MONASH University

ETC3250 Business Analytics: Classification with Support Vector Machines

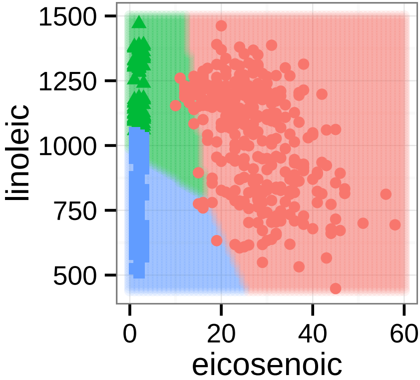
Souhaib Ben Taieb, Di Cook, Rob Hyndman

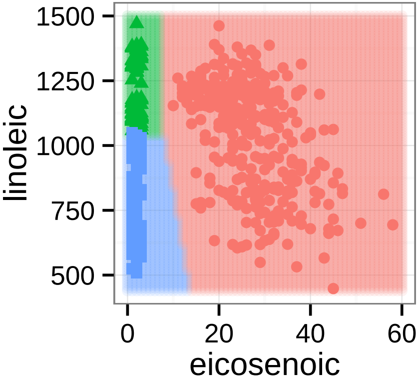
September 21, 2015

LDA to SVM

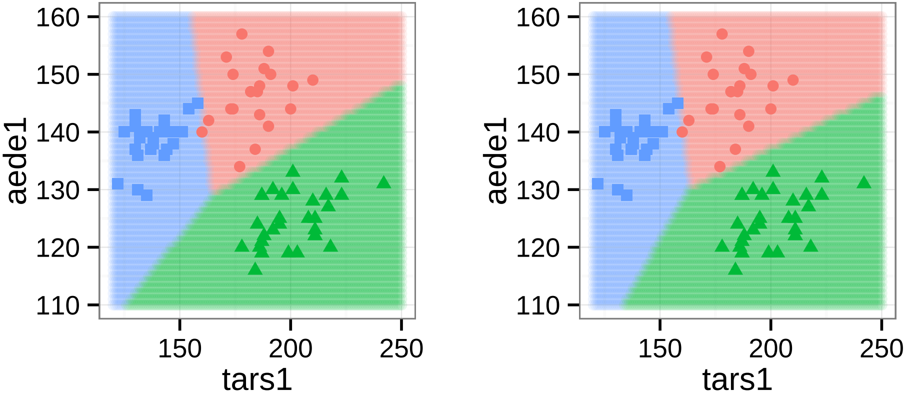
- Support vector machines build a classifier by finding **gaps** between clusters
- Primarily only work on two classes at a time, so multiclass problems need some tweaking of approach
- Let's look at how it works on the same three examples as used for LDA

