Homework 1 Machine Learning

Zheng WAN_B00758336

Question 1

(a) Stochastic gradient descent performs less computation per update than gradient descent.
Answer: True.
Justification: The per-update computational complexity of SGD is $O(d)$, while that of GD is $O(nd)$.

(b) Both PCA and linear regression can be thought of as algorithms for minimizing a sum of squared errors.
Answer: True.
Justification: PCA aims to minimize the squared reconstruction error after projecting data onto lower-dimensional space. Linear regression minimizes the squared difference between observed and predicted values.

(c) Let $A \in \mathbb{R}^{m \times n}$ be the matrix representation of our data. Let's assume that we project our data on the k-dimensional space using Principal Component Analysis, where k equals the rank of A. Then, no loss is incurred in the reconstruction of the data.
Answer: True.
Justification: If k equals the rank of A, then all the variance in the data is captured by the first k principal components, and no information is lost during projection.

(d) Let $y_i = \log(x_i^1 x_i^2) + \varepsilon_i$ be a model, where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ corresponds to Gaussian noise. Then, the maximum likelihood parameters of the model (α) can be learned using linear regression.
Answer: True.
Justification: By taking the logarithm of both sides, the equation can be transformed into a linear form suitable for linear regression.

(e) The eigenvectors of AA^T and A^TA are the same.
Answer: False.
Justification: Let $A \in \mathbb{R}^{m \times n}$, the eigenvectors of AA^T belong to $\mathbb{R}^m$ and those of A^TA belong to $\mathbb{R}^n$. They are not the same.

Question 2

Answer to (a):

$$
\begin{aligned}
(MM^T)^T &= (M^T)^T M^T \\
&= MM^T
\end{aligned}
$$

$$
\begin{aligned}
(M^TM)^T &= (M^T)^T M^T \\
&= MM^T
\end{aligned}
$$

Both MM^T and M^TM are symmetric. MM^T is $m \times m$ (square) and M^TM is $n \times n$ (square). They are real if matrix M has real entries.

Answer to (b):

Eigenvalues: Let v be an eigenvector of M^TM with eigenvalue λ. This means: $M^TMv = \lambda v$

Multiplying both sides of the equation by matrix M, we get:

$$M(M^TMv) = \lambda Mv$$

$$MM^T(Mv) = \lambda Mv$$

This shows that Mv is an eigenvector of MM^T with the same eigenvalue λ.

Eigenvectors: While the eigenvalues of MM^T and M^TM are the same, their eigenvectors are generally not the same, because they belong to different vector spaces.

Answer to (c): Using the SVD decomposition $M = U\Sigma V^T$:

$$\begin{aligned} M^TM &= (U\Sigma V^T)^T U\Sigma V^T \\ &= V\Sigma^T U^T U\Sigma V^T \\ &= V\Sigma^2 V^T \end{aligned}$$

Similarily,we conclude:

$$MM^T = U\Sigma^2 U^T$$

Answer to (d): The eigenvalues of M^TM are the squared singular values of M. This means that if λ is an eigenvalue of M^TM, then $\sqrt{\lambda}$ will be a singular value of M.

In other words,The singular values of M are the square roots of the eigenvalues of MM^T and M^TM

$$\lambda_i(M^TM) = \lambda_i(MM^T) = \sigma_i^2$$

Question 3

(a) Formulate the least squares problem in terms of U_k, x, and w.

The projection is given by U_kw. The reconstruction error between the original data point and its projection is $x - U_kw$.

We want to minimize the square of this error:

$$\min_w \|x - U_kw\|^2$$

This problem resembles a linear regression problem, where U_k is like the design matrix, and w are the regression coefficients.

(b) Show that the solution of the least squares problem is equal to U_k^Tx, which is the projection of x onto the span of the columns of U_k.

From the previous answer, we have:

$$\min_w \|x - U_kw\|^2$$

To find the value of w that minimizes this expression, differentiate with respect to w and set the result to zero:

$$\frac{d}{dw}\|x - U_k w\|^2 = 0$$

$$\Rightarrow -2U_k^T(x - U_k w) = 0$$

$$\Rightarrow U_k^T x - U_k^T U_k w = 0$$

Given that U_k consists of orthonormal eigenvectors (from PCA), we have:

$$U_k^T U_k = I$$

Where I is the identity matrix in $\mathbb{R}^k$. Substituting this in, we get:

$U_k^T x - w = 0 \Rightarrow w = U_k^T x$

Thus, the solution to the least squares problem w is $U_k^T x$.

