Lecture #10: **Introduction to Support Vector Machines**

Mat Kallada

STAT2450 - Introduction to Data Mining with R

Outline for Today

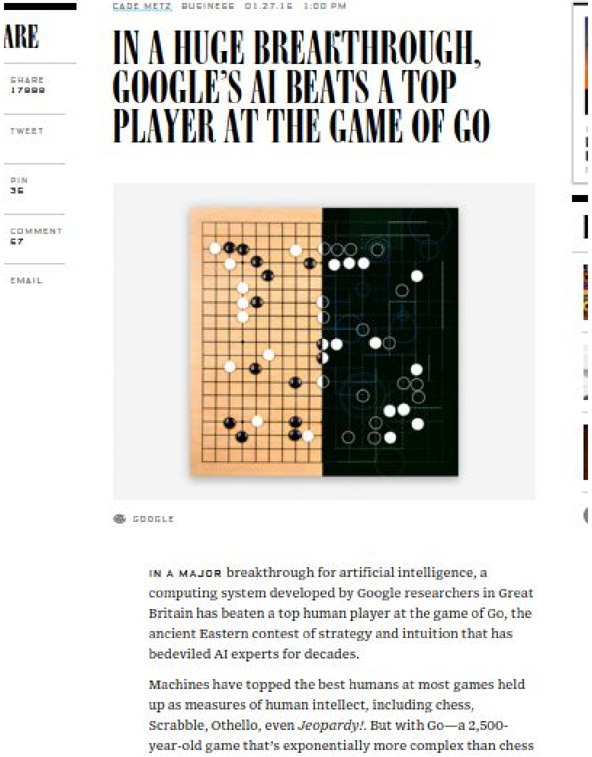
Support Vector Machines - Another way to draw lines

Multi-class Support Vector Machines

Kernels and Support Vector Machines

Support Vector Machines for Regression

Data Mining Classifiers to Play Go: Google's AlphaGo



CADE METZ BUSINESS 01.27.16 1:00 PM

ARE

SHARE 17999

TWEET

PIN 36

COMMENT 67

EMAIL

IN A HUGE BREAKTHROUGH, GOOGLE'S AI BEATS A TOP PLAYER AT THE GAME OF GO

GOOGLE

IN A MAJOR breakthrough for artificial intelligence, a computing system developed by Google researchers in Great Britain has beaten a top human player at the game of Go, the ancient Eastern contest of strategy and intuition that has bedeviled AI experts for decades.

Machines have topped the best humans at most games held up as measures of human intellect, including chess, Scrabble, Othello, even *Jeopardy!*. But with Go—a 2,500-year-old game that's exponentially more complex than chess

Remember - **Deep Blue's Win** in 1997?

They didn't use data mining

But still cool

Data Visualization Strategies

We’ve seen five so far:

- **Scatter Plots**: Data Points on Cartesian Plane
- **Line Plots**: Change of Numerical value against Numerical value
- **Bar Graphs**: Categorical against Numerical values
- **Histograms:** Count distribution of values
- **Heatmaps**: Categorical variable against another Categorical Variables

Outline for Today

Support Vector Machines - Another way to draw lines ←

Multi-class Support Vector Machines

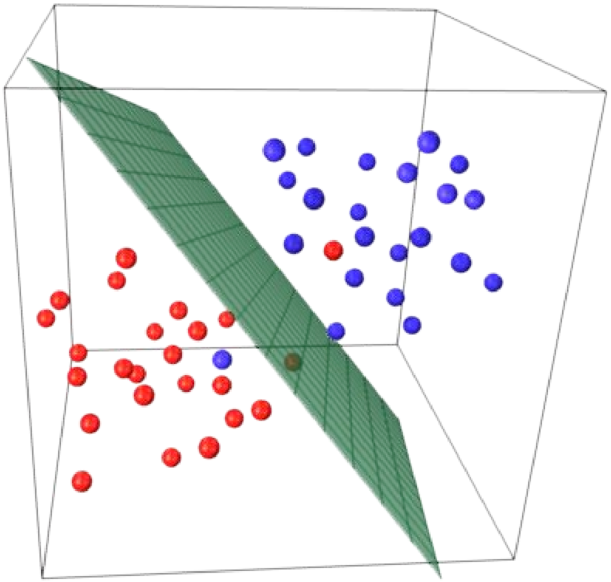
Kernels and Support Vector Machines

Support Vector Machines for Regression

Let’s change gears for a bit...

Remember, We learned **two ways to draw lines**

To solve regression or classification tasks



Ways to Create Predictive Models

(I.e. Methods to solve the Supervised Data Mining Setup)

Decision Trees

Construct a decision tree which chops on the vector space

“Chops” are feature splits which minimize error

K-Nearest Neighbours

Look at the K-closest Points in Training Data

Supervised Data Mining: The Line Drawing Contest

Who can draw the most “realistic” line?

Supervised Data Mining: The Line Drawing Contest

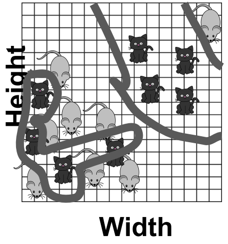
Who can draw the most “realistic” line?

Data (in a table form)

K-nearest Neighbours

Grid

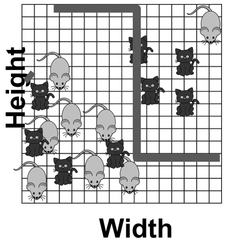
K = 5 to 10



Decision Trees (“rpart”)

Grid

cp = 0.01 to 0.10



Supervised Data Mining: The Line Drawing Contest

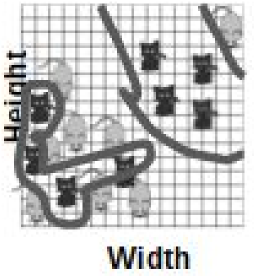
To evaluate whether these lines/curves actually work

Let’s use the one with highest performance

We need to use either **hold-out validation** or K-Fold Cross-Validation

Test Set

Model Built with K=3



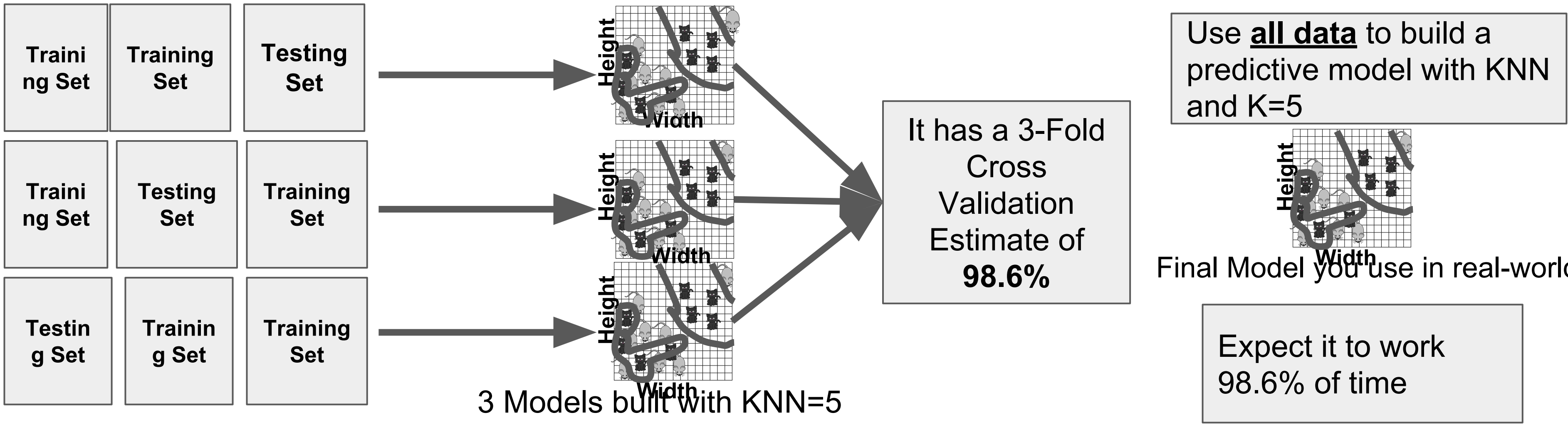
It got **99.5%** right on test set!

