New-age classification:
random forests & support vector machines

Ty Stanford

August 27, 2014

SCHOOL OF
MATHEMATICAL SCIENCES

THE UNIVERSITY
OF ADELAIDE
AUSTRALIA
Outline

Motivation
  Example data
  Why SVM, RF?

RandomForests
  Classification Trees
  Example with Trees
  RandomForests made up of trees

Support Vector Machines (SVMs)
  Optimal separating hyperplanes
  Two-dimensional formulation
  It gets more complicated
Motivation

We have data we want to classify into known classes/groups.

We have data with known labels, the $X/Y$ components: predictor/outcome

Model to create model to estimate $Y$ using only $X$

e.g. Prognostic test, $X$ is easier/cheaper/faster to obtain, can we create sensitive and specific rule to infer $Y$
Introducing the SRBCT (Small Round Blue-Cell Tumours) Data\textsuperscript{1}.

- cDNA Microarray Data
- $n = 83$ patients
- $p = 2308$ gene expression values
- $k = 4$ classes (4 types of SRBCT)

The four classes are:

- EWS: Ewing family of tumors \((n_1 = 29)\)
- RMS: Rhabdomyosarcoma \((n_2 = 25)\)
- NB: Neuroblastoma \((n_3 = 18)\)
- BL: Burkitt lymphoma \((n_4 = 11)\)
To give us a feel for the data here are some scatter plots of randomly selected pairs of genes.
Why SVM/RF?

Other standard methods (of classification)
- LDA/QDA/logistic regression + variants
- KNN
- Classification trees
- Neural networks

The why
- The \( n << p \) problem
- Doesn’t assume distribution about data (kinda)
- Emphasis on boundary values LDA statistical and all points affect the boundary
- SVM produces best results for my data
We have data

\[ \mathbf{X}_i = (X_{i1}, X_{i2}, \ldots, X_{ip}), \quad i = 1, \ldots, n \]

with known group membership in one of \( K \) classes \( G_1, G_2, \ldots, G_K \)

We need to create a decision rule if we get an observation \( \mathbf{X}_{n+1} \) of unknown class. What group do we assign it to?
Briemann [1984] was the first literature to bring them into the statistician’s realm.
Aim of classification trees

- From the root node successively split the data at nodes

- for new observations $x_{n+1}$ you simply traverse the tree down the nodes until you reach a leaf node - thus classify $x_{n+1} \in \{G_1, G_2, \ldots, G_K\}$
How to do this

There are two things that need to be considered:

1. Splitting rules (node impurity)
2. Determining the final leaf nodes (stopping rule)
Splitting rules

- Splitting is done on a feature/variable from the possible features (feature set).
- Say a splitting value \( c_m \) for feature \( v_m \) on node \( m \) is used.
- Observations with values \( x_{vm} \) less than \( c_m \) are classified to the left daughter node.
Node impurity

Before talking about the variable we use at each node to split the data...

We need to talk about node impurity

- a measure (or a function) of the misclassification at a node
- minimise
- unfortunately there is no one measure which is considered best (e.g. regression trees use least squares)
- this measure will help us to chose the best variable to use as the next split
The two favoured impurity measures $i_m(T)$ are

\[ i_m(T) = \sum_{j=1}^{K} \hat{p}_{mj} (1 - \hat{p}_{mj}) \]

\[ i_m(T) = - \sum_{j=1}^{K} \hat{p}_{mj} \log_2 \hat{p}_{mj} \]

where $\hat{p}_{mj}$ is the proportion of observations in group $j$ at node $m$. 
Consider the gini function as code for the 3 class case:

\[
gini<-function(x) \text{return}(\text{sum}(x*(1-x)))
\]

Let’s try some examples:

- \( \text{gini}(c(1/3,1/3,1/3)) = 2/3 \)
- \( \text{gini}(c(0.4,0.3,0.3)) = 0.66 \)
- \( \text{gini}(c(0.5,0.4,0.1)) = 0.58 \)
- \( \text{gini}(c(0.7,0.2,0.1)) = 0.46 \)
- \( \text{gini}(c(0.9,0,0.1)) = 0.18 \)
Now consider the entropy function as code for the 3 class case:

\[
\text{entr} \leftarrow \text{function}(x) \text{ return}(\text{sum}(-x\text{log2}(x)))
\]

