Relatório sobre Support Vector Machines

Lucas Avila Moreira de Paula

2024-07-12

Teoria

Vamos trabalhar com os dados de tal forma que $\vec{x}_i$ são n vetores de dimensão p, assim, temos p covariáveis e n observações. Temos também que $y_i = -1$ caso pertença ao grupo A, e $y_i = 1$ se pertencer ao grupo B.

$$\text{Dados} = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Intuição do algoritmo

Vamos considerar um caso mais simples onde p = 2 e temos exatamente dois grupos. Conforme a imagem abaixo mostra, queremos traçar uma reta (por serem apenas 2 dimensões, se p = 3, teriamos um plano, se p = 4, teriamos um hiperplano de 3 dimensões, e assim por diante) que separe os grupos. A reta ideal é aquela cuja menor distância para qualquer um dos grupos seja maximizada. Essa distância da reta ao ponto mais próximo de cada grupo é chamada de margem.

Em que $\vec{w} \cdot \vec{x} - b = w_1 \cdot x_1 + w_2 \cdot x_2 \cdots w_p \cdot x_p - b = 0$, onde w são os coeficientes desse hiperplano, e x_i os valores da x-ésima covariável observada.

Agora, vamos escrever matematicamente qual o tamanho de uma margem qualquer. Para tanto, vamos supor que $\vec{x}$ está na reta de decisão, então queremos saber quantas unidades preciso andar (ou seja, qual o valor de k) na direção w (que nesse caso está dividido pela norma de w para ser um vetor unitário), de tal forma que eu chegue na linha superior da figura 1 (por exemplo), isto é

