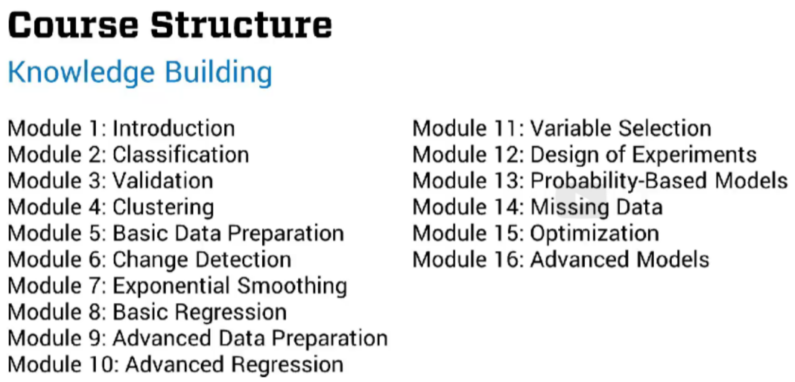
ISYE6501x Mid Term Quiz 1 Revision

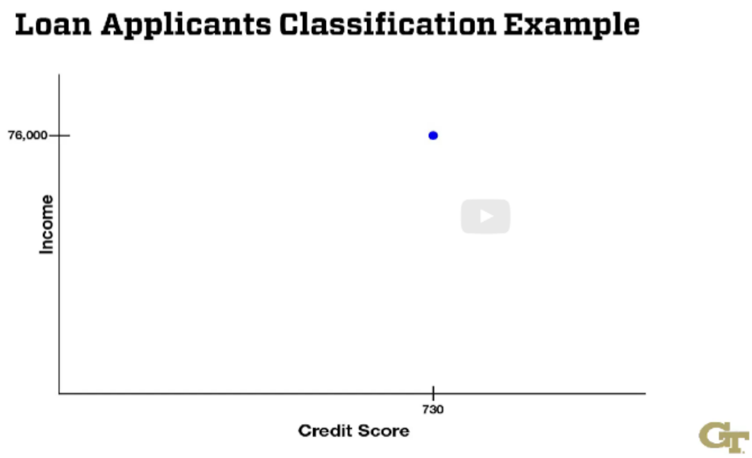


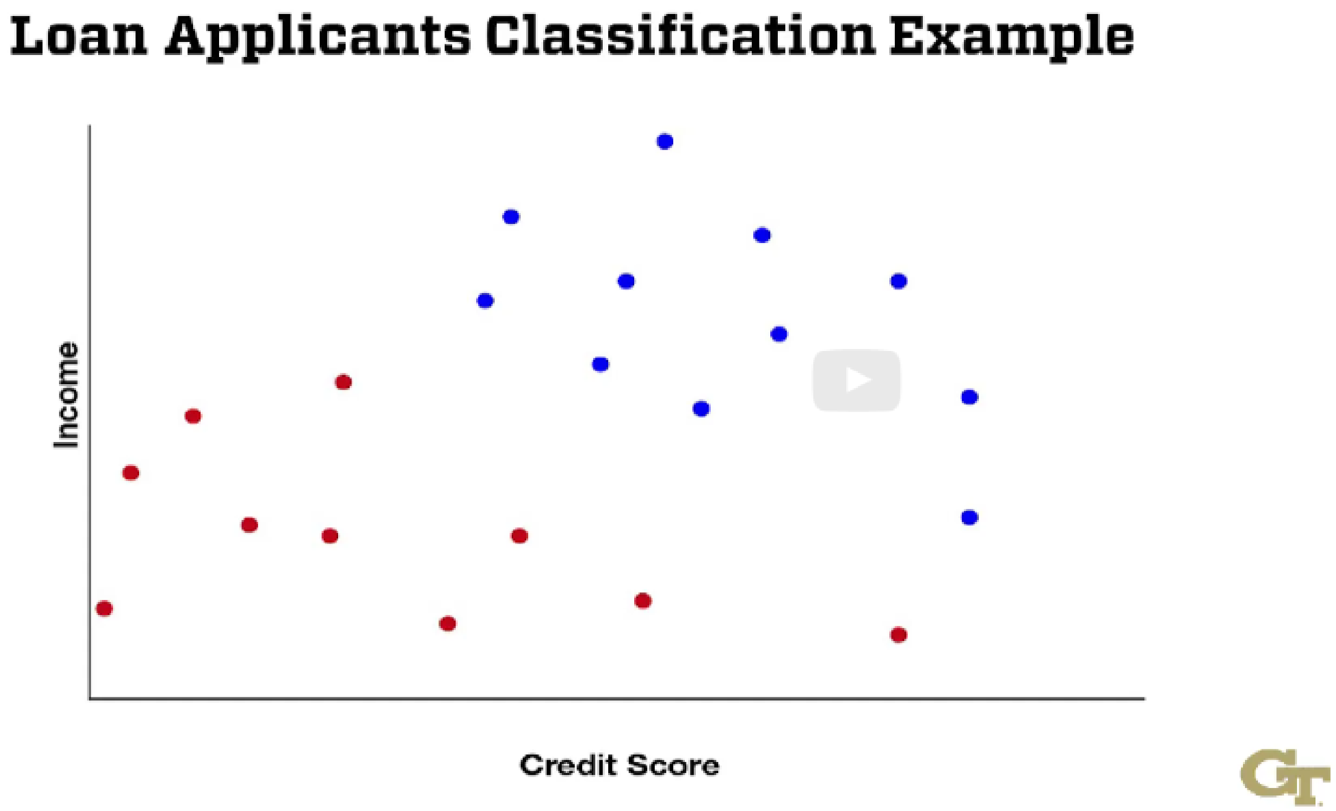
Module 1: Introduction

