Statistical Learning, Homework #3

Jacop Manenti, id: 247279

2024-05-18

GitHub Repository

1 Introduction and Data Exploration

In this study, we will use support vector machines (SVMs) to classify leukemia patients into two subgroups. We will evaluate different SMV models and select the one with the highest AUC (area under the curve) making usage of the ROC curve.

Data, store in `gene_expr.tvs`, has been collected from 79 patients and for each it contains 2000 gene expressions. Patients have been divided into those with chromosomal translocations (labeled "1") and cytogenetically normal (labeled "-1"). Chromosomal translocations are known to be associated with specific types of leukemia and can significantly impact prognosis and treatment response. Identifying patients with these translocations is essential for personalized medicine.

Once the dataset has been imported, we drop the "sampleID" column as it refers to each patient id and is not needed for the scope of the analysis. Since we are performing a classification task, we encode the response variable `y` (the groups) as a factor variable. Each of all the remaning columns represent a gene expression. We then identifying and addressing any missing data.

```
[1] "Number of na values:  0"
```

```
# A tibble: 4 x 2,001
  X31771_at X1030_s_at X161_at X1824_s_at X34200_at X35615_at X32597_at
      <dbl>      <dbl>   <dbl>      <dbl>     <dbl>     <dbl>     <dbl>
1      3.47       7.85    2.77       6.68      7.74      6.72      8.02
2      3.55       7.15    2.93       6.51      7.05      7.13      7.93
3      3.36       8.54    2.86       5.90      7.44      6.02      8.56
4      3.20       7.98    2.61       7.25      7.59      7.38      8.45
# i 1,994 more variables: X38486_at <dbl>, X379_at <dbl>, X41370_at <dbl>,
#   X940_g_at <dbl>, X40205_g_at <dbl>, X36559_g_at <dbl>, X34785_at <dbl>,
#   X35233_r_at <dbl>, X40236_at <dbl>, X41710_at <dbl>, X37388_at <dbl>,
#   X36514_at <dbl>, X34025_at <dbl>, X39683_at <dbl>, X1957_s_at <dbl>,
#   X38120_at <dbl>, X39618_at <dbl>, X1299_at <dbl>, X35043_at <dbl>,
#   X40503_at <dbl>, X34070_s_at <dbl>, X39221_at <dbl>, X31747_g_at <dbl>,
#   X36722_s_at <dbl>, X1366_i_at <dbl>, X33757_f_at <dbl>, ...
```

2 Support Vector Machine

In this section, we will fit four SVMs to perform the classification task. SVMs find the hyperplane that best separates data points of different classes. Depending on their assumptions, SVMs are categorized into:

- **Maximal Margin Classifier** (hard svm): assumes perfectly separable classes. They try to find a linear decision surface by maximizing the distance between the hyperplane and the nearest data point to each class (margin).
- **Support Vector Classifier** (soft svm): assumes a linear decision surface but allows some misclassification, meaning classes may not be perfectly separable.
- **Support Vector Machine**: extends the support vector classifier to non-linear decision surfaces by mapping features into an higher-dimensional space where points become linearly separable. Depending on the type of higher-dimensional space assumed, we will fit two models using respectivelly `radial kernel` and the `polynomial kernel`.

For each of this model, we will search for the optimal configuration through cross-validation and select the best one based on prediction performance employing ROC Curve.

In our scenario, the hyperparameters that need tuning are: **cost**, represents the penalty for a margin violation. A smaller value results in wider margins, allowing more misclassifications during fitting; **gamma**, in radial SVM, determines the influence range of a single training example, with low values allowing more flexibility in distances; **degree**, specifies the degree of the polynomial kernel.

Before diving into the search, note that 2 out of 4 models assume linear separability of classes (of different flexibility). To test this assumption and anticipate potential evaluation results, we plotted observations to check if it holds. If it does not, we might expect a low AUC value, thus poor classification performance. Since the dataset has more than three dimensions, we created pairwise plots for the first seven predictors, showing each pair of predictors.

