $\hat{\theta} = (X^T X)^{-1} X^T Y$. We can also see that $\mathbb{E}[\hat{\theta}]$ is basically $\mathbb{E}[\theta\hat{/}X]$, as calculate this is supervised learning where we calculate / estimate the value of θ with respect to the known inputs X.
Note that , equation for variables in matrix form will be of the type //

$$\mathbf{Y}/\mathbf{X} \sim \mathcal{N}(\mathbf{X}\theta, \sigma^2 I) \tag{4}$$

$$\begin{aligned}
\mathbb{E}[\hat{\theta}/X] &= \mathbb{E}[(X^T X)^{-1} X^T Y/X] \\
&= (X^T X)^{-1} X^T \mathbb{E}[Y/X] \\
&= (X^T X)^{-1} X^T \mathbb{E}[Y/X] \\
&= (X^T X)^{-1} X^T \mathbf{X}\theta \\
&= \theta
\end{aligned}$$

(b) Using the normal equation, derive the variance Var$[\hat{\theta}]$. [6 pts]

This part now, will be similar to what we did above, Thus

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

Note that here we used the fact that $(K^{-1})^T = (K^T)^{-1}$, thus $((X^T X)^{-1})^T = (X^T X)^{-1}$
We also used that if A is constant then, $Var(AY) = A^2 Var(Y) = A * A^T Var(Y)$

(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]

We know that MSE is the sum of the bias squared and the variance. Since bias is zero here $(E(\theta) - \theta)$, we see that if X is nxd matrix then $X^T * X$ is a dxd matrix , whose value increases with increasing the n that is the number of input points.
Thus, for MSE(sum of the bias squared (which is zero) and the variance)we have that the d d MSE matrix $\sigma^2 (X^T X)^{-1}$ decreases to the zero matrix as n tends to ∞ (the entries of $(X^T X)$ get bigger as n increases and therefore the entries of $\sigma^2 (X^T X)^{-1}$ decrease).
We also see that the MSE is linearly related to the inherent noise level σ^2 of ϵ . We have shown above that conditioned on X the vector Y is multivariate normal. Since $\hat{\theta}$ is a linear function of Y it is also multivariate Gaussian (since a linear transform of a multivariate normal RV is a multivariate normal RV). We therefore have

$$\hat{\theta} \sim \mathcal{N}(\theta, \sigma^2 (X^T X)^{-1}) \tag{5}$$

(d) Weighted linear regression

Suppose we keep the independence (no covariance) assumption but remove the same variance assumption. Then, data points would be still sampled independently, but now they may have different variance σ_i. Thus,

the covariance matrix would be still diagonal, but with different values:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}. \tag{6}$$

Derive the normal equation for this problem using matrix-vector notations with Σ. [10 pts]

We know and have derived above that Y /X follows Gaussian Distribution,

$$\mathbf{Y}/\mathbf{X} \sim \mathcal{N}(\mathbf{X}\theta, \sigma^2 I) \tag{7}$$

In this case, as each of the points have different variance, thus we will have equation of the form

$$Y_i/X_i \sim \mathcal{N}(\theta^T X_i, \sigma_i^2 I) \tag{8}$$

Now as each points are iid's , thus probability of Y can be calculated by taking the product of their individual probabilities.

Inverse of Covariance Matrix

$$\Sigma_k^{-1} = 1/\sigma^2 * I$$

, So its only effect is to scale vector product by $1/\sigma^2$

$$p(Y|x, \theta, \Sigma) = \prod_{i=1}^{n} \mathcal{N}(\theta^T X_i, \sigma_i^2 I) \tag{9}$$

Putting the value of

$$p(Y_i/X_i) = \frac{1}{(2\pi)^{d/2}\sigma} \exp\left(-\frac{(Y_i - \theta^T X_i)^t (Y_i - \theta^T X_i)}{2\sigma_i^2}\right).$$

, we get

$$p(Y|x, \theta, \Sigma) = \frac{1}{(2\pi)^{nd/2}\sigma^n} \exp\left(\sum_{i=1}^{n} -\frac{(Y_i - \theta^T X_i)^t (Y_i - \theta^T X_i)}{2\sigma_i^2}\right) \tag{10}$$