Boundaries for Olive oils: LDA, SVM

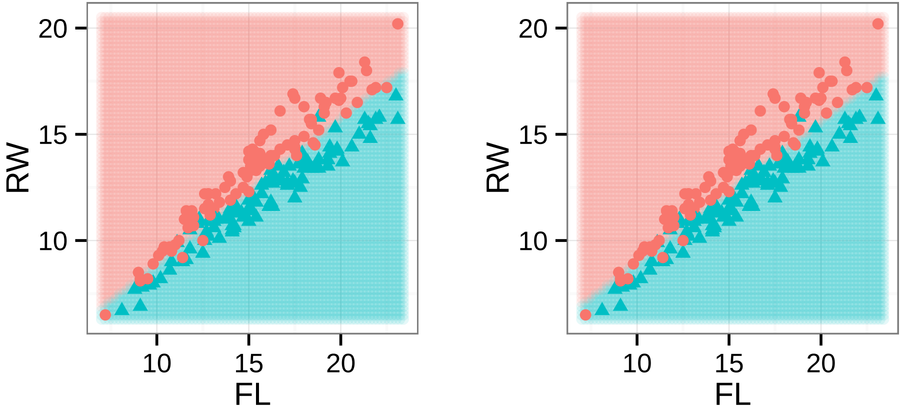




Boundaries for Beetles: LDA, SVM



Boundaries for Crabs: LDA, SVM



Comparison

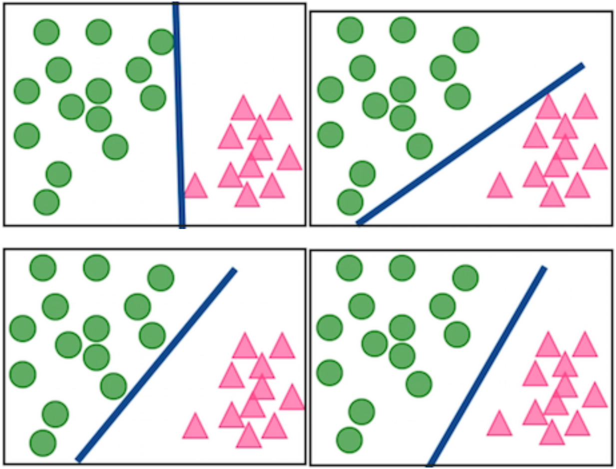
- Which boundaries look better?
- Why?

How does SVM work?

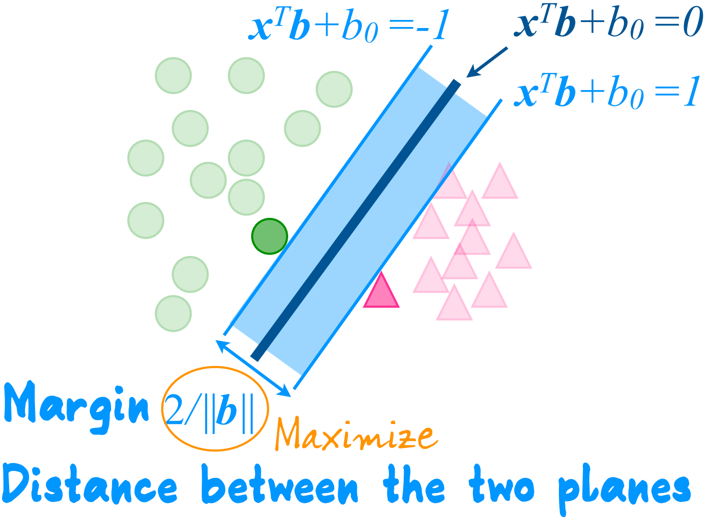
- Variables $(x_1, ..., x_p)$ need to be standardized
- Class (y) is coded as ± 1
- Separating hyperplane defined to be $\{x : x^T b + b_o = 0\}$ $(x, b$ are $p-$dimensional vectors)
- where $b = \sum_{i=1}^{s} (\alpha_i y_i) x_i$
- s is the number of support vectors
- estimated by maximizing margin $M = 2/||b||$ subject to $\sum_1^p b_i^2 = 1$, $y_i(x_i^T b + b_o) \geq 1, i = 1, ..., n$

Best separating hyperplane

- All are separating hyperplanes. Which is best?

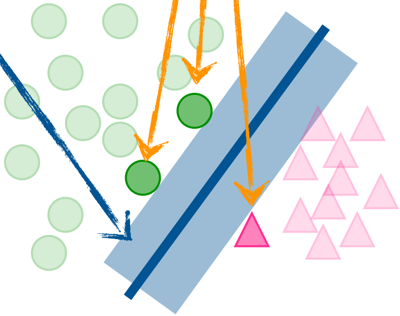


Maximum margin

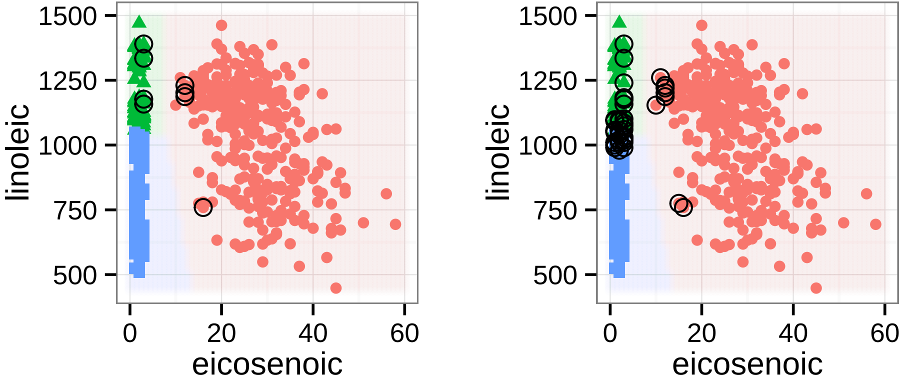


Support vectors

- Hyperplane is defined by a subset of the point, the support vectors
- Distance between these is what is maximized



