Classification: Naive Bayes and SVM

Torben

August 24, 2018

Classification

Today we will discuss different types of classification methods. One is based on a probabilistic argument, the other on separable *hyperplanes* (which is just a fancy word for a plane that divides the feature space).

In classification we are set in a supervised learning situation. That is, we have some training data for which we have some features X and a response Y. In classification we always has that Y is categorical - i.e. it has a certain number of levels. They may be ordered, but ordinal situations is out of our scope.

Typically, we begin with the case of assuming that Y is binary – it one has two levels, e.g. $\{0, 1\}$ or $\{-1, 1\}$. Most classification methods have been derived from this simplest case, and is then extended to deal with more than two classes.

Model based approach

For the model based approaches, the typical procedure is to compute a *posterior probability* for a class given the features. That is, $P(Y = y \mid X = x)$, where Y is the *stochastic variable* and y is *a* state, e.g $y = 0$ or $y = 1$ in the binary case. Similarly, X is the random quantity and for a specific observation X has the value(s) x.

A posterior probability is the probability of Y *given* we have seen the feature information X. The *a priori probability*, or short *prior*, reflects the believe we have about Y *before* we see the features.

Bayes classifier

A classifier that assigns the class to be the most probable is called a *Bayes classifier*:

$$\hat{k} = \arg\max_k P(Y = k \mid X = x_0),$$

for some features x_0.

Binary case

First, we observe that $P(Y = 0 \mid X) + P(Y = 1 \mid X) = 1$, which implies $P(Y = 0 \mid X) = 1 - P(Y = 1 \mid X)$. Hence, we can focus just on $P(Y = 1 \mid X)$, say.

$$P(Y = 1 \mid X) = \frac{P(Y = 1, X)}{P(X)} = \frac{P(Y = 1, X)}{P(X)} \frac{P(Y = 1)}{P(Y = 1)} = \frac{P(X \mid Y = 1)P(Y = 1)}{P(X)}$$

For the binary case, we have that a Bayes classifier will assign the class to be 1 if $P(Y = 1 \mid X) > P(Y = 0 \mid X)$, because we know they sum to 1 (there are only two outcomes). Furthermore, we can write

$$\frac{P(Y = 1 \mid X)}{P(Y = 0 \mid X)} > 1$$

And since the above factorisation of $P(Y = 1 \mid X)$ also holds for $P(Y = 0 \mid X)$ we have

$$\frac{P(Y = 1 \mid X)}{P(Y = 0 \mid X)} = \frac{P(X \mid Y = 1)}{P(X \mid Y = 0)} \frac{P(Y = 1)}{P(Y = 0)}$$

Hence, we just need a model for $P(X \mid Y = y)$ and then assign a *prior* probability to $P(Y = 1)$. Often this is (maybe confusingly) denoted $\pi = P(Y = 1)$ with $P(Y = 0) = 1 - \pi$. Note, π is in this case **not** 3.1415927.

Naive Bayes

Modelling $P(X \mid Y = y)$ may not be easy as X can high (extremely) high-dimensional. However, one model assumption that simplifies this dramatically is that of (conditional) independence:

$$P(X \mid Y = y) = P(X_1 \mid Y = y)P(X_2 \mid Y = y)\cdots P(X_p \mid Y = y) = \prod_{i=1}^{p} P(X_i \mid Y = y)$$

This turns a complicated model for $P(X \mid Y = y)$ into a product of simple models – one for each X_i.

Including this in the expression above yields for $X = x_0$,

$$\frac{P(Y = 1 \mid X = x_0)}{P(Y = 0 \mid X = x_0)} = \frac{\pi}{1 - \pi} \prod_{i=1}^{p} \frac{P(X_i = x_{0i} \mid Y = 1)}{P(X_i = x_{0i} \mid Y = 0)}$$

Hence, if the ratio above is greater than 1, we classify a new observation x_0 as $\hat{Y} = 1$. Otherwise, classify $\hat{Y} = 0$.

The model has several advantages:

- Simple to estimate parameters (no need for iterative procedures)
- Insensitive to missing data (the term just disappears)
- Works for $n \ll p$

`naiveBayes` in R

The package `e1071` contains several methodologies, including `naiveBayes` and `svm` (which we will use here).

