Statistical Learning, Homework 3

Virginia Leombruni 247160

Introduction

The analysis will be conducted on a gene expression dataset from 79 leukaemia patients, divided into two subgroups: patients with chromosome translocation (indicated as '1') and cytogenetically normal patients (indicated as '-1'). The data come from the gene_expr.tsv file, which includes the expression of 2,000 genes and an additional column with patient labels. A supervised subgroup prediction analysis will be performed using support vector machines.

Split the data into training and testing sets

```r
set.seed(1)  # for reproducibility
train_indices <- sample(1:nrow(data), 0.7 * nrow(data))  # 70% for training

# Split the data into training and testing sets
train_index <- sample(1:nrow(data), 0.7 * nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
test_data <- test_data[, -1]
train_data <- train_data[, -1]

table(data$y)
```

```
-1  1
42 37
```

The results indicate that random sampling preserved the distribution of labels fairly well between the training and test sets, ensuring that both sets contained a similar representation of labels as the original dataset.

The dataset has 42 instances of label `-1` and 37 instances of label `1`, indicating a slightly imbalanced dataset. The training set has 29 instances of label `-1` and 26 instances of label `1`. The testing set has 13 instances of label `-1` and 11 instances of label `1`.

The label proportions in the training set are consistent with the overall dataset. Approximately 53.8% of the training set is label `-1` (29 out of 55), and 46.2% is label `1` (26 out of 55). This is close to the overall proportions in the dataset (53.2% for -1 and 46.8% for 1). The testing set also maintains a similar distribution with 54.2% label -1 (13 out of 24) and 45.8% label 1 (11 out of 24).

The label distribution in both the training and testing sets closely matches the overall distribution of labels in the dataset. This indicates that the sampling process has maintained the proportionality of labels, which is crucial for training a model that generalizes well to unseen data. This stratification helps ensure that the model will not be biased towards a particular label due to imbalanced data.

Linear kernel

```r
set.seed(11)
cost_range <- c(0.001, 0.01, 0.1, 1, 5, 10, 100)
tune.out <- tune(svm, y ~ ., data = train_data, kernel = "linear", ranges = list(cost = cost_rang
tune.summary <- summary(tune.out)
# tune.summary

bestmod_linear <- tune.out$best.model
# summary(bestmod_linear)

ypred_linear <- predict(bestmod_linear, test_data, decision.values = TRUE)  # our usual confusion
table(predict = ypred_linear, truth = test_data$y)
```

```
       truth
predict -1  1
     -1 12  2
     1   2  8
```

```r
accuracy <- mean(ypred_linear == test_data$y)
print(paste("Accuracy of SVM model for linear kernel:", accuracy))
```

```
[1] "Accuracy of SVM model for linear kernel: 0.833333333333333"
```

The tuning process identified a cost parameter of 0.01 as optimal for the linear SVM. The model's best performance during cross-validation was an error rate of 34.67%. When applied to the test set, the model achieved an accuracy of 83.33%. The confusion matrix suggests that the model is capable of distinguishing between the two classes with reasonable accuracy, with a low number of misclassifications. The consistency in accuracy between this and previous experiments (both achieving around 83.33%) indicates the robustness of the model under different seeds and suggests reliable performance in practical applications.

Radial kernel

```r
cost_range <- c(0.1, 1, 10, 100, 1000)
gamma_range <- c(0.5, 1, 2, 3, 4)
tune.out <- tune(svm, y ~ ., data = train_data, kernel = "radial", ranges = list(cost = cost_rang
    gamma = gamma_range))
# summary(tune.out)

bestmod_radial <- tune.out$best.model
# summary(bestmod_radial)

ypred_radial <- predict(bestmod_radial, test_data, decision.values = TRUE)  # our usual confusion
# table(predict=ypred_radial, truth=test_data$y) Evaluate accuracy
accuracy <- mean(ypred_radial == test_data$y)
print(paste("Accuracy of SVM model for radial kernel:", accuracy))
```

```
[1] "Accuracy of SVM model for radial kernel: 0.583333333333333"
```

The tuning process identified a cost parameter of 0.1 as optimal for the radial SVM. However, the best performance achieved during cross-validation was an error rate of 63.33%, indicating a poor fit to the training data. When applied to the test set, the model achieved an accuracy of 58.33%, further confirming poor performance. The confusion matrix suggests that the model is heavily biased towards classifying instances as -1 and fails to correctly identify instances of class 1. This indicates that the radial kernel SVM with the chosen parameters is not appropriate for this dataset and may require further tuning or alternative approaches to improve performance.