Nothing much here. Just course introductions.

Module 2: Classification

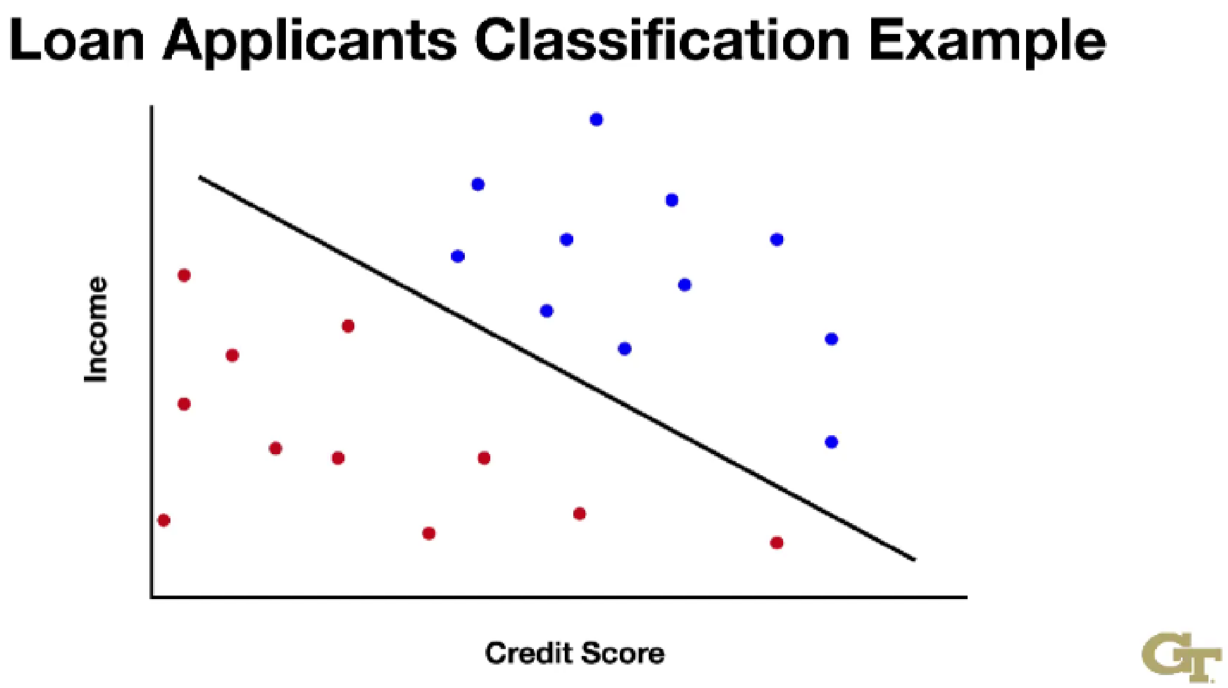
Lesson 2.1: Introduction to Classification