$$\vec{w} \cdot (\vec{x} + k\frac{w}{||w||}) - b = 1$$

Desenvolvendo a equação anterior, temos que

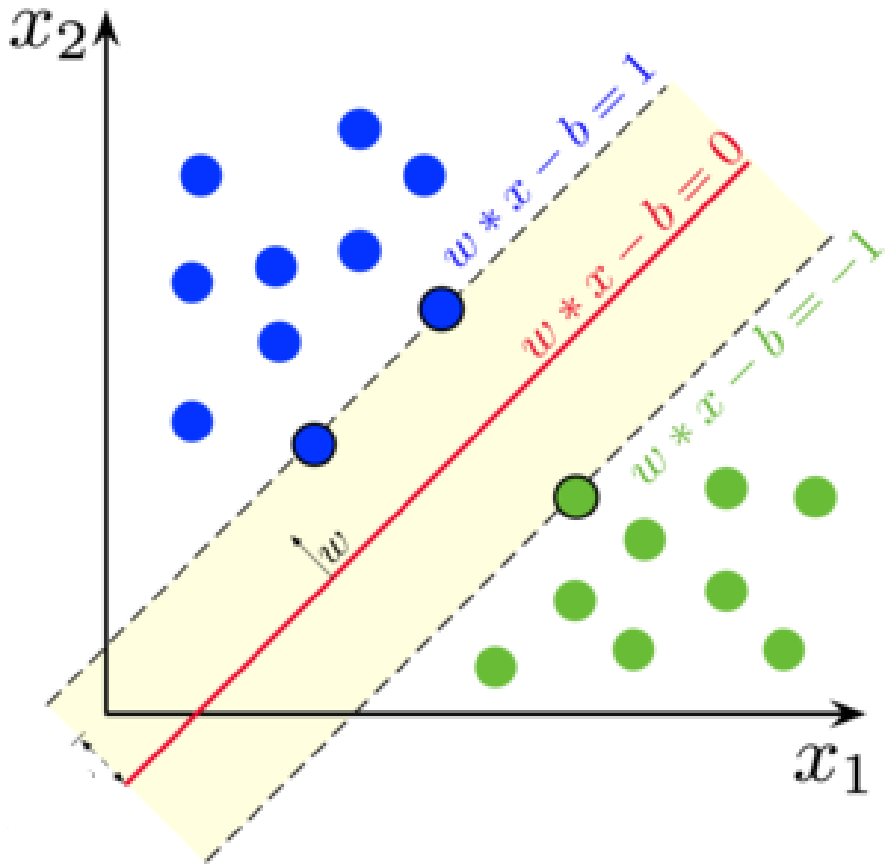


Figure 1: Exemplo intuitivo SVM

$$\vec{w} \cdot \vec{x} + \vec{w} \cdot k\frac{\vec{w}}{||\vec{w}||} - b = 1$$

$$\vec{w} \cdot k\frac{\vec{w}}{||\vec{w}||} = 1 \rightarrow k = \frac{1}{||\vec{w}||}, \text{ então a margem vale} \frac{2}{||\vec{w}||}$$

E como queremos maximizar essa margem, então queremos minimizar $||\vec{w}||$

Contudo, não queremos uma reta qualquer que maximize a margem, devemos ter uma restrição, e a restrição será a de que os grupos devem ser separados corretamente. Repare que, conhecido os valores de $\vec{w}$ e b, uma observação $\vec{x}$ qualquer terá como resultado da função $\vec{w} \cdot \vec{x} - b$ um valor menor ou igual a -1 se pertencer ao grupo A (codificado como -1), e valor maior ou igual a 1 se pertencer ao grupo B (codificado como 1). Matematicamente, dizemos queremos que:

$$\text{Se } y_i = 1 \Rightarrow \vec{w} \cdot \vec{x}_i - b \geq 1$$
$$\text{Se } y_i = -1 \Rightarrow \vec{w} \cdot \vec{x}_i - b \leq -1$$

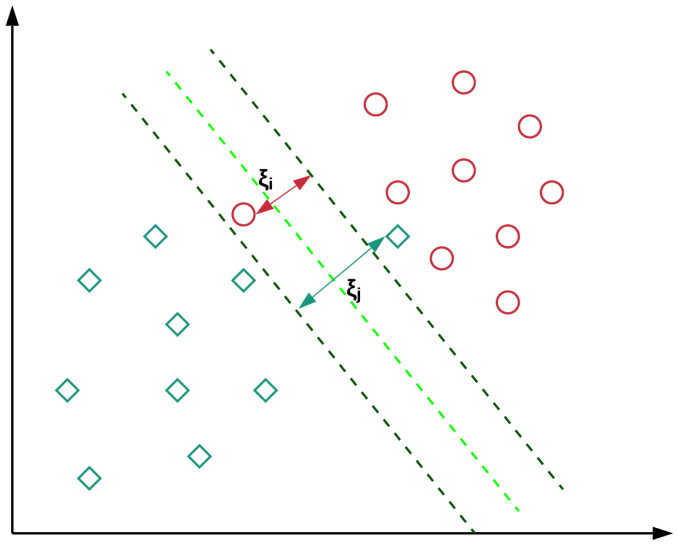
Analogamente

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$$

Ou seja, queremos minimizar $||\vec{w}||$ ao mesmo tempo que para todos os pontos $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$

Soft Margin

No caso da figura 1, os grupos podem ser completaente separados por uma reta, mas nem sempre isso será verdade. Assim, devemos aplicar penalidades para os valores que estejam sendo designados ao grupo errado. Essa penalidade também deve ser proporcional ao qual longe da margem esse ponto errôneamente classificado está. Matematicamente, definimos a penalidade hinge-loss então como



Logo, queremos na verdade minimizar

$$\min_{\vec{w},b} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)) + \lambda \vec{w}$$

Em que λ é um hiperparâmetro que pondera esse trade-off entre tamanho da margem e tolerância com erros. Se lambda for muito grande, toleramos mais erros, e focamos apenas em diminuir o tamanho da margem. Por outro lado, se lambda for muito pequeno o tamanho da margem pouco importará, dando maior importância para os erros cometidos.

Kernel Trick

Nem sempre será possível separar os grupos com uma reta. Contudo, se transformarmos os dados, talvez seja possível.