Supervised Data Mining: The Line Drawing Contest

To evaluate whether these lines/curves actually work

Let’s use the one with highest performance

We need to use either hold-out validation or **K-Fold Cross-Validation**



Supervised Data Mining: The Line Drawing Contest

To evaluate whether these lines/curves actually work

Let’s use the one with highest performance

We need to use either hold-out validation or K-Fold Cross-Validation

Both work but K-fold Cross-Validation is more robust (no “easy examples”)

Supervised Data Mining: The Line Drawing Contest

Why are lines a big deal again?

The actual underlying hypothesis is unknown

Lots of features in our dataset makes it difficult to draw them by hand

Simulating intelligent behaviour has complex lines

AlphaGo was just a very complex model which predicted the next move to make

Supervised Data Mining: The Line Drawing Contest

Why are lines a big deal again?

The actual underlying hypothesis is unknown

Supervised Data Mining: The Line Drawing Contest

Why are lines a big deal again?

The actual underlying hypothesis is unknown

Infinitely many ways we can create lines

Lots of features in our dataset makes it difficult to draw them by hand

Supervised Data Mining: The Line Drawing Contest

Simulating truly intelligent behaviour has complex lines/curves

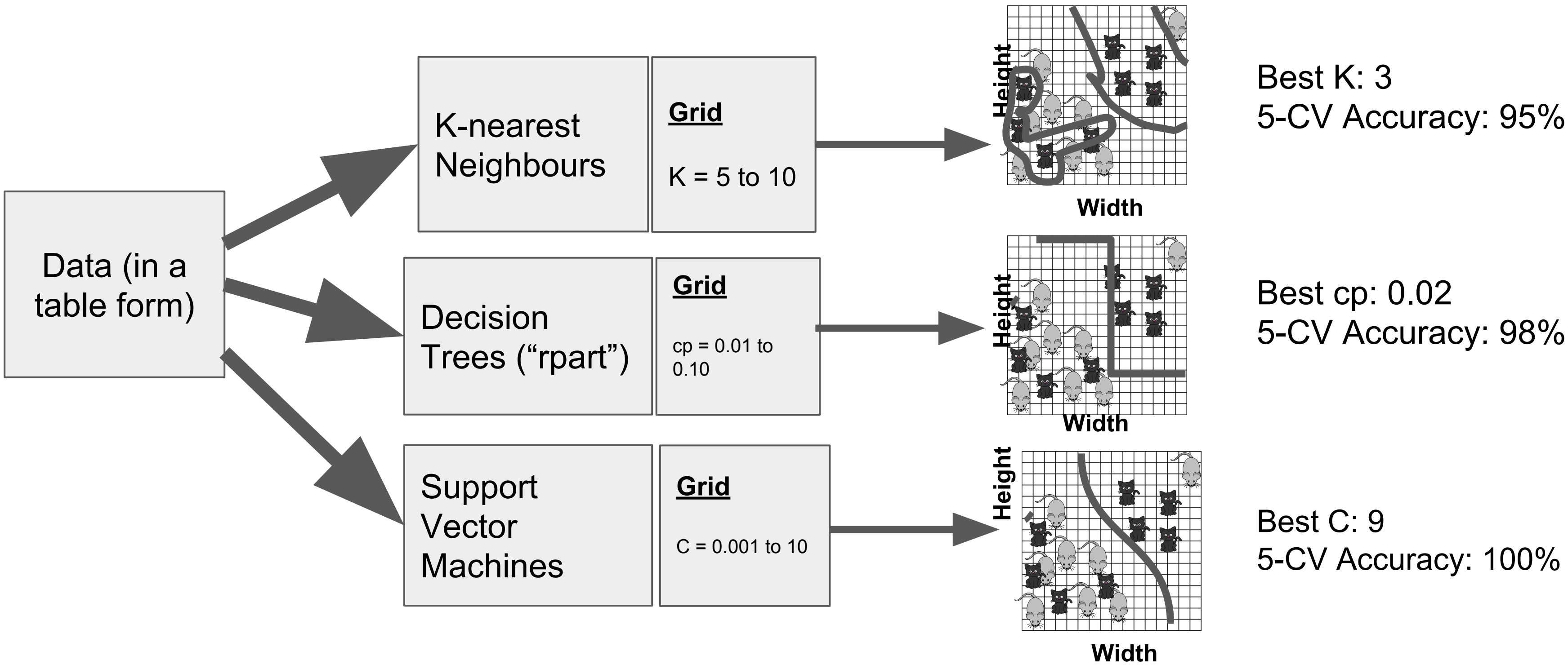
Keep this in mind:

AlphaGo was just a very complex predictive model which predicted the next move to make in Go

It took them **months** with a supercomputer to build this model

Supervised Data Mining: The Line Drawing Contest

Who can draw the most “realistic” line?



Support Vector Machines: How do they draw lines?

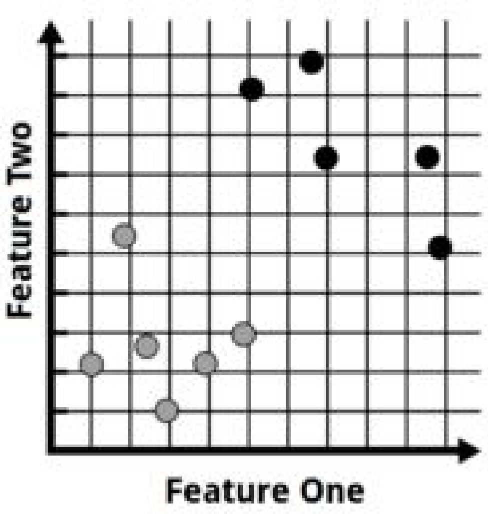
They are another supervised data mining technique used for either **regression** or **classification**

Invented by **Vladimir Vapnik** (now at Facebook)

Support Vector Machines: How do they draw lines?

Let's look at two-class classification first.

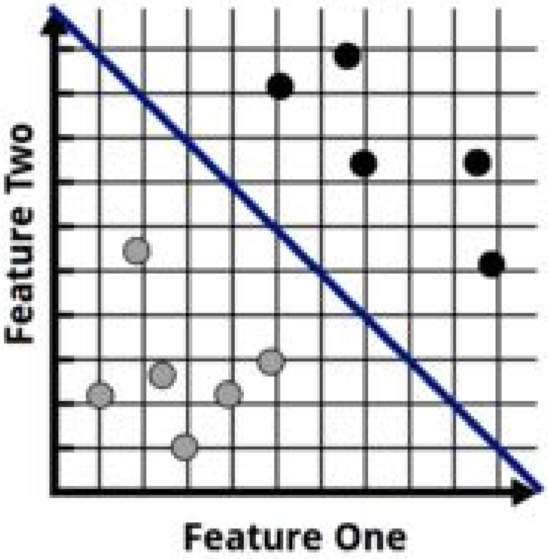
Consider the classification scenario below:



How could we create a classifier which best divides the two classes?

Support Vector Machines: How do they draw lines?

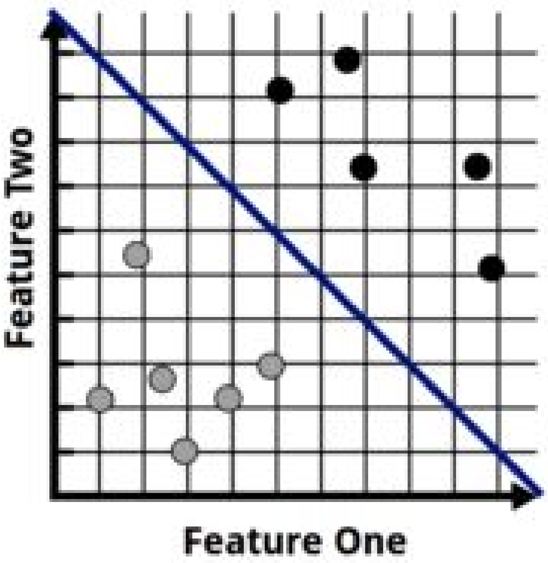
Let's look at classification first. Consider the classification scenario below:



Hmm - probably right there.

