**INTRODUCTION TO SUPPORT VECTOR MACHINES**

**by Simon Moss**

|  |
| --- |
| **Introduction** |

**Example**

 Imagine a researcher who wants to develop an algorithm or app that can predict which research candidates are likely to complete their thesis on time. Specifically, the researcher collates information on 1000 candidates who had enrolled at least 8 years ago and thus should have completed their thesis. An extract of data appears in the following table. Each row corresponds to one individual. The columns represent

* whether the candidates completed their thesis on time
* the grade point average of this candidate during their undergraduate studies—as a percentage of the maximum GPA
* age
* IQ, and
* EQ—or emotional intelligence—as measured by a battery of tests on a 10 point scale

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Complete on time | GPA | Age | IQ | EQ |
| Yes | 85% | 34 | 113 | 7 |
| No | 90% | 27 | 104 | 6 |
| No | 70% | 56 | 142 | 8 |
| Yes | 65% | 71 | 107 | 4 |
| … | … | … | … |  |

 The researcher then utilises these data to construct a model, like a series of formulas, that can be used to predict which applicants in the future are likely to complete their thesis on time. To construct this model, researchers could apply a variety of techniques such as

* logistic regression
* ROC analysis
* the K nearest neighbor algorithm
* decision trees, and
* support vector machines.

This document outlines one of these techniques: support vector machines. Compared to many other techniques, support vector machines

* often generate accurate predictions, even when the amount of data is not particularly extensive
* are thus popular and respected
* but, like the majority of machine learning algorithms, do not generate simple conclusions—like “EQ enhances the likelihood of completion”.

**Prerequisites**

 This document, although straightforward, does assume you have developed some knowledge of machine learning, especially cross validation. If you have not developed this knowledge, perhaps read the document about machine learning and K nearest neighbours, available on the CDU research websites, in the section called “Choosing research methodologies and methods”.

|  |
| --- |
| **Introduction to the rationale: Example with 1 predictor** |

 To introduce a few relevant terms and concepts, consider the following illustration. In this graph

* the red circles represent the IQ of individuals who did not complete on time
* the green circles represent the IQ of individuals who did complete on time



 Now consider an applicant, who applied today, and whose IQ is 107. Should you classify this person as a red circle or a green circle? Is this person more likely to complete or not complete the thesis on time. To answer this question, the following display illustrates a reasonable strategy. Specifically, as this display shows

* the short vertical line represents a point midway between the last red circle and the first green circle—two circles that are called the **edges**
* this point can be utilized as a threshold; that is, data to the left of this threshold are classified as red; data to the right of this threshold are classified as green
* the black circle, representing the applicant, lies to the right of this threshold and should thus be classified as green. This person is thus likely to complete the thesis on time



 Although this approach is obvious and straightforward, scholars have still managed to develop convoluted terms to explain these procedures. For example

* the **margin** refers to the distance between the threshold point and the closest observations or circles
* the decision to insert the threshold midway between the two edges is called the **maximal margin classifier**, because this threshold maximizes the margins

**Outliers**

Unfortunately, in practice, two obstacles complicate the decision. The first complication is that most datasets comprise outliers—data points that deviate from the expected pattern. To illustrate, in the following display

* one of the red circles is located closer to the green circle
* if we applied the maximum marginal classifier, the threshold would lie between the two edges
* the black circle would thus be classified as red—as someone who will not complete the thesis
* this conclusion seems misguided, because the black circle seems closer, overall, to the green circles than to the red circles.



To override this problem, researchers may need to abandon the maximum marginal classifier. Instead, they might need to insert a threshold that permits one or more misclassifications. To illustrate, in the following display

* the researcher disregards the outlier
* the threshold is inserted midway between the red circle and green circle that are closest together—after disregarding the outlier
* consequently, the black circle would be classified as green—as someone who would complete the thesis.



So, how would the researcher decide which circles are outliers? In essence

* the researcher would attempt a variety of thresholds—after disregarding one, two, three, and more circles respectively
* the researcher could use cross validation, discussed in another document, to decide which threshold is the best
* that is, the researcher could choose the threshold that predicts outcomes in the testing data most accurately
* in practice, computer software would complete all these tests.

