Homework3_Orlandi

Damiano Orlandi

Introduction

Leukemia, an heterogeneous group of hematologic disorders, presents significant diagnostic and prognostic challenges in clinical practice. A critical area of focus within leukemia research is the distinction between different subgroups, specifically those characterized by chromosomal translocations and those that are cytogenetically normal. This distinction bears direct implications on the therapeutic strategies and prognostic outcomes for patients. As with any cancer, early and accurate subgroup classification can profoundly impact treatment efficacy and patient survival rates. In this study, I harness the potential of gene expression data from 79 patients with leukemia to explore and predict subgroup classifications. This endeavor is not only aimed at enhancing our understanding of leukemia's molecular basis but also at improving diagnostic precision which, in turn, could lead to more tailored and effective treatment approaches.

My analytical approach employs Support Vector Machines (SVM), a sophisticated machine learning technique known for its efficacy in classification tasks, particularly when dealing with high-dimensional data like gene expressions. In this study, to ascertain the most effective SVM kernel for classifying leukemia subgroups based on gene expression profiles, I conduce a comparative analysis using three widely used kernels: linear, polynomial and radial basis function (RBF).

In addition to utilizing SVM, this study introduces a filtering process where only the most variable genes-those whose expression varies the most across the sample-are retained for downstream analysis. This step is predicated on the assumption that genes with higher variability are more likely to carry significant information for class distinction.

This report will detail the methodology employed to handle the gene expression data, the rationale behind the choice of SVM and the specific adjustments made to refine the analysis. Furthermore, it will discuss the results of the classification, providing insights into their implications for further research and clinical practice.

Data exploration

These, the used libraries:

- library(readr)
- library(e1071)
- library(dplyr)
- library(caret)
- library(pROC)

The gene expression dataset consists of 79 rows and 2002 columns where each row corresponds to a patient, while the columns represent gene expression levels of 2000 genes, plus an identifier for each sample (sampleID) and a label column (y) indicating the subgroup of leukemia (taking 1 or -1 values).

Overall, there are:

- SampleId: an integer identifier for each patient sample that for standard use takes character values.

- Gene expression features: 2000 continuous variables (floating-point numbers) representing the expression levels of different genes, denoted as X'gene_id'_'suffix'

- Subgroup lables: a categorical integer variable indicating the leukemia subgroup of each patient, where '1' marks patients with chromosomal translocations and '-1' for cytogenetically normal patients.

By running `sum(is.na(dataset))` I can easily notice the lack of NAs, which simplifies the preprocessing stage by removing the need for imputation methods and indicates the optimal condition for the research analysis.

Due to the far gap between the number of features (2000) and the number of samples (79), the dataset appears to be high-dimensional. This is a crucial aspect which may lead to few statistical issues like overfitting and curse of dimensionality. In the first case, with many features and relatively few samples, SVM could too closely fit the noise in the training data, leading to poor generalization to new data. In the second one, as the dimensionality increases, the volume of the space increases exponentially, and the available data becomes sparse. This sparsity makes it difficult for models to estimate relationships between features and responses effectively. Considering these risks, a popular approach in gene expression analysis with high-dimensional dataset aiming to reduce overfitting, curse of dimensionality, and computational power, is to filter out genes that do not vary much across samples and retain only the most variable ones. In this way, the model performance should improves by focusing on features that are more likely to be informative. In order to be coherent with the comparative approach of this research, I firstly apply the methods on the entire dataset, and then on the filtered version taking into account only those features whose standard deviation is among the top 5%. Therefore I can effectively constate the potential risks on the actual dataset and define the overall best model.

Data preparation

At this stage, I prepare the dataset for the models' computations and I do build the filtered dataset by selecting the only genes whose standard deviation is among the top 5%.

Firstly, I operate on the full dataset.

