Practice Midterm (Sample Solution)

Summer, 2023

Instructor: Ziang Zhang

Student Name: ____________________________

Student ID: ____________________________

Instructions

- Time allowed: 3 hours; Total points: 100.
- One non-programmable calculator is allowed.
- Students are expected to complete *all* the questions within the space provided. Extra space is provided at the end of the exam. If the extra space is used, please indicate clearly which question is being answered.
- Write your answers clearly and show your detailed steps. If the examiner cannot read an answer or follow your steps, it will be marked as incorrect.
- Good luck!

Multiple Choices (20 pts):
Solution: BBBCD, CDDCC, ADCDD, DCDDC

1. (15 pts) For each of the following statements in linear algebra, either prove it is true using the basic properties and definitions discussed in class, or provide a counter-example showing it is false.

   (a) (5 pts) If $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix with eigenvalues $\{\lambda_i\}_{i=1}^n$, then its determinant can be computed as $|A| = \prod_{i=1}^n \lambda_i$.

   Solution: True.
   Apply the eigen-decomposition to A: $A = UDU^T$, then by the properties of determinant:

$$\begin{aligned} |A| &= |UDU^T| \\ &= |U||D||U^T| \\ &= |D||U||U^{-1}| \\ &= |D||U||U|^{-1} \\ &= |D| = \prod_{i=1}^n \lambda_i, \end{aligned} \tag{1}$$

   because $U^{-1} = U^T$.

   (b) (5 pts) If $A \in \mathbb{R}^{n\times n}$ is an invertible matrix, $\mathrm{tr}(A)$ cannot be zero.

   Solution: False.
   Counter example is $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. A is invertible but $tr(A) = 1 - 1 = 0$.

   (c) (5 pts) If $A \in \mathbb{R}^{n\times n}$ is a positive semi-definite matrix, A^p will also be a positive semi-definite matrix for any positive integer p.

   Solution: True.
   Let $A = UDU^T$ using its eigen-decomposition, where $D_{ii} = \lambda_i$, then:

$$\begin{aligned} A^2 = AA &= UDU^TUDU^T \\ &= UD(U^TU)DU^T \\ &= U(DD)U^T \\ &= UD^2U^T. \end{aligned} \tag{2}$$

   where D^2 has diagonal value being λ_i^2. The case for general $p \in \mathbb{Z}^+$ follows by the same logic.

2. (20 pts) (*Regularized Regression*) Suppose the data is generated from the following linear regression model:

$$y_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i,$$

where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$.

Denote the training data as $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$ where n is the sample size. Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ denote the design matrix, with its ith row being $\boldsymbol{x}_i^T$.

(a) (3 pts) Let $\mathbf{y} = (y_1, ..., y_n)^T \in \mathbb{R}^n$. What is the conditional distribution of $\mathbf{y}$ given all the features $\mathbf{X}$? Can you also conclude the marginal distribution of $\mathbf{y}$ based on the information above?

Solution: The conditional distribution of $\mathbf{y}$ given all the features $\mathbf{X}$ is

$$\mathbf{y}|\mathbf{X} \sim N_p(\mathbf{X}\boldsymbol{\beta}, \sigma^2 I_n).$$

We don't have enough information to determine the marginal distribution of $\mathbf{y}$, we do not know the distribution of the features.

(b) (5 pts) Assume $\mathbf{X}^T\mathbf{X}$ is invertible, derive the expression of the MLE $\hat{\boldsymbol{\beta}}_{ML}$ for $\boldsymbol{\beta}$. Recall for a random variable $\mathbf{z} \sim N(\mu, \boldsymbol{\Sigma})$, its density is:

$$f(\mathbf{z}) = \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \boldsymbol{\mu})\right),$$

Solution:

$$f(\mathbf{y}|\mathbf{X}) \propto \exp\left(- \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right). \tag{3}$$

Hence

$$\begin{aligned} l(\boldsymbol{\beta}) &= -\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}). \\ \nabla_{\boldsymbol{\beta}} l &= -\frac{1}{2\sigma^2}(2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - 2\mathbf{X}^T\mathbf{y}) = 0 \\ &\Rightarrow \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y} \\ &\Rightarrow \hat{\boldsymbol{\beta}}_{ML} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{4}$$

(c) (5 pts) Unfortunately you cannot collect too much data due to the tight budget, hence $n < p$. Show that $\hat{\boldsymbol{\beta}}_{ML}$ above is not uniquely defined (*hint: you just need to show* $\mathbf{X}^T\mathbf{X}$ *is not invertible*).

