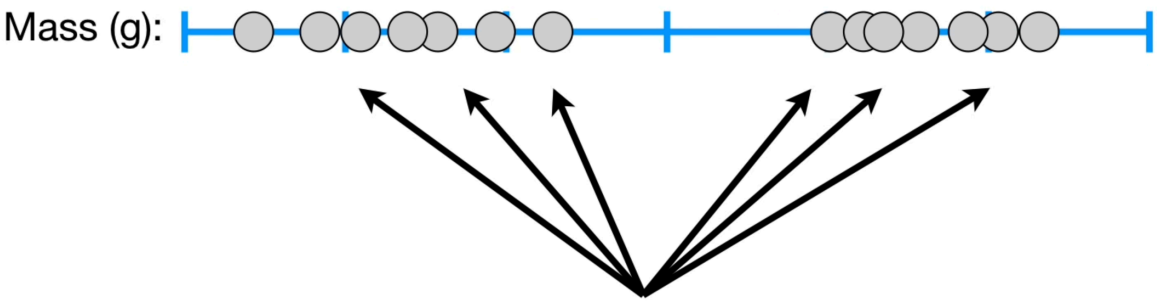
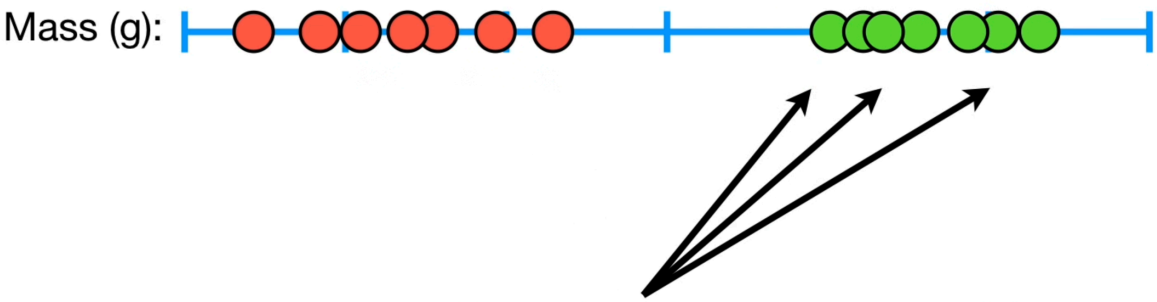
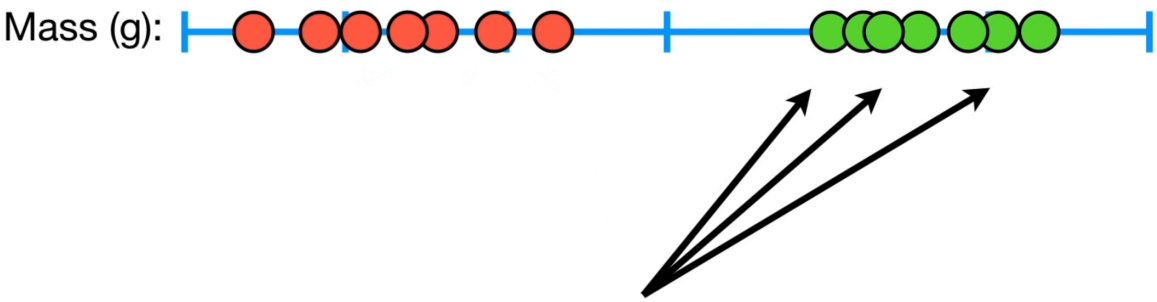