I encode the response as factor variable, otherwise `svm()` will perform a support vector regression, remove the SampleId variable and create a dataframe of the dataset to easily manipulate and analyze the data, saving it as `full_df`. Then, I split it into train and test dataset (75% - 25%) by applying the `sample` function, so that I will be able to test the model on unseen data. For reproducibility I set the seed at 18.

```r
set.seed(18)

dataset$y <- as.factor(dataset$y)
dataset <- subset(dataset, select = -sampleID)
full_df <- as.data.frame(dataset)
```

```
# Split dataset into training and testing subsets
train_indices <- sample(nrow(full_df), 0.75* nrow(full_df))  # 75% for training
train_data <- full_df[train_indices, ]
test_data <- full_df[-train_indices, ]
```

Secondly, I focus on the filtered dataset.

I compute the standard deviation of the gene expression features, set a threshold using the quantile function at 95% in order to select only the top 5%, filter them and create a new dataframe named `filtered_df` . Thanks to this process I obtain 100 features. Subsequently, I also split the dataset into train and test (75 - 25) by applying the `sample` function, in order to test the model on unseen data. For reproducibility I set the seed at 18.

```
set.seed(18)

#standard deviations
gene_stds <- apply(dataset[, -which(names(dataset) == "y")], 2, sd)
threshold <- quantile(gene_stds, 0.95)

# Filter
high_var_genes <- names(gene_stds[gene_stds >= threshold])

# Filtered dataset including only the most variable genes and the target
filtered_df <- dataset[c("y", high_var_genes)]


# Split dataset into training and testing subsets
train_indices_filt <- sample(nrow(filtered_df), 0.75* nrow(filtered_df))
# 75% for training
train_data_filtered <- filtered_df[train_indices_filt, ]
test_data_filtered <- filtered_df[-train_indices_filt, ]
```

Methods

Considering the binary classification task of dividing leukemia patients into two groups, I adopt three different kernels for the SVM, initially setting the following parameters:

- Linear (cost=1)
- Polynomial (cost=1, degrees=2)
- Radial Basis Function (cost=1, gamma=0.1)

The choice of these three methods has been based on the assumption of ensuring that no potential model configuration is overlooked while the idea behind these fix parameters is given by the needing to have a preliminary idea of the classification.

Afterwards, to avoid overfitting risks and to have the best parameters, I apply the cross validation method with k=10 and then I tune cost for the linear kernel, cost and degree for the polynomial kernel, and cost and gamma for the RBF.

The structure will follow this schema:

- SVM models (firsty on full training set, secondly on filtered training set)
- Cv and tuning process

- SVM optimized models
- Evaluation using ROC-AUC (firstly on full test set, secondly on filtered test set)

Full training set

Linear

Function applied selecting “`linear`”.

```r
set.seed(18)

# Linear Kernel SVM - full dataset
svm_linear <- svm(y ~ ., data = train_data,
                  kernel = "linear", cost = 1, scale = FALSE)
```

By creating a cost range (between 0.001 and 100) I have a wide volume of parameters to tune the model with. By setting up the cross-validation function with k=10, I can avoid overfitting and provide a more accurate estimate of out-of-sample performance.

```r
tune_control <- tune.control(sampling = "cross", cross = 10)

cost_range <- c(0.001, 0.01, 0.1, 1, 5, 10, 100)
```

```r
# Tuning process on the full training data
tuned_model_full <- tune(svm, y ~ ., data = train_data, kernel = "linear",
                    ranges = list(cost=cost_range),
                    tunecontrol = tune_control)
```

The tuned-function shows a great optimization’s impact, reaching a classification of 55 support vectors with the best cost found set at 0.001.

```r
# Best model - Full training data
best_model <- tuned_model_full$best.model
best_cost <- tuned_model_full$best.parameters$cost

# Train the final Linear SVM model with the best parameters
svm_final <- svm(y ~ ., data = train_data, kernel = "linear",
                 cost = best_cost, scale = FALSE)

summary(svm_final)
```

```


Call:
svm(formula = y ~ ., data = train_data, kernel = "linear", cost = best_cost,
    scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
```

```
       cost:  0.001

Number of Support Vectors:  55

 ( 28 27 )


Number of Classes:  2

Levels:
 -1 1
```

Polynomial

The applied function is set as “`polynomial`” with the initial fixed parameter of cost=1, degree=2. The choice of degree=2 is motivated by a computational’s power issue: instead of using the default value (cost=3), I prefer to gradually increase it, so that I do not have to reuse that value in the tuning process.