Support Vector Machines: How do they draw lines?

It's **<u>"right in between"</u>** both classes and divides them both pretty well.



It is a line **<u>equidistant</u>** from the "outside" points of each class

Support Vector Machines: How do they draw lines?

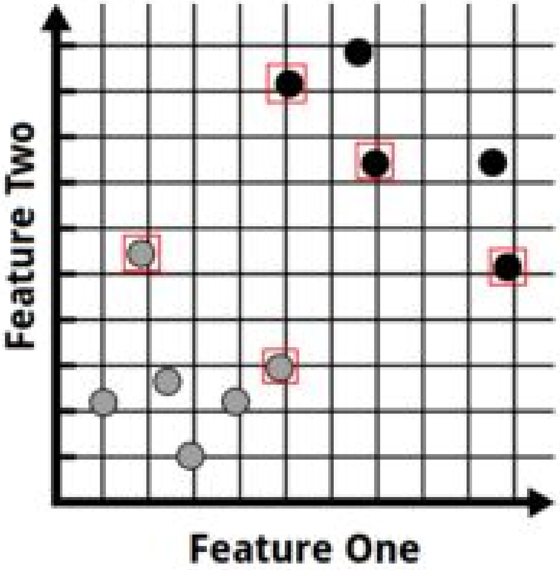
There are **two steps** for this:

1. Identify these “outside” points

2. Draw a line equidistant between both sets of outside points

Support Vector Machines: How do they draw lines?

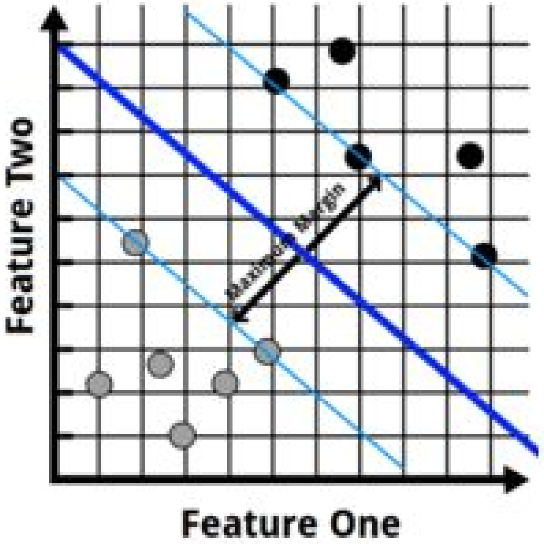
Step 1: Identify these "outside" points.



They are called the "support vectors" in the SVM model.

Support Vector Machines: How do they draw lines?

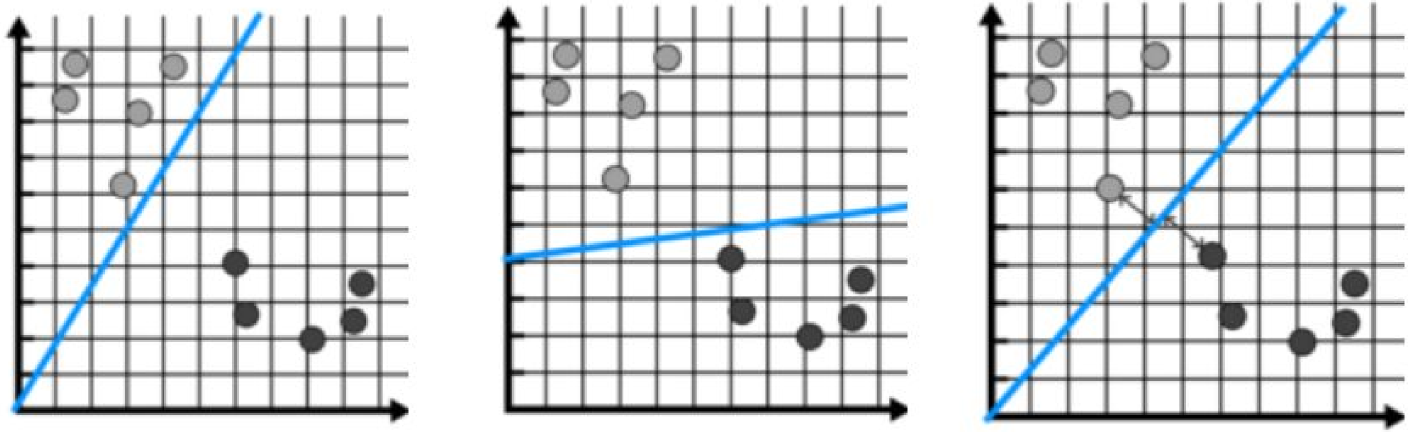
<u>Step 2:</u> Draw a line equidistant between support vectors



Draw the dividing line which is perpendicular to the margin with the furthest distance between the boundaries of the support vectors

Support Vector Machines: How do they draw lines?

Step 2: Draw a line equidistant between support vectors



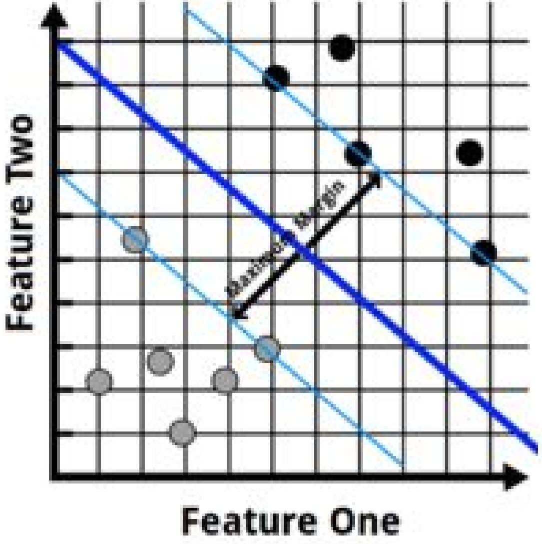
(e) This doesn't seem right. The decision boundary is too close to one class.

(f) Still no - with so many possible solutions - which one should we pick?

(g) This one. Why not use the line with a margin having the maximal distance away from each classes outer-point(s)? Seems like the most probable to me.

Support Vector Machines: How do they draw lines?

<u>Step 2:</u> Draw a line equidistant between both classes



<u>Final Exam:</u> I'll ask a question related to how/why does the SVM draw this line.

<u>Step 2:</u> Draw a line equidistant between both classes

To find this **'middle-ground' line**

Consider that we need to find the appropriate slope and intercept of the line with respect to an optimization task of **<u>maximizing the margin</u>** between support vectors.

Step 2: Draw a line equidistant between both classes

The idea is easy to understand, but there is beautiful math behind the scenes to find this line.

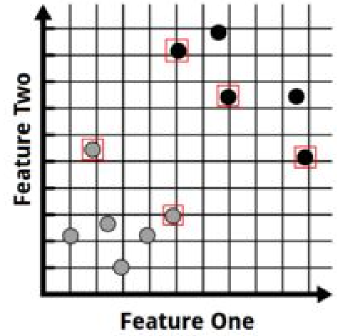
If you are interested in this, please have a look at Lecture Notes.

Support Vector Machines: How do they draw lines?

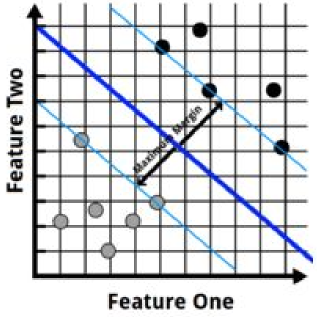
In a nutshell, a predictive model built with SVM has two steps:

Step 1: Find the "outside" data points (called the support vectors)

Step 2: Draw line equidistant between them.