Solution:
Approach One (Using SVD): Let $\mathbf{X} = UDV^T$ then

$$\mathbf{X}^T\mathbf{X} = VD^TU^TUDV^T = VD^TDV^T.$$

Let $D^* = D^TD$, since $n < p$, then:

$$\begin{aligned} D^* = D^TD &= \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{p\times n} \begin{bmatrix} d_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n & 0 & \cdots & 0 \end{bmatrix}_{n\times p} \\ &= \begin{bmatrix} d_1^2 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & d_2^2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n^2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}_{p\times p} . \end{aligned} \tag{5}$$

The matrix $\mathbf{X}^T\mathbf{X}$ is not invertible, as $|\mathbf{X}^T\mathbf{X}|$ will be zero.

Approach Two (Using rank-nullity theorem):
If you recall the rank-nullity theorem from your linear algebra course, for a matrix X, the sum of its nullity and its rank must be its number of columns (p).
The matrix X has n rows, so the rank must be less than or equal to n. Hence there exists at least one none zero vector $\mathbf{v}$ such that $\mathbf{Xv} = 0$ hence $\mathbf{X}^T\mathbf{Xv} = 0$. The matrix $\mathbf{X}^T\mathbf{X}$ therefore must be non-invertible.

(d) (7 pts) To solve the above problem, you decide to use a Ridge regression to obtain the estimate $\hat{\boldsymbol{\beta}}_{Reg}$. Derive the formula of $\hat{\boldsymbol{\beta}}_{Reg}$ and explain why it is well-defined even if $\mathbf{X}^T\mathbf{X}$ is not invertible (*hint:* when deriving $\hat{\boldsymbol{\beta}}_{Reg}$, you can take the original loss function to be either the negative log-likelihood or the RSS).

Solution:
Your loss function could be either negative log-likelihood function or the RSS. Their results are essentially equivalent. The derivation using the RSS is the same as in the lecture slides, here we show the derivation using the negative log-likelihood:

$$l_{Reg}(\boldsymbol{\beta}, \sigma^2) = \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

$$\nabla_{\boldsymbol{\beta}} l_{Reg}(\boldsymbol{\beta}, \sigma^2) = -\frac{1}{\sigma^2}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta} = 0$$

$$\hat{\boldsymbol{\beta}}_{Reg} = (\mathbf{X}^T\mathbf{X} + 2\sigma^2\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

To show that $(\mathbf{X}^T\mathbf{X} + 2\sigma^2\lambda\mathbf{I})$ is always invertible, note that:

$$\begin{aligned} |\mathbf{X}^T\mathbf{X} + 2\sigma^2\lambda\mathbf{I}| &= |UDU^T + 2\sigma^2 I| \\ &= |UDU^T + 2\sigma^2 UIU^T| \\ &= |U(D + 2\sigma^2 I)U^T| \\ &= |(D + 2\sigma^2 I)| > 0 \end{aligned} \tag{6}$$

where UDU^T is the eigen-decomposition of $\mathbf{X}^T\mathbf{X}$.

3. (15 pts) (*PCA*) Assume you have collected n pairs of *iid* feature vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ in $\mathbb{R}^p$. They are stacked together into one design matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$. You want to develop a predictive model using the PC regression method. For simplicity, assume the feature vectors are already centered to have zero means.

   (a) (3 pts) Your collaborator told you that if you only want to keep k terms in your final model, performing a best-subset selection with size k on the original features or a PC regression using the first k PCs will be equivalent since the first k PCs are designed to optimize the variance explained. Do you agree with your collaborator? If not, explain using no more than two sentences why your collaborator is wrong.