As for the linear kernel, I define a range of values, in this case for cost and degree.

```r
cost_range_poly <- c(0.001, 0.01, 0.1, 1, 5, 10, 100)
degree_range <- c(3, 4, 5)
```

The tuned model shows a classification of 48 vectors, with a cost of 1 and the highest possible degree=5.

```

Call:
svm(formula = y ~ ., data = train_data, kernel = "polynomial", cost = best_poly_params$cost,
    degree = best_poly_params$degree, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  5
     coef.0:  0

Number of Support Vectors:  48

 ( 24 24 )


Number of Classes:  2

Levels:
 -1 1
```

Radial basis function

I set the function as “`radial`” fixing cost=1 and gamma=0.1.

The cost and gamma range are set as:

```r
cost_range_rb <- c(0.001, 0.01, 0.1, 1, 5, 10, 100)
gamma_range <- c(0.001, 0.01, 1, 5, 10)
```

By tuning the model, the classification includes 54 vectors, with the best parameters of cost=5 and gamma=0.001 (best_rbf_params$gamma).

```

Call:
svm(formula = y ~ ., data = train_data, kernel = "radial", cost = best_rbf_params$cost,
    gamma = best_rbf_params$gamma, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  5

Number of Support Vectors:  54

 ( 27 27 )


Number of Classes:  2

Levels:
 -1 1
```

Evaluation

ROC curve and AUC have been identified as optimal evaluation metrics, due to the quantification of the model's performance across all classification thresholds performed in providing a robust measure of its discriminative power.

The schema applied for all the methods (both on full and filtered dataset) is:

- Enabling the probability estimation by using the best parameters on the three SVM models svm_linear_prob <- svm(y ~ ., data = train_data, kernel = "linear", cost = best_cost, scale = FALSE, probability = TRUE)

- Computing prediction probability on test data, probabilities_linear <- attr(predict(svm_linear_prob, test_data, probability =TRUE), "probabilities")[,2]

- Computing the roc curve and auc, roc_linear <- roc(test_data$y,probabilities_linear) auc_linear <- auc(roc_linear