Question 4

(A)

The regularization term λ in logistic regression serves as a trade-off between the complexity of the model and the fitting to the training data.With larger dataset, less regularization needed since the data itself can offer better generalization. On the other hand, with a smaller dataset, the risk of overfitting increases. The first model is on 100 million users and then re-trained on smaller dataset:We need more regularization with the smaller dataset to avoid overfitting. As a result, the regularization hyperparameter λ_2 is expected to be greater than λ_1 .

1.[True] while **2,3,4 [False]** λ_2 is expected to be greater than λ_1.
4.5.[False] As there's no linear relationship implied between the dataset size and regularization strength.

(B)

1. **Using polynomial features**: Adding polynomial features captures more complexities, typically decreasing training error.
2. **Using Ridge to reduce model complexity**: Ridge might slightly increase training error by adding constraints but can improve generalization.
3. **Using Lasso to encourage sparse coefficients**: Lasso can increase training error by forcing coefficients to zero but can prevent overfitting.
4. **Normalizing the data points**: Normalization doesn't change data relationships, hence doesn't increase training error.

Correct choices: 1 and 4.

(C)

1. **The variance of the method decreases if λ increases enough**: Correct. As λ increases, the model becomes more regularized, leading to lower complexity and thus lower variance.
2. **There might be multiple solutions for w^***: Incorrect. Regularization ensures a unique solution, especially when X is not full rank.

3. **The bias of the method decreases if λ increases enough**: Incorrect. As λ increases, the model becomes more biased towards simpler solutions.

4. $w^* = X^+Y$**, where** X^+ **is the pseudoinverse of** X: Incorrect. This is the solution without the regularization term. With regularization, the solution changes.

Correct answer is: 1.

Question 5

(a) Given the cost function for ridge regression:

$$J(\theta) = ||y - X\theta||_2^2 + \lambda||\theta||_2^2$$

Compute the gradient of $J(\theta)$ with respect to θ:

$$\nabla_\theta J(\theta) = -2X^T(y - X\theta) + 2\lambda\theta$$

To find the minimum of $J(\theta)$, set the gradient to zero:

$$-2X^T(y - X\theta) + 2\lambda\theta = 0$$

Rearrange terms:

$$X^T(y - X\theta) = \lambda\theta$$

$$X^Ty - X^TX\theta = \lambda\theta$$

$$X^Ty = (X^TX + \lambda I)\theta$$

Solve for θ:

$$\theta = (X^TX + \lambda I)^{-1}X^Ty$$

Where:

- X is the design matrix.
- y is the vector of target values.
- λ is the regularization parameter.
- I is the identity matrix.

(b) Ridge Regression penalizes large coefficients, which in turn constrains the model complexity. When models have too many features or when features are correlated, least-squares can fit noise, leading to overfitting. The regularization in ridge regression shrinks the coefficients and helps in preventing this overfitting.

Question 6

(a) Likelihood function $L(\theta; X_1, \ldots, X_n)$: For a single random variable X_i,

$$p(X_i) = \theta^{X_i}(1 - \theta)^{1-X_i}$$

Given that X_i are i.i.d., the likelihood function is:

$$L(\theta; X_1, \ldots, X_n) = \prod_{i=1}^{n} \theta^{X_i}(1-\theta)^{1-X_i}$$

(b) Starting with the likelihood function:

$$L(\theta; X_1, \ldots, X_n) = \prod_{i=1}^{n} \theta^{X_i}(1-\theta)^{1-X_i}$$

Apply the natural logarithm to both sides:

$$\ln L(\theta) = \ln \prod_{i=1}^{n} \theta^{X_i}(1-\theta)^{1-X_i}$$

Use the logarithm properties. When taking the logarithm of a product, it becomes a sum of the logarithms:

$$\ln L(\theta) = \sum_{i=1}^{n} \ln[\theta^{X_i}(1-\theta)^{1-X_i}]$$

Separate the terms inside the logarithm and utilize properties of logarithms:

$$\ln L(\theta) = \sum_{i=1}^{n} [X_i \ln \theta + (1-X_i)\ln(1-\theta)]$$

(c) To find the value of θ that maximizes the log-likelihood function, we differentiate it and set it to zero:

$$\frac{d \ln L(\theta)}{d\theta} = \sum_{i=1}^{n} \left(\frac{X_i}{\theta} - \frac{1-X_i}{1-\theta} \right) = 0$$