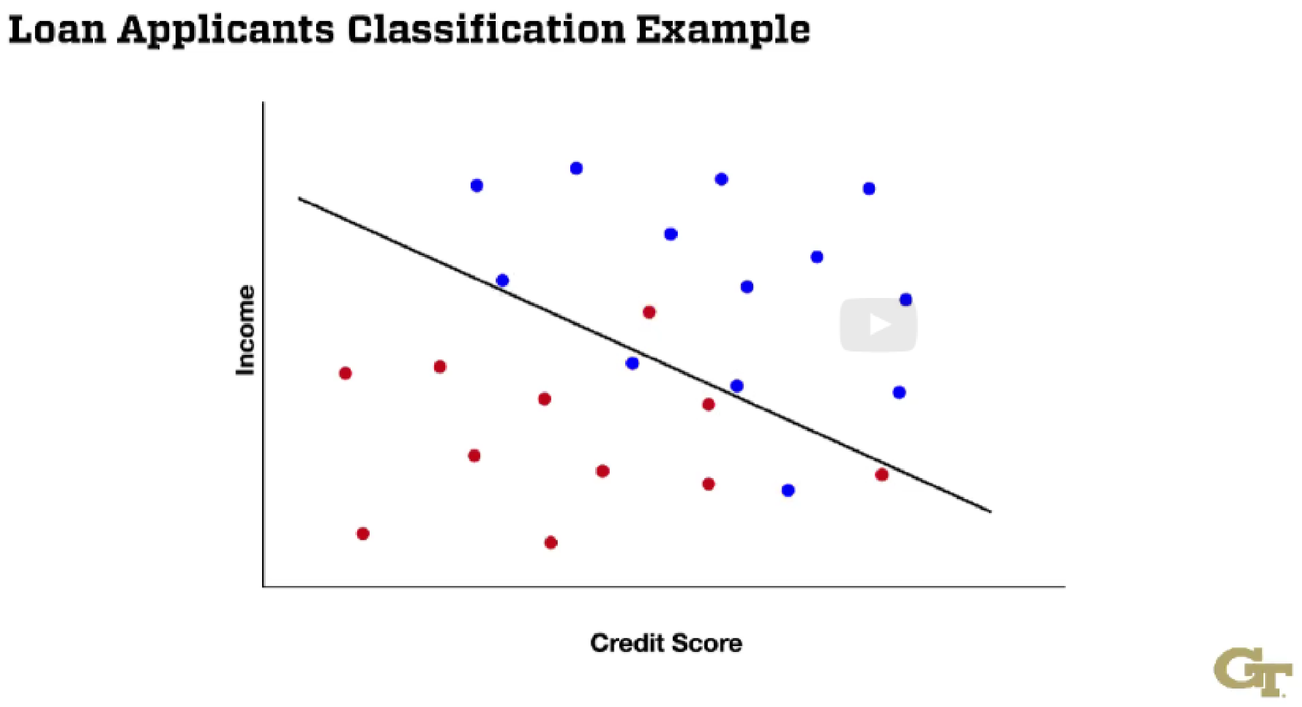
Blue - Loan Repaid
Red - Defaulted

Lesson 2.2 (M): Choosing a Classifier



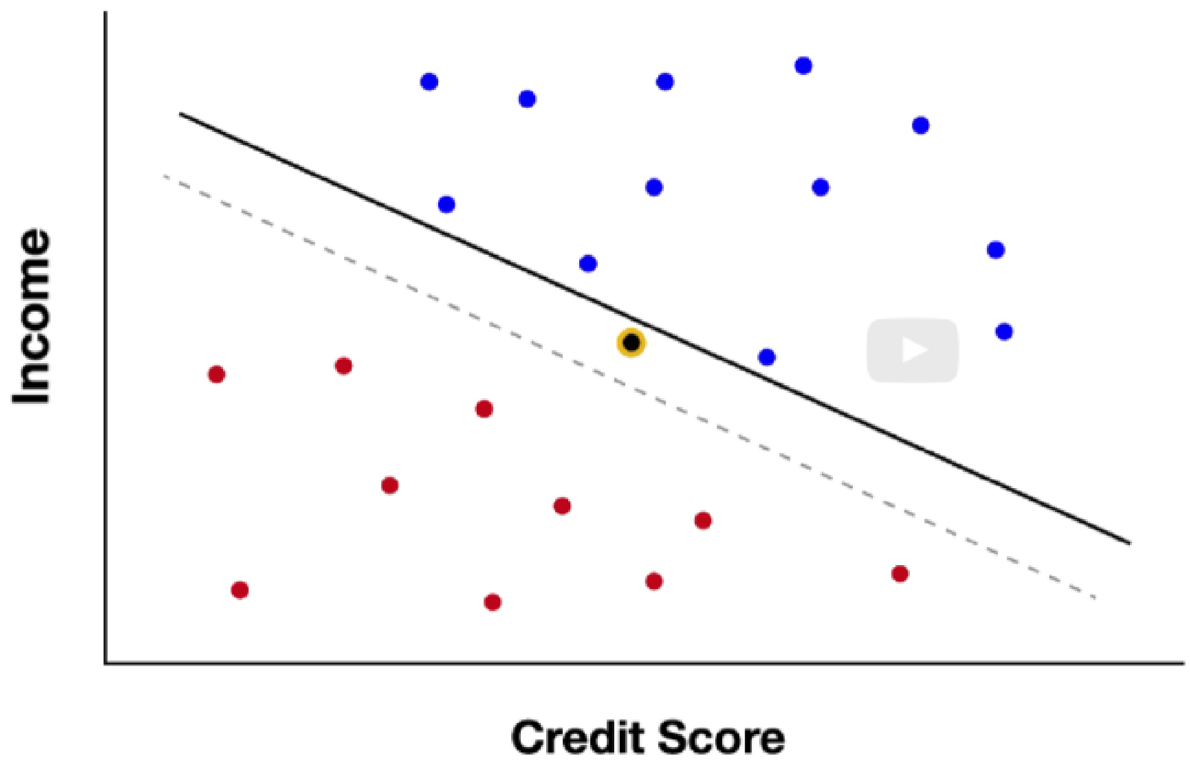
We can see where the new applicant's data is relative to the line, and classify it accordingly.

A Soft Classifier is used when you cannot perfectly separate the points.



In most cases we are unable to perfectly separate the two classes. So we pick a (soft) classifier that minimizes the number of incorrectly classified points.

We have to weigh the cost of actual mistakes and near mistakes.



Let’s say that the cost of making a bad loan is twice as high as the cost of turning away a good loan, we should shift the line so it is closer to the blue points than to the red points.