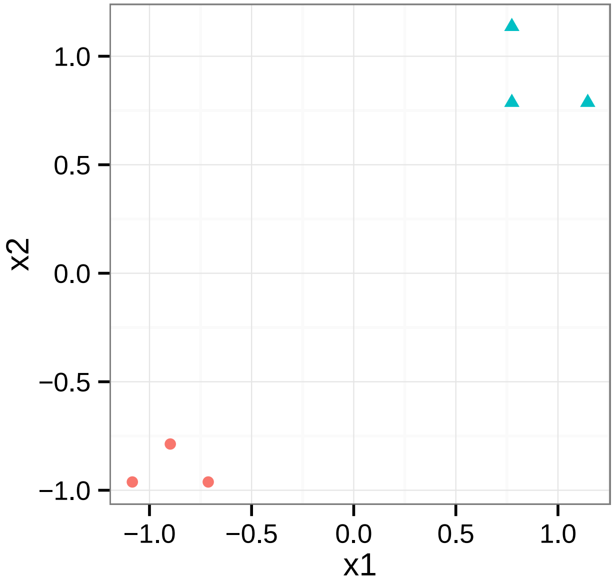
Support vectors for the olive oil classification



Simulation example

```
##            x1         x2  y
## 1  0.7741756  0.7869346  1
## 2  1.1457799  0.7869346  1
## 3  0.7741756  1.1366833  1
## 4 -0.8980437 -0.7869346 -1
## 5 -1.0838459 -0.9618089 -1
## 6 -0.7122416 -0.9618089 -1
```

Simulation example



Fit the classifier

```
df.svm <- svm(y~., data=df, kernel="linear")
df.svm$SV
df.svm$index
df.svm$coefs
```

```
##           x1         x2
## 1  0.7741756  0.7869346
## 4 -0.8980437 -0.7869346
## 6 -0.7122416 -0.9618089
```

```
## [1] 1 4 6
```

```
##            [,1]
## [1,]  0.3094284
## [2,] -0.1404977
## [3,] -0.1689307
```

Calculate

$b = \sum_{i=1}^{s}(\alpha_i y_i)x_i$

```
t(as.matrix(df.svm$coefs))%*%df.svm$SV
```

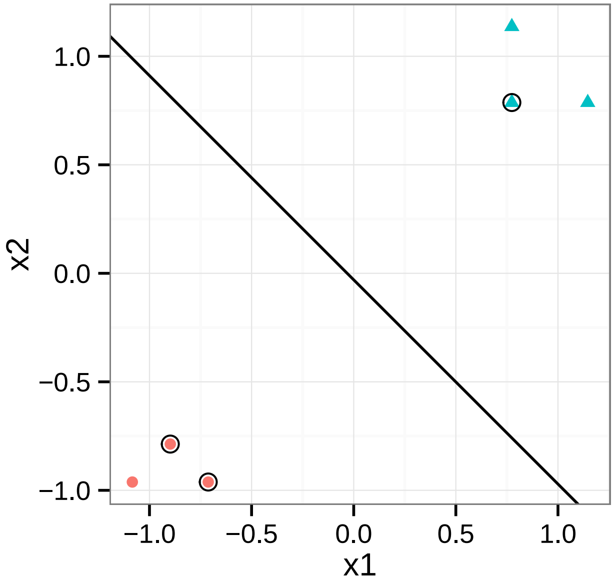
```
##              x1        x2
## [1,] 0.4860445 0.5165415
```

$\{x : 0.486x_1 + 0.517x_2 + b_o = 0\}$

```
df.svm$rho
```

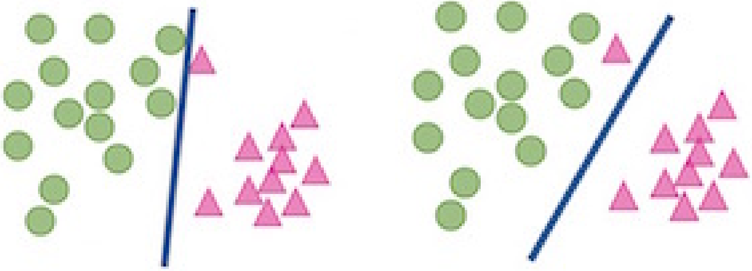
```
## [1] -0.03010995
```

Look at it



Non-separable, and outliers

- Outliers can overly influence a strict boundary



Non-separable

- estimated by maximizing margin $M = 2/||b||$
- subject to $\sum_1^p b_i^2 = 1$, $y_i(x_i^T b + b_o) \geq M(1 - \epsilon_i), i = 1, ..., n$, where $b = \sum_{i=1}^s (\alpha_i y_i) x_i$
- $\epsilon_i \geq 0$, and $\sum_1^n \epsilon_i < C$ where C is a non-negative tuning parameter