Let’s try some examples:

\[
\begin{align*}
\text{entr}(c(1/3,1/3,1/3)) &= 1.59 \\
\text{entr}(c(0.4,0.3,0.3)) &= 1.57 \\
\text{entr}(c(0.5,0.4,0.1)) &= 1.36 \\
\text{entr}(c(0.7,0.2,0.1)) &= 1.16 \\
\text{entr}(c(0.9,0,0.1)) &= 0.47
\end{align*}
\]
Create a tree

We now have the machinery to create a classification tree:

1. for each (as yet unused) feature, find the value of that variable that minimises the node impurity
2. then scan all features for the feature that has the smallest minimised node impurity
3. this feature and splitting value creates a left and right daughter node
4. repeat until all nodes have ‘pure’ data.
Classification Tree (Entropy impurity)

Decision Tree: Entropy

- Gene742 < 1.047
- Gene509 < 0.7168
- Gene123 ≥ 0.9917

Tree Decision Process

Log_2 Expression Ratio

-3 -2 -1 0 1 2 3
Classification Tree (Gini impurity)

Decision Tree: Gini
- Gene1955 < 0.2585
- Gene545 ≥ 0.1841
- Gene153 < -0.2133

Tree Decision Process
- Log_2 Expression Ratio
- Gene742
- Gene545
- Gene1955
To understand the Random Forest algorithm, consider the 2 class problem.

- We build $N_T$ trees (to make a “forest”).
- Each of these $N_T$ trees are constructed as follows...
m observations selected from the total n observations WITH replacement

At each node, randomly select a (new) subset of q variables (where q << p) to inform a recursive split of the data
Classification with random forests

Each tree votes for what an input’s classification is, the class with the most votes wins.
Pros/Cons

Advantages:

- It can handle multi-class data easily.
- Makes use of the framework of trees, but takes away some of the disadvantages of trees via the imposed randomness.
- Feature importance takes into account the sometimes complex relationships with other variables.
- There are other feature importance measures in-built that can be used.
Pros/Cons

Disadvantages:

- As we are fitting an entire forest we now no longer have an easily understandable/visualised method.
- It’s not computational cheap.
- Selection of the training set may drastically alter the model.
Feature Selection

How do we get variable importance from the Random Forest algorithm?

We have \( N \) trees and we have \( n_{OOB} \) 'Out-Of-the-Bag' (not sampled in the tree construction) observations for each tree.

We aggregate the error of each tree with the OOB test sets.

Now to test the \( i \)th feature's importance:

1. Randomly permute the values of the \( i \)th feature in the OOB test sets.
2. Now get the errors of the trees with these altered OOB test sets.
3. The increase in error from the original OOB test error is an indication of the feature's importance.
How do we get variable importance from the Random Forest algorithm?

- We have $N_T$ trees and we have $n_{OOB}$ ‘Out-Of-the-Bag’ (not sampled in the tree construction) observations for each tree.
- We aggregate the error of each tree with the OOB test sets.
- Now to test the $i^{th}$ features importance:
How do we get variable importance from the Random Forest algorithm?

- We have $N_T$ trees and we have $n_{OOB}$ ‘Out-Of-the-Bag’ (not sampled in the tree construction) observations for each tree.
- We aggregate the error of each tree with the OOB test sets.
- Now to test the $i^{th}$ features importance:
  - Randomly permute the values of the $i^{th}$ feature in the OOB test sets.
  - Now get the errors of the trees with these altered OOB test sets.
  - The increase in error from the original OOB test error is an indication of the feature’s importance.
Feature Selection
A terrible motivation: optimal separating hyperplane

\[ f(x) = 0 \]

\[ 2W \]
A terrible motivation: support vectors

\[ f(x) = 0 \]

\[ \xi_1 \]

\[ \xi_2 \]
Classification

We have training data \((y_1, x_1), (y_2, x_2), \ldots, (y_n, x_n)\) with \(x_i \in \mathbb{R}^p\) and

\[
y_i = \begin{cases} 
1 & \text{if from population 1} \\
-1 & \text{if from population 2.}
\end{cases}
\]
Optimisation problem