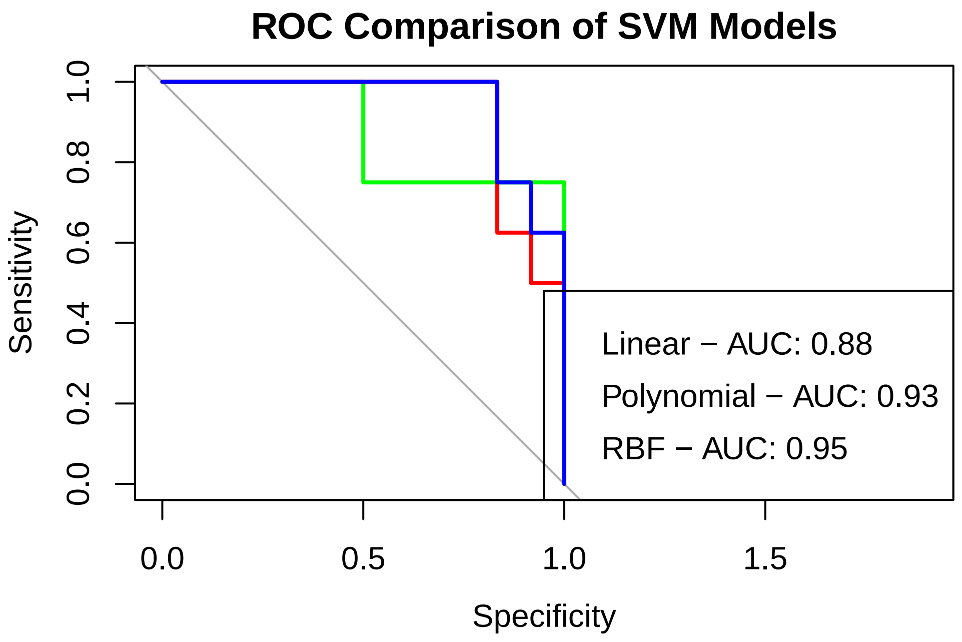
- Method evaluation plot

The ROC curve visually represents the trade-offs between the true positive rate (sensitivity), considered for the class -1, and the false positive rate (1 - specificity), considered for the class 1.

On the graph, each curve represents a kernel type with its corresponding Area Under Curve (AUC) score, which quantifies the overall ability of the model to discriminate between the classes. Higher AUC values suggest a model with better predictive accuracy.

- The **Linear** kernel shows a decent performance with an AUC of 0.88, suggesting that it has a moderate ability to distinguish between classes.
- The **Polynomial** kernel improves on this with an AUC of 0.93, indicating a higher level of discrimination capability.
- The **RBF (Radial Basis Function)** kernel outperforms the other two with an AUC of 0.95, highlighting its superior performance in handling the classification task in this specific scenario.

In conclusion, the RBF model shows best performance over the other two, with the best parameters cost=5 and gamma=0.01. This suggests that the non-linear decision boundaries modeled by the RBF kernel are more suitable for this dataset's complexity.



Filtered dataset

Linear SVM

As previosuly indicated, the following analysis is performed on the filtered dataset with the same functions and parameters as for the full dataset.

```
# Linear Kernel SVM - filtered dataset
svm_linear_filtered <- svm(y ~ ., data = train_data_filtered,
                           kernel = "linear", cost = 1, scale = FALSE)
```

The tuned model on the filtered dataset optimizes the classification reaching 42 support vectors, with best cost at 0.01.

```
Call:
```

```
svm(formula = y ~ ., data = train_data_filtered, kernel = "linear",
    cost = best_cost_ft, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  42

 ( 22 20 )


Number of Classes:  2

Levels:
 -1 1
```

Polynomial

The optimized method classifies 48 support vectors with best cost=1 and degree=5.

```
Call:
svm(formula = y ~ ., data = train_data, kernel = "polynomial", cost = best_poly_params$cost,
    degree = best_poly_params$degree, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  5
     coef.0:  0

Number of Support Vectors:  48

 ( 24 24 )


Number of Classes:  2

Levels:
 -1 1
```

Radial basis function

The optimized method classifies 43 support vectors using as best parameters: cost=5 and gamma=0.01 (best_rbf_params_ft$gamma).