**Bias versus variance**

 This example illustrates a vital tension, prevalent in the field of machine learning. In this example, if a maximum marginal classifier is applied and the threshold is thus inserted between the two edges

* the model represents the existing data perfectly; that is, all red circles appear to the left of this threshold and all green circles appear to the right of this threshold
* proponents of machine learning call models that represent the data closely as low in **bias**
* however, this model might not classify future cases accurately; that is, the black circle may be misclassified
* proponents of machine learning call models that do not predict outcomes accurately as high in **variance** because the outcomes vary appreciably from the predictions

Interesting, models that are low in bias are often high in variance and vice versa. To illustrate, consider the two scatterplots in the following display

* in the left scatterplot, the green circles deviate appreciably from the straight line
* the straight line, therefore, does not represent the data accurately; bias is high
* but the straight predicts the black circle closely: variance is low; most future cases are likely to be reasonably close to this straight line
* in the right scatterplot, the red circles are very close to the jagged line; bias is thus low
* however, this jagged line does not predict the black circle closely; many future cases are likely to deviate appreciably from this jagged line

|  |  |
| --- | --- |
|  |  |

This example is designed to instil a vital intuition about machine learning. Specifically

* simple models tend to increase bias but reduce variance: these models predict future cases better than perhaps they represent past data
* complex models tend to decrease bias but amplify variance; these models do not predict future cases as well as they represent past data

This technique, in which outliers are identified and thus misclassifications are permitted, is

* called a soft margin classifier, because the distance between the threshold and the closest circle that is not an outlier is called a **soft margin** rather than a margin
* or called a support vector classifier, because these closest circles are also called **support vectors** especially if represented in more than one dimension—a concept discussed later.

**Overlapping clusters**

 The second complication is that, in many circumstances, researchers might not be able to separate the two clusters—such as the red circles and green circles—with a single point or line, even after outliers are excluded. To illustrate, in following display

* the individuals who do not complete their thesis—the red circles—are either very low or very high in IQ
* perhaps, if their IQ is too low, they cannot understand the materials. If their IQ is too high, they might feel bored
* regardless of the reason, irrespective of where the threshold is inserted, many circles will be misclassified
* in this example, the black circle will be classified as green, but is closer to red cases



Fortunately, researchers have discovered an intriguing method to solve this problem. In essence, they apply various formulas that, essentially, increase the number of dimensions until the clusters of data can be separated with a straight line, plane, or hyperplane. To illustrate, in the following display, the x or horizontal axis represents the IQ of each circle, as previously, but the y or vertical axis represents the square of these IQ values. To illustrate

* IQ of the leftmost circle is 85
* the square of 85 is 7225
* hence, this circle appears at the coordinates 85 and 7225



Remarkably, when the data is transformed from a one-dimensional graph to a two-dimensional graph, a single line can now separate the red circles and green circles. That is

* circles above this line, such as the black circle in this example, will be classified as red or someone who is not predicted to complete the thesis
* circles below this line will be classified as green or someone who is predicted to complete the thesis

Indeed, in this example, this simple line represents all the existing data and is thus low in bias. Furthermore, because the line is simple, the corresponding formula is likely to classify future cases correctly as well and thus be low in variance.

**Conclusion from this example**

This document has, thus far, merely introduce this approach rather than explained the principles in detail. Nevertheless, even this simple example has illustrated some vital principles

* after removing outliers, the method is designed to uncover a point, line, plane, or hyperplane that separates the two clusters of data points—such as candidates who completed the thesis on time and candidates who did not complete the thesis
* if no point, line, or plane can separate the clusters of data points, the method transforms the data by essentially increasing the number of dimensions—from one dimension to two dimensions, for example
* the method continues to increase the number of dimensions until identifying a line, plane, hyperplane that separates the two clusters of data points
* a hyperplane is a plane, but represented in more than three dimensions—a concept that is actually impossible to imagine
* this line, plane, or hyperplane can then be applied to classify future cases
* this method is called support vector machines

|  |
| --- |
| **Introduction to the rationale: Polynomial kernels** |

 In the previous example, the researcher, or at least a software program, constructed a scatterplot that represented both the original IQ values and the square of these values and then identified a line that separates the red circles and green circles. But, this formula, in which the original values are squared, will not always be effective. How can the software program determine which formula will be effective? As a preliminary answer

* the program applies a formula: **kernel = (a x b + c)d**
* the software continues to vary c and increase d—two parameters explained later—until the model is effective
* that is, the software continues until the model or formula accurately classifies the data
* this model or approach is called polynomial kernels

This section presents more details about polynomial kernels. You do not need to understand these details. Nevertheless, if you read this section, you might be able to appreciate support vector machines more.