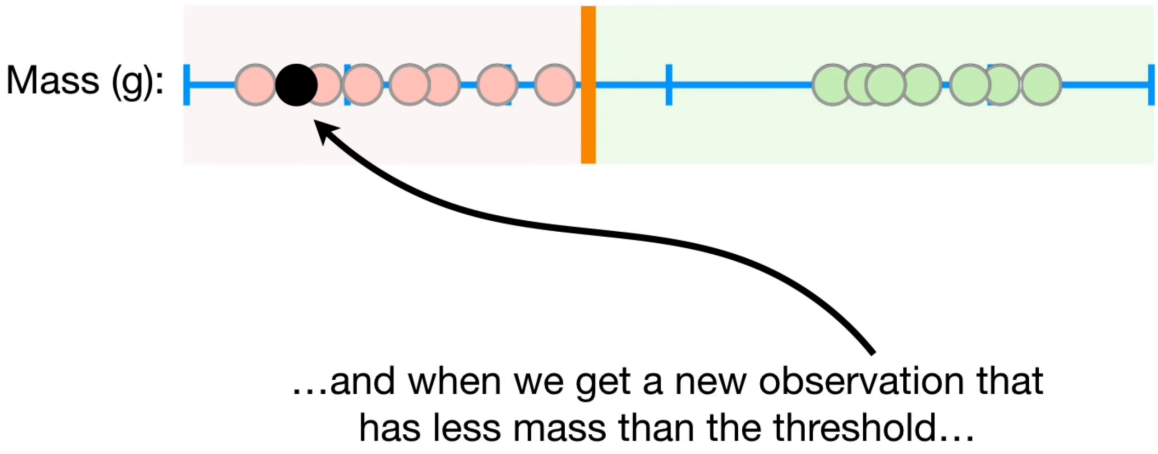
Let's start by imagining we measured
the mass of a bunch of mice…

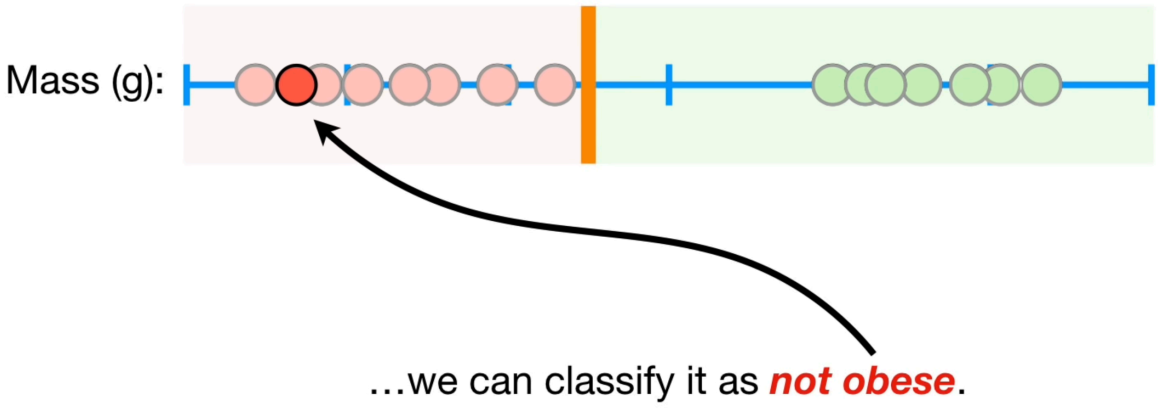


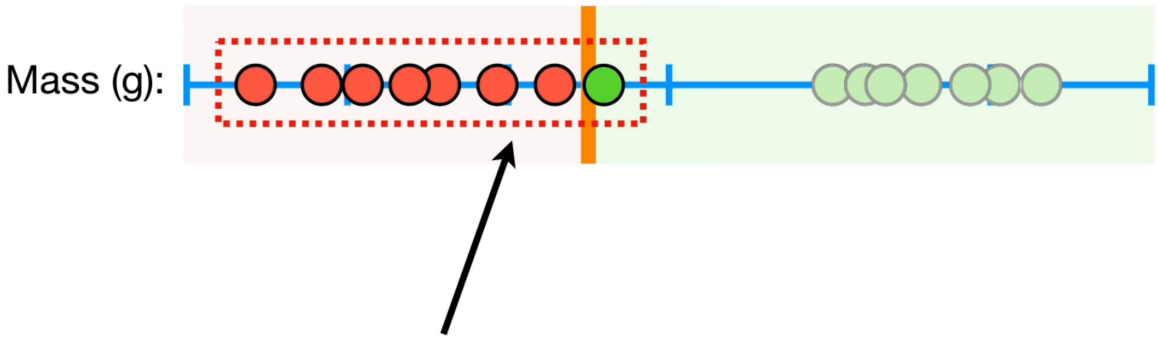
…and the **green dots** represent mice are ***obese***.