Nonlinear separability

- Variables $(x_1, ..., x_p)$ could be expanded to include $(x_1^2, ..., x_p^2)$
- then proceed with building classifier in the expanded space
- maximize margin $M = 2/||b||$ subject to $\sum_1^p b_{1i}^2 + \sum_1^p b_{2i}^2 = 1$, $y_i((x_i^2)^T b_2 + x_i^T b_1 + b_o) \geq M(1 - \epsilon_i)$

Kernels - make nonlinear classification easy to define

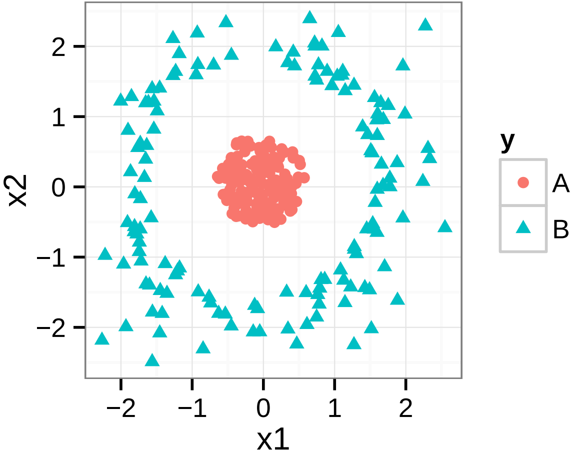
- Because $b = \sum_{i=1}^{s}(\alpha_i y_i)x_i$
- $y_i(x_i^T b + b_o)$ can be written as
- $y_i(\alpha_i x_i^T x_i + b_o)$
-
- $x_i^T x_i$ can be wrapped into a kernel function $K(x_i^T x_i)$ which enables building nonlinear boundaries

Common kernels

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Name	Function
Polynomial	$(\lVert \boldsymbol{x}_i{}^T \boldsymbol{x}_j \rVert + d)^p$
Gaussian radial basis	$exp(-\lVert \boldsymbol{x}_i - \boldsymbol{x}_j \rVert^2 / 2\sigma^2)$
Sigmoid	$tanh(a \lVert \boldsymbol{x}_i{}^T \boldsymbol{x}_j \rVert + d)$