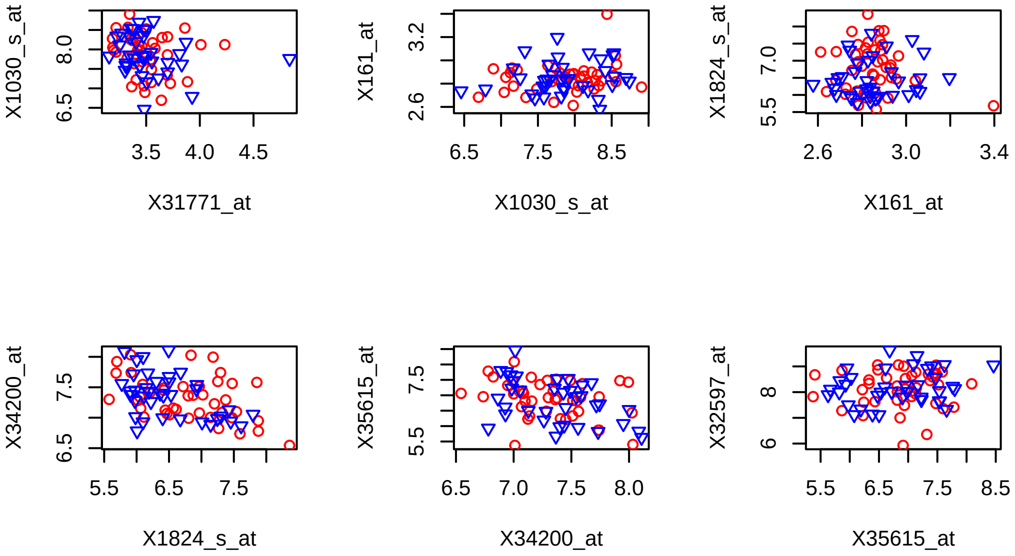


Figure 1: The figure shows pairwise class comparison between the first 7 predictors. Classes do not seem to be linearly separable.

The utility functions for obtaining probabilities to construct the ROC curve and for plotting the ROC curve are the following:

```
# get the probabilities
evaluate_model <- function(mod, x_ts) {
  preds_ts <- predict(mod, x_ts, type="prob", probability =TRUE)
  cm_ts <- table(x_ts$y, preds_ts)
  acc_ts <- mean(x_ts$y == preds_ts)
  err_ts <- mean(x_ts$y != preds_ts)
  return(preds_ts)
}
```

```
# plot ROC curve
rocplot <- function(pred, truth,x, y, name, ...){
    predob <- prediction(attr(pred, "probabilities")[,2], truth)
    perf <- performance(predob, "tpr", "fpr")
    # AUC
    auc <- performance(predob, measure = "auc")
    auc_value <- auc@y.values[[1]]
    plot(perf, ...)
    text(x, y, paste("AUC ", name, ":", round(auc_value, 3)), col = "black",
     ↪  cex = 0.9)
}
```

With everything set, we can now proceed with model selection and evaluation:

```
set.seed(1)
# split the dataset into training and test set
train_size <- 0.7
my_split <- initial_split(leukemia, strata = y, prop = train_size)
leukemia_trn <- training(my_split)
leukemia_tst <- testing(my_split)

# setting the hyperparameters' value
cost_range <- c(0.001, 0.01, 0.1, 1, 5, 10, 100)
gamma_range <- c(0.001, 0.01, 0.1, 0.5, 1, 3, 10)
poly_range <- c(1, 2, 3,4,5,6,10)

# Start time
tic()
# tain control
tc <- tune.control(cross = 5,random = TRUE)

## Inner loop
# hard svm
hard_svm <- svm(y ~ ., data=leukemia_trn, kernel="linear", cost=10e5,
 ↪  scale=FALSE, probability = TRUE)

# soft smv
tune.soft <- tune(svm, y ~ ., data=leukemia_trn, kernel="linear",
 ↪  ranges=list(cost=cost_range),tunecontrol = tc)
opt_cost  <-  tune.soft$best.parameters$cost
soft_svm_out <- svm(y ~ ., data=leukemia_trn, kernel="linear", cost=opt_cost,
 ↪  scale=FALSE, probability = TRUE)

# radial svm
tune.radial <- tune(svm, y ~ ., data=leukemia_trn,
 ↪  kernel="radial",ranges=list(cost=cost_range,
 ↪  gamma=gamma_range),tunecontrol = tc)
opt_radial_cost<-  tune.radial$best.parameters$cost
```

```
opt_gamma<-  tune.radial$best.parameters$gamma
radial_svm <- svm(y ~ ., data=leukemia_trn, kernel="radial", gamma=opt_gamma,
 ↪  cost=opt_radial_cost, probability = TRUE)


# poly svm
tune.poly <- tune(svm, y ~ ., data = leukemia_trn, kernel = "poly", ranges =
 ↪  list(cost = cost_range, degree = poly_range ), tunecontrol = tc)
opt_poly_cost<-  tune.poly$best.parameters$cost
opt_degree<-  tune.poly$best.parameters$degree
ply_svm <- svm(y ~ ., data=leukemia_trn, kernel="poly", cost=opt_poly_cost,
 ↪  degree=opt_degree, probability=TRUE)

# prediction

fitted.soft.opt <- evaluate_model(soft_svm_out,leukemia_tst)
fitted.hard.opt <- evaluate_model(hard_svm,leukemia_tst)
fitted.radial.opt <- evaluate_model(radial_svm,leukemia_tst)
fitted.poly.opt <- evaluate_model(ply_svm,leukemia_tst)

# time
elapsed_time <- toc(log = TRUE)
```

```
423.88 sec elapsed
```

```
time <- round((elapsed_time$toc - elapsed_time$tic),3)
```