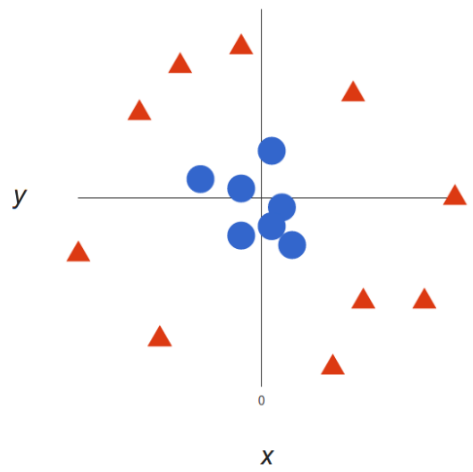


Figure 2: Dados originais

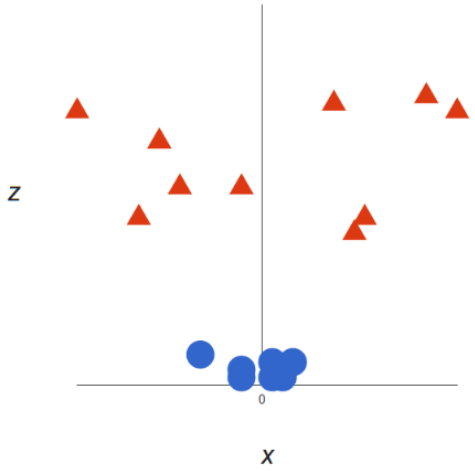


Figure 3: Dados transformados

Repare na figura 2, é evidente que o grupo azul e vermelho não podem ser separados por uma reta. Por outro lado, se criarmos uma nova variável $z = x^2 + y^2$ temos um novo eixo de altura, que indica, nesse caso, a distância de cada ponto da figura 2 à origem. Agora, na figura 3, que, apesar de mostrarmos apenas duas dimensão é tridimensional, é fácil ver que podemos traçar um plano que separa perfeitamente os grupos.

Nesse caso, se quisermos aplicar o modelo Support Vector Machine no R, fariamos da seguinte maneira

```
library(e1071)
svm(y ~ x, data = dados, kernel = "radial")
```

Nesse pacote temos outras opções de kernel para utilizar, são elas:

- Linear: que mantém a ideia original de traçar uma reta que separe os dados originais. Argumento usado seria kernel = 'linear'
- Polinomial: que ao invés de uma reta traçaria uma curva. O grau do polinômio pode ser escolhido através do argumento 'degree', ao mesmo tempo que kernel = 'polynomial'.
- Radial: argumento já exemplificado, vale notar que esse é um kernel não paramétrico, ao contrário do kernel linear por exemplo.
- sigmoid: que cria uma faixa 'em formato de S' para separar os grupos. argumento kernel = 'sigmoid'.

Assim, os grupos podem ser em formato de círculos concêntricos, podem estar separados por retas, por polinômios de qualquer grau ou podem ter formato sigmoide.

Veja a simulação a seguir para demonstrar a diferença dos kernels

```r
 set.seed(123)
 x = matrix(rnorm(40), 20, 2)
 set.seed(42)
 y = rep(c(-1, 1), c(10, 10))
 x[y == 1,] = x[y == 1,] + 1
 dat = data.frame(x, y = as.factor(y))

svmlinear = svm(y ~ ., data = dat,
             kernel = "linear",
             scale = FALSE)
svmpolinomial = svm(y ~ ., data = dat,
             kernel = "polynomial", degree = 3,
             scale = FALSE)
svmradial = svm(y ~ ., data = dat,
             kernel = "radial",
             scale = FALSE)
svmsigmoid = svm(y ~ ., data = dat,
             kernel = "sigmoid",
             scale = FALSE)

plot(svmlinear, dat, formula = X1 ~ X2)
plot(svmpolinomial, dat, formula = X1 ~ X2)
plot(svmradial, dat, formula = X1 ~ X2)
plot(svmsigmoid, dat, formula = X1 ~ X2)
```

Complexidade do algortimo

Para o SVM, teremos duas complexidades de algoritmo diferente. No caso do SVM linear, ou seja, sem usar o kernel trick, a complexidade é de $O(n * p)$, enquanto o não linear tem complexidade de $O(n^2 * d)$ ou $O(n^3)$, sendo o primeiro caso se o número de grupos for pequeno e o segundo caso seja grande.