Step 1: **Find Support Vectors**



Step 2: **Draw Equidistant Line**

Support Vector Machines: Some issues

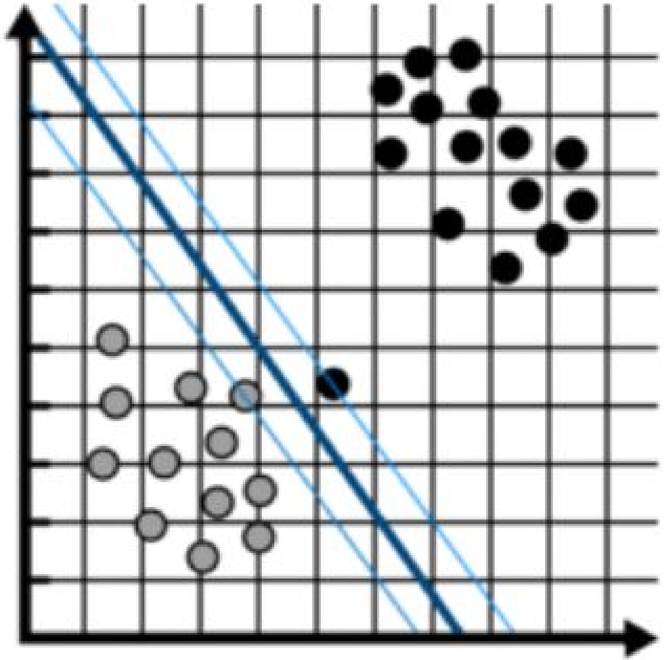
What about **noisy observations**?

Deformed cats may ruin the equidistant line

What if noise were chosen as a support vector?

Noisy Observations can ruin the line

If the support vectors were noise, we may get something like this...



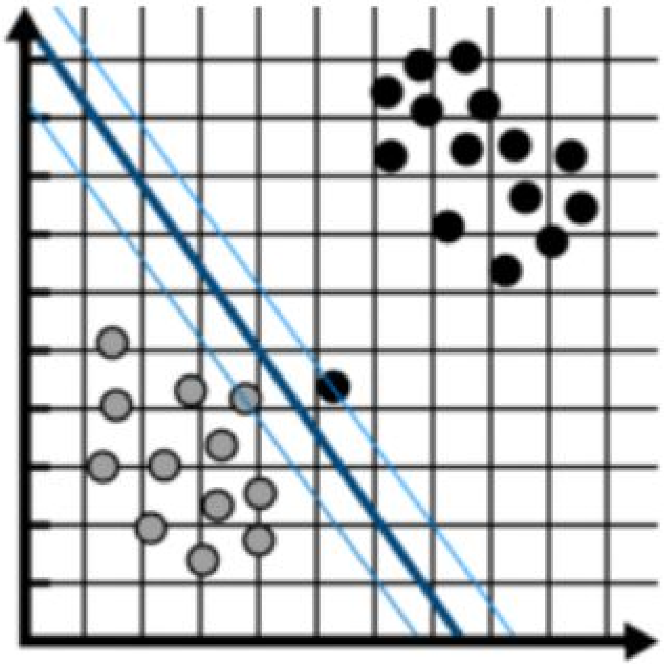
Noisy Observations can ruin the line

A **<u>single noisy example</u>** messes up everything

The support vector is incorrectly chosen.

Our model is <u>invalid</u> and has **overfit**

It wouldn't generalize very well to real-world cases



Support Vector Machines: The Cost Parameter

Like K in K-nearest Neighbours

Like “cp” and “max depth” in Decision Trees

We have a hyperparameter to control complexity of the model

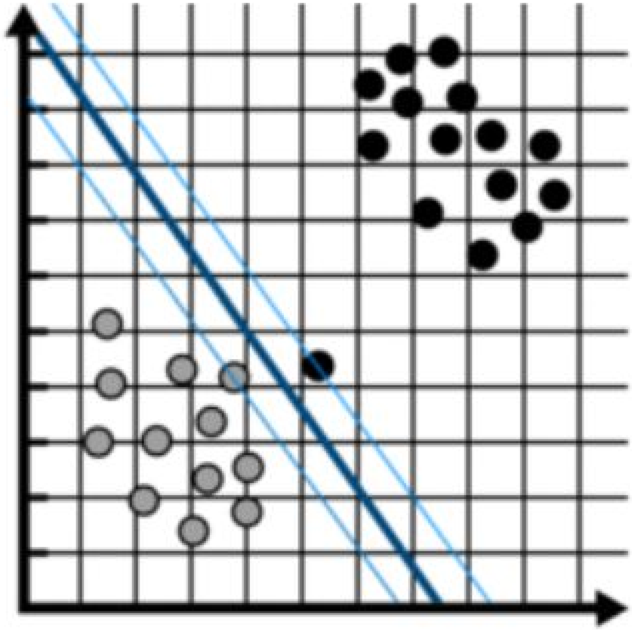
That is, how tolerable it is to noisy observations in our data

Support Vector Machines: The Cost Parameter

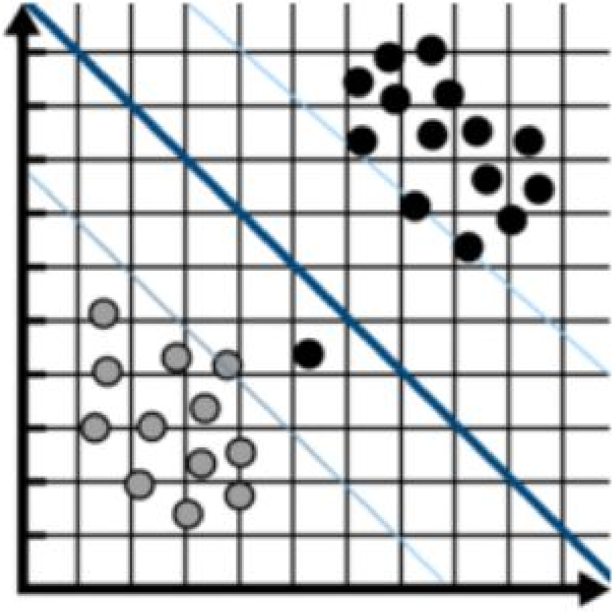
We can specify the “**Cost**” hyperparameter or “C” to avoid noise.

A way of determining the **resistance** of the chosen support vectors to noise.

Support Vector Machines: The Cost Parameter



C = 1,000



C = 0.001

Support Vector Machines: The Cost Parameter

Cost (C): When picking the support vectors, the “cost” of incorrectly classifying a data point.

Support Vector Machines: The Cost Parameter

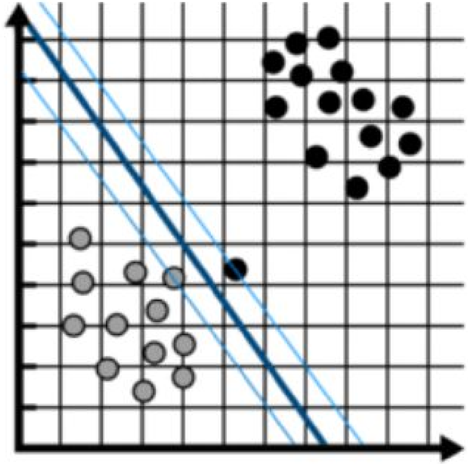
Cost (C): When picking the support vectors, the “cost” of incorrectly classifying a data point.

Higher C values means that there is a **higher** cost to incorrectly classifying a training point. Too high means we’ll underfit.

Lower C values means that there is a **lower** cost to incorrectly classifying a training point. Too low means we’ll overfit.

Support Vector Machines: The Cost Parameter

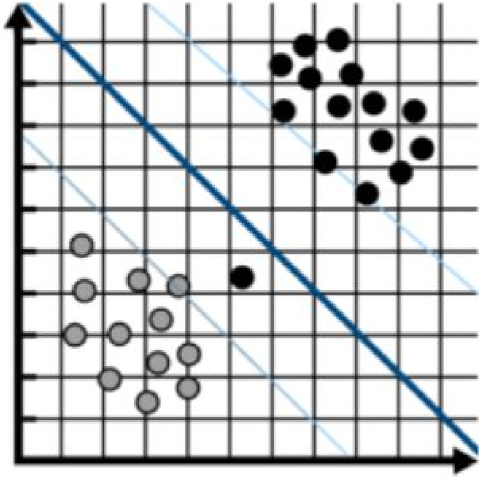
C = 1,000



The cost is **<u>high</u>** to make mistakes on the training data.
Since the cost is high, we can’t make mistakes
Let’s draw a line here

Support Vector Machines: The Cost Parameter

C = 0.001



The cost is **<u>low</u>** to make mistakes on the training data.
Since the cost is low, we can make some mistakes
Let’s draw a line here

Support Vector Machines: The Cost Parameter

“Cost” for SVMs is like K for KNN (or cp for Decision Trees)

We just try a bunch of different cost values until we find a good one

The one that will develop a model that works well.

