CSE 6740 Homework 3

Anqi Wu, Spring 2025

Deadline: 9:30 am ET, Thursday, 27th March

- There are 2 sections in gradescope: Homework 3 and Homework 3 Programming. Submit your answers as a PDF file to Homework 3 (including report for programming) and also submit your code in a zip file to Homework 3 Programming.
- All Homeworks are due by the beginning of class. Homework is penalized by 20% for each day that it is late (this applies additively, meaning that no credit is gained after 5 late days).
- We strongly encourage the use of LaTeX for your submission. Unreadable handwriting is subject to zero credit.
- Explicitly mention your collaborators if any.
- Recommended reading: PRML[1] Section 3.1, 3.2
- Python and Matlab are allowed.
- When submitting to Gradescope, please make sure to mark the page(s) corresponding to each problem/sub-problem. **Note**: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions. Please check this link for additional information on submitting to Gradescope.
- Wherever an expression is asked for, a full derivation is expected. Simply stating the final answer will not count.

1 Linear Regression [20 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,

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

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, $\theta \in \mathbb{R}^d$, 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.

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

a	b	c	K
1	0	1	1
0	1	1	0
0	0	1	1
1	1	1	1
1	1	0	1
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	0

Table 1: Data with boolean inputs and outputs for question 3.1.

(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. [5 pts]

(b) Similarly, derive the variance $\mathbf{Var}[\hat{\theta}]$. [5 pts]

(c) Under the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, prove that $\hat{\theta}$ follows a multivariate Gaussian distribution. Then, derive the variance expression for $\hat{\theta}$ when each data point has a different noise variance. How do these two cases affect the efficiency and validity of OLS? [6 pts]

(d) For linear regression, please answer whether each of the statements below is true or false? And Why? [4 pts]

1. **Gradient descent is likely to converge to a local minimum rather than a global minimum.**
2. **One iteration will not change θ if it is initialized at the global minimum.**

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 $\mathbf{y} \sim \mathcal{N}(X\theta, \sigma^2 I)$.

(a) 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. [12 pts]

(b) Different values of regularization parameter λ will yield different Ridge Regression model performances. Please design an algorithm that uses K-Fold Cross-Validation to select the best value of λ from $\Lambda = (\lambda^1, ...\lambda^n)$, which yields minimised MSE in Ridge Regression. [3 pts]

3 Bayes Classifier [20 pts]

3.1 Joint Bayes vs Naive Bayes Classifier

Suppose you are given the set of data, as in Table 1, with the three boolean input variables a, b, and c, and a single Boolean output variable K.

(a) According to the joint Bayes classifier, what is the value of the following expressions: (i) $P(K = 1 \mid a = 1 \wedge b = 1 \wedge c = 0)$**; and (ii)** $P(K = 0 \mid a = 1 \wedge b = 1)$**. [2.5 pts]**

(b) According to the naive Bayes classifier, what is the value of the expressions in part (a). [2.5 pts]

3.2 Bayes Classifier with Gaussian Class Conditional Distribution

Let us consider a binary classification problem $X \to Y$ such that $Y \in \{-1, +1\}$. As discussed in Lecture 10, Bayes classifier (or decision rule) for such a classification problem is given as follows:

$$f(x) = \text{sign}\left(\log\left(\frac{p(x \mid y = 1)p(y = 1)}{p(x \mid y = -1)p(y = -1)}\right)\right)$$

Furthermore, we discussed how geometric shape of decision boundaries depends on the type of class conditional distribution. In the following sub-problems, given a pair of class conditional distributions, you are required to formally deduce the geometric shape of decision boundaries.

(a) Suppose the class conditional distributions are Gaussians, with non-identical covariance matrices and means. Write the Bayes classifier as $f(x) = \textbf{sign}(h(x))$ **and simplify** h **as much as possible. What is the geometric shape of the decision boundary? [5 pts]**

(b) Repeat (a) but assume the two Gaussians have identical covariance matrices. What is the geometric shape of the decision boundary? [5 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? [5 pts]

4 Basics of optimization[10 pts]

(a) Show that the binary cross-entropy loss function is convex in z when $y \in \{0, 1\}$:

$$L(z) = -y \log(\sigma(z)) - (1 - y) \log(1 - \sigma(z)),$$

where $\sigma(z)$ is the sigmoid function. [5 pts]

(b) (Jensen's inequality) Use the definition of a concave function, f, to show that

$$f\left(\sum_{i=1}^{m} \alpha_i x_i\right) \geq \sum_{i=1}^{m} \alpha_i f\left(x_i\right)$$

where $\sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0$. [5 pts]

5 Logistic Regression[10 pts]

5.1 Binary Classification

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) Please derive the decision boundary equation and explain its geometric interpretation. [5 pts]

5.2 Multi-class Classification

Let us do multi-class classification: assign input vector $x^i, i = 1, \ldots, m$ into one of classes $c, c = 1, 2, \ldots, C$. Rather than doing "one against all", you will use the softmax probability,

$$P\left(y^i = c \mid x^i, \theta_1, \ldots, \theta_C\right) = \frac{\exp\left(\theta_c^\top x^i\right)}{\sum_{c'=1}^{C} \exp\left(\theta_{c'}^\top x^i\right)}$$