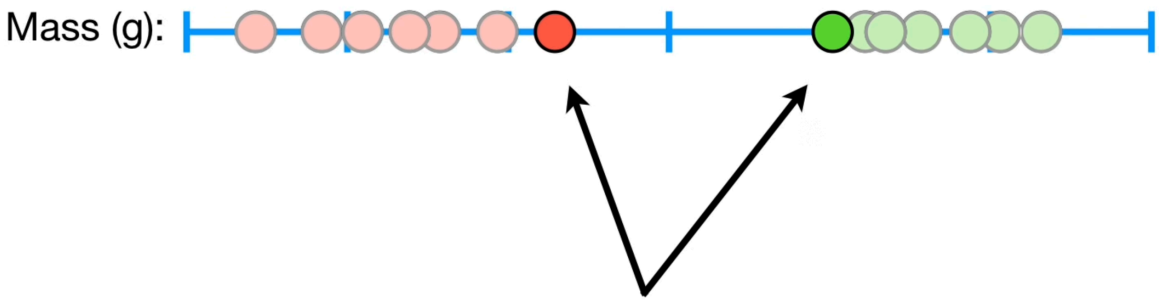
…and the **green dots** represent mice are ***obese***.


Mass (g):
…and when we get a new observation that
has less mass than the threshold…

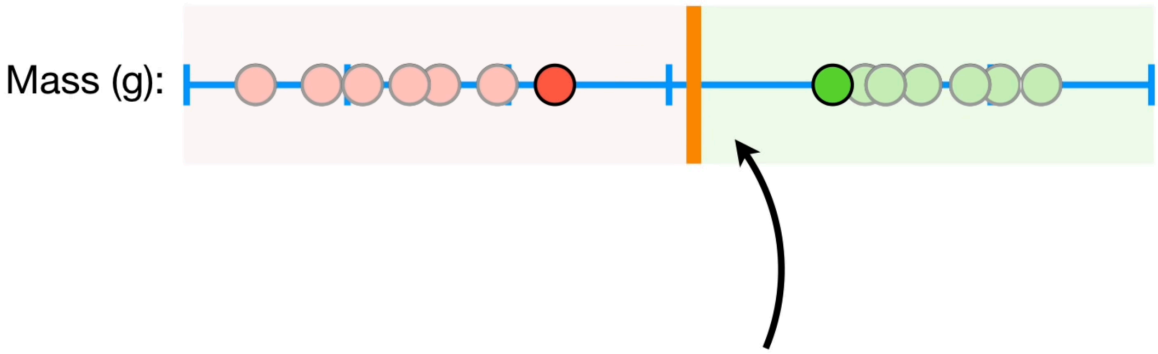

Mass (g):
…we can classify it as not obese.

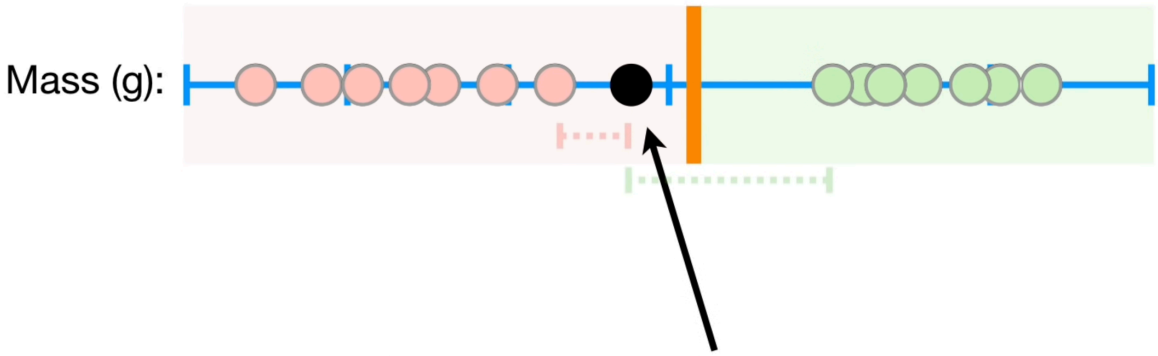


But that doesn't make sense, because it is much closer to the observations that are ***not obese***.