The `naiveBayes` assumes that for numerical features, x_i follows a normal distribution, with group specific mean and variance: μ_y and σ_y^2.

For categorical cases we simply tabulate and estimate the associated probabilities from the counts.

Example: `Titanic`

A small example with the survival of Titanic passengers.

```r
Titanic_tbl <- Titanic %>%
  as_tibble() %>%
  mutate(n = as.integer(n)) %>%
  mutate_if(is.character, factor) %>%
  mutate_at(c("Age", "Survived"), funs(fct_rev))
Titanic_tbl %>% kable()
```

Class	Sex	Age	Survived	n
1st	Male	Child	No	0
2nd	Male	Child	No	0
3rd	Male	Child	No	35
Crew	Male	Child	No	0
1st	Female	Child	No	0

Class	Sex	Age	Survived	n
2nd	Female	Child	No	0
3rd	Female	Child	No	17
Crew	Female	Child	No	0
1st	Male	Adult	No	118
2nd	Male	Adult	No	154
3rd	Male	Adult	No	387
Crew	Male	Adult	No	670
1st	Female	Adult	No	4
2nd	Female	Adult	No	13
3rd	Female	Adult	No	89
Crew	Female	Adult	No	3
1st	Male	Child	Yes	5
2nd	Male	Child	Yes	11
3rd	Male	Child	Yes	13
Crew	Male	Child	Yes	0
1st	Female	Child	Yes	1
2nd	Female	Child	Yes	13
3rd	Female	Child	Yes	14
Crew	Female	Child	Yes	0
1st	Male	Adult	Yes	57
2nd	Male	Adult	Yes	14
3rd	Male	Adult	Yes	75
Crew	Male	Adult	Yes	192
1st	Female	Adult	Yes	140
2nd	Female	Adult	Yes	80
3rd	Female	Adult	Yes	76
Crew	Female	Adult	Yes	20

Fitting a `naiveBayes` (we use the tabular form of the data here)

```
library(e1071)
nb_titanic <- naiveBayes(Survived ~ ., data = Titanic)
```

The estimates of *prior* and $P(x_i \mid y)$

```
nb_titanic$apriori
```

```
## Survived
##   No  Yes
## 1490  711
```

```
nb_titanic$tables
```

```
## $Class
##         Class
## Survived        1st        2nd        3rd       Crew
##      No  0.08187919 0.11208054 0.35436242 0.45167785
##      Yes 0.28551336 0.16596343 0.25035162 0.29817159
##
## $Sex
##         Sex
## Survived       Male     Female
##      No  0.91543624 0.08456376
##      Yes 0.51617440 0.48382560
```

```
##
## $Age
##          Age
## Survived      Child      Adult
##      No  0.03489933 0.96510067
##      Yes 0.08016878 0.91983122
```

Which variables are more informative to predict survival?

```
## Class:
p_class <- nb_titanic$tables$Class
p_class["Yes",]/p_class["No",]
```

```
##       1st       2nd       3rd      Crew
## 3.4870074 1.4807516 0.7064847 0.6601422
```

```
# Sex
p_sex <- nb_titanic$tables$Sex
p_sex["Yes",]/p_sex["No",]
```