Expanding $\frac{d \ln L(\theta)}{d\theta}$ and rearranging, we get:

$$\sum_{i=1}^{n} X_i = \theta \sum_{i=1}^{n} (1-X_i) + \theta \sum_{i=1}^{n} X_i$$

Further simplifying, we find:

$$\theta = \frac{1}{n} \sum_{i=1}^{n} X_i$$

This is consistent with the given $\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} X_i$, thus completing the proof.

Question 7

(a) For $p(x_i = 1|y)$, the Maximum Likelihood Estimation (MLE) is given by:

$$\frac{\text{Number of times } x_i = 1 \text{ when } y \text{ takes a particular value}}{\text{Total number of times } y \text{ takes that particular value}}$$

	$y = +1$	$y = 0$	$y = -1$
$x_1 = 1$	0.5	0.5	1/3

	$y = +1$	$y = 0$	$y = -1$
$x_2 = 1$	1	0.5	1/3
$x_3 = 1$	0.5	1	1/3
$x_4 = 1$	0.5	1	2/3

(b) Compute the MLE

$$p(y = k) = \frac{\text{Number of times } y = k}{\text{Total number of data points}}$$

Using the above formula:

- $p(y = +1) = 2/7$
- $p(y = 0) = 2/7$
- $p(y = -1) = 3/7$

(c)

For naive Bayes, the class assigned to a new data point is the one that maximizes the following:

$$p(y = k|\mathbf{x}) \propto p(y = k) \prod_{i=1}^{n} p(x_i|y = k)$$

Where n is the number of features, and in this case $n = 4$.

We have three classes $y = +1, y = 0$, and $y = -1$. We will calculate the above value for each class and see which class gives the maximum value.

1. For $y = +1$:

$$p(y = +1|\mathbf{x}) \propto p(y = +1) \times p(x_1 = 1|y = +1) \times p(x_2 = 1|y = +1) \times p(x_3 = 1|y = +1) \times p(x_4 =$$

$$p(y = +1|\mathbf{x}) \propto \frac{2}{7} \times 0.5 \times 1 \times 0.5 \times 0.5 = \frac{1}{28}$$

1. For $y = 0$:

$$p(y = 0|\mathbf{x}) \propto p(y = 0) \times p(x_1 = 1|y = 0) \times p(x_2 = 1|y = 0) \times p(x_3 = 1|y = 0) \times p(x_4 = 1|y = 0)$$

$$p(y = 0|\mathbf{x}) \propto \frac{2}{7} \times 0.5 \times 0.5 \times 1 \times 1 = \frac{1}{14}$$

1. For $y = -1$:

$$p(y = -1|\mathbf{x}) \propto p(y = -1) \times p(x_1 = 1|y = -1) \times p(x_2 = 1|y = -1) \times p(x_3 = 1|y = -1) \times p(x_4 =$$

$$p(y = -1|\mathbf{x}) \propto \frac{3}{7} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} = \frac{2}{189}$$

The maximum value is $\frac{1}{7}$ which corresponds to $y = 0$.

Thus, the new data point with the given feature values is classified as class $y = 0$.