Essa discussão pode ser vista melhor em https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf e https://leon.bottou.org/publications/pdf/lin-2006.pdf. O segundo link foi o que mais utilizei, o primeiro contém discussões muito técnicas e difíceis de entender.

Aplicação

Dados wine do pacote rattle

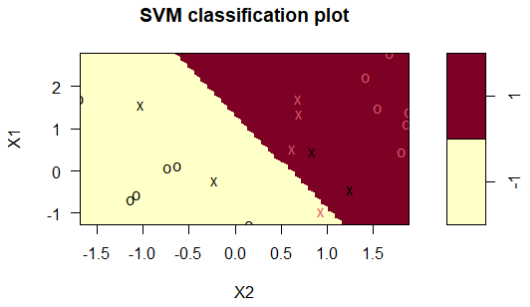


Figure 4: SVM Linear

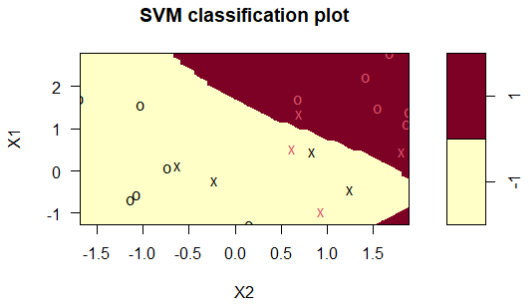


Figure 5: SVM Polinomial de Grau 3

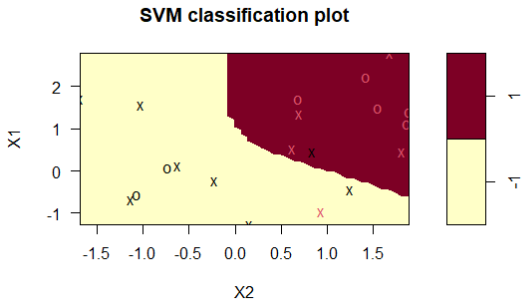


Figure 6: SVM com Kernel Radial

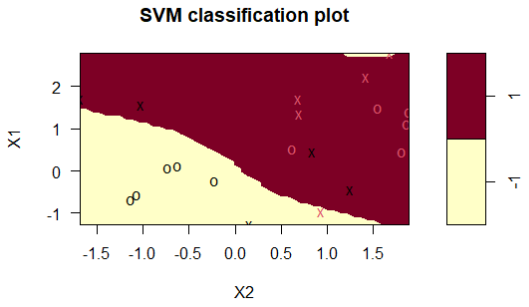
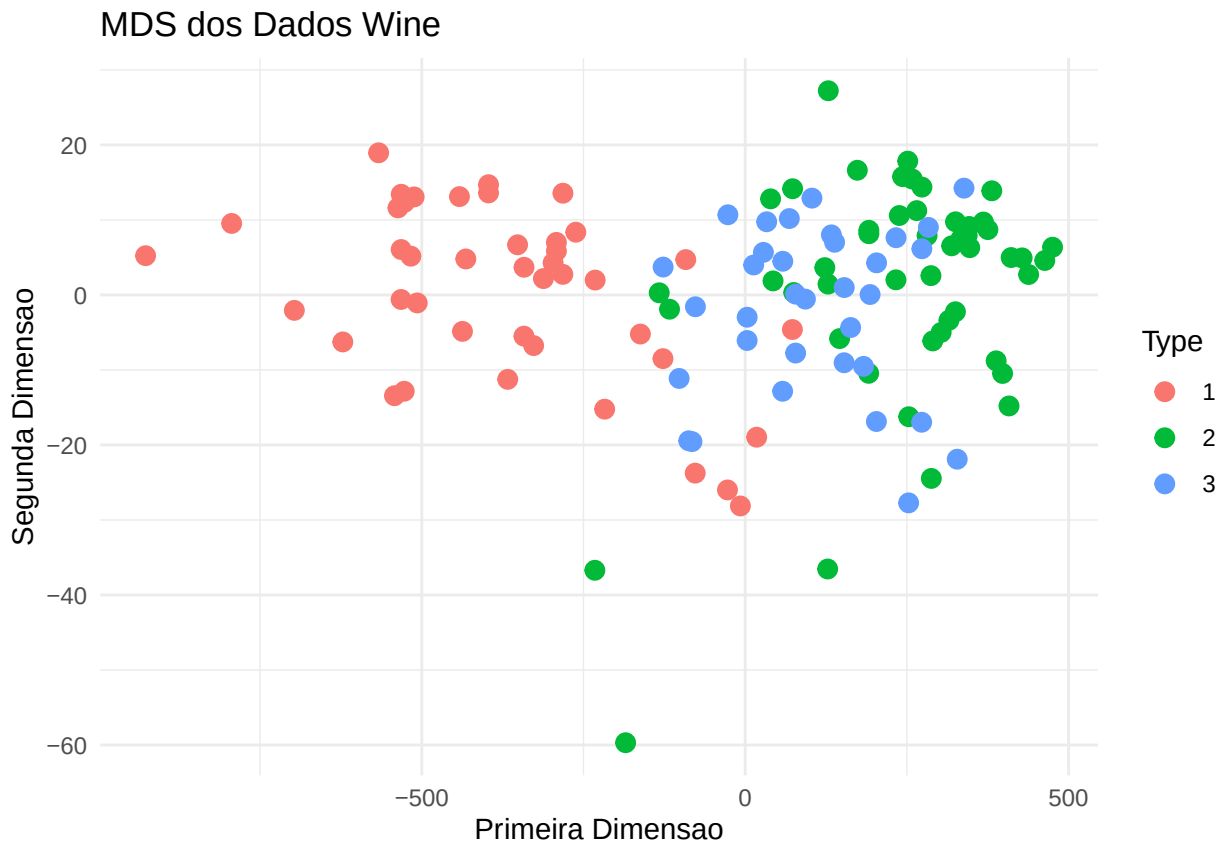


Figure 7: SVM com Kernel Sigmoide

```
library(rattle)
library(xtable)
library(ggplot2)
library(MASS)
library(reshape2)
library(gridExtra)
library(MASS)
library(reshape2)
library(gridExtra)
library(caret)
library(e1071)
```