   Solution:
   No. While both methods select k features, best-subset selection retains k original features, whereas PCA regression uses k principal components, which are linear combinations of all the original features. Hence, they are not equivalent.

   (b) (6 pts) Derive the expressions of the first (sample) PC direction $\mathbf{v}_1$. Given the kth PC direction $\mathbf{v}_k$, how will you compute the kth PC score z_{ik} of the observation $\mathbf{x}_i$?

   Solution:
   Note that $Var(\mathbf{v}^T\mathbf{x}) = \mathbf{v}^T\Sigma_x\mathbf{v} \approx \frac{1}{n}\mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v}$. The definition of the kth sample PC direction is:

$$\mathbf{v}_k = \underset{\substack{\|\mathbf{v}\|=1 \\ \mathbf{v}^T\mathbf{v}_j=0,\ \forall j<k}}{\arg\max} \quad \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v}. \tag{7}$$

   To compute the first sample direction, note that:

$$\mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} = \mathbf{v}^T\mathbf{U}\mathbf{D}\mathbf{U}^T\mathbf{v} = \tilde{\mathbf{v}}^T\mathbf{D}\tilde{\mathbf{v}}, \tag{8}$$

   where $\mathbf{U}\mathbf{D}\mathbf{U}^T$ is the eigen-decomposition of the PSD matrix $\mathbf{X}^T\mathbf{X}$, and $\tilde{\mathbf{v}} = \mathbf{U}^T\mathbf{v}$. Note that $\tilde{\mathbf{v}}^T\tilde{\mathbf{v}} = \mathbf{v}^T\mathbf{U}^T\mathbf{U}\mathbf{v} = \mathbf{v}^T\mathbf{v} = 1$. Hence the optimization problem just becomes optimizing $\tilde{\mathbf{v}}^T\mathbf{D}\tilde{\mathbf{v}}$.
   Since $\tilde{\mathbf{v}}^T\mathbf{D}\tilde{\mathbf{v}} = \sum_{i=1}^p \lambda_i\tilde{\mathbf{v}}_i^2 \leq \sum_{i=1}^p \lambda_1\tilde{\mathbf{v}}_i^2 = \lambda_1$. This implies the optimal choice will be $\tilde{\mathbf{v}} = \mathbf{e}_1$, and hence $\mathbf{v} = \mathbf{U}\mathbf{e}_1 = \mathbf{u}_1$, where $\mathbf{u}_1$ is the first eigenvector of $\mathbf{X}^T\mathbf{X}$.

(c) (6 pts) Now assume the covariance matrix $\mathbf{\Sigma_X}$ is known, and you have computed (population) PC directions $\{\mathbf{v}_k\}_{k=1}^{p}$ using the true covariance. Prove that the two resulting (population) PC scores z_{ik} and z_{ij} (where $k \neq j$) are uncorrelated. Do we need any additional assumptions to claim that they are independent?

Solution:
The PC scores can be computed as $z_{ik} = \mathbf{v}_k^T \mathbf{x}_i$ and $z_{ij} = \mathbf{v}_j^T \mathbf{x}_i$. Hence:

$$\begin{aligned} Cov(z_{ik}, z_{ij}) &= Cov(\mathbf{v}_k^T \mathbf{x}_i, \mathbf{v}_j^T \mathbf{x}_i) \\ &= \mathbf{v}_k^T \mathbf{\Sigma}_x \mathbf{v}_j \\ &= \mathbf{v}_k^T \mathbf{U}\mathbf{D}\mathbf{U}^T \mathbf{v}_j \\ &= \mathbf{e}_k^T \mathbf{D} \mathbf{e}_j \\ &= \lambda_j \mathbf{e}_k^T \mathbf{e}_j \\ &= 0, \end{aligned} \tag{9}$$

where λ_i denotes the ith diagonal value in $\mathbf{D}$.