Question 8

In [35]:
```
#import necessary libraries

import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [36]:
```
# read JSON
path = 'fantasy_100.json'
df = pd.read_json(path, lines = True)
```

In [37]:
```
df.info()
```

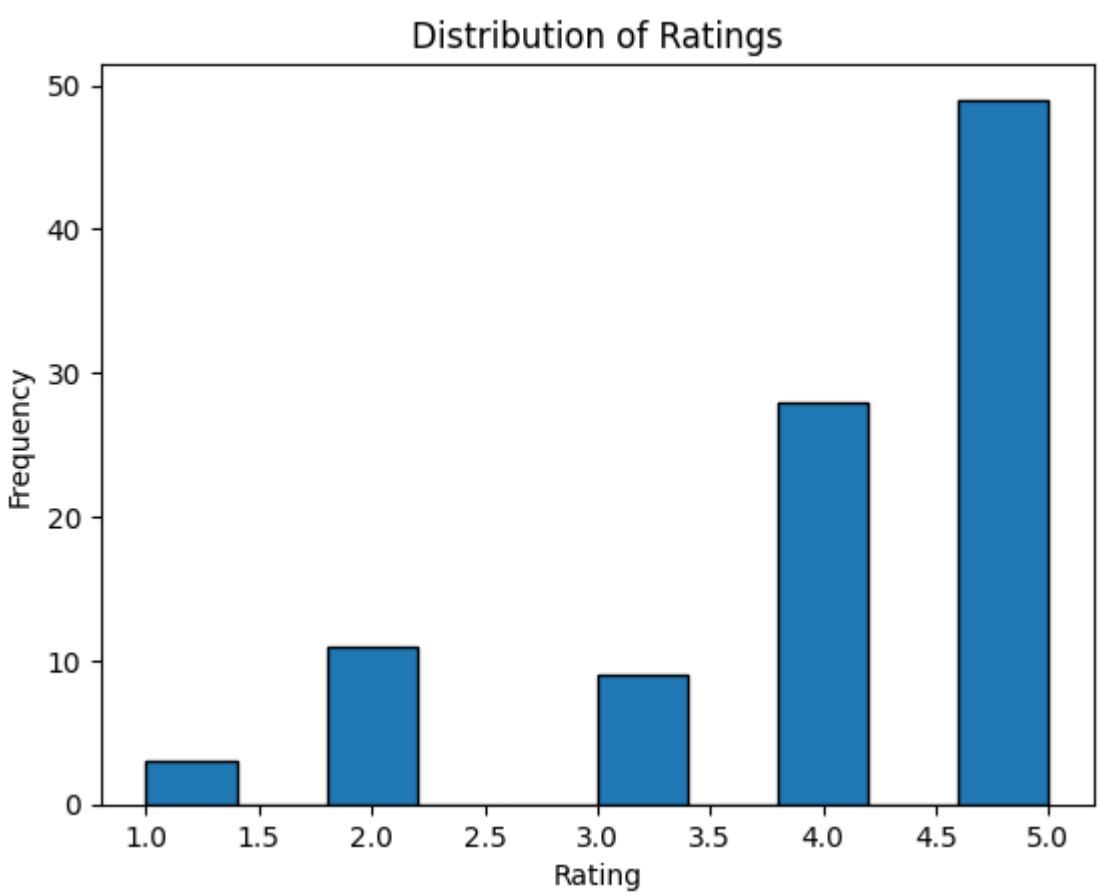
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   user_id       100 non-null    object
 1   book_id       100 non-null    int64 
 2   review_id     100 non-null    object
 3   rating        100 non-null    int64 
 4   review_text   100 non-null    object
 5   date_added    100 non-null    object
 6   date_updated  100 non-null    object
 7   read_at       100 non-null    object
 8   started_at    100 non-null    object
 9   n_votes       100 non-null    int64 
 10  n_comments    100 non-null    int64 
dtypes: int64(4), object(7)
memory usage: 8.7+ KB
```

In [38]:
```
df['rating'].describe()
```

Out[38]:
```
count    100.000000
mean       4.090000
std        1.137914
min        1.000000
25%        4.000000
50%        4.000000
75%        5.000000
max        5.000000
Name: rating, dtype: float64
```

(a)

In [39]:
```
# (a) plot the distribution of rating
plt.hist(df['rating'], bins=10, edgecolor='black')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



(b)

In [40]:
```
#compute the length of each review and save as a new col
df['review_length'] = df['review_text'].apply(len)

# Define predictor (X) and target (y)
X = df[['review_length']]
Y = df['rating']

#Train the model
model_1 = LinearRegression()
model_1.fit(X,Y)

#Prediction
predictions = model_1.predict(X)

# Compute the Mean Squared Error (MSE)
mse = mean_squared_error(Y, predictions)

#compute the parameters
theta_0 = model_1.intercept_
theta_1 = model_1.coef_[0]

print(f"Theta_0 (Intercept): {round(theta_0,6)}")
print(f"Theta_1 (Slope): {round(theta_1,6)}")
print(f"Mean Squared Error: {round(mse,6)}")
```