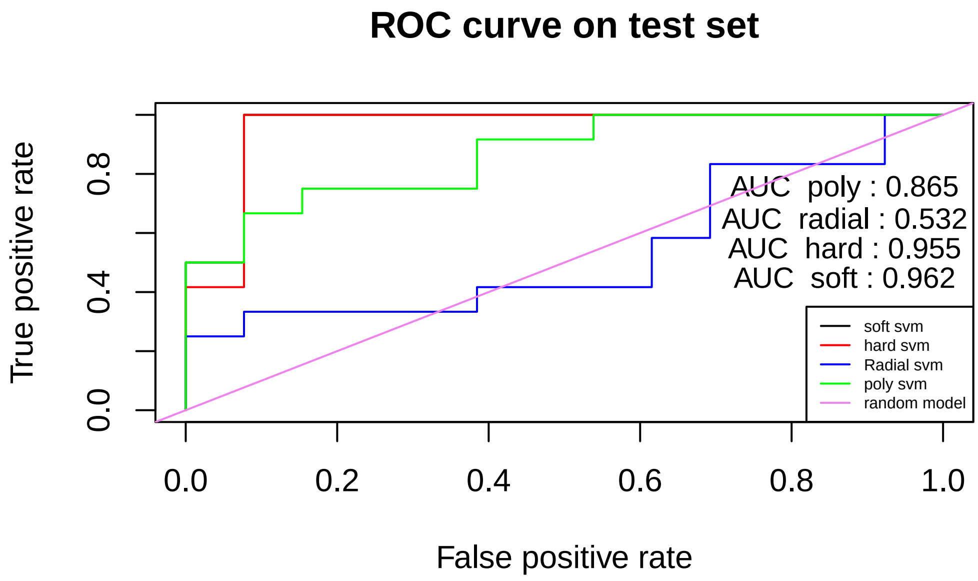


Figure 2: The figure shows ROC curve obtained by fitting and evaluating models on **leukemia** dataset. We have also reported the AUC value of each model.

In the procedure above we used 5-fold cross-validation and random hyperparameter search for efficiency, reducing runtime from ≈ 20 to 7.065 minutes compared to 10 folds and grid search.

Figure 2 shows the ROC curve, which compares model performance across all decision thresholds. It highlights whether a model produces too many false positives or negatives, crucial in medical diagnosis. The main diagonal represents performances of a random model. The best model on test data has the largest AUC, which in this case corresponds to the **Soft SVM**. It better discriminates best between patients with and without chromosomal translocations, minimizing false positives and negatives.

As the dataset contains 2000 predictors, poor performance of other models (like radial) might be related to the "curse of dimensionality"—issues: it refers to a family of problems that arise when analyzing and organizing data in high-dimensional spaces, such as increased computational cost, overfitting, and reduced generalization capability. To mitigate this effect, in the next section we will perform feature selection.

3 Feature selection

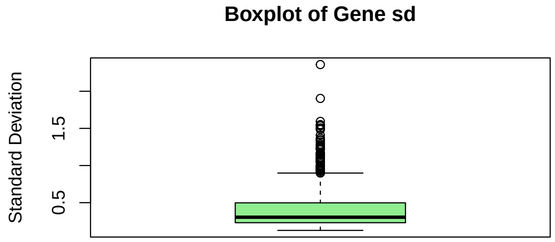
To reduce the dimensionality of the `leukemia` dataset, we will use a common technique in gene expression analysis: discarding genes with low expression variability. This variability is measured by the standard deviation. We will first compute it for each gene and then filter the dataset, retaining only those in the top 5%.

```r
# compute standard deviation
gene_sd<- apply(leukemia[, 1:2000], 2, sd )
# cut off 5%
cutoff <- quantile(gene_sd, 0.95)
# select top 5%
top_genes <- which(gene_sd >= cutoff)
top_sd <- gene_sd[top_genes]
# filter the dataset and include column y
leukemia_sd <- leukemia %>%
  select(, c(top_genes, 2001))

# get dimensions
d <- dim(leukemia_sd)
print(paste("After having filtered, the dataset has", d[1],d[2],
 ↪  "dimension"))
```

```
[1] "After having filtered, the dataset has 79 101 dimension"
```

We are now ready to repeat the analysis conducted in section Section 2 using the filtered dataset `leukemia_sd` and the same algorithm. This time, we used 10-fold cross-validation and grid search because the reduced dimensionality of the dataset significantly decreases the optimization time to ≈ 27 seconds.