Taking the log

$$\log p(Y|x, \theta, \Sigma) = \frac{nd}{2}\log \pi + n \log \sigma + \sum_{i=1}^{n} -\frac{(Y_i - \theta^T X_i)^t (Y_i - \theta^T X_i)}{2\sigma_i^2} \tag{11}$$

Taking the derivative wrt θ and setting it to 0 we get

$$\begin{aligned} 0 &= \sum_{i=1}^{n} \frac{(Y_i - \theta^T X_i)(X_i)^t}{2\sigma_i^2} \\ \sum_{i=1}^{n} \frac{(Y_i)(X_i)^t}{2\sigma_i^2} &= \sum_{i=1}^{n} \frac{(\theta^T X_i)(X_i)^t}{2\sigma_i^2} \end{aligned} \tag{12}$$

Note that

$$\frac{(X_i)(X_i)^t}{2\sigma_i^2} \tag{13}$$

is basically can be written in vector form as

$$[X_1, X_2.....] \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \end{bmatrix} \tag{14}$$

, Thus in matrix notation , we can write this as

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

2 Ridge Regression [15 pts]

For linear regression, it is often assumed that $y = \theta^\top \mathbf{x} + \epsilon$ 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 $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$, we define $\mathbf{y} = (y^1, ..., y^n)^\top$ and $X = (\mathbf{x}^1, ..., \mathbf{x}^n)^\top$. 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.

According to the question , we can write the likelihood of Y

$$\mathbf{Y}/\mathbf{X} \sim \mathcal{N}(\mathbf{X}\theta, \sigma^2 I) \tag{15}$$

We are also given the prior

$$\theta \sim \mathcal{N}(0, \tau^2 I) \tag{16}$$

If we assume that our input data x is already centered , so that we can ignore the intercept term θ_0. The posterior distribution of ridge regression given Y and X can be written using Bayes theory as:

$$p(\theta/X, Y) = \frac{p(Y/X, \theta) * p(\theta)}{Z} \tag{17}$$

Here Z which is Z(Y,X),is a normalization constant and is independent of θ, thus we can ignore the denominator.

Probability of Normal Gaussian Distribution for a general x is given by :

$$p(x) = \frac{1}{(2\pi)^{d/2}\Sigma^{1/2}} \exp\left(-\frac{(x-\mu)^t\Sigma^{-1}(x-\mu)}{2}\right).$$

Inverse of Covariance Matrix

$$\Sigma_k^{-1} = 1/\sigma^2 * I$$

, So its only effect is to scale vector product by $1/\sigma^2$

$$p(x) = \frac{1}{(2\pi)^{d/2}\sigma} \exp\left(-\frac{(x-\mu)^t(x-\mu)}{2\sigma^2}\right).$$

Thus after plugging the values in Bayes equation we get the following ,

$$p(\theta/X,Y) = \frac{1}{Z}\frac{1}{(2\pi)^{d/2}\sigma} \exp\left(-\frac{(y-\mathbf{X}\theta)^t(y-\mathbf{X}\theta)}{2\sigma^2}\right) * \frac{1}{(2\pi)^{d/2}\tau} \exp\left(-\frac{(\theta)^t(\theta)}{2\tau^2}\right) \tag{18}$$

Taking the log form of posterior distribution, we get

$$\log p(\theta/X,Y) = -logZ - logK - \frac{(y-\mathbf{X}\theta)^t(y-\mathbf{X}\theta)}{2\sigma^2} - \frac{(\theta)^t(\theta)}{2\tau^2} \tag{19}$$

Here, K corresponds to other terms outside of exp., which are independent of θ. Taking the derivative of this log we get, wrt to θ and setting it to 0

$$\begin{aligned}
0 &= \frac{X^t(y-\mathbf{X}\theta)}{\sigma^2} - \frac{(\theta)}{\tau^2} \\
0 &= X^t(y-\mathbf{X}\theta) - \frac{(\theta)\sigma^2}{\tau^2} \\
0 &= X^t y - X^t\mathbf{X}\theta) - \frac{(\theta)\sigma^2}{\tau^2} \\
0 &= X^t y - (X^t X + \frac{\sigma^2}{\tau^2})\theta \\
\theta &= (X^t X + \frac{\sigma^2}{\tau^2})^{-1} X^t y
\end{aligned} \tag{20}$$