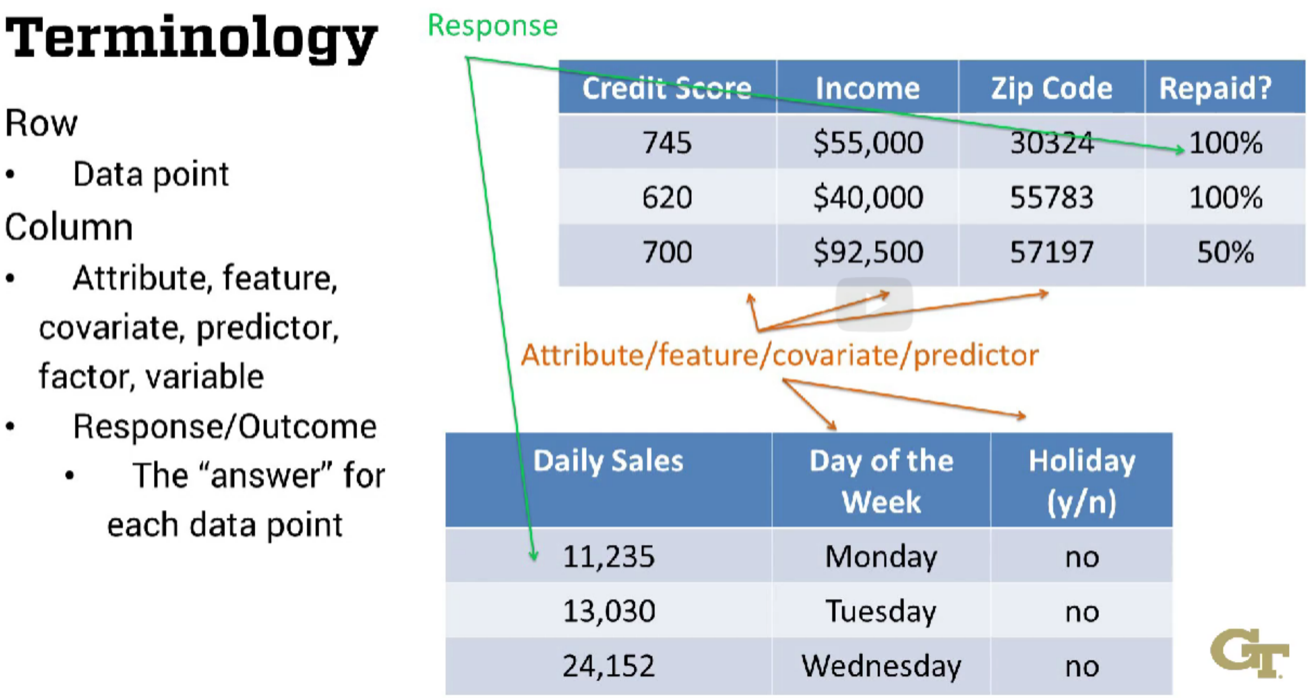
Given that realistically it is impossible to separate with no mistakes, we might be more willing to accept one type of mistake than another.

Lesson 2.3 (C): Data Definitions

Row: Data Point (A data point is all the information about one observation)

Column:

- Attribute, feature, covariate, predictor, factor, variable
- Response/Outcome (the "answer" for each data point



Structured Data

Data that can be stored in a structured way

- Quantitative: credit score, age, sales, etc
- Categorical: M/F, Hair Colour, etc