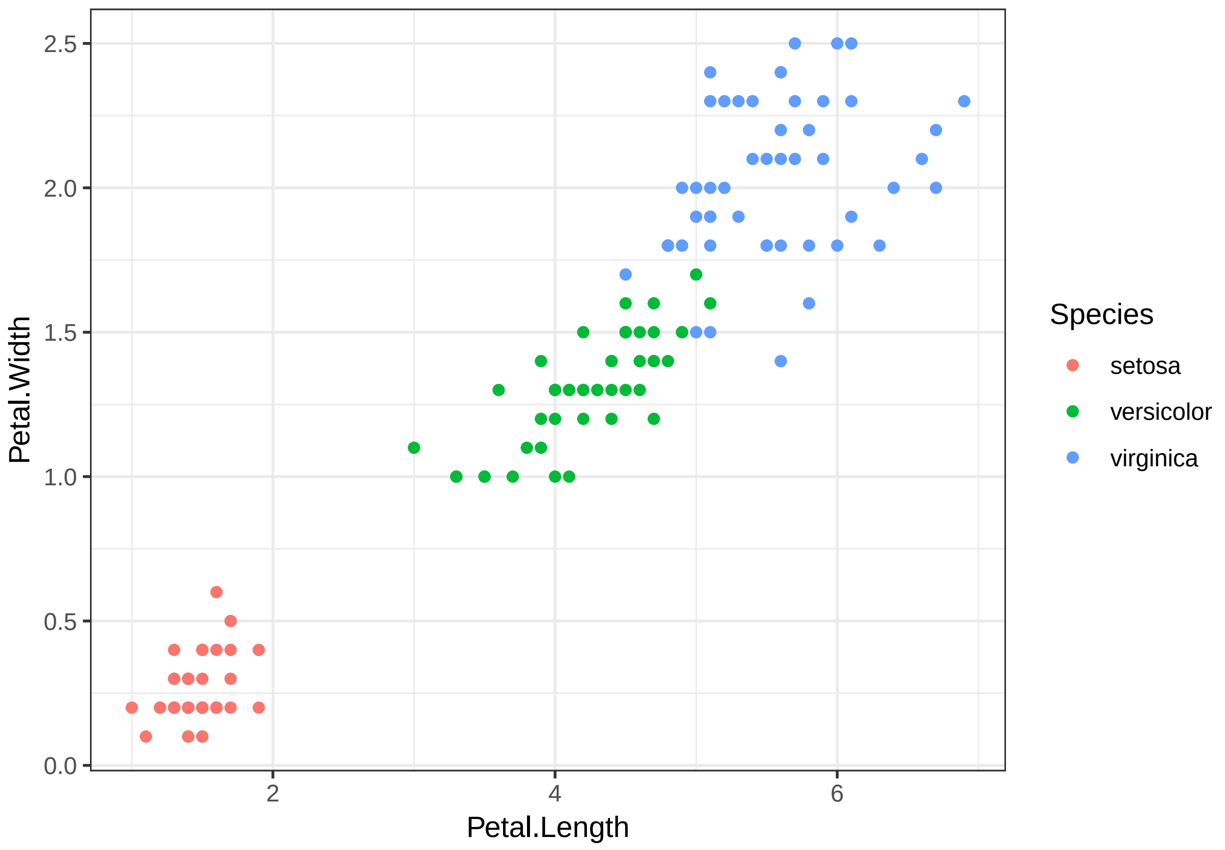
```
##      Male    Female
## 0.5638562 5.7214297
```

```
# Age
p_age <- nb_titanic$tables$Age
p_age["Yes",]/p_age["No",]
```

```
##     Child     Adult
## 2.2971438 0.9530935
```

Exercise:

```
iris %>% ggplot(aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point()
```



- Use `naiveBayes` to classify the `iris` flowers into their three classes.
- What information does the `$tables` from the fit contain?
- Try to identify the most informative variables for predicting the classes.
- Are the assumptions about normality satisfied for each of $x_i \mid y$?
- Can you visualise the decision boundaries in the `Petal`-plane (i.e `x = Petal.Length, y = Petal.Width`)? *Tip:* Create a grid (cf below), make the prediction for* each* point in the grid and plot this.

```
iris_petal_grid <- iris %>%
  expand(
    Petal.Length = seq(min(Petal.Length), max(Petal.Length), len = 100),
    Petal.Width = seq(min(Petal.Width), max(Petal.Width), len = 100)
  ) %>%
  mutate(Sepal.Length = NA, Sepal.Width = NA)
```

Support Vector Machines: SVM

See slides `day-5-SVM.pdf` for introduction to SVMs

```
iris_svm_linear <- svm(Species ~ ., data = iris, cost = 1, kerner = "linear")
iris_svm_linear
```

```
##
## Call:
## svm(formula = Species ~ ., data = iris, cost = 1, kerner = "linear")
##
```