Setting the ridge regression parameter $\lambda = \frac{\sigma^2}{\tau^2}$, we see that the above solution is equivalent to ridge regression. When τ^2 is small (meaning we have prior knowledge to believe θ is close to zero), then λ is large (meaning large θ will be penalized). When σ^2 is small (meaning that observations y are not noisy), we will focus on fitting the data (small λ).
Now, since posterior is product of two gaussians , therefore it will also be gaussian. Let m_b and Σ_b, be its mean and variance, then its exponent will be ,
Expansion of squared Mahalanobis distance.

$$\begin{aligned}
&(\theta - m_b)^t\Sigma_b^{-1}(\theta - m_b) \\
&\qquad = \theta^t\Sigma_b^{-1}\theta - 2(\Sigma_b^{-1}m_b)^t\theta + m_b^t\Sigma_b^{-1}m_b
\end{aligned} \tag{21}$$

Equating the second order θ terms between eq 11 and 13 we get

$$\begin{aligned}
\theta^t\Sigma_b^{-1}\theta &= \frac{\theta^t X^t X\theta}{\sigma^2} + \frac{\theta^t\theta}{\tau^2} \\
\Sigma_b^{-1} &= \frac{1}{\sigma^2}(X^t X + \frac{\sigma^2}{\tau^2} I)
\end{aligned} \tag{22}$$

Then you can show by equating the value of θ ,we can obtain the mean by plugging in

$$\begin{aligned}
m_b &= (X^t X + \frac{\sigma^2}{\tau^2} I)^{-1} X^t y \\
&= \frac{\Sigma_b X^t y}{\sigma^2}
\end{aligned} \tag{23}$$

3 Bias - Variance Decomposition [15 pts]

Suppose x is a d-dimensional vector. The estimator S_j^2 defined as

$$S_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_j^i - \bar{\mathbf{x}}_j)^2$$

where $i = 1, \ldots, n$, $j = 1, \ldots, d$ and $\bar{\mathbf{x}}_j = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_j^i$, is used to estimate the diagonal of covariance matrix, that is, $diag(Cov(X))$. Show that S_j^2 is an unbiased estimator.

Estimator is said to be unbiased $E(\hat{\theta}) = \theta$ Thus ,to prove that an estimator $\hat{\theta}$ for the parameter θ is unbiased, we need to show that the bias of $\hat{\theta}$ is $Bias(\hat{\theta}) = E(\hat{\theta}) - \theta = 0$,
Now, iwe are given that x is a d-dimensional vector, and S_j^2 is the estimator which is used to estimate the diagonal of covariance matrix, that is, $diag(Cov(X_j))$. Here, $X = \{x_j^i\}$ where $i = 1, \ldots, n$ & $j = 1, \ldots, d$.
Finally, we need to show that S_j^2 is an unbiased estimator.

Thus, we need to show that $E(S_j^2) = diag(Cov(X_j))$ where $j = 1, \ldots, d$, ie $E(S^2) = ES^2 = diag(Cov(X))$.

We are given that:

$$\begin{aligned}
\bar{\mathbf{x}}_j &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_j^i \\
E(\bar{\mathbf{x}}_j) &= E(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_j^i) \\
&= \sum_{i=1}^{n}\frac{E(\mathbf{x}_j^i)}{n} \\
&= \frac{nE(\mathbf{x}_j)}{n} = E(\mathbf{x}_j)
\end{aligned}$$

Let us consider the following result:

$$\begin{aligned}
\sum_{i=1}^{n}(\mathbf{x}_j^i - \bar{\mathbf{x}}_j)^2 &= \sum_{i=1}^{n}(\mathbf{x}_j^i)^2 - 2\bar{\mathbf{x}}_j\sum_{i=1}^{n}\mathbf{x}_j^i + n\bar{\mathbf{x}}_j^2 \\
&= \sum_{i=1}^{n}(\mathbf{x}_j^i)^2 - 2n\bar{\mathbf{x}}_j\bar{\mathbf{x}}_j + n\bar{\mathbf{x}}_j^2 \\
&= \sum_{i=1}^{n}(\mathbf{x}_j^i)^2 - n\bar{\mathbf{x}}_j^2
\end{aligned}$$

To prove that S^2 is unbiased we will show that it is unbiased in the one dimensional case i.e., S_j^2 are scalars (if this holds, we can apply this result to each component separately to get unbiasedness of the vector S^2).