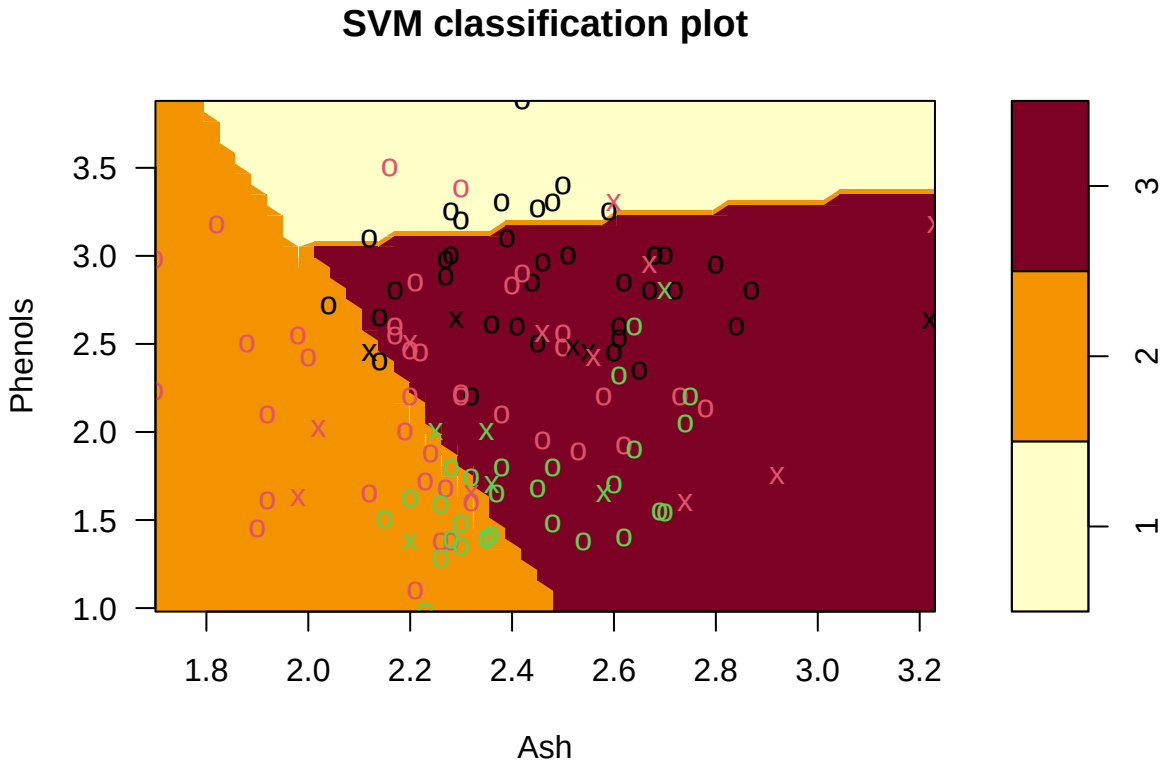
```
wine$Type <- as.factor(wine$Type)


set.seed(123)
# 70% treino, 30% teste
trainIndex <- createDataPartition(wine$Type, p = .7, list = FALSE, times = 1)
wineTrain <- wine[trainIndex, ]
```

```r
wineTest <- wine[-trainIndex, ]

distances <- dist(wineTrain[, -1], method = "euclidean")

# Aplicar MDS para visualizar os grupos
mds <- cmdscale(distances, k = 2)
mds_data <- as.data.frame(mds)
mds_data$Type <- wineTrain$Type

mds_plot <- ggplot(mds_data, aes(x = V1, y = V2, color = Type)) +
  geom_point(size = 3) +
  labs(title = "MDS dos Dados Wine",
       x = "Primeira Dimensao",
       y = "Segunda Dimensao") +
  theme_minimal()

print(mds_plot)
```



Se quisermos visualizar como o SVM separa os grupos com base nas variáveis Phenols e Ash, podemos fazer que

```
modelo <- svm(Type ~., data = wineTrain,
              kernel = "linear",
             cost = 1)

plot(modelo, wineTrain, formula = Phenols ~ Ash)
```