Example: The amount of money in a person’s bank account

Unstructured Data

- data not easily described and stored
- example: Written text

Example: The contents of a person’s Twitter feed

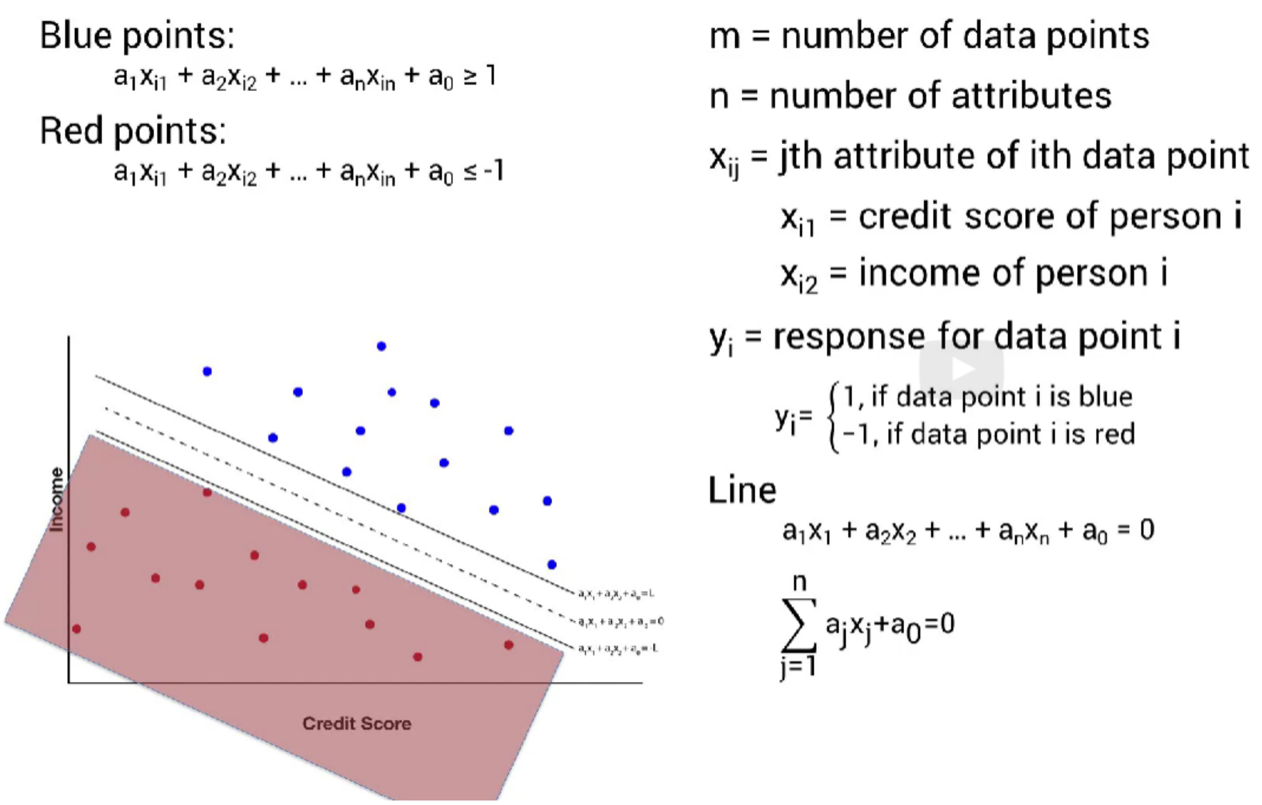
Time Series Data

- same data recorded over time
- often recorded in equal intervals (doesn't have to be)
- Eg: Daily sales, stock prices, child's height on each birthday, The average cost of a house in the United States every year since 1820

Lesson 2.4 (M): Support Vector Machines (SVM)

SVM is a type of Classification Models

Extra reading on SVM Classifier ([http://pyml.sourceforge.net/doc/howto.pdf (http://pyml.sourceforge.net/doc/howto.pdf)](http://pyml.sourceforge.net/doc/howto.pdf))



We want to find values of a_0, a_1 up to a_n that classify the points correctly and have the maximum gap or margin between the parallel lines.

Since we defined y_i to be 1 for blue points and negative 1 for red points, we can combine these two expressions to get the following:

All points:

$$(a_1x_{i1} + a_2x_{i2} + \ldots + a_nx_{in} + a_0)y_i \geq 1$$

The above inequality will hold true in the case of a correct classification, ie. when a data point is on the correct side of the line.