Here c' is a possible label and y^i is the discrete training label $\in \{1, 2, 3, ..., C\}$. Given all the input data $\left(x^1, y^1\right), \left(x^2, y^2\right), \ldots, \left(x^m, y^m\right)$, the log-likelihood that results from using the softmax probability can be expressed as:

$$l(\theta) = \sum_{i=1}^{m}\sum_{c=1}^{C} y_c^i \theta_c^T x^i - \sum_{i=1}^{m}\sum_{c=1}^{C} y_c^i \log \sum_{c'=1}^{C} \exp\left(\theta_{c'}^T x^i\right)$$

(a) To improve generalization and prevent overfitting, we introduce L2 regularization to the loss function (Negative log-likelihood). Please write the new regularized loss function by adding an L2 penalty term and derive the gradient of this regularized loss function with respect to θ_c. [5 pts]

6 Programming: Recommendation System [25 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

$$M_{u,i} \approx \sum_{k=1}^{r} U_{u,k} V_{i,k}. \tag{2}$$

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

$$E(U, V) = \sum_{(u,i) \in M} (M_{u,i} - U_u^T V_i)^2 = \sum_{(u,i) \in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 \tag{3}$$

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. Recall that we minimize in linear regression:

$$E(\theta) = \sum_{i=1}^{m} (Y^i - \theta^T x^i)^2 = \sum_{i=1}^{m} (Y^i - \sum_{k=1}^{r} \theta_k x_k^i)^2 \tag{4}$$

where m is the number of training data points. Let's compare (3) and (4). $M_{u,i}$ in (3) corresponds to Y^i in (4), in that both are the observed labels. $U_u^T V_i$ in (3) corresponds to $\theta^T x^i$ in (4), 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 (3), while only θ is learned in (4). 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 (3), there is no closed form solution, unlike linear regression case. Thus, we need to use gradient descent:

$$U_{v,k} \leftarrow U_{v,k} - \eta \frac{\partial E(U, V)}{\partial U_{v,k}}, \qquad V_{j,k} \leftarrow V_{j,k} - \eta \frac{\partial E(U, V)}{\partial V_{j,k}}, \tag{5}$$

where η is a hyper-parameter deciding the update rate. It would be straightforward to take partial derivatives of $E(U, V)$ in (3) 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 (5).

(a) Derive the update formula in the gradient descent equation above by computing the partial derivatives of $E(U, V)$ with respect to each element $U_{v,k}$ and $V_{j,k}$. [3 pts]

(b) To avoid overfitting, we add regularization terms that penalize large values in U and V. Redo part (a) using the following regularized objective function: [3 pts]

$$E(U,V) = \sum_{(u,i)\in M} \left(M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right)^2 + \lambda \left(\sum_{u,k} U_{u,k}^2 + \sum_{i,k} V_{i,k}^2 \right), \tag{6}$$

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

(c) Extend the model to incorporate bias features [4 pts].

In the enhanced model, the predicted rating for user u on item i is given by:

$$\hat{M}_{u,i} = \mu + b_u + b_i + \sum_{k=1}^{r} U_{u,k} V_{i,k}, \tag{7}$$

where μ is the global bias (typically computed as the average of all observed ratings), b_u is the bias specific to user u, and b_i is the bias specific to item i. Consequently, the objective function becomes:

$$E(U,V,b_u,b_i) = \sum_{(u,i)\in M} \left(M_{u,i} - \mu - b_u - b_i - \sum_{k=1}^{r} U_{u,k} V_{i,k} \right)^2 + \lambda \left(\sum_{u,k} U_{u,k}^2 + \sum_{i,k} V_{i,k}^2 + \sum_{u} b_u^2 + \sum_{i} b_i^2 \right). \tag{8}$$

Derive the gradient descent update equations for the bias terms b_u and b_i, assuming that μ is fixed.

(d) Complete `recommender_system_hw3.ipynb` by filling the gradient descent part.

You are given a skeleton code `recommender_system_hw3.ipynb`. Using the training data `rateMatrix`, you will implement your own recommendation system of rank `lowRank`. In the gradient descent part, repeat your update formula in (b) and (c), 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 η 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 (number of users) $\times$ (lowRank).
- **V**: The item profile matrix of dimension (number of items) $\times$ (lowRank).
- b_u: The bias vector for users.
- b_i: The bias vector for items.
- μ: The global bias.

Evaluation [10 pts]

Upload your completed `recommender_system_hw3.ipynb` file. The notebook should output both training and test RMSE (Root Mean Squared Error) defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|M|} \sum_{(u,i) \in M} (M_{u,i} - f(u,i))^2}, \tag{9}$$

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 0.95 without cheating (that is, training on the test set), you may expect to see the similar performance on the unseen test set as well.

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 most 1.00 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 [5 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).

Deliverables

You need to upload the following files to Gradescope - Programming

- Completed `recommender_system_hw3.ipynb`
- Q.6 PDF file with the answers to Q.6 (a), (b), (c) and report

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 .py files. 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.