Therefore using the result from above,

$$
\begin{aligned}
E\left(\sum_{i=1}^{n}(\mathbf{x}_j^i - \bar{\mathbf{x}}_j)^2\right) &= E\left(\sum_{i=1}^{n}(\mathbf{x}_j^i)^2 - n\bar{\mathbf{x}}_j^2\right) \\
&= \sum_{i=1}^{n} E((\mathbf{x}_j^i)^2) - nE(\bar{\mathbf{x}}_j^2) = nE((\mathbf{x}_j)^2) - nE(\bar{\mathbf{x}}_j^2)
\end{aligned} \tag{24}
$$

Now, using the relations between variance and Expectations know that:

$$
\begin{aligned}
Var(X) &= E(X^2) - (E(X))^2 \\
E(X^2) &= Var(X) + (EX)^2
\end{aligned}
$$

Similarly, we get:

$$
\begin{aligned}
Var(\bar{X}) &= E(\bar{X}^2) - (E(\bar{X}))^2 \\
E(\bar{X}^2) &= \frac{Var(X)}{n} + (E\bar{X})^2
\end{aligned}
$$

Now, substituting the expectations $E(X^2)$ and $E(\bar{X}^2)$, we will get:

$$
\begin{aligned}
E\left(\sum_{i=1}^{n}(\mathbf{x}_j^i - \bar{\mathbf{x}}_j)^2\right) &= nE((\mathbf{x}_j)^2) - nE(\bar{\mathbf{x}}_j^2) \\
&= n\left(Var(X) + (EX)^2\right) - n\left(\frac{Var(X)}{n} + (E\bar{X})^2\right) \\
&= (n-1)Var(X) \\
\frac{E\left(\sum_{i=1}^{n}(\mathbf{x}_j^i - \bar{\mathbf{x}}_j)^2\right)}{n-1} &= Var(X) \\
E\left(\frac{\sum_{i=1}^{n}(\mathbf{x}_j^i - \bar{\mathbf{x}}_j)^2}{n-1}\right) &= Var(X) \\
E(S_j^2) &= diag(Cov(X_j))
\end{aligned}
$$

Now, similarly to the multivariate case, this implies $E(S_j^2) = diag(Cov(X_j))$ for all $j = 1, ..., d$, and therefore:

$$
ES^2 = diag(Cov(X)) \tag{25}
$$

As Expectation is just equal to the variance, thus it proves that the S^2 is unbiased.

4 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

$f : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$, 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

$$M_{u,i} \approx \sum_{k=1}^{r} U_{u,k} V_{i,k}. \tag{26}$$

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

$$E(U, V) = \sum_{(u,i) \in M} (M_{u,i} - U_u^T V_i)^2 = \sum_{(u,i) \in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 \tag{27}$$

where U_u is the uth row of U and V_i is the ith row of V. We observe that this looks very similar to the linear regression, which we learned in the class (from slide 24 in lecture 15). Recall that we minimize in linear regression:

$$E(\theta) = \sum_{i=1}^{m} (Y^i - \theta^T x^i)^2 = \sum_{i=1}^{m} (Y^i - \sum_{k=1}^{r} \theta_k x_k^i)^2 \tag{28}$$

where m is the number of training data points. Let's compare (30) and (28). $M_{u,i}$ in (30) corresponds to Y^i in (28), in that both are the observed labels. $U_u^T V_i$ in (30) corresponds to $\theta^T x^i$ in (28), 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 (30), while only θ is learned in (28). 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 (30), there is no closed form solution, unlike linear regression case. Thus, we need to use gradient descent:

$$U_{v,k} \leftarrow U_{v,k} - \mu \frac{\partial E(U, V)}{\partial U_{v,k}}, \qquad V_{j,k} \leftarrow V_{j,k} - \mu \frac{\partial E(U, V)}{\partial V_{j,k}}, \tag{29}$$

where μ is a hyper-parameter deciding the update rate. It would be straightforward to take partial derivatives of $E(U, V)$ in (30) 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 (29).