We need to **maximise** the margin of separation (distance) between both parallel lines in the classifier, which means the following:

Distance between solid lines

$$= \frac{2}{\sqrt{\Sigma_j (a_j)^2}} \text{ So, } \underset{a}{\text{Minimize}} \sum_j (a_j)^2$$

The above is basically the Euclidean (Orthogonal) Distance between the two parallel lines.

$$\underset{a_0,\ldots,a_n}{\text{Minimize}} \sum_{j=1}^{n} (a_j)^2$$

Subject to

$$(a_1 x_{i1} + a_2 x_{i2} + \ldots + a_n x_{in} + a_0) y_i \geq 1$$

for each data point i

As mentioned above, the following inequalities will hold true:

Correct side of the line:

$$\left(\sum_{j=1}^{n} a_j x_{ij} + a_0 \right) y_i - 1 \geq 0$$

Wrong side of the line:

$$\left(\sum_{j=1}^{n} a_j x_{ij} + a_0 \right) y_i - 1 < 0$$

The error for the data point i is as follows:

Error for data point i:

$$\max\left\{0, 1 - \left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i\right\}$$

The total error we want to minimise can be written as the sum over all data points i of the following:

Total error:

$$\sum_{i=1}^{m} \max\left\{0, 1 - \left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i\right\}$$

We experience a tradeoff between the **ERROR** and **MARGIN** as can be seen below:

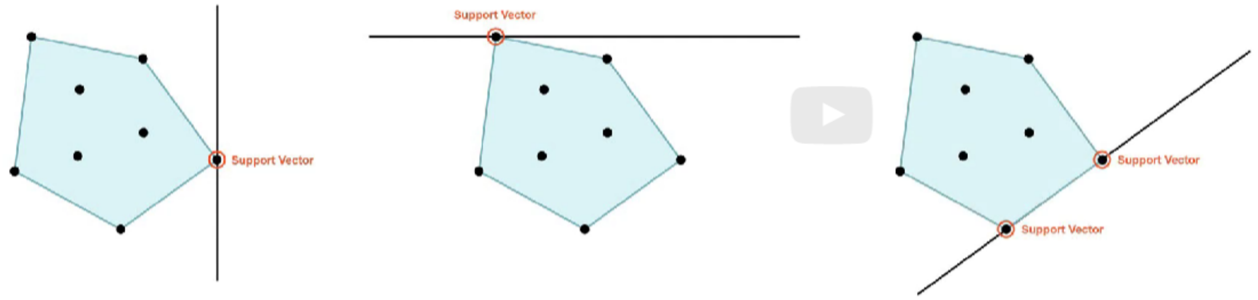
$$\underset{a_0,\ldots,a_n}{\text{Minimize}} \sum_{i=1}^{m} \max\left\{0, 1 - \left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i\right\} + \lambda \sum_{j=1}^{n} \left(a_j\right)^2$$

We can pick a value of Lambda (durin hyperparameter tuning) and minimise the combination of error minus margin.