Polynomial kernel

```r
cost_range <- c(0.1, 1, 10, 100, 1000)
degree_range <- c(1, 2, 3, 4)
tune.out <- tune(svm, y ~ ., data = train_data, kernel = "polynomial", ranges = list(cost = cost_
    degree = degree_range))
# summary(tune.out)

bestmod_polynomial <- tune.out$best.model
# summary(bestmod_polynomial)

ypred_polynomial <- predict(bestmod_polynomial, test_data, decision.values = TRUE)  # our usual c
# table(predict=ypred_polynomial, truth=test_data$y) Evaluate accuracy
accuracy <- mean(ypred_polynomial == test_data$y)
print(paste("Accuracy of SVM model for polynomial kernel:", accuracy))
```

```
[1] "Accuracy of SVM model for polynomial kernel: 0.833333333333333"
```

The tuning process identified a cost parameter of 10 and a polynomial degree of 1 as optimal for the polynomial SVM. The model's best performance during cross-validation was an error rate of 26.33%. When applied to the test set, the model achieved an accuracy of 83.33%. The confusion matrix suggests that the model is capable of distinguishing between the two classes with reasonable accuracy, with a low number of misclassifications. The performance is comparable to the linear kernel SVM, indicating that a polynomial kernel with degree 1 (effectively a linear model) is suitable for this dataset. This suggests that the data might be well-separated linearly, and higher-degree polynomial transformations did not significantly improve performance.

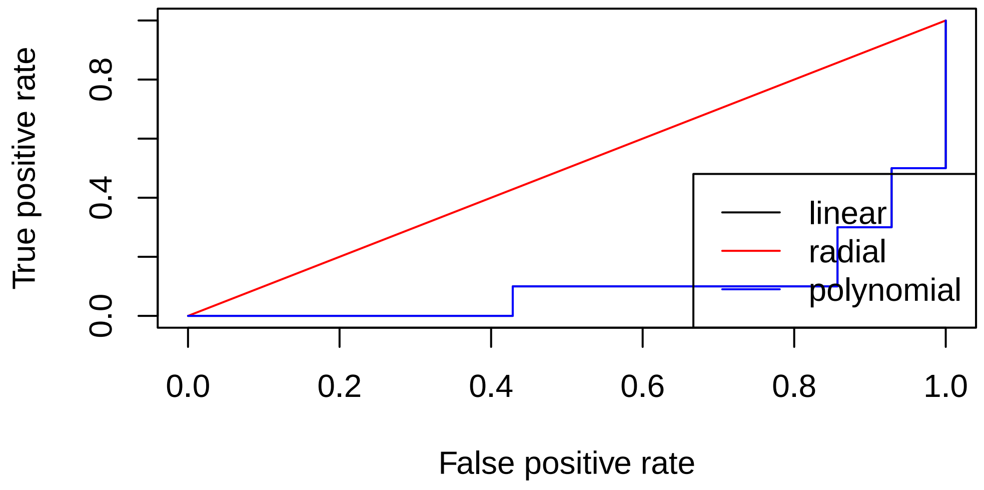
ROC curve

```r
library(ROCR)

rocplot <- function(pred, truth, ...) {
    # we use '…' to capture additional arguments to be passed to plot(), such
    # as add=T
    predob <- prediction(pred, truth)
    perf <- performance(predob, "tpr", "fpr")
    plot(perf, ...)
}

fitted.opt_linear <- attributes(ypred_linear)$decision.values
fitted.opt_radial <- attributes(ypred_radial)$decision.values
fitted.opt_polynomial <- attributes(ypred_polynomial)$decision.values
```

Generate ROC curve



Conclusion

The overlapping ROC curves confirm that the linear and polynomial (degree 1) SVM models perform similarly, likely due to the linear nature of the data. The radial kernel's performance should be further examined and potentially optimized for better results.