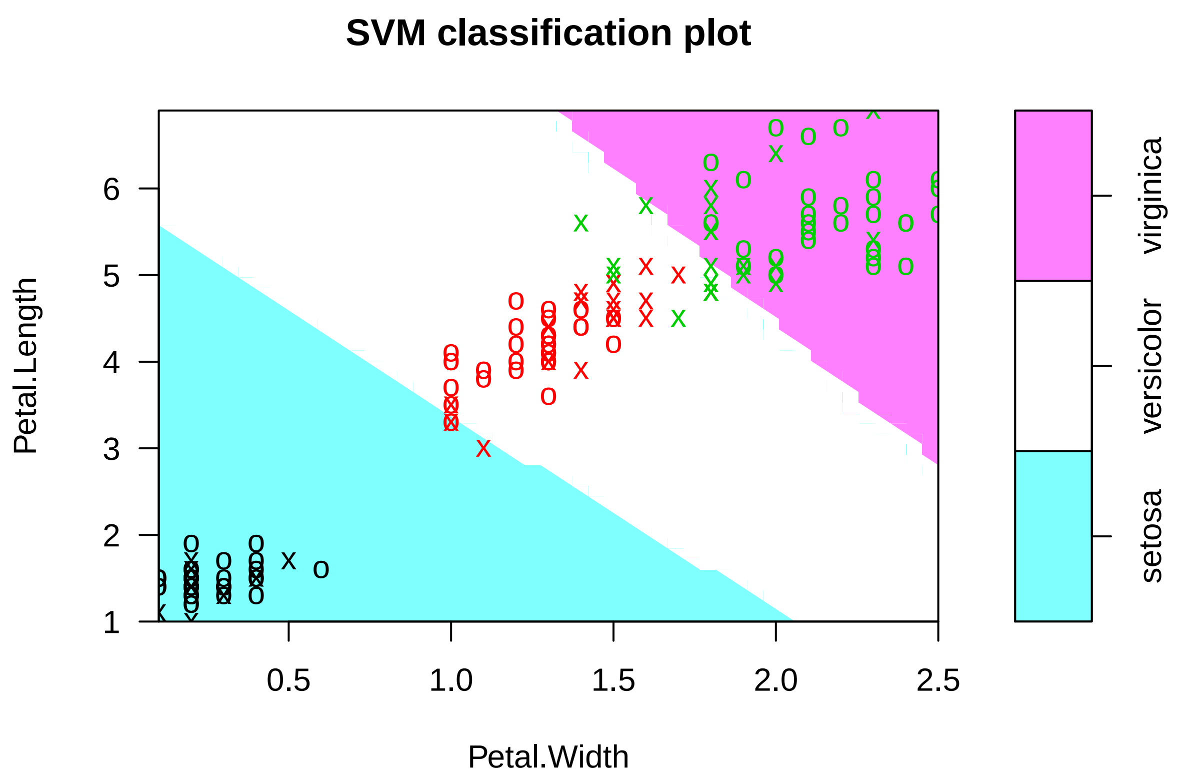
```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##
## Number of Support Vectors:  51
```

```r
summary(iris_svm_linear)
```