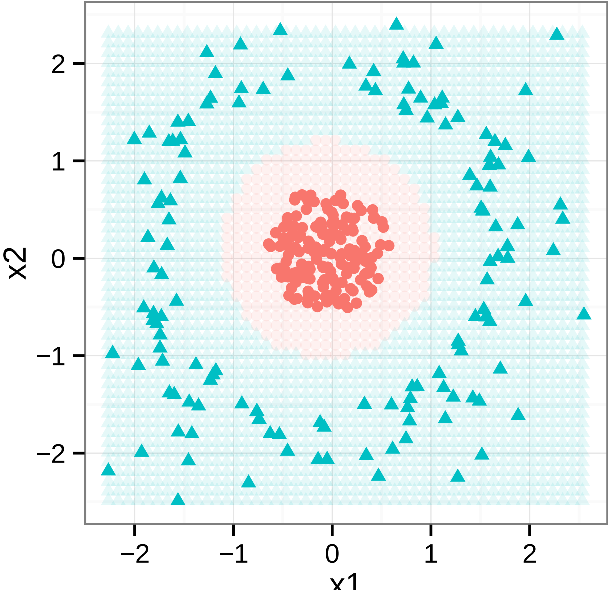
Examples



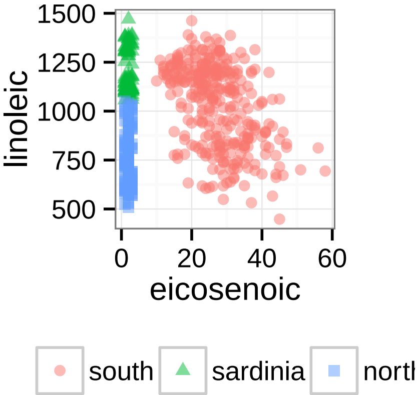
Examples

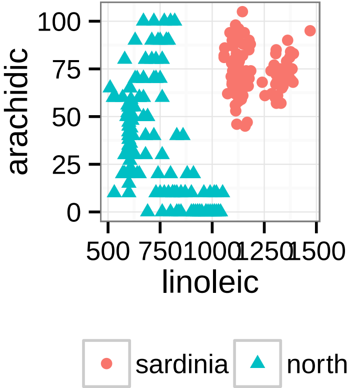
```
##
## Call:
## svm(formula = y ~ ., data = df)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  16

##             x1         x2
## 147  0.5036281  0.3744985
## 148 -0.6434527  0.1471393
## 150  0.2449530 -0.4603625
```

Examples



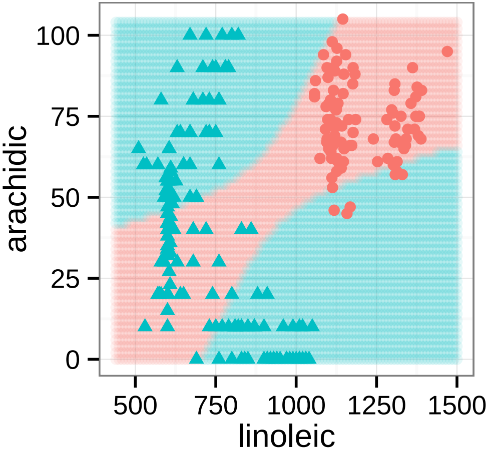
Examples: olive oils





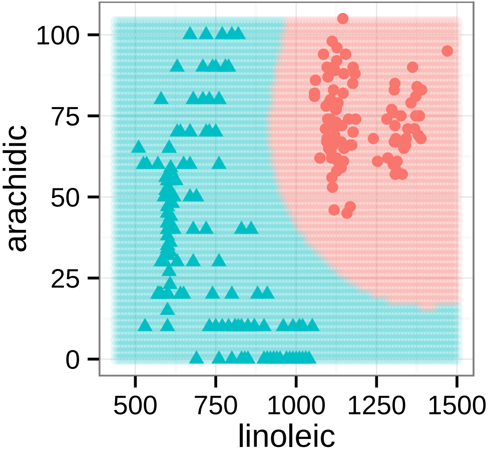
Examples: olive oils

```
olive.svm <- svm(region~linoleic + arachidic, data=olive.sub,
  kernel="polynomial", degree=2)
```



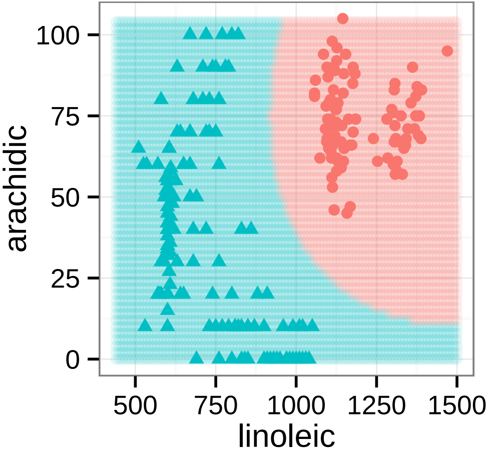
Examples: olive oils

```
olive.svm <- svm(region~linoleic + arachidic, data=olive.sub,
  kernel="radial")
```



Examples: olive oils

```
olive.svm <- svm(region~., data=olive.sub[,-c(2,10)],
  kernel="radial")
```



High-dimensional data

- For high-dimension low sample size problems (more variables than samples) SVM cannot properly estimate the coefficients for the separating hyperplane
- Even fitting a linear kernel is a problem
- The same is true for LDA
- Dimension reduction, or penalisation, needs to be used in association with the classifiers

Links to ggobi videos

- How do boundaries look in high dimensions?
- This video is a basic intro to visualising the SVM model
- This video shows boundaries for a radial kernel fitted to 3D data
- This video shows boundaries for a polynomial kernel fitted to 5D data
- This video another video illustrating looking at boundaries for an SVM model

Multiclass

The available procedures are:

- One-vs-one (also called all-vs-all) or one-vs-all.
- One-vs-one, makes all pairwise classifiers. Predictions are made by a voting scheme.
- One-vs-all does A vs not A, B vs not B, ... Predictions are made by picking the best "positive" class (A, B, ...).