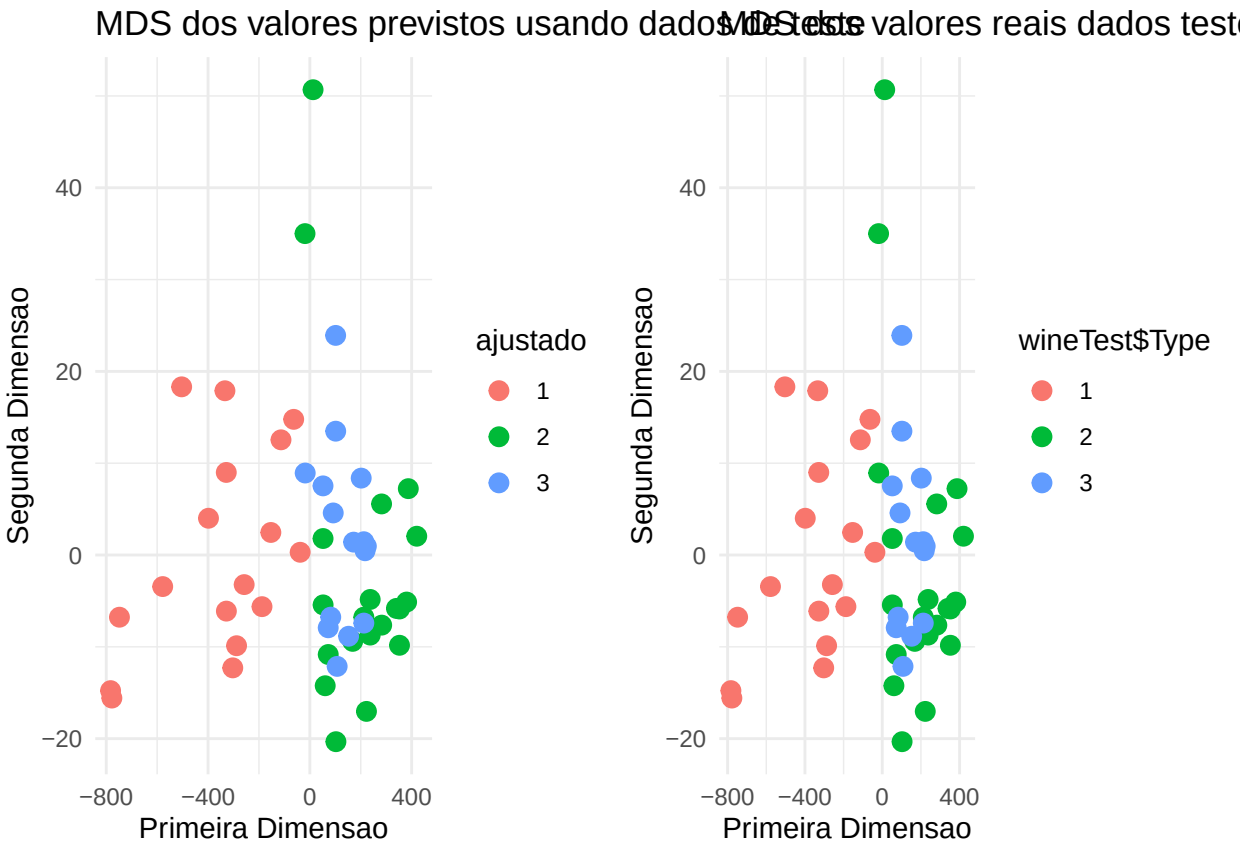


O formato de X indica que essas variáveis são os vetores de suporte, enquanto que a cor deles indica a qual grupo pertence. Analogamente, a cor do plano de fundo indica a qual grupo uma variável que está na região será designado.

Para visualizar como ficaram os grupos formados pelo SVM e como eles são na realidade, podemos aplicar um MDS, assim, temos que:

```
distances <- dist(wineTest[, -1], method = "euclidean")

mds <- cmdscale(distances, k = 2)

mds_data <- as.data.frame(mds)
mds_data$ajustado <- predict(modelo, wineTest[,-1])

mds_plot <- ggplot(mds_data, aes(x = V1, y = V2, color = ajustado)) +
  geom_point(size = 3) +
```

```
  labs(title = "MDS dos valores previstos usando dados de teste",
       x = "Primeira Dimensao",
       y = "Segunda Dimensao") +
  theme_minimal()
mds_plot2 <- ggplot(mds_data, aes(x = V1, y = V2, color = wineTest$Type)) +
  geom_point(size = 3) +
  labs(title = "MDS dos valores reais dados teste",
       x = "Primeira Dimensao",
       y = "Segunda Dimensao") +
  theme_minimal()

grid.arrange(mds_plot, mds_plot2, ncol = 2)
```



A qualidade do ajuste pode ser verificada nessa matriz de confusão, onde podemos notar que o ajuste foi muito bem feito

```
cm <- confusionMatrix(mds_data$ajustado, wineTest$Type)
cm$table
```

```
##            Reference
## Prediction  1  2  3
```

```
##           1 17  0  0
##           2  0 20  0
##           3  0  1 14
```

E se fizermos LDA, chegamos ao seguinte resultado

```
linear <- lda(Type~., wineTrain)
p <- predict(linear, wineTest)
p2 <- predict(linear, wineTest)$class
tab1 <- table(Predicted = p2, Actual = wineTest$Type)

tab1
```

```
##           Actual
## Predicted  1  2  3
##         1 17  0  0
##         2  0 21  1
##         3  0  0 13
```

