Additional Material for Support Vector Machine

Contents

A kernel is a function that quantifies the similarity of two observations. The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way using kernels(James et al. (2021)).

Before demonstrating the various kernels, let us first perform all the initial steps as done in the SVM tutorial.

```
library("e1071")
library("caret")
data("iris")
x <- subset(iris[,-c(1,2,5)])
y <- iris$Species
```

In the Spoken Tutorial, we implemented the SVM model using a **radial kernel**. In this additional material, we will look at 3 other kernels that can be used:

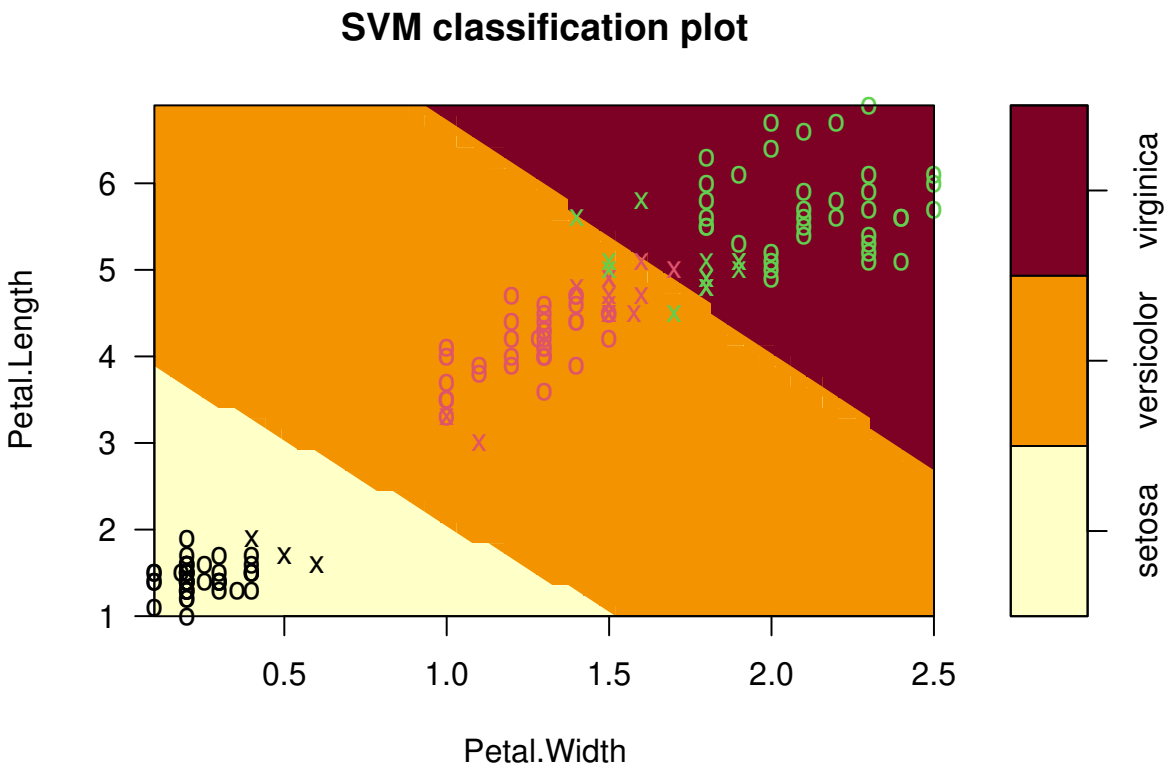
Linear Kernel: This is the simplest kernel function that can be used. The linear kernel essentially quantifies the similarity of a pair of observations using Pearson (standard) correlation(James et al. (2021)).

To use the **linear kernel**, the SVM command should be modified as follows:

```
svm_linear=svm(Species~Petal.Length+Petal.Width,data=iris,kernel="linear")
```

Now let us visualise the predictions:

```
pred_lnr <- predict(svm_linear,x)
plot(svm_linear, data=iris,formula = Petal.Length~Petal.Width)
```