Find one that avoids both overfitting and underfitting.

Outline for Today

Support Vector Machines - Another way to draw lines

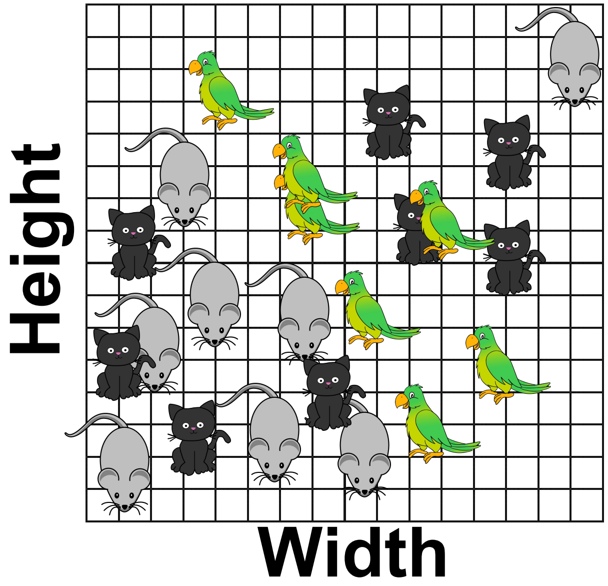
Multi-class Support Vector Machines ←

Kernels and Support Vector Machines

Support Vector Machines for Regression

Support Vector Machines: Multi-class Problems

What if we had more than two classes?



How would we draw the line now?

Support Vector Machines: Multi-class Problems

K-Nearest Neighbours and Decision Trees are naturally made to handle multi-class tasks

SVMs are not made for classification tasks with multiple classes.

The “**One-vs-One**” trick for Multi-Class SVMs

We **can’t use SVMs** by themselves for multi-class problems

We can use a trick for SVMs to solve multiclass problems

The "**One-vs-One**" trick for Multi-Class SVMs

We train three classifiers for all combination of classes:

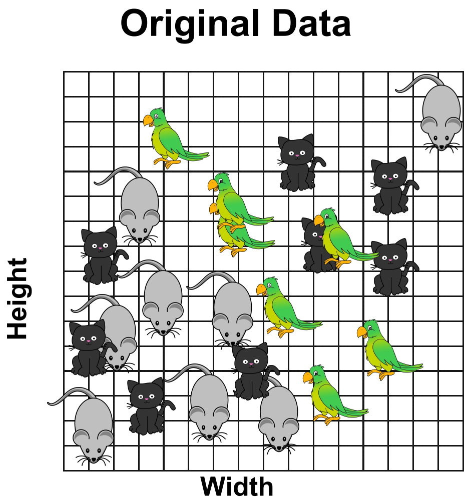
- Cat vs. Parrot
- Cat vs. Mouse
- Mouse vs. Parrot

Run SVM three different times

When an unknown observation comes in, we evaluate the point with each classifier.

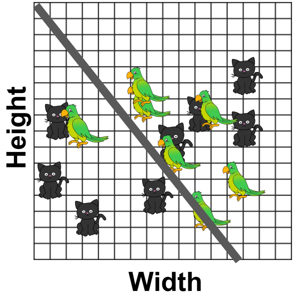
Do a majority vote as final prediction

The “One-vs-One” trick for Multi-Class SVMs

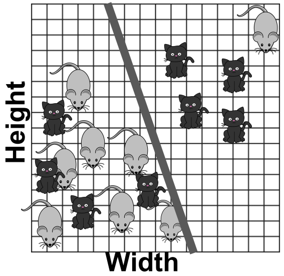


We must create three separate SVM models

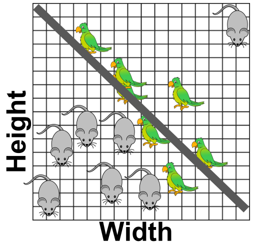
Cat vs Parrot



Cat vs Mouse



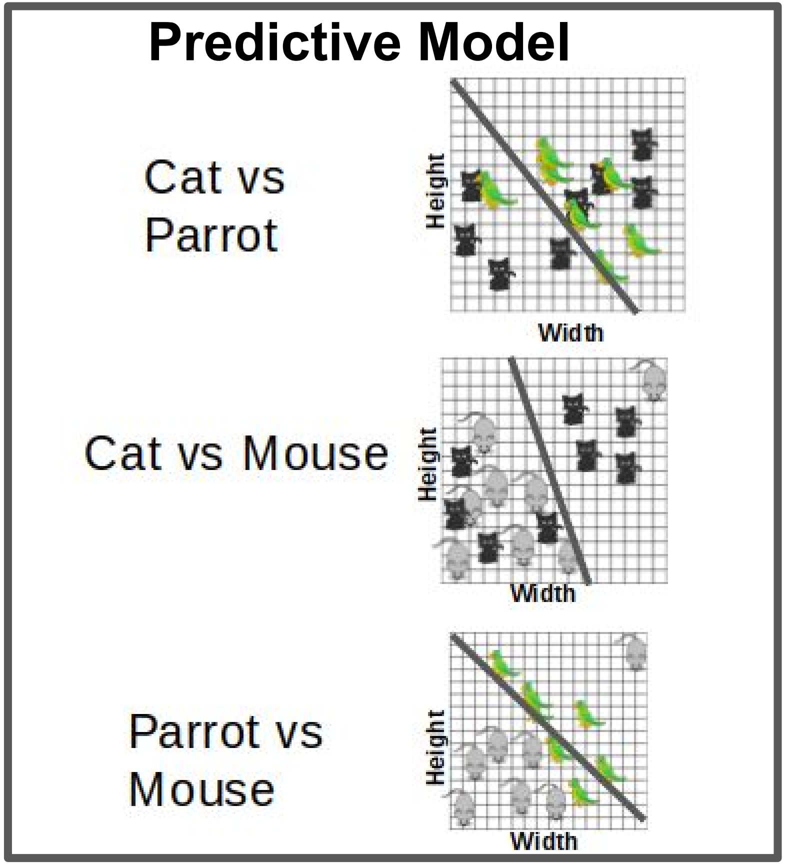
Parrot vs Mouse



The “One-vs-One” trick for Multi-Class SVMs

Our predictive model is composed of sub-models.

Three different SVM sub-models



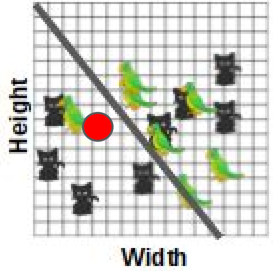
The “One-vs-One” trick for Multi-Class SVMs

What Species is this?

<4.2, 5.4>

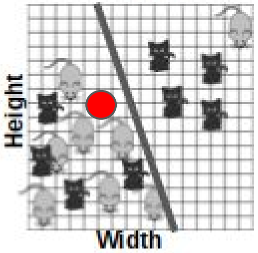
Predictive Model

Cat vs Parrot



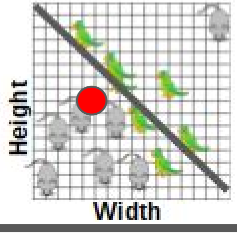
→ *Cat*

Cat vs Mouse



→ *Mouse*

Parrot vs Mouse



→ *Mouse*

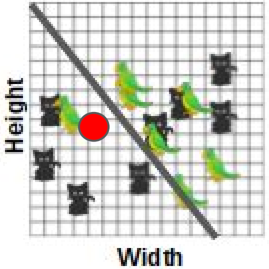
The “One-vs-One” trick for Multi-Class SVMs

What Species is this?