Separates the two classes:

\[ x_2 = 3x_1 - 3 - \frac{3}{2}x_1 + x_2 = 0 \]

As we are to generalise to higher dimensions:

\[ f(x_1, x_2) = \beta_0 + \beta^T x = 0 \]

\[ \Rightarrow \beta_0 = 3 \text{ and } \beta = \begin{bmatrix} -3 \\ -\frac{3}{2} \\ 1 \end{bmatrix} \]
Optimisation problem

Separates the two classes:

\[ x_2 = \frac{3}{2} x_1 - 3 \]
\[ 3 - \frac{3}{2} x_1 + x_2 = 0 \]
\[ f(x_1, x_2) = 0 \]

As we are to generalise to higher dimensions:

\[ f(x) = \beta_0 + \beta^T x = 0 \]

\[ \Rightarrow \beta_0 = 3 \text{ and } \beta = \begin{bmatrix} -\frac{3}{2} \\ 1 \end{bmatrix} \]
Geometric interpretation
Optimisation problem

Try $x_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$\Rightarrow f(x_1) = 3 - \frac{3}{2}(2) + (1)$

$= 3 - 3 + 1 = 1.$
Now try $x_2 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$

$\Rightarrow f(x_2) = 3 - \frac{3}{2}(4) + (2) = 3 - 6 + 2 = -1$. 
As \( W = c\beta \) and if we take \( x_0 \) on the separating hyperplane/affine and \( x_1 \) on the margin

\[
\begin{align*}
f(x_1) - f(x_0) &= 1 \\
\beta^T x_1 - \beta^T x_0 &= 1 \\
\beta^T (x_1 - x_0) &= 1 \\
\beta^T (c\beta) &= 1 \\
c &= \frac{1}{\|\beta\|^2}.
\end{align*}
\]

\[
\therefore W = c\|\beta\| = \frac{1}{\|\beta\|} = \frac{2}{\sqrt{13}}.
\]
Formulate optimisation

\[
\begin{align*}
\max_{\beta_0, \beta} & \quad W \\
\text{s.t.} & \quad y_i \left( \beta_0 + \beta^T x_i \right) \geq W & \forall \ i = 1, 2, \ldots, n.
\end{align*}
\]
We have found for
\[ \beta_0 = 3 \text{ and } \beta = \begin{bmatrix} -3 \\ 1 \end{bmatrix} \]
\[ W = \frac{2}{\sqrt{13}} \]
But we can see if we multiply \( f(x) \) by a constant \( k \) then
\[ W' = kW \]
Why would you say that?

This is why, we set

\[
f(x) = 0
\]

\[
\therefore kf(x) = k \times 0 = 0
\]

\[
= k\beta_0 + k\beta^T x = 0
\]
We need to set $||\beta||$ to a constant...

$\ldots \ ||\beta|| = 1$
New formulation

\[
\max_{\beta_0, \beta, \|eta\| = 1} W \\
\text{s.t.} \quad y_i \left( \beta_0 + \beta^T x_i \right) \geq W \quad \forall \ i = 1, 2, \ldots, n.
\]
\[
\max_{\beta_0, \beta} \quad W \\
\text{s.t.} \quad \frac{1}{||\beta||} y_i \left( \beta_0 + \beta^T x_i \right) \geq W \quad \forall \quad i = 1, 2, \ldots, n.
\]
But $W = \frac{1}{\|\beta\|}$ so

$$\max_{\beta_0, \beta} \frac{1}{\|\beta\|}$$

s.t.

$$\frac{1}{\|\beta\|} y_i \left( \beta_0 + \beta^T x_i \right) \geq \frac{1}{\|\beta\|} \quad \forall \ i = 1, 2, \ldots, n.$$
So we arrive at

$$\min_{\beta_0, \beta} \|\beta\|$$

s.t. $$y_i \left( \beta_0 + \beta^T x_i \right) \geq 1 \quad \forall \quad i = 1, 2, \ldots, n.$$
Now make it a quadratic minimisation problem

\[
\min_{\beta_0, \beta} \frac{1}{2} \|\beta\|^2
\]

\[
\text{s.t. } y_i \left( \beta_0 + \beta^T x_i \right) \geq 1 \quad \forall \ i = 1, 2, \ldots, n.
\]