**Dot products**

To learn about polynomial kernels, you need to understand dot products. To illustrate a dot product, consider two sets of numbers, such as [3, 5] and [2, 7]. Each set might represent the number of pets and number of people in one household. Or each set might represent one point on a scatterplot, for example

* By definition, the dot product of [3, 5] and [2, 7] is [3 x 2, 5 x 7] or [6, 35]
* That is, the first terms and the second terms are multiplied separately
* The dot product of [3, 5] and [2, 7] is written as [3, 5] . [2, 7]
* Conversely [6, 35] can thus be reduced to [3, 5] . [2, 7]

**Example of a kernel**

 Now that you have been introduced to dot products, you can learn about polynomial kernels. To illustrate, consider the following display.



We will now apply the formula **kernel = (a x b + c)d .** In this formula

* a and b refer to the two circles labelled a and b—such as about 84 and 92 respectively
* c is a parameter that can vary. This parameter changes the scale.
* d is the number of dimensions

To apply this formula, we first choose an arbitrary value of c, such as 0.5. We initially set d to 2. Therefore, kernel

* = (a x b + 0.5)2
* = ab2 + 0.5 x ab + 05 x ab + 0.25
* = ab + ab2 + 0.25 after some rearranging
* = [a, a2, 0.5] . [b, b2, 0.5] as implied by discussion about dot products before
* = [84, 7056, 0.5] . [92, 8464, 0.5] because a = 84 and b = 92
* These two sets of numbers [84, 7056, 0.5] and [92, 8464, 0.5] can be represented on a scatterplot.
* Specifically, the a and b circles in the following scatterplot represent these two sets of numbers. Note that sets of numbers are often called vectors
* This figure comprises only two dimensions. The 0.5 has not been represented. Perhaps you can assume the graph equals 0.5 on a z axis that you cannot see.



 So far, the formula has merely converted one dimensional data—such as the numbers 84 and 92—to two dimensions. But, in addition, using the formula, we can calculate the kernel—a value that roughly indicates the distance between two points. That is

* kernel or distance = (a x b + 0.5)2
* kernel or distance = (84 x 92 + 0.5)2
* kernel or distance = 59 729 712

The computer can then apply this procedure to every pair of circles—to generate the distance between each pair of circles. So, why does the computer calculate the distance between each pair of circles? What is the benefit of this information? Roughly speaking

* remember that support vector machines attempt to identify a line, plane, or hyperplane that separates the various clusters—such as the red circles and green circles
* specifically, this machine is designed to maximize the distance between this line, or plane and the closest points in each cluster
* information about the distance between points helps calculate the position of this line, plane, or hyperplane

**Overview reiterated**

 To reiterate, to conduct support vector machines using polymomial kernels, the software

* begins with a specific value of c and d in the formula (a x b + c)d .
* uses this formula to estimate the distance between all the data points—ultimately to locate a line or plane that separates the clusters of data points optimally
* tests the degree to which the training data are consistent with this model
* iteratively and systematically changes values of c and increases levels of d to improve this model

The researcher can then utilise cross-validation to assess this model. Later, one of the sections will demonstrate how you can complete these tasks.

|  |
| --- |
| **Extensions Radial kernel and multiple dimensions** |

**Multiple dimensions**

 In the previous examples, IQ, the only predictor, was represented in one dimension, on the x axis. The formulas then converted these data to two dimensions, and then three dimensions, and so forth, until the clusters—such as the red circles and green circles—could be separated. Note

* if the data comprises two predictors, you would need to construct a scatterplot in two dimensions to represent these data, as illustrated in the following display
* in this instance, you cannot draw a line that separates the red circles and green circles
* but, if the data were then transformed to three or more dimensions, you might be able to draw a plane surface to separate the red circles and green circles
* the rationale is the same, regardless of how many predictors or dimensions you distinguish



If the data comprises four or more predictors, you need to draw a plot in four or more dimensions. Actually, nobody can imagine four or more dimensions. But, the same rationale and formulas still apply.