(a) Derive the update formula in (29) by solving the partial derivatives. [10 pts]

We know that:

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

By Calculating the partial derivatives wrt to $U_{v,k}$:

$\frac{\partial E(U,V)}{\partial U_{v,k}}$ is given by:

$$\begin{aligned} \frac{\partial E(U,V)}{\partial U_{v,k}} &= \frac{\partial}{\partial U_{v,k}} \left(\sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 \right) \\ &= \sum_{(u,i)\in M} \frac{\partial}{\partial U_{v,k}} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right)^2 \\ &= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) \frac{\partial}{\partial U_{v,k}} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) \\ &= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) \left(\frac{\partial M_{u,i}}{\partial U_{v,k}} - \frac{\partial}{\partial U_{v,k}} \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) \\ &= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) (0 - V_{i,k}) \\ &= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) V_{i,k} \\ &= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) V_{i,k} \\ &= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) V_{i,k} = -2(\mathbf{M} - \mathbf{U}\mathbf{V}^T)\mathbf{V} \end{aligned}$$

$\frac{\partial E(U,V)}{\partial V_{j,k}}$ is given by:

$$
\begin{aligned}
\frac{\partial E(U,V)}{\partial V_{j,k}} &= \frac{\partial}{\partial V_{j,k}} \left(\sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 \right) \\
&= \sum_{(u,i)\in M} \frac{\partial}{\partial V_{j,k}} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right)^2 \\
&= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) \left(\frac{\partial M_{u,i}}{\partial V_{j,k}} - \frac{\partial}{\partial V_{j,k}} \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) \\
&= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) (0 - U_{u,k}) \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right) U_{u,k} \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} = -2(\mathbf{M} - \mathbf{U}\mathbf{V}^T)^T \mathbf{U}
\end{aligned}
$$

So, update rules for $U_{v,k}$&$V_{j,k}$ as follows:

$$
\begin{aligned}
U_{v,k} &= U_{v,k} - \mu \frac{\partial E(U,V)}{\partial U_{v,k}} \\
&= U_{v,k} - \mu \left(-2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) V_{i,k} \right) \\
&= U_{v,k} + 2\mu \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) V_{i,k}
\end{aligned}
$$

$$
\begin{aligned}
V_{j,k} &= V_{j,k} - \mu \frac{\partial E(U,V)}{\partial V_{j,k}} \\
&= V_{j,k} - \mu \left(-2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} \right) \\
&= V_{j,k} + 2\mu \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k}
\end{aligned}
$$

(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.)
Now, we will calculate the partial derivatives as follows:

$\frac{\partial E(U,V)}{\partial U_{v,k}}$ is given by: (Using the partial derivative results from the first part , we get)

$$\begin{aligned}
\frac{\partial E(U,V)}{\partial U_{v,k}} &= \frac{\partial}{\partial U_{v,k}} \left(\sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k})^2 + \lambda \sum_{u,k} U_{u,k}^2 + \lambda \sum_{i,k} V_{i,k}^2 \right) \\
&= \frac{\partial}{\partial U_{v,k}} \left(\sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k})^2 \right) + \frac{\partial}{\partial U_{v,k}} \left(\lambda \sum_{u,k} U_{u,k}^2 \right) + \frac{\partial}{\partial U_{v,k}} \left(\lambda \sum_{i,k} V_{i,k}^2 \right) \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} + \lambda \sum_{u,k} \frac{\partial}{\partial U_{v,k}} \left(U_{u,k}^2 \right) + \lambda \sum_{i,k} \frac{\partial}{\partial U_{v,k}} \left(V_{i,k}^2 \right) \\
&= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k} \right) (0 - V_{i,k}) + 2\lambda U_{u,k} \\
&= -2(\mathbf{M} - \mathbf{U}\mathbf{V}^T)\mathbf{V} + 2\lambda \mathbf{U}
\end{aligned}$$