<4.2, 5.4>

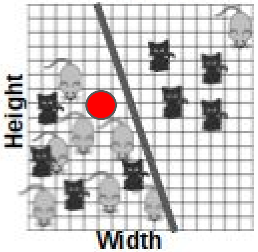
Predictive Model

Cat vs Parrot



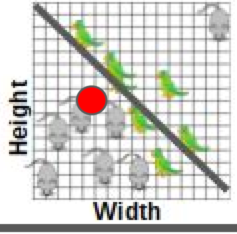
→ *Cat*

Cat vs Mouse



→ *Mouse*

Parrot vs Mouse



→ *Mouse*

→ *Mouse*

The “One-vs-One” trick: Summary

This is sort-of like cheating

But SVMs cannot handle multi-classes by themselves

The “One-vs-One” trick: Summary

SVM uses the One-vs-One trick for multi-class problems

Sub-models are built each possible class combination

Majority Vote afterwards for final prediction

R does this trick for us in the background

Outline for Today

Support Vector Machines - Another way to draw lines

Multi-class Support Vector Machines

Kernels and Support Vector Machines ←

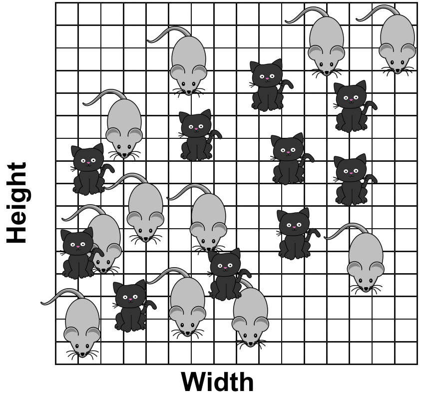
Support Vector Machines for Regression

Support Vector Machines

A support vector machine can find a **linear** decision boundary between two classes

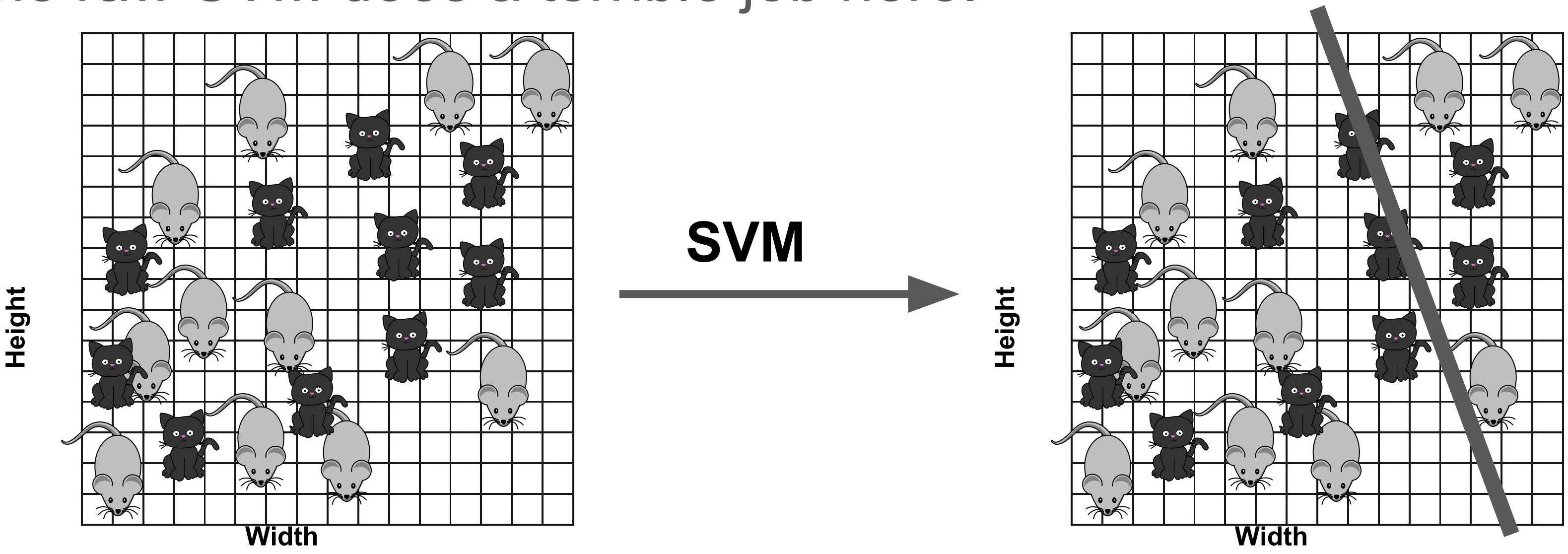
But what if the underlying function of our data is *non-linear*?

I.e. It is a curvy decision surface?



Support Vector Machines

The raw SVM does a terrible job here.



The underlying data cannot be linearly separated

Pre-processing: Non-Linear Kernels

To use the support vector classifier with non-linear data, there is only one twist to what we have seen earlier.

We have to “**pre-process**” the given data with a non-linear transformation function.

Hopefully after this transformation, the regular SVM will work properly.

Non-Linear Support Vector Machines

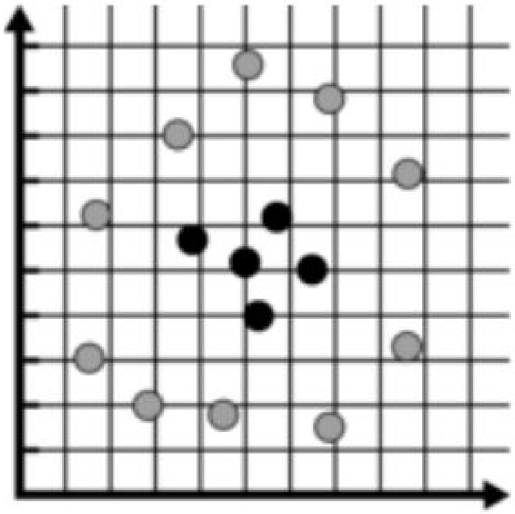
There are **four steps** now.

1. **Pre-process the data with a non-linear function**
2. Identify support vectors

3. Draw a line equidistant between both classes

4. **Project the line back onto original space**

Non-Linear Support Vector Machines

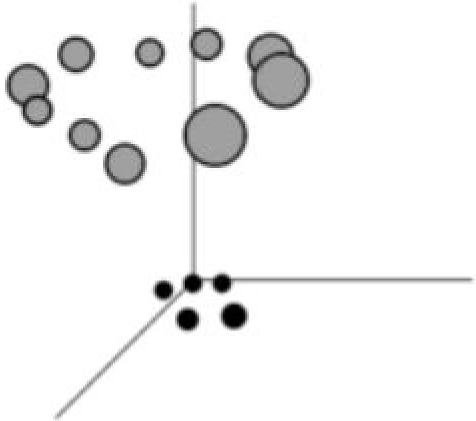


Our original data not linearly separable.

Using a plain SVM here would give us a predictor with terrible performance.

Non-Linear Support Vector Machines

Transform the given data using a **kernel function**.



We **hope** that after applying a non-linear kernel to the data, we can apply a regular SVM and get good accuracy

Non-Linear Support Vector Machines

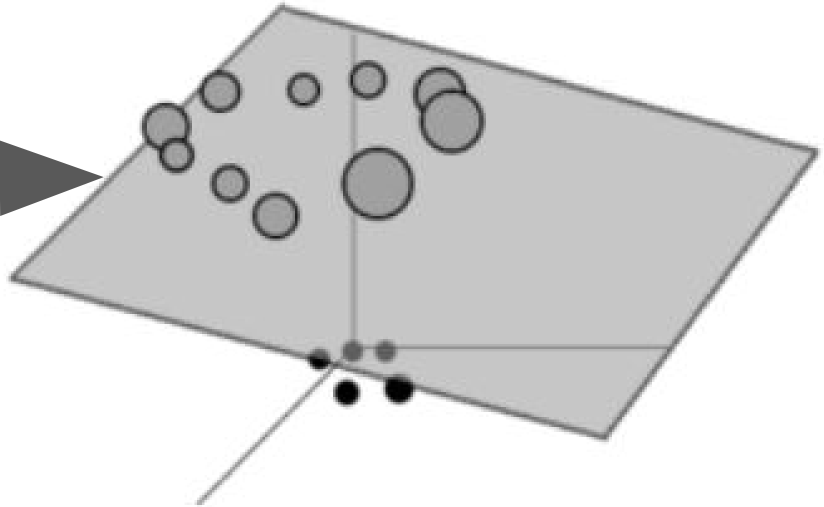
The support vector classifier is applied in transformed feature space