**Radial kernels**

Rather than polynomial kernels to model the data, the software can use another formula, called radial kernels. These kernels extend the polynomial kernels to infinite dimensions. Radial kernels are harder to imagine but tend to be more powerful than polynomial kernels. But to illustrate the principle of radial kernels, consider the following display

 

IQ

80 90 100 110 120 130

a

b

c

 Suppose you wanted to classify the black circle. That is, you wanted to ascertain whether this circle should be classified as red—and thus unlikely to complete the thesis—or green—and thus likely to complete the thesis. To answer this question

* radial kernels consider all the training data—all the other circles—to classify this black circle
* however, radial kernels assign greater weight to the data that is closer to this black circle
* for example, radial kernels would assign considerable weight to the red circle labelled b, because this circle is quite close to the black circle
* radial kernels would assign negligible weight to the green circle labelled c, because this circle is farther from the black circle
* after aggregating this information, the radial kernel would most likely classify the black circle as red.

To achieve this goal, radial kernels apply the following formula to weight the circles. The key feature of this formula is that, as the distance between the two points—a and b—increases, the weight diminishes. For more information, watch <https://www.youtube.com/watch?v=Qc5IyLW_hns>

|  |
| --- |
| Radial kernel = e to the power of – (a - b)2 |

|  |
| --- |
| **How to conduct support vector machines. Step 1: Install and use R** |

You can use a variety of statistical packages to apply this technique. This document will show you how to conduct this technique in software called R. If you have not used R before, you can download and install this software at no cost. To achieve this goal

* visit <https://www.cdu1prdweb1.cdu.edu.au/files/2020-08/Introduction%20to%20R.docx> to download an introduction to R
* read the section called Download R and R studio
* although not essential, you could also skim a few of the other sections of this document to familiarize yourself with R.

|  |
| --- |
| **How to conduct support vector machines. Step 2: Upload the data file** |

Your next step is to upload the data into R. To achieve this goal

* open Microsoft Excel
* enter your data into Excel; you might need to copy your data from another format. Or your data might already have been entered into Excel

In particular, as the following example shows

* each column should correspond to one variable
* ensure the first column corresponds to the key outcome—in this instance, whether participants have completed the thesis or not
* each row should correspond to one unit—such as one person, one animal, one specimen, and so forth
* the first row labels the variables
* to prevent complications, use labels that comprise only lowercase letters—although you could end the label with a number, such as age3



To convert this file into a csv file—such as a file called research.data—and then to upload this file into R studio

* visit <https://www.cdu1prdweb1.cdu.edu.au/files/2020-08/Introduction%20to%20R.docx> to download the introduction to R—unless you have already downloaded this document
* read the section called “Upload some data”

|  |
| --- |
| **How to conduct support vector machines. Step 3: Enter the code and interpret the results** |

 To conduct support vector machines, you need to enter some code. The code might resemble the following display. At first glance, this code looks absolutely terrifying. But actually this code is straightforward once explained.

|  |
| --- |
| install.packages("caret") library(caret)anyNA(research.data)set.seed(3033)intrain <- createDataPartition(y = research.data$completion, p =0.7, list= FALSE)training <- research.data[intrain, ]testing <- research.data[-intrain, ]training[["completion"]] = factor(training[["completion"]]) testing[["completion"]] = factor(testing[["completion"]])train.control <-trainControl(method = "repeatedcv", number = 10, repeats = 3) set.seed(3233)model1 <- train(completion ~., data =  training, method = "svmPoly",  trControl = train.control, preProcess = c ("center", "scale"), tunelength = 10) test\_pred <- predict(model1, newdata = testing)testing[["completion"]] = factor(testing[["completion"]])confusionMatrix(table, test\_pred, testing$completion) |

 To enter code, you could write one row, called a command, at a time in the Console. But, if you want to enter code more efficiently,

* in R studio, choose the File menu and then “New File” as well as “R script”
* in the file that opens, paste the code that appears in the left column of the following table
* to execute this code, highlight all the instructions and press the “Run” button—a button that appears at the top of this file