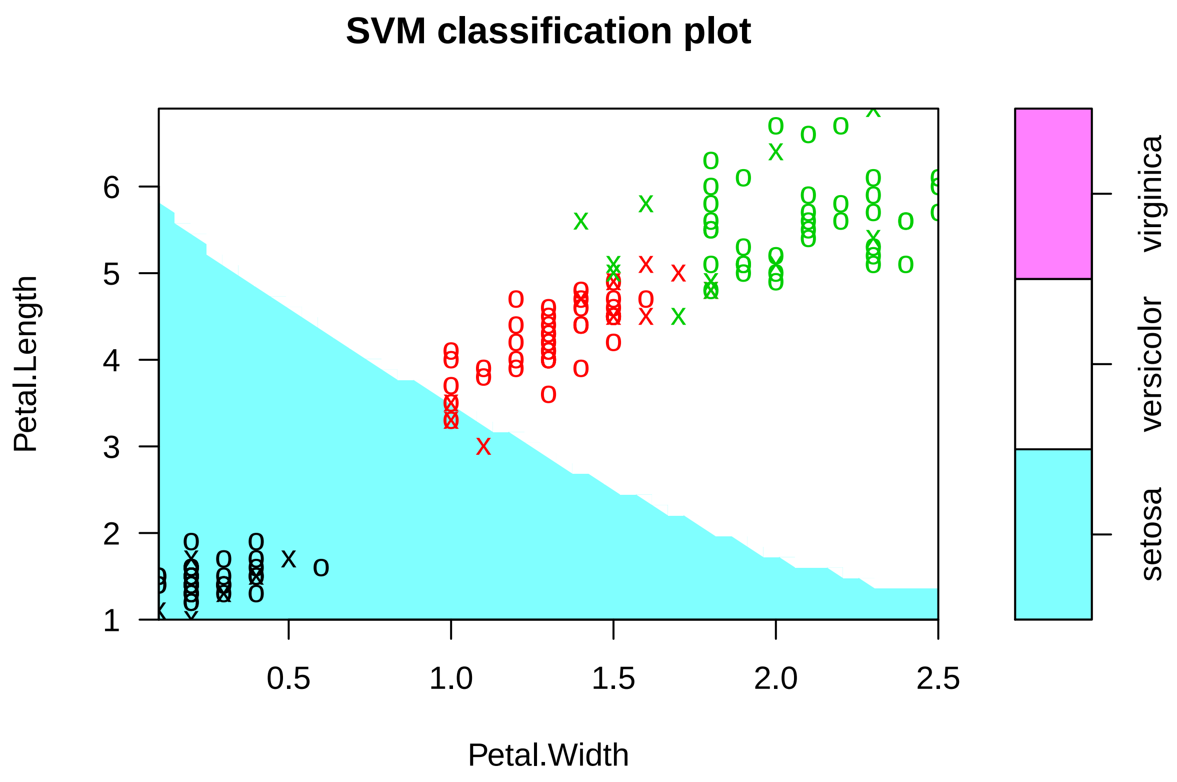
```
##
## Call:
## svm(formula = Species ~ ., data = iris, cost = 1, kerner = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##
## Number of Support Vectors:  51
##
##  ( 8 22 21 )
##
##
## Number of Classes:  3
##
## Levels:
##  setosa versicolor virginica
```

```r
plot(iris_svm_linear, formula = Petal.Length ~ Petal.Width, data = iris,
     slice = list(Sepal.Width = 3, Sepal.Length = 4))

iris_svm_radial <- svm(Species ~ ., data = iris, cost = 1, kerner = "radial")
plot(iris_svm_radial, formula = Petal.Length ~ Petal.Width, data = iris,
     slice = list(Sepal.Width = 3, Sepal.Length = 4))
```