```
Theta_0 (Intercept): 3.983948
Theta_1 (Slope): 0.000119
Mean Squared Error: 1.265752
```

Interpretation: It is obvious that the length of the review has no impact over the rating of a movie

The values are as follows:

- Θ_0 (Intercept): 3.983948
- Θ_1 (Slope): 0.000119
- Mean Squared Error (MSE): 1.265752

Interpretation:

- Θ_0 :if a review had a length of 0 (which isn't practically possible for a genuine review), the predicted rating would be approximately 4.
- Θ_1 (Slope) indicates the change in the predicted 'rating' for a unit increase in 'review_length'.This value is very small, suggesting that the length of the review has a minimal effect on the rating. (Almost no impact)
- The MSE is a measure of how well the model's predictions match the actual values. A smaller MSE indicates a better fit of the model to the data. In this context, the MSE provides a quantified measure of the average squared difference between the predicted ratings and the actual ratings.

(c)

In [41]:
```
# Define predictor (X1) (X2) and target (Y)
#Y = df['rating']
X2 = df[['review_length', 'n_comments']]
Y2 = df['rating']

model_2 = LinearRegression()
model_2.fit(X2,Y2)

predictions = model_2.predict(X2)

mse2 = mean_squared_error(Y2, predictions)

theta0 = model_2.intercept_
theta1, theta2 = model_2.coef_

print(f"Theta_0 (Intercept): {round(theta0,6)}")
print(f"Theta_1 (Coefficient for review length): {round(theta1,6)}")
print(f"Theta_2 (Coefficient for number of comments): {round(theta2,6)}")
print(f"Mean Squared Error: {round(mse,6)}")
```

```
Theta_0 (Intercept): 3.954321
Theta_1 (Coefficient for review length): 7.2e-05
Theta_2 (Coefficient for number of comments): 0.108071
Mean Squared Error: 1.265752
```

Observations:

- The change in Θ_1 when introducing the number of comments as an additional predictor is due to the shared variance between review length and the number of comments and the model's attempt to attribute the explained variance in the star rating to each predictor effectively.

(c)

In [42]:
```
# Sample data (Replace this with your data)
# X - Feature matrix with columns 'review length' and 'number of comments'
# y - Target vector (e.g., star rating)
X = df[['review_length', 'n_comments']].values
y = df['rating'].values

# Polynomial feature expansion function
def polynomial_expansion(X):
    x1 = X[:, 0]  # review length
    x2 = X[:, 1]  # number of comments

    # Calculate polynomial features
    x1_squared = x1**2
    x2_squared = x2**2
    interaction = x1 * x2

    # Create new feature matrix
```

```
    X_poly = np.column_stack((x1, x2, x1_squared, x2_squared, interaction))

    return X_poly

# Transform the original features
X_poly = polynomial_expansion(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_st

# Create and train the linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predictions and MSE
y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", round(mse,3))
```

```
Mean Squared Error: 1.423
```

Feature Vector:

1. Intercept term: 1 (this is to account for the bias term, Θ_0)
2. Original `review length` : x_1
3. Original `number of comments` : x_2
4. Square of `review length` : x_1^2
5. Square of `number of comments` : x_2^2
6. Interaction term between `review length` and `number of comments` : $x_1 \times x_2$

polynomial feature vector:

$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 \times x_2 \end{bmatrix}$$

This vector now contains the original features, their squares, and the interaction term, leading to a more flexible model that can capture non-linear relationships and interactions between the features.

Mean Squared Error (MSE): 1.423

This is the average squared difference between the observed outcomes and the model's predictions.

Observations:

1. **Complexity**: Introducing polynomial features increases the complexity of the model. This allows the model to fit more complex, non-linear relationships in the data.
2. **Risk of Overfitting**: While polynomial regression can capture non-linear patterns, it also comes with an increased risk of overfitting, especially with high-degree polynomials.
3. **Interactions**: The interaction term $x_1 \times x_2$ captures the combined effect of `review length` and `number of comments` on the star rating. For instance, it helps understand scenarios where longer reviews with many comments have a different impact on the star rating than what would be predicted by considering each feature independently.