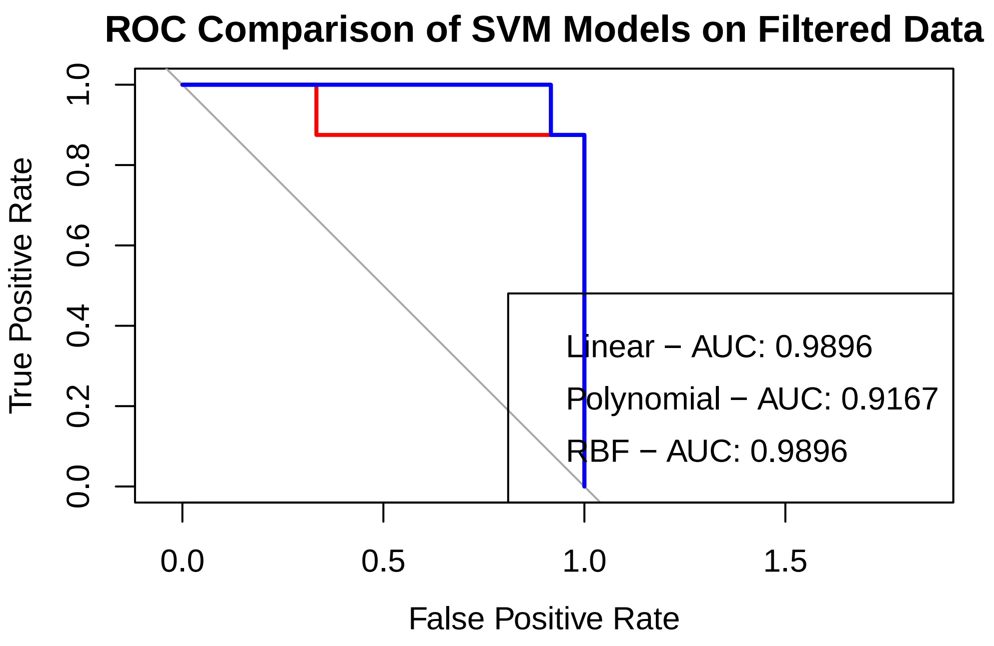
```
Call:
svm(formula = y ~ ., data = train_data_filtered, kernel = "radial",
    cost = best_rbf_params_ft$cost, gamma = best_rbf_params_ft$gamma,
    scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  5

Number of Support Vectors:  43

 ( 22 21 )


Number of Classes:  2

Levels:
 -1 1
```

Evaluation on the filtered dataset

In the ROC curve comparison of SVM models on the filtered dataset, the performance of the different kernels is :

- **Linear Kernel:** Exhibits exceptional predictive accuracy with an AUC of 0.9896. This high score suggests that the linear model is highly effective at discriminating between the classes, particularly in scenarios where the underlying data relationships may be more linear, as is often accentuated by the feature selection in the filtered dataset.
- **Polynomial Kernel:** Shows a robust performance with an AUC of 0.9167. This indicates a high discrimination capability, though not as strong as the linear kernel. The polynomial kernel, designed to model more complex relationships than the linear kernel, may not be as effective in this scenario due to the reduced complexity of the filtered dataset.
- **RBF Kernel:** Matches the linear model's performance in the filtered dataset with an AUC of 0.9896, highlighting its robustness in capturing complex patterns even when the dataset's dimensionality is reduced.

In conclusion, in this filtered dataset scenario, the linear SVM model not only outperforms the more complex kernels but also highlights the importance of feature selection in enhancing model accuracy. The simpler relationships captured by the linear model post-filtering emphasize the effectiveness of focusing on the most informative features.



Comparison

The two evaluations shows different results based on the dataset complexity. While operating on a high dimensional dataset the best performance has been achieved by the Radial Basis Function with cost=1 and gamma=0.01 and an AUC curve equal to 0.95; the linear kernel function applied on the filtered dataset reached an AUC value of 0.98 with cost=0.001, as well as the Radial basis Function with cost=5 and gamma=0.01. These outocomes could be explained due to the respective role of each function. The RBF kernel is particularly well-suited for complex, high-dimensional data because it maps data into a higher-dimensional space where a linear separator might be found. Its performance generally excels when the relationship between class labels and attributes is nonlinear. On the other hand, on the filtered dataset, where only the top 5% most variable genes are retained, the underlying structure of the data might become more linearly separable. This reduction in dimensionality simplifies the relationships among features, making it easier for a linear model to perform exceptionally well.

Conclusion

This scenario illustrates the critical importance of matching the kernel type to the dataset's nature. While RBF kernels are generally robust and capable of handling complex patterns, in a situation where feature selection has simplified the data's relationships, a simpler linear model can achieve equivalent (filtered dataser) or even superior performance (full dataset). This effectiveness of the linear kernel in the filtered dataset also highlights the benefit of focusing on the most informative features, as it enhances model accuracy and efficiency without the need for more complex modeling approaches that might be better suited to datasets with inherent non-linearities not present here.