You should not change the bold characters in the left column. You might change the other characters, depending on the name of your data file, the number of variables, and so forth. The right column of the following table explains this code. You do not, however, need to understand all the code.

|  |  |
| --- | --- |
| Code to enter | Explanation or clarification |
| **install.packages("caret")** **library(caret)** | * R comprises many distinct sets of formulas or procedures, each called a package
* *caret* is one of these packages and can be used to conduct a range of machine learning algorithms including support vector machines
* *install.packages* merely installs this package onto the computer
* *library* then activates this package
* the quotation marks should perhaps be written in R rather than Word; the reason is that R recognises this simple format— " —but not the more elaborate format that often appears in Word, such as “ or ”.
* in addition, sometimes you need to restart R after you install packages; otherwise, you might receive some error messages
 |
| **anyNA**(research.data) | * This code determines whether or not the data file called research.data contains missing data
* When conducting support vector machines, you should perhaps omit cases with missing data or use a strategy that substitutes missing data with reasonable approximations
 |
| **set.seed**(3033) | * Later, the computer will be asked to identify some random numbers
* This code, however, instructs the computer to begin these random numbers at position 3033
* Consequently, you could, if you wanted, identify the same random numbers again
 |
| intrain <- **createDataPartition(y =** research.data$completion, **p =**0.7, **list=** FALSE) | * This code randomly selects 0.7 or 70% of the rows in the data file *research.data*—or at least 70% of the rows that contain data in the column called *completion*
* These rows are then stored in a container or variable called *intrain*
* For example, *intrain* might contain numbers like 1, 3, 4, 6, 9, 10, 13, and 16—representing 70% of the rows
 |
| training <- research.data[intrain, ] | * This code extracts all the rows in the data file called *research.data* that correspond to *intrain*—such as rows 1, 3, 4, 6, 9, 10, 13, and 16
* This subset of rows is stored in a data file or container called *training*
* Thus *training* represents the training data
 |
| testing <- research.data[-intrain, ] | * This code extracts all the rows in the data file called *research.data* that does not correspond to *intrain*—such as rows 2, 5, 7, 8, 11, and 12
* This subset of rows is stored in a data file or container called *testing*
* Thus *testing* represents the testing data
 |
| training[["completion"]] = **factor**(training[["completion"]]) testing[["completion"]] = **factor**(testing[["completion"]]) | * R might assume that *completion—*a column that contains 1s and 0s—is a numerical variable
* But, to undertake support vector machines, R must recognize these 1s and 0s actually represent categories, such as individuals who completed their thesis and individuals who did not complete their thesis
* This code merely converts the column labelled completion to a categorical variable—sometimes called a factor.
 |
| train.control **<-trainControl(method =** "repeatedcv", **number =** 10, **repeats =** 3)  | * This code is designed to stipulate a few parameters that will be utilized later to conduct the support vector machine
* You do not need to understand this code
* But, to illustrate, *repeatedcv* refers to a resampling method called repeated cross validation; number is the number of iterations permitted to optimize the cross-validation, and so forth
 |
| **set.seed(**3233) |  |
| model1 <- **train**(completion ~., **data =**  training, **method =** "svmPoly",  **trControl =** train.control, **preProcess = c** ("center", "scale"), **tunelength =** 10)  | * This code actually conducts the support vector machine
* For example, *completion ~ .* indicates that completion is the outcome variable and all the other variables are the predictors
* Training data are used to generate the model—and hence the data file is *training*
* The method is *svmPoly*—or support vector machines with polynomial kernels
* If you wanted to use radial kernels instead, the method would be *svmRadial*
* You do not need to change the other parameters
* The model or formulas that classify the cases most effectively is stored in a contained called model1
 |
| test\_pred **<- predict(**model1, **newdata =** testing) | * This code now applies this model to predict the outcome that corresponds to the testing data—stored in a data file called testing
* Thus *test\_predict* will contain predictions on whether the individuals in the testing data file will complete the thesis on time or not
 |
| **confusionMatrix(table**, test\_pred, testing$completion) | * This code then generates a confusion matrix to compare these predictions with the actual outcome in the testing data file

If this code does not work, you can apply other instructions to compare the predicted and actual outcomes, such ass* percentCorrect <-100 \* sum(testing$completion == test\_pred)/NROW(testing)
* percentCorrect
 |