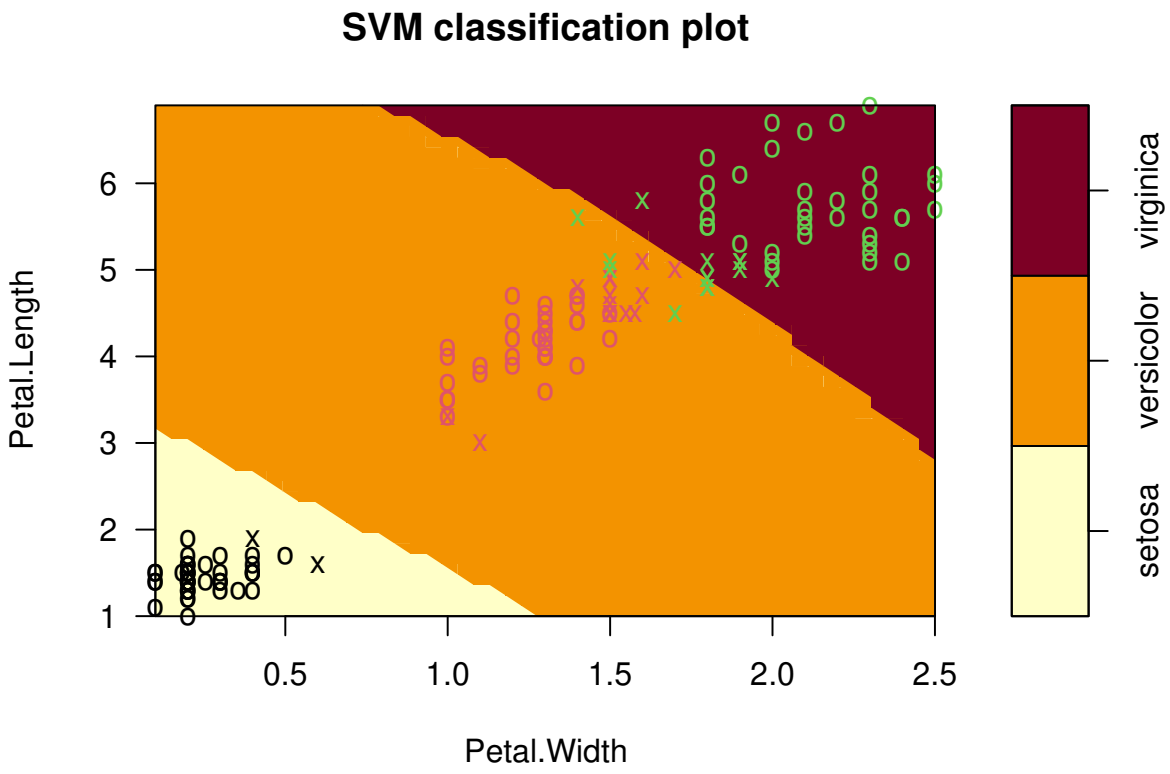
We can see that the decision boundaries are mostly linear.

Polynomial Kernel: Compared to the linear kernel,using the polynomial kernel involves fitting the support vector classifier in a higher dimensional space. The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized (Souza (2010)).

```
svm_poly=svm(Species~Petal.Length+Petal.Width,data=iris,kernel="polynomial")
```

Let us now visualise the results:

```
pred_poly <- predict(svm_poly,x)
plot(svm_poly, data=iris,formula = Petal.Length~Petal.Width)
```