The line is drawn separating the two classes apart

Step 2 and Step 3

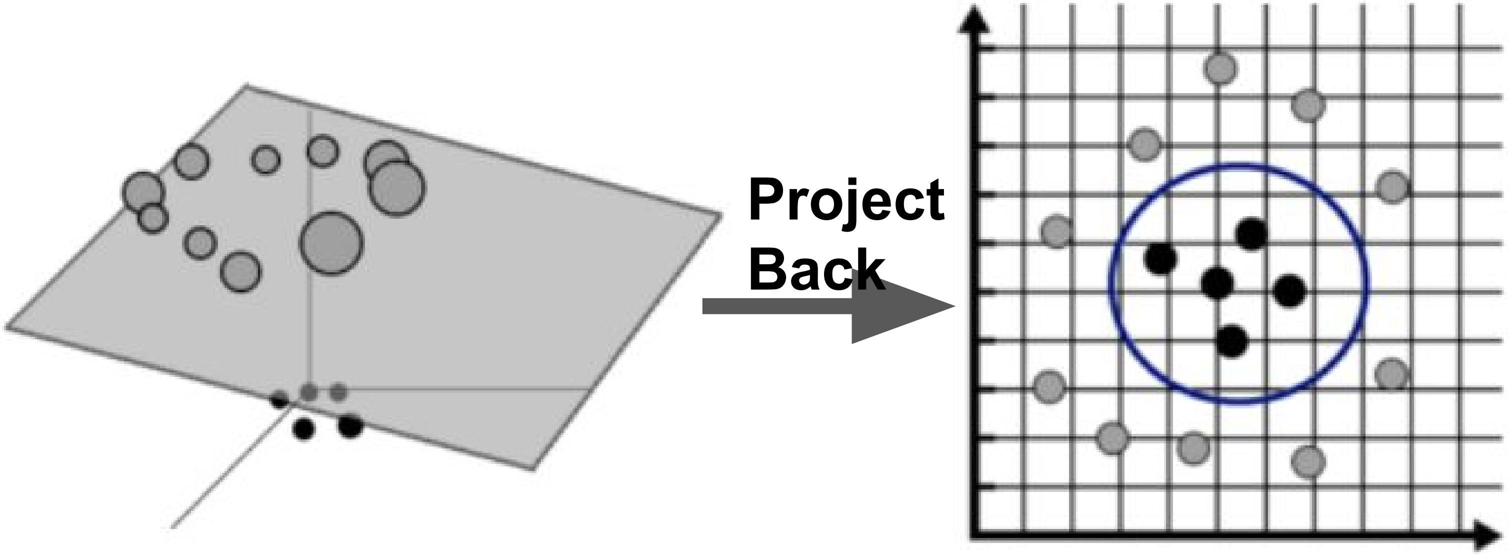
Apply the regular SVM in this transformed space.

Find the "middle" ground line



Non-Linear Support Vector Machines

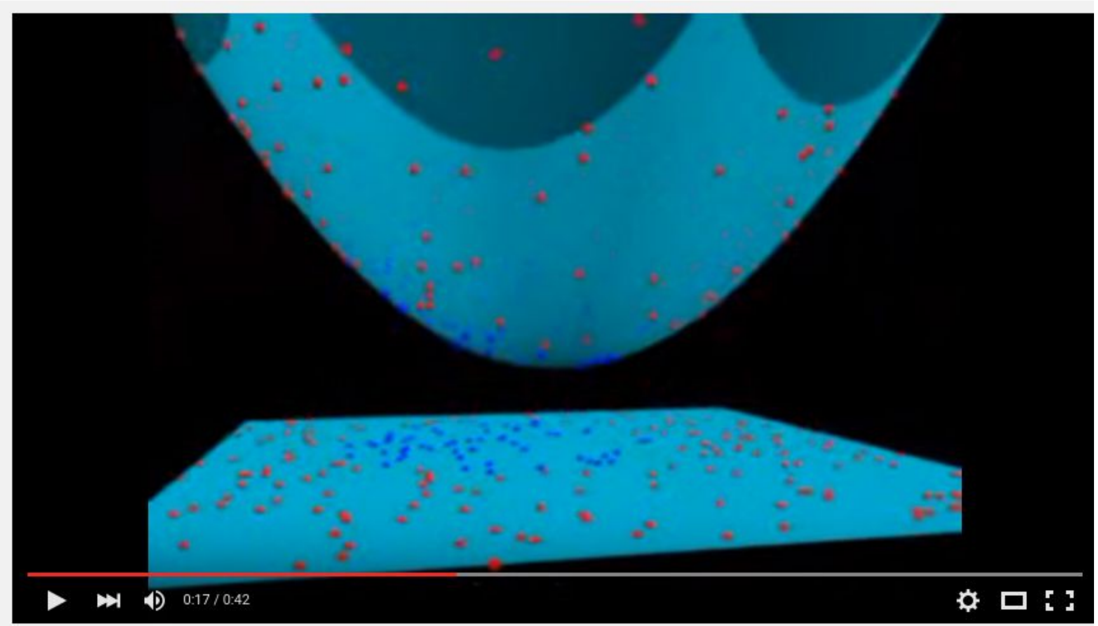
Step 4: Projecting the decision surface back onto our original feature space



We get a non-linear decision boundary

Non-Linear Support Vector Machines

This is an awesome video that gives better intuition on how kernels work.



Video: https://www.youtube.com/watch?v=3liCbRZPrZA

Outline for Today

Support Vector Machines - Another way to draw lines

Multi-class Support Vector Machines

Kernels and Support Vector Machines

Support Vector Machines for Regression ←

Support Vector Machines for Regression

We talked about SVM for **classification**

Building predictive models that predict categories

What about for **regression**?

Building predictive models that predict numbers

Support Vector Machines for Regression

Basically the same procedure.

<u>Step 1.</u> Identify Support Vectors

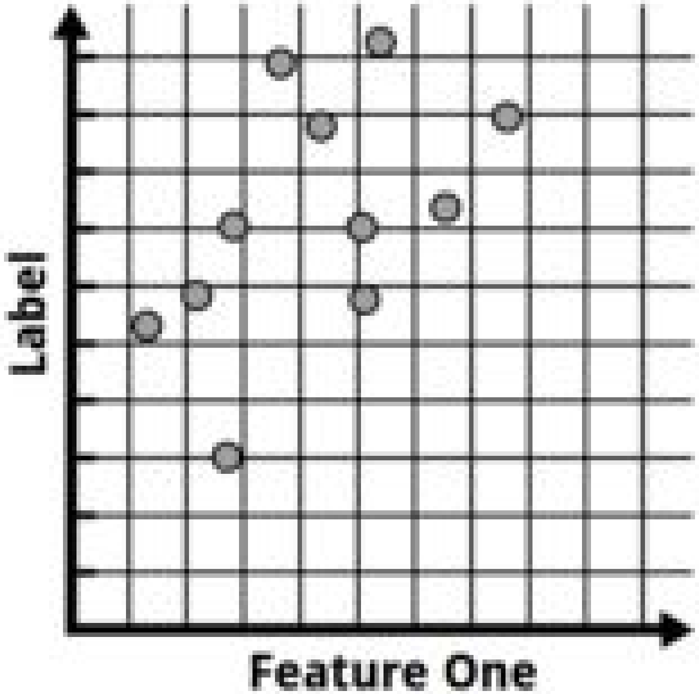
<u>Step 2.</u> Draw "Middle" Line

* Maybe a few more steps if using a kernel transformation

But now the **<u>support vectors</u>** are "outside points" of our entire data

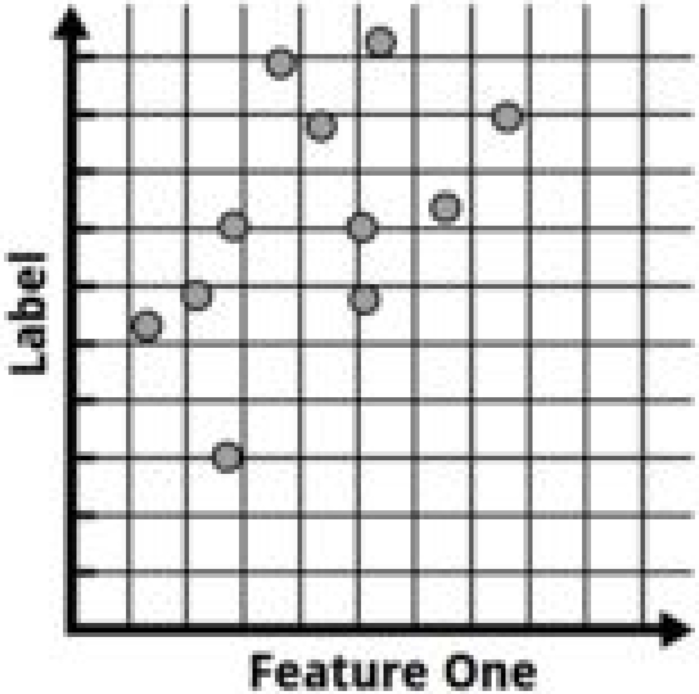
Support Vector Machines for Regression

This is our data set that we are given.