Um resultado muito parecido. Portanto, esse caso de grupos linearmente separáveis, não temos diferenças significativas entre os modelos

Com grandes bases de dados

- Vamos comparar o desempenho do LDA com o do SVM com uma base de dados de maior dimensão
- Os dados em questão são os referentes a detecção de fraude no cartão de crédito, dados disponíveis em https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
- São 284807 observações e 31 variáveis
- 284315 observações do grupo 1 e 492 do grupo 2

Podemos ver como os grupos estão separados via componentes principais no gráfico abaixo

E a comporação do SVM e LDA gera a seguinte tabela

```
library(kableExtra)
resultado <- data.frame(LDA = c(15.87, 0.9997,0.7551, 0.7813), SVM = c(610.97, 0.9999, 0
kable(resultado, caption = "Resultados de LDA e SVM")
```

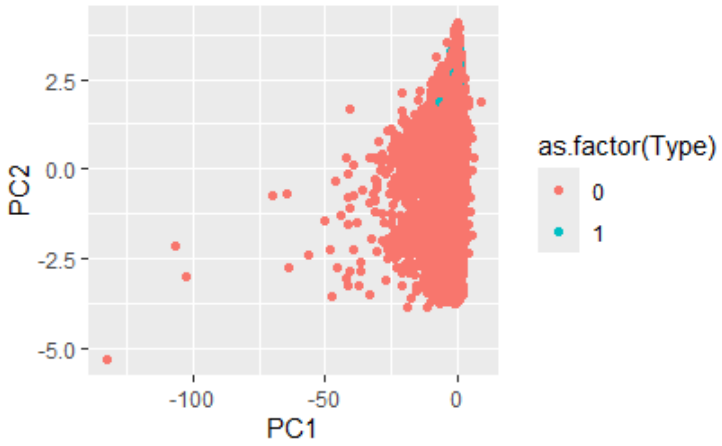


Table 1: Resultados de LDA e SVM

	LDA	SVM
Tempo (s)	15.8700	610.9700
Sensitividade	0.9997	0.9999
Especificidade	0.7551	0.6327
Kappa	0.7813	0.7651

Vemos que, apesar de uma sensitividade menor, a sensibilidade da LDA foi muito superior a do SVM, o que, dado o contexto, é muito mais importante, já que é preferível não conceder crédito a um bom pagador do que conceder crédito a um mau pagador. Não suficiente, o LDA foi muito melhor em tempo de processamento. Essa é uma desvantagem evidente do SVM: para amostras grandes, o custo computacional é muito alto.

Conclusões

Assim, podemos concluir que o SVM é uma técnica que deve ser utilizado em amostras de tamanho pequeno, sem necessidade de especificar a distribuição das variáveis explicativas, e que pode ser utilizada quando o número de covariáveis é maior que o de observações. Além disso, é muito útil para grupos não linearmente separáveis, como por exemplo o caso da figura 2. Por outro lado, suas desvantagem dizem respeito à falta de interpretabilidade, a dependência da escolha de hiperparâmetros (como escolha do Kernel, escolha do peso que quer se dar para o trade-off de tamanho da margem e tolerância a erros), sensibilidade a outliers extremos e custo computacional alto para grandes amostras.

Trabalhos futuros

- library(penalizedSVM) para seleção de variáveis via SCAD

- SVM com pesos para melhor análise de grupos desbalanceados

Referências

- Classification And Analysis Of Clustered Non-Linear Separable Data Set Using Support Vector Machines - Pavithra C
- Introduction to Statistical Learning with Applications in R - Gareth Jame
- Applied Predictive Modeling - Max Kuhn
- Hinge-loss & relationship with Support Vector Machines - geeksforgeeks.org
- Variable Selection for Support Vector Machines in Moderately High Dimensions - Xiang Zhang
- https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf
- https://leon.bottou.org/publications/pdf/lin-2006.pdf