Assignment-01 ADM

Chandima Attanayake

2025-03-02

Part A

#QA1. Purpose of Regularization

Regularization is used to prevent overfitting in predictive modeling by adding a penalty to the complexity of the model. This ensures that the model generalizes in good manner to unseen data by preventing excessive complexity. In this case, the regularization techniques such as Lasso (L1) and Ridge (L2) regression fulfill the above requirement by shrinking coefficients. Therefore, it is improving the model performance on new data.

As an example, we can take the linear regression model to predict the house prices. Without regularization, the model may assign large weights to less relevant features such as zip code and therefore, it could lead to poor generalization. Accordingly, regularization helps to control such unnecessary complexity and generalize the data more effectively.

#QA2. Role of Loss Function

The loss function measures the extent of predicted values match the actual values in a supervised learning model. It is a key component in optimization which guides the model learning process. In this case, common loss function is working for regression as well as classification as below.

Common Loss Functions for Regression

1. Mean Squared Error – This loss function corrects large errors more heavily than small ones by squaring the differences between predicted and actual values.
2. Mean Absolute Error - This calculates all differences between predictions and actual values and treats all errors equally. However, this can be less stable than MSE.

Common Loss Functions for Classification

1. Cross-Entropy Loss – This is commonly using for logistic regression and deep learning. This calculates the possibility of variance of the actual class label from the predicted probability distribution.
2. Hinge Loss – This is used in Support Vector Machines. Hinge loss ensures that the points are classified correctly in order to encourage that they to be at a safe margin from the decision boundary.

#QA3. Trust in a Model with Low Training Error on a Small Dataset

A model with an extremely small training error on a small dataset is likely overfitting. In this case, overfitting arises when a model captures noise rather than underlying patterns which creates an unreliable for new data.

As an example, if a deep neural network perfectly classifies 50 training samples, it could fail on new data as it has memorized rather than learned patterns. Therefore, to address this issue, it is required to use cross validation or regularization techniques to improve generalization.

#QA4. Role of Lambda in Regularized Linear Models

In regularized models like Lasso and Ridge regression, the lambda parameter () controls the degree of regularization.

A higher lambda value increases regularization, reducing coefficient sizes and leading to a simpler model. A lower lambda value reduces regularization, allowing the model to fit the data more closely. The optimal lambda is determined through cross-validation to balance model complexity and predictive accuracy.

In Lasso regression, increasing lambda from 0.01 to 0.1 can shrink small coefficients to zero, effectively selecting only the most important features.

Part B: Practical Implementation in R

```r
# Load required libraries
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
# Select relevant attributes
Carseats_Filtered <- Carseats %>% select(Sales, Price, Advertising, Population, Age, Income, Education)

# Prepare data
set.seed(123)  # For reproducibility
preProc <- preProcess(Carseats_Filtered[, -1], method = c("center", "scale"))
scaled_data <- predict(preProc, Carseats_Filtered[, -1])

X <- as.matrix(scaled_data)
Y <- Carseats_Filtered$Sales
```

```r
# Split data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(Y, p = 0.8, list = FALSE)
X_train <- X[trainIndex, ]
X_test <- X[-trainIndex, ]
Y_train <- Y[trainIndex]
Y_test <- Y[-trainIndex]
```

QB1. Building a Lasso Regression Model

```r
# Train Lasso model
cv.lasso <- cv.glmnet(X_train, Y_train, alpha = 1)
best_lambda_lasso <- cv.lasso$lambda.min

# Output best lambda
best_lambda_lasso
```

```
## [1] 0.00451294
```

The best lambda value of 0.00451294 is selected using cross validation. This value minimizes the validation error and it provides the best balance between bias and variance. A lower lambda allows more flexibility in fitting the data while a higher lambda may overly constrain the model the reducing accuracy.

QB2. Coefficient of Price in the Best Model

```r
# Extract coefficients
lasso_coef <- coef(cv.lasso, s = best_lambda_lasso)
lasso_coef["Price", ]
```

```
## [1] -1.287537
```

The coefficient of `Price` is -1.287537, which means that for every unit increase in price could decrease the sales by 1.29 units. This negative relationship aligns with economic principles, where higher prices typically lead to lower demand.

QB3. Effect of Lambda on Variable Selection

```r
# Coefficients at lambda = 0.01
lasso_coef_001 <- coef(glmnet(X_train, Y_train, alpha = 1, lambda = 0.01))
num_nonzero_001 <- sum(lasso_coef_001 != 0)

# Coefficients at lambda = 0.1
lasso_coef_01 <- coef(glmnet(X_train, Y_train, alpha = 1, lambda = 0.1))
num_nonzero_01 <- sum(lasso_coef_01 != 0)

list(lambda_001 = num_nonzero_001, lambda_01 = num_nonzero_01)
```

```
## $lambda_001
## [1] 7
##
## $lambda_01
## [1] 6
```

- At **lambda = 0.01**, 7 attributes remain in the model, meaning most features are retained.
- At **lambda = 0.1**, only 6 attributes remain, indicating that some coefficients have been shrunk to zero.
- As lambda increases, fewer variables remain in the model because Lasso penalizes smaller coefficients, forcing some to become exactly zero, effectively performing feature selection.

QB4. Elastic Net Model with Alpha = 0.6

```
# Train Elastic Net model with alpha = 0.6
cv.elasticnet <- cv.glmnet(X_train, Y_train, alpha = 0.6)
best_lambda_elasticnet <- cv.elasticnet$lambda.min

# Output best lambda
best_lambda_elasticnet
```

```
## [1] 0.0252093
```

The best lambda for the Elastic Net model is 0.0252093, balancing the penalties from both L1 (Lasso) and L2 (Ridge) regularization. This helps in retaining correlated variables while shrinking less important coefficients and making the model more robust compared to pure Lasso or Ridge regression.