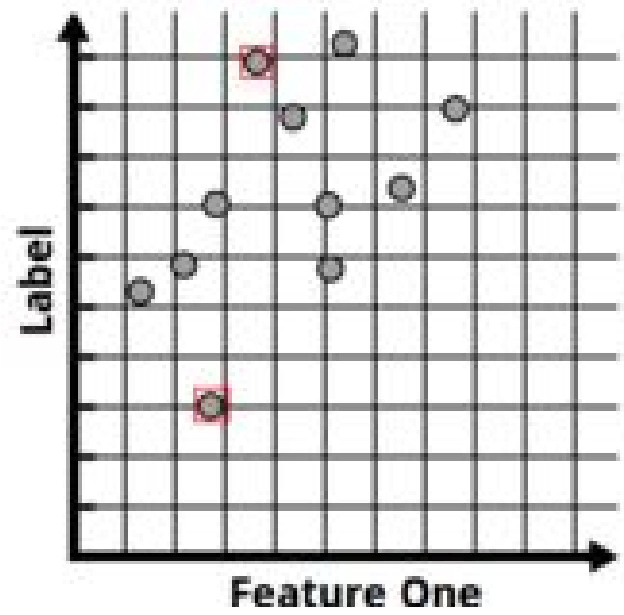
Support Vector Machines for Regression

This is our data set that we are given.



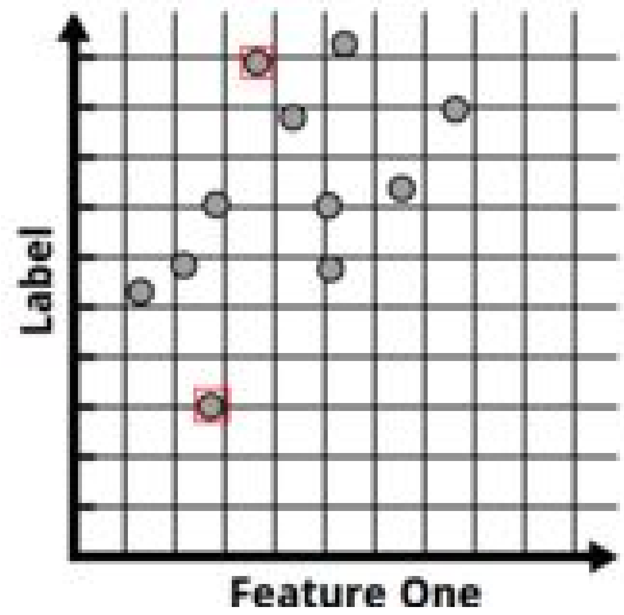
Support Vector Machines for Regression

Step 1: Identify Support Vectors (Outside Points)



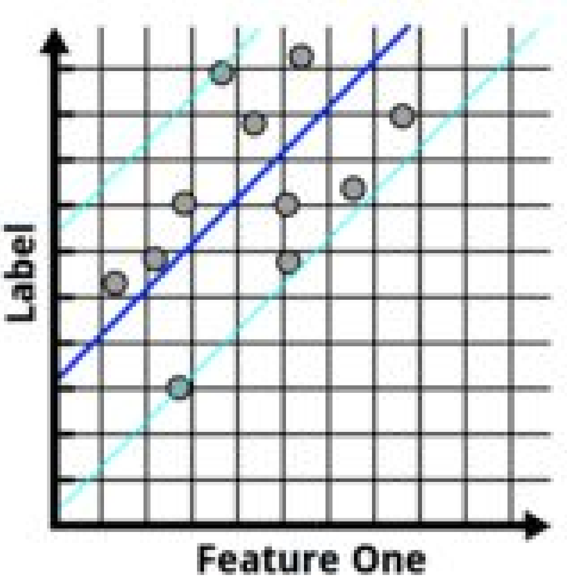
Support Vector Machines for Regression

<u>Step 1</u>: Identify Support Vectors (Outside Points)

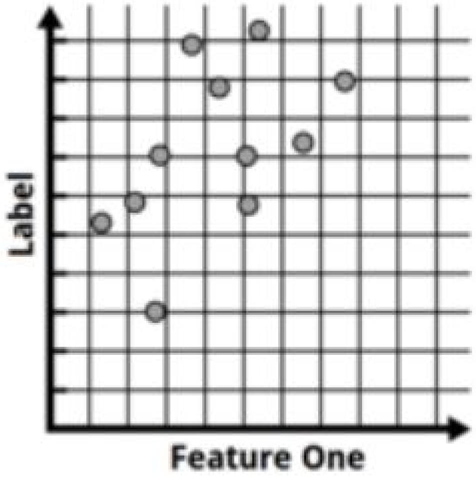


Support Vector Machines for Regression

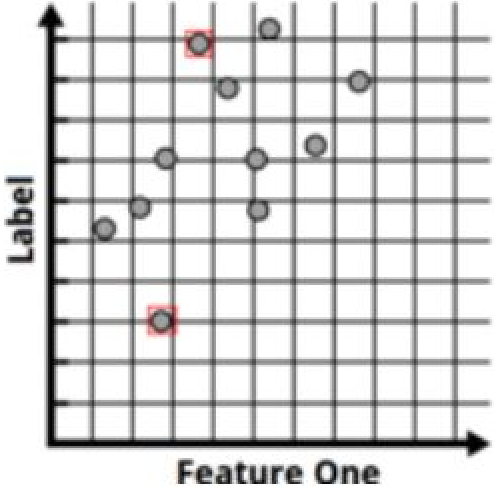
Step 2: Draw line equidistant from support vectors



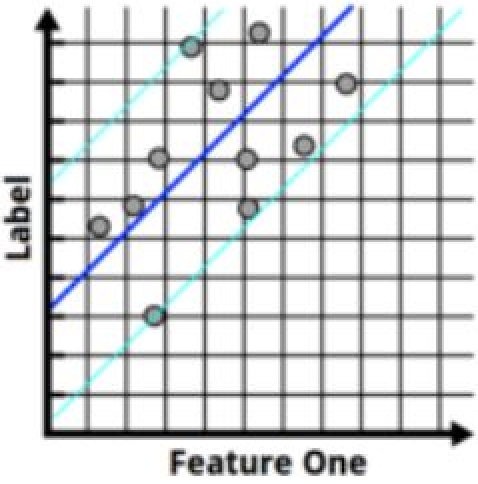
Support Vector Machines: For Regression



(n) 1. Our raw data where our feature is plotted against the numerical label.



(o) 2. Firstly, we identify the support vectors (which are the observations that lie on the outer surface of our data).



(p) 3. The optimal hyperplane (our regression function) that maximizes the distances between the support vectors is drawn.

Outline for Today

Support Vector Machines - Another way to draw lines

Multi-class Support Vector Machines

Kernels and Support Vector Machines

Support Vector Machines for Regression

Assignment 2 Review

Any questions/confusions/worries?

Lemme know! =)

Link: http://web.cs.dal.ca/~kallada/stat2450/assignments/Assignment2.pdf

That’s all for today

Assignment 2 is due next Tuesday!

I will be away next <u>Monday</u>.