4. (15 pts) *(Generative Method)* Assume you want to derive a generative classifier for an outcome variable y that takes two possible levels $k = 0, 1$. Conditional on $y = k$, the feature vector $\mathbf{x} \in \mathbb{R}^2$ is normally distributed with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, with density

$$f(\mathbf{x}|y=k) = \frac{1}{|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right).$$

(a) (5 pts) Derive the quadratic discriminant function $\delta_k(\mathbf{x})$, and show maximizing δ_k is equivalent to maximizing the probability $P(y = k|\mathbf{x})$.

Solution: By Bayes theorem:

$$\begin{aligned} P(y=k|\mathbf{x}) &= \frac{f(\mathbf{x}|y=k)P(y=k)}{f(\mathbf{x})} \\ &\propto f(\mathbf{x}|y=k)P(y=k) \end{aligned} \tag{10}$$

Here, $f(\mathbf{x})$ is the marginal likelihood that does not depend on k.
We want to find the class k that maximizes $P(y = k|\mathbf{x})$, which can be done by equivalently maximizing its log:

$$\begin{aligned} &\log\left[f(\mathbf{x}|y=k)P(y=k)\right] = \log f(\mathbf{x}|y=k) + \log P(y=k) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{p}{2}\log(2\pi) + \log P(y=k) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log \pi_k + C, \end{aligned} \tag{11}$$

where C is a constant that does not depend on k.
Therefore, maximizing $P(y = k|\mathbf{x})$ over k is equivalent to maximizing

$$\delta_k = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log \pi_k$$

over k.

(b) (10 pts) Assume you have observed $n = 10$ data points, with 4 observations in the 0 class, 6 observations in the 1 class.

The estimated means are $\bar{\boldsymbol{\mu}}_0 = [-3/2, 3/2]^T, \bar{\boldsymbol{\mu}}_1 = [1/2, 1]^T$.

The estimated covariances are $\hat{\mathbf{\Sigma}}_0 = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}$ and $\hat{\mathbf{\Sigma}}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$.

Compute the estimated discriminant function $\hat{\delta}_k(\mathbf{x})$ for each k. Estimate the Bayes decision boundary between the two classes. Finally, construct a Bayes classifier to classify $\boldsymbol{x}_1^* = [2, 4]^T$ and $\boldsymbol{x}_2^* = [4, 1]^T$.

Solution: The estimate $\hat{\pi}_k = 0.4$ or 0.6 for $k = 0$ or 1.
The estimated discriminant functions are hence:

$$\begin{aligned} \hat{\delta}_0 &= \log(0.4) - \frac{1}{2}\log(5) - \frac{1}{2}\begin{bmatrix} x_1 + \frac{3}{2} & x_2 - \frac{3}{2} \end{bmatrix} \begin{bmatrix} 1/5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 + \frac{3}{2} \\ x_2 - \frac{3}{2} \end{bmatrix} \\ &= \log(0.4) - \frac{1}{2}\log(5) - \frac{1}{2}\left[\frac{1}{5}(x_1 + \frac{3}{2})^2 + (x_2 - \frac{3}{2})^2\right] \end{aligned} \tag{12}$$

and

$$\begin{aligned} \hat{\delta}_1 &= \log(0.6) - \frac{1}{2}\log(5) - \frac{1}{2}\begin{bmatrix} x_1 - \frac{1}{2} & x_2 - 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1/5 \end{bmatrix} \begin{bmatrix} x_1 - \frac{1}{2} \\ x_2 - 1 \end{bmatrix} \\ &= \log(0.6) - \frac{1}{2}\log(5) - \frac{1}{2}\left[(x_1 - \frac{1}{2})^2 + \frac{1}{5}(x_2 - 1)^2\right]. \end{aligned} \tag{13}$$

The Bayes decision boundary is where $\hat{\delta}_0(\mathbf{x}) = \hat{\delta}_1(\mathbf{x})$.
A Bayes classifier assigns observation to the class where the discriminant function is maximized.
The computation details are skipped here: $\hat{\delta}_0(\mathbf{x}_1^*) = -6.07$ and $\hat{\delta}_1(\mathbf{x}_1^*) = -3.34$; $\hat{\delta}_0(\mathbf{x}_2^*) = -4.87$ and $\hat{\delta}_1(\mathbf{x}_2^*) = -7.44$.
Hence $\hat{y}(\mathbf{x}_1^*) = 1$ and $\hat{y}(\mathbf{x}_2^*) = 0$.