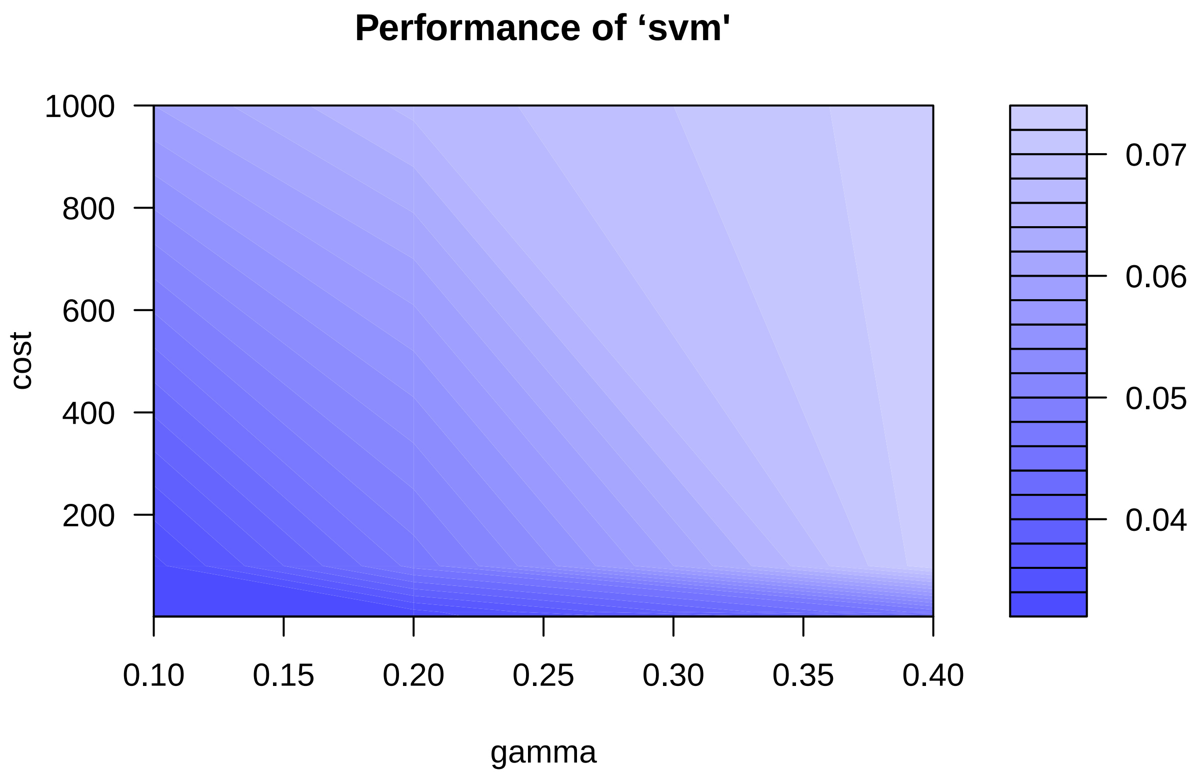


```
iris_svm_poly_3 <- svm(Species ~ ., data = iris, cost = 100, kerner = "polynomial", degree = 5)
plot(iris_svm_poly_3, formula = Petal.Length ~ Petal.Width, data = iris,
     slice = list(Sepal.Width = 3, Sepal.Length = 4))
```



Tuning

```r
ncol_data <- ncol(iris)
iris_radial_tune <- tune.svm(Species ~ ., data = iris, kerner = "radial",
                             cost = 10^(0:3), gamma = 1/(ncol_data*c(0.5,1,2)))
plot(iris_radial_tune)
```



```r
summary(iris_radial_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.2    1
##
## - best performance: 0.03333333
##
## - Detailed performance results:
##    gamma cost      error dispersion
## 1    0.4    1 0.04000000 0.04661373
## 2    0.2    1 0.03333333 0.04714045
## 3    0.1    1 0.03333333 0.04714045
## 4    0.4   10 0.04666667 0.05488484
## 5    0.2   10 0.03333333 0.04714045
## 6    0.1   10 0.03333333 0.04714045
## 7    0.4  100 0.07333333 0.07336700
## 8    0.2  100 0.04666667 0.06324555
## 9    0.1  100 0.03333333 0.05665577
## 10   0.4 1000 0.07333333 0.07336700
## 11   0.2 1000 0.06666667 0.07027284
## 12   0.1 1000 0.06000000 0.05837300
```

Topics not covered

- ROC curves
  - Used to decide on an optimal threshold value. That is, the threshold of 0.5 may not be optimal
- k-Nearest Neighbouhrs
  - A 'simple' technique where a *test sample* is classified based by a majority vote among its k closest data points in the *training data*
  - This is called on *online* or *lazy* learner as it does not fit a model to data, but uses all the training data for each new classification task.
  - Typically cross-validation is used to decide on k
- Imbalanced training data case
  - When samples of one type is much more dominant in the training data. One approach is to use weights for methods that allows/incorporates this.
- Linear (LDA) and quadratic discriminant analysis (QDA)
  - The predecessors of SVM
  - Relies on an assumption of multivariate normality of the data (given the class)
- Ordinal classification
  - Situations where the levels of Y has an ordering to them, e.g. *low*, *mid* and *high*
  - See the `ordinal` package for regression methods to deal with this type of analysis.
  - See the `rpartOrdinal` or `rpartScore` for extentions to `rpart` for classification trees with ordinal responses.
  - See `glmnetcr` for a `glmnet` like approach to ordinal response prediction