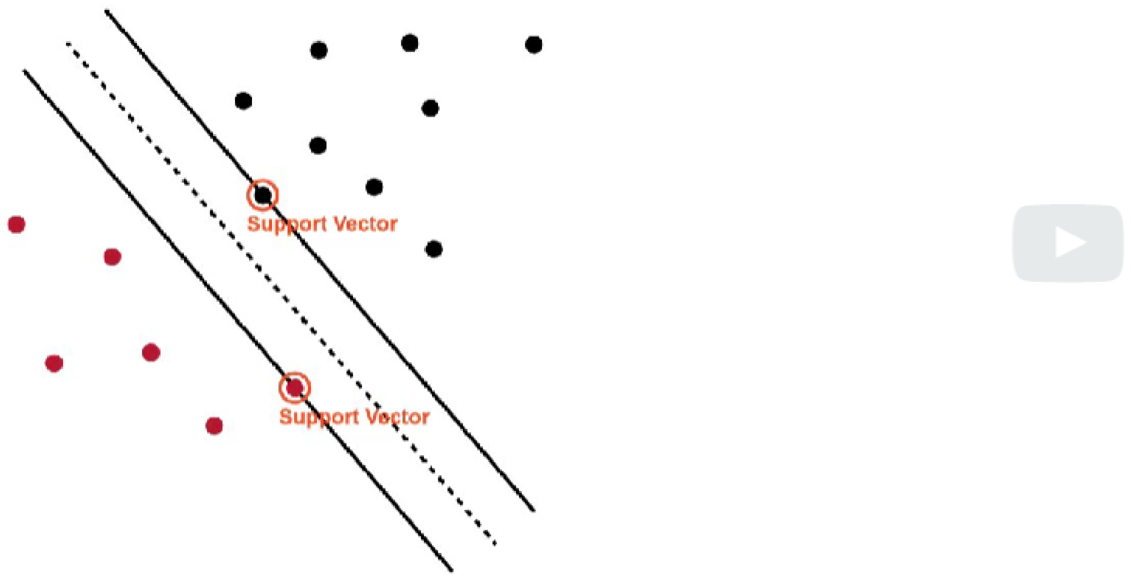
Question: In Lesson 2.4

- why did they say at 6:09 that "the margin we want to maximise is the sum of a_ij squared"? Shouldn't it be that when we want to maximise the margin, we should then minimise a_ij squared?
- "as lambda gets large, this term gets large, so the importance of a larger margin outweighs avoiding mistakes in classifying known data points" isn't the sum of a_ij just the denominator and not the actual distance between the two parallel lines?
- it seems to contradict what is said in Lesson 2.6

Lesson 2.5 (M): SVM: What the Name Means



- Point that holds up shape = support vector
  - Support vectors can support sides, top, etc.



- Support Vector Machine model
  - Determines "support vectors"
  - Automatically from data (hence, "machine")

The **classifier** it returns is actually not one of the lines touching a support vector.

Lesson 2.6 (M): Advanced SVM

Hard Margin

$$\underset{a_0,\ldots,a_m}{\text{Minimize}} \sum_{i=1}^{m} (a_i)^2$$

Subject to

$$(a_1x_1 + a_2x_2 + \ldots + a_mx_m + a_0)y_j \geq 1$$

for each data point i

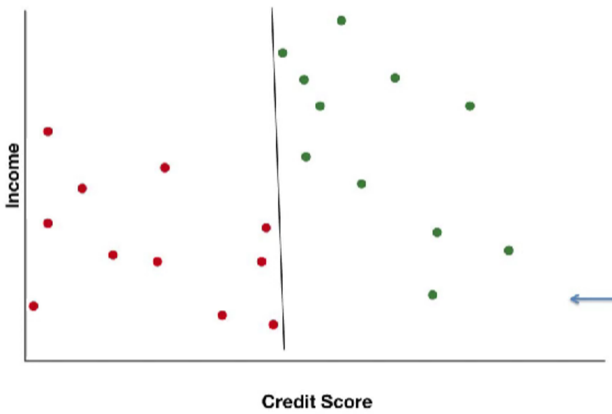
Soft Margin

Which trades off reducing errors and enlarging the margin

$$\underset{a_0,\ldots,a_m}{\text{Minimize}} \quad \sum_{j=1}^{n} \max\left\{0, 1 - \left(\sum_{i=1}^{m} a_i x_{ij} + a_0\right) y_j\right\} + \lambda \sum_{i=1}^{m} (a_i)^2$$

Classification:

Support Vector Machines



$$a_1x_1 + a_2x_2 + \ldots + a_mx_m + a_0 = 0$$

Near-zero coefficients are probably not relevant for classification

Income is not helpful for classifying