This can be solved by Lagrange multiplier method
Non-separable case
Non-separable case formulation

So now we can still construct a hyperplane with the new formulation

\[
\min_{\beta_0, \beta} \frac{1}{2} \|\beta\|^2 + \gamma \sum_{i=1}^{n} \xi_i
\]

s.t. \( y_i(\beta_0 + \beta^T x_i) \geq 1 \quad \forall \ i = 1, 2, \ldots, n \)
\( \xi_i \geq 0 \quad \forall \ i = 1, 2, \ldots, n \)

where \( \gamma \) is a constraining function on the misclassified observation distances, \( \xi_1, \xi_2, \ldots, \xi_n \).
But the questions remain

- Can we create a non-linear boundary?
- How do we choose $\gamma$?
- What about if we have more than two classes?
Non-linear SVM

It turns out the Lagrange mult method gives us the solution

$$\hat{\beta} = \sum_{i=1}^{n} \lambda_i y_i x_i.$$ 

Which means our classifier function is

$$f(x) = \beta_0 + \hat{\beta}^T x$$

$$= \beta_0 + \sum_{i=1}^{n} \lambda_i y_i x_i^T x$$

$$= \beta_0 + \sum_{i=1}^{n} \lambda_i y_i K(x_i, x)$$

$K(y, y')$ is the dot-product or more flexible function!

- Instead of fitting a linear boundary in the original space
  - A linear boundary in the enlarged space is fit
- When transformed back into the original space is non-linear.
The kernel function is at the discretion of the user.

Two of the most used kernel functions are

Radial basis kernel: \( K(x, x') = e^{-\frac{1}{c}\|x-x'\|^2}, \quad c > 0 \)

d\textsuperscript{th} degree polynomial kernel: \( K(x, x') = (1 + \langle x, x' \rangle)^d, \quad d > 0. \)
Two parameters to be optimised, $\gamma$ and $c$, in the SVM model using the radial kernel.

- We have used the solution grid

\[
\gamma \in \{2^{-12}, 2^{-10}, 2^{-8}, 2^{-6}, 2^{-4}\} \text{ and } c \in \{2^{-4}, 2^{-2}, 2^0, 2^2, 2^4\}.
\]

- The axes are on the $\log_2$ scale, i.e. $-4 \rightarrow 2^{-4}$. 
More than two classes...

Consider $k = 3$ and with new point $x$ to be classified.
Multiclass case

To handle more than two classes in the data, \( K > 2 \), multiple SVMs need to be created.

The simplest method to handle the multiclass case is the one-vs-all method.

- Create \( K \) pairwise models comparing
  - one group \( C_k \)
  - all other groups combined \( C_1 \cup \ldots \cup C_{k-1} \cup C_{k+1} \cup \ldots \cup C_K \).

\[ \Rightarrow \] the largest value is the predicted class.
More than two classes...

Step 1 of the one-vs-all method....

Blue vs Not Blue: Blue
More than two classes...

Step 2 of the one-vs-all method....
More than two classes...

Step 3 of the one-vs-all method....
Multiclass case

An alternative: the one-vs-one method (and used as default in LIBSVM)

\[
\binom{K}{2} = \frac{K(K-1)}{2}
\]

- models comparing each pairwise combination
- The class that wins the most pairwise comparisons is the predicted class.

Requires more SVM models for \( K > 3 \) than the one-vs-all method
- a more stable classifier.
More than two classes...

How a one-vs-one method would classify the new point....

Red vs Green: Green
Blue vs Green: Blue
Blue vs Red: Blue

... $G(x) = \text{Blue}$
SVM pros

▶ very flexible
▶ variables are scaled but no distributional assumptions
▶ $n \ll p$ with ease

SVM cons

▶ over fit the data?
▶ the choice of kernel - is there a reason one is best?
▶ optimising the model parameters can take a while
Sorry about the overload of information

Not discussed:

- Regression with these black boxes
- Feature selection
- $k$-fold cross-validation
- Training and test data