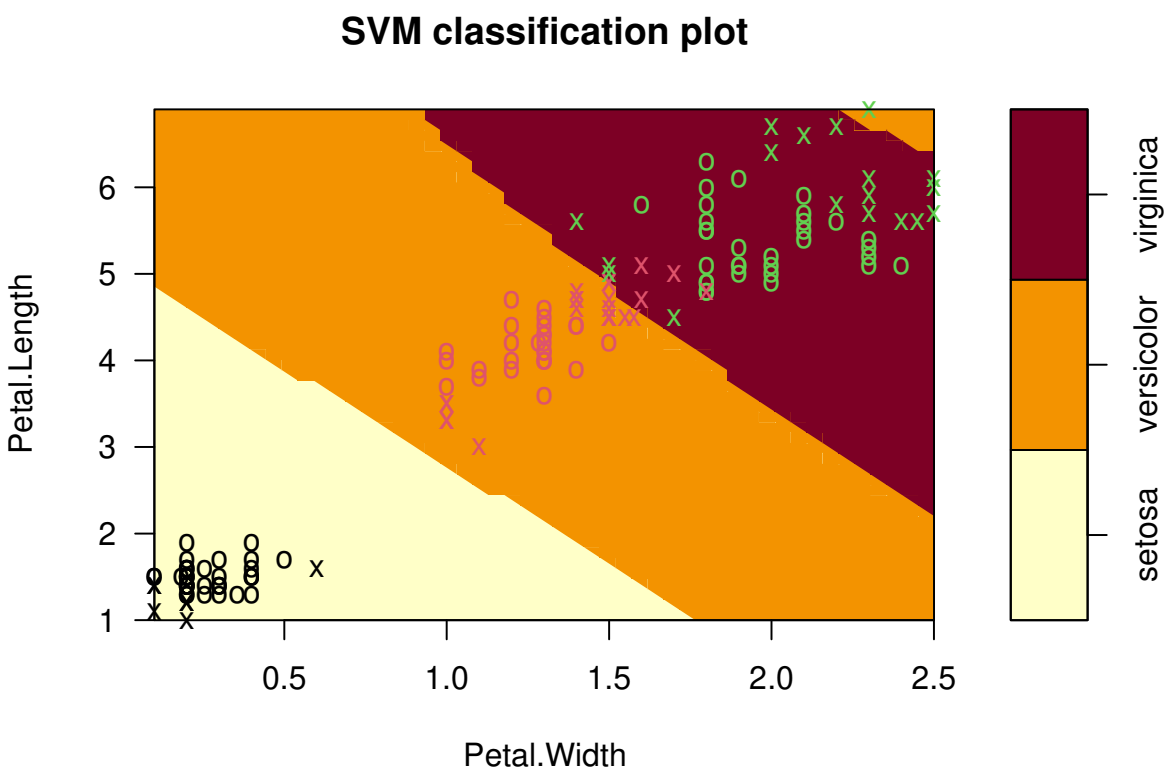
We can see that the decision boundary is much more flexible.

Sigmoid: The Sigmoid Kernel begins from the Neural Networks field, where the bipolar sigmoid limit is consistently used as an order work for counterfeit neurons(Souza (2010)).

```
svm_sgmd=svm(Species~Petal.Length+Petal.Width,data=iris,kernel="sigmoid")
```

Let us now visualise the predictions:

```
pred_sgmd <- predict(svm_sgmd,x)
plot(svm_sgmd, data=iris,formula = Petal.Length~Petal.Width)
```



Now let us compare the accuracies of these 3 kernels on the iris dataset:

```
confusionMatrix(pred_lnr,y)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         47         4
##   virginica       0          3        46
##
## Overall Statistics
##
##                Accuracy : 0.9533
##                  95% CI : (0.9062, 0.981)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.93
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9400           0.9200
```

```
## Specificity                 1.0000            0.9600           0.9700
## Pos Pred Value              1.0000            0.9216           0.9388
## Neg Pred Value              1.0000            0.9697           0.9604
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3133           0.3067
## Detection Prevalence        0.3333            0.3400           0.3267
## Balanced Accuracy           1.0000            0.9500           0.9450
```

```
confusionMatrix(pred_poly,y)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         50         6
##   virginica       0          0        44
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.94
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.8800
## Specificity                 1.0000            0.9400           1.0000
## Pos Pred Value              1.0000            0.8929           1.0000
## Neg Pred Value              1.0000            1.0000           0.9434
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.2933
## Detection Prevalence        0.3333            0.3733           0.2933
## Balanced Accuracy           1.0000            0.9700           0.9400
```

```
confusionMatrix(pred_sgmd,y)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         44         1
##   virginica       0          6        49
##
## Overall Statistics
##
##                Accuracy : 0.9533
##                  95% CI : (0.9062, 0.981)
```

```
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.93
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.8800           0.9800
## Specificity                 1.0000            0.9900           0.9400
## Pos Pred Value              1.0000            0.9778           0.8909
## Neg Pred Value              1.0000            0.9429           0.9895
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.2933           0.3267
## Detection Prevalence        0.3333            0.3000           0.3667
## Balanced Accuracy           1.0000            0.9350           0.9600
```

We can see that for the iris dataset the polynomial kernel is the best performing out of the 3, as it has the highest accuracy of 96%.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2021. "Statistical Learning." In *An Introduction to Statistical Learning*, 15–57. Springer.

Souza, César R. 2010. "Kernel Functions for Machine Learning Applications." *Creative Commons Attribution-Noncommercial-Share Alike* 3 (29): 1–1.