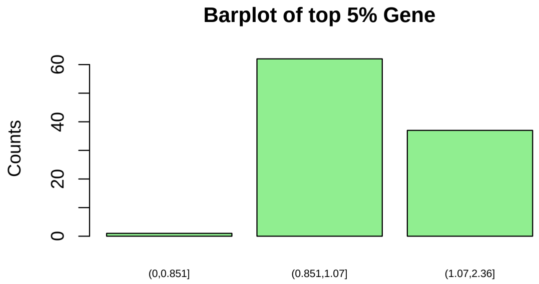


Figure 3: The figure shows a boxplot of the genes' standard deviations. The values up to the third quartile (Q3) do not exceed 0.5, indicating that most predictors have low expression level.

Figure 4: The barplot shows the top 5% of genes by standard deviation, categorized into three groups based on their standard deviation values. The majority of these high-variability genes fall into the second group.

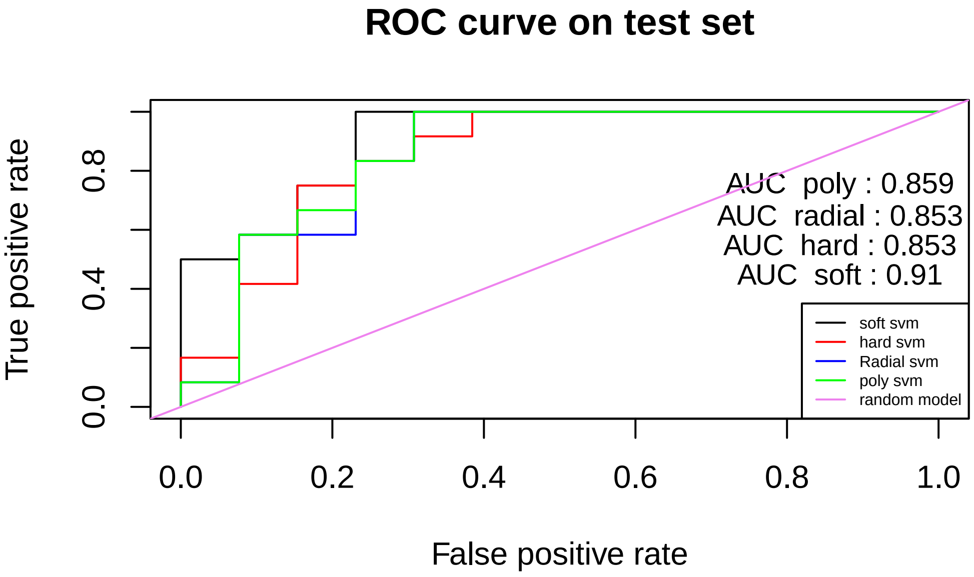


Figure 5: The figure shows the ROC curve obtained by fitting and evaluating models on **leukemia_sd**. We have also reported the AUC value of each model.

Table 1: Performance evaluation

(a) Confusion matrix for the optimal soft svm

	true -1	true 1
pred -1	10	3
pred 1	2	10

(b) Accuracy, True positive and False positive rate for the optimal soft svm

	soft svm
acc_ts	0.8000000
tpr	0.7692308
fpr	0.1666667

The new ROC curve in Figure 4 leads to similar conclusions as before, with the **Soft SVM** remaining the optimal model. However, note that the performance of the Radial SVM has improved significantly. Previously, it was only slightly better than a random model, but now it shows a substantial divergence. This confirms our hypothesis that the curse of dimensionality negatively impacted performances due to the presence of noisy predictors.

4 Conclusions

In this study, we examined four methods for predicting the group to which a leukemia patient belongs. We began by optimizing and selecting models using the leukemia dataset with all predictors. We then tested the hypothesis of the curse of dimensionality by filtering the dataset to retain only genes with high expression. Finally, we repeated the model optimization and selection process to identify the best model for the classification task.

The final model is a **Soft Margin SVM** with the optimal hyperparameter cost =0.01. We will now use this model to classify points into their respective classes. The final results are reported in Table 1. The `leukemia_sd dataset` was used, with a 70/30 split for training and testing, respectively.

```r
set.seed(1)
# split the dataset into training and test set
train_size <- 0.7
my_split <- initial_split(leukemia_sd, strata = y, prop = train_size)
leukemia_trn <- training(my_split)
leukemia_tst <- testing(my_split)

# fit soft svm
soft.svm <- svm(y ~ ., data=leukemia_trn, kernel="linear", cost=opt_cost)
# predictions
soft.pred <- predict(soft.svm, newdata=leukemia_tst)
```

```r
# print optimal soft svm summary
print(summary(soft.svm))
```

```
Call:
svm(formula = y ~ ., data = leukemia_trn, kernel = "linear", cost = opt_cost)


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

Number of Support Vectors:  38

 ( 20 18 )


Number of Classes:  2

Levels:
 -1 1
```