5. (15 pts) (*Discriminative Method*) Recall the K-Nearest Neighborhood (KNN) method which we covered in the lecture as a way to do regression. It can also be used as a way to construct a discriminative classifier. For simplicity, assume that we have one feature $x \in \mathbb{R}$ to predict an outcome variable $y = 0$ or 1.

   (a) (5 points) Given an integer K, write the algorithm or pseudo-code to describe how can you use the training data $\{x_i, y_i\}_{i=1}^n$ to estimate $P(y = 1|x = x_0)$, and hence construct a Bayes classifier $\hat{y}(x_0)$ for a new feature value x_0.

   Solution: The algorithm can be described as the following:

   i. For each training feature x_i, compute the distance $d_i = |x_i - x_0|$.
   ii. Sort the distances from small to big, and take the first K indices $\mathcal{N}(x_0)$.
   iii. Now, calculate the conditional probability $P(y = 1|x = x_0)$ as the proportion of instances in $\mathcal{N}(x_0)$ where $y_i = 1$.
   iv. Construct a Bayes classifier $\hat{y}(x_0)$: Classify x_0 to the class 1 if $P(y = 1|x = x_0) > 0.5$, else classify x_0 to the class 0.

(b) (5 points) Write the algorithm or pseudo-code to perform the leave-one-out CV to tune the hyper-parameter K based on the overall (testing) error rate (misclassification rate) for your classifier above.

Solution:

i. Initialize the range of K to be from 1 to n.

ii. For each value of K in the range:
   - Initialize the total error number E_{total} as 0.
   - Loop through the training data $\{x_i, y_i\}_{i=1}^{n}$:
     - Remove the i-th instance from the training data, and let it be the test instance.
     - Use the rest of the data as the training data to construct a Bayes classifier $\hat{y}(x_i)$ using KNN as described before.
     - Predict the class of the i-th instance, and compare it with y_i. If $\hat{y}(x_i)$ is incorrect, increment E_{total} by 1.
   - After going through all instances, calculate the overall error rate $E_{\text{overall}} = \frac{E_{\text{total}}}{n}$ for this value of K.

iii. After going through all values of K, choose the K that gives the smallest E_{overall}.

(c) (5 points) The following R code aims to perform a 3-fold CV to estimate the error rate for the previous KNN classifier with $K = 5$. However, there is a lethal bug in the code that invalidates the estimated (testing) error rate. Explain what is the bug in the code, and in which way it bias the error rate.
Note: the function `knn_classification` was used to obtain the prediction for the testing set using the model fitted in the training set. You can assume this function is correct.

```r
K_fold_CV_wrong <- function(K_KNN = 5, K_CV = 3, train_data){
  indx <- 1:nrow(train_data)
  # Split the indices into K folds
  kfolds <- createFolds(y = indx, k = K_CV, list = TRUE,
   returnTrain = FALSE)
  error_rate <- c()
  for (i in 1:K_CV) {
    # The training data in the i-th iteration
    selected_training_data <- train_data[-kfolds[[i]], , drop = F
   ]
    # The validation data in the i-th iteration
    selected_validation <- train_data
    # Compute the prediction
    i_th_pred <- knn_classification(k = K_KNN,
                                     training = selected_training_data,
                                     testing = selected_validation)
    # Compute the error rate in the i-th iteration
    new_error_rate <- mean(selected_validation$y == i_th_pred)
    error_rate <- c(new_error_rate, error_rate)
  }
  # Compute the mean error rate
  mean(error_rate)
}
```

Solution:
The selection of validation set in each iteration is incorrect, it should be:
`selected_validation <- train_data[kfolds[[i]], , drop = F]`.
With this bug, the model is validated on a set in which some observations have been used in the training procedure, hence the error rate is likely underestimated.