$\frac{\partial E(U,V)}{\partial V_{j,k}}$ is given by:

$$\begin{aligned}
\frac{\partial E(U,V)}{\partial V_{j,k}} &= \frac{\partial}{\partial V_{j,k}} \left(\sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k})^2 + \lambda \sum_{u,k} U_{u,k}^2 + \lambda \sum_{i,k} V_{i,k}^2 \right) \\
&= \frac{\partial}{\partial V_{j,k}} \left(\sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k})^2 \right) + \frac{\partial}{\partial V_{j,k}} \left(\lambda \sum_{u,k} U_{u,k}^2 \right) + \frac{\partial}{\partial V_{j,k}} \left(\lambda \sum_{i,k} V_{i,k}^2 \right) \\
&= \sum_{(i:M(u,i)>0)} 2 \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k} \right) (0 - U_{u,k}) + 2\lambda V_{j,k} \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k} \right) U_{u,k} + 2\lambda V_{j,k} \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} + 2\lambda V_{j,k} \\
&= -2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} + 2\lambda V_{j,k} \\
&= -2(\mathbf{M} - \mathbf{U}\mathbf{V}^T)^T \mathbf{U} + 2\lambda \mathbf{V}
\end{aligned}$$

So, now that we have found the gradients, we will update $U_{v,k}$&$V_{j,k}$ as follows:

$$\begin{aligned}
U_{v,k} &= U_{v,k} - \mu \frac{\partial E(U,V)}{\partial U_{v,k}} \\
&= U_{v,k} - \mu \left(-2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) V_{i,k} + 2\lambda U_{u,k} \right) \\
&= (1 - 2\lambda) U_{v,k} + 2\mu \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) V_{i,k}
\end{aligned}$$

$$
\begin{aligned}
V_{j,k} &= V_{j,k} - \mu \frac{\partial E(U,V)}{\partial V_{j,k}} \\
&= V_{j,k} - \mu \left(-2 \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k} + 2\lambda V_{j,k} \right) \\
&= (1-2\lambda) V_{j,k} + 2\mu \sum_{(i:M(u,i)>0)} \left(M_{u,i} - U_u^T V_i \right) U_{u,k}
\end{aligned}
$$

(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, directly referring to 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).

Discussion

Performance (RMSE) for the following svd values are: I am able to get RMSE around.92-.93 for SVD3 and SVD5. Time taken is around 30 seconds per SVD. Insert a Table here.

SVD	TrainRMSE	DevRMSE	Time
SVD-1	0.9171	0.9474	36.81
SVD-3	0.8664	0.9349	38.42
SVD-5	0.8393	0.9360	38.59
SVD-7	0.8266	0.9354	41.96
SVD-9	0.8140	0.9288	40.71

Discuss
I had tried two approaches and depending on that I took different values of μ, λ. In the first case, I kept learningRate = 0.00025; regularizer = 0.03 . In this case I kept on decreasing the value of learning rate marginally as I was getting more and more close to the data ,but kept the regularizer same. I was getting around .93 to .94 for both SVD3 and SVD5. (code in myRecommendation1.m). Its needs to be noted that RMSE on training data decreased as we increased the SVD number. It also decreased as we increased the number of iteration, but it had not impact for improvement in DevRMSE. Thus it was doing overfitting on test data. I tried to combat this by increasing regularizer,not in every iteration but it every 100 iterations.This approach was run 450 times or when the error decreases below a bound.Values for this approach are as follows.

SVD-1 0.9171 0.9474 36.81
bhatia, parminder
SVD-3 0.8664 0.9349 38.42
bhatia, parminder
SVD-5 0.8393 0.9360 38.59
bhatia, parminder
SVD-7 0.8266 0.9354 41.96
bhatia, parminder
SVD-9 0.8140 0.9288 40.71

Next approach, which I used is Bolt Driver approach, where we keep on increasing learning rate by 1.2 times till the time error is decreasing and as soon as we find an increase in error, we decrement the learning rate by half. We initialized the values for learningRate = 0.000003 regularizer = 3 . Note that this approach reduced the RMSE for test data very well but was overfitting and performance and time consumed were slightly higher than the earlier approach.
Bhatia, Parminder

SVD-1 0.9222 0.9554 49.58
Bhatia, Parminder
SVD-3 0.8479 0.9280 52.64
Bhatia, Parminder
SVD-5 0.8061 0.9357 52.87
Bhatia, Parminder
SVD-7 0.7655 0.9457 52.37
Bhatia, Parminder
SVD-9 0.7376 0.9634 52.90

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 defalut 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 achives 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.