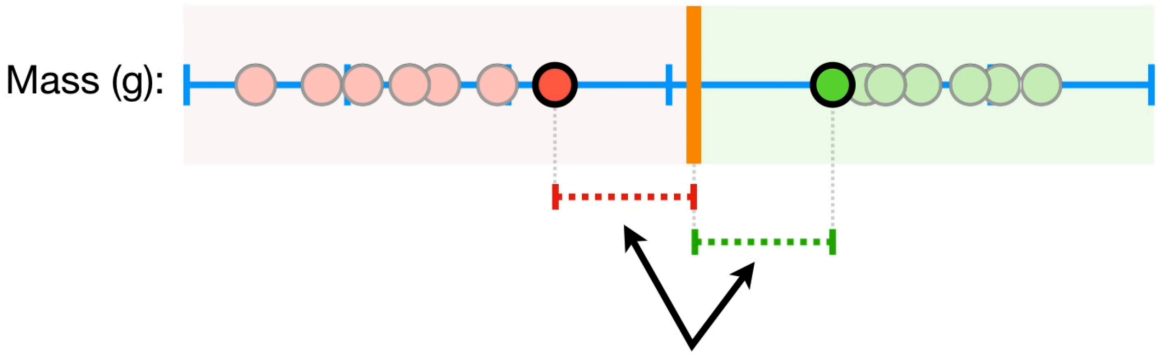


…we can focus on the observations on the edges of each cluster…

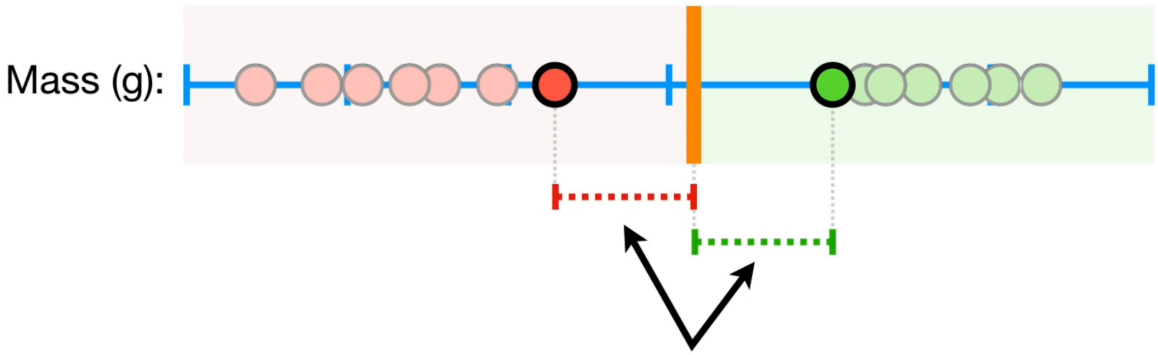

Mass (g):
…and use the midpoint between
them as the threshold.



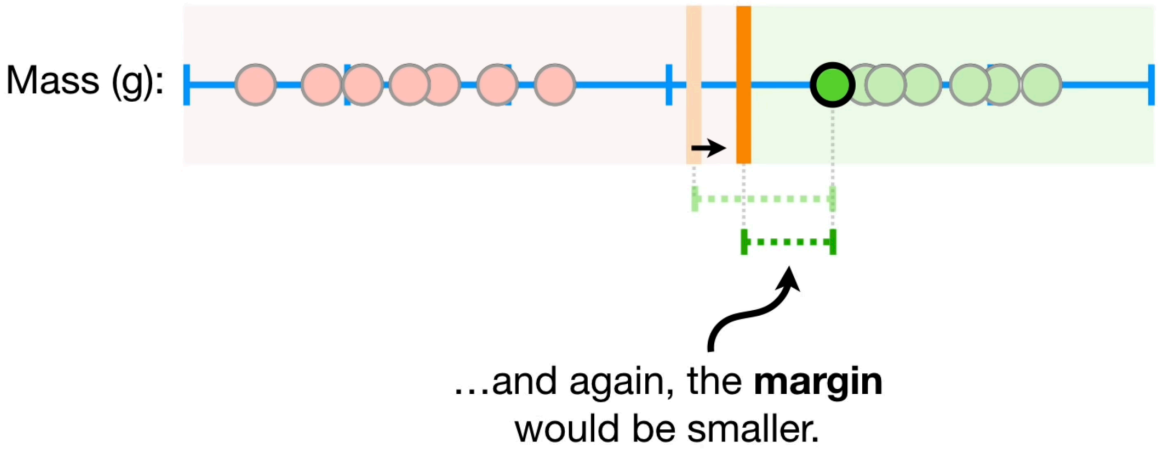
So it makes sense to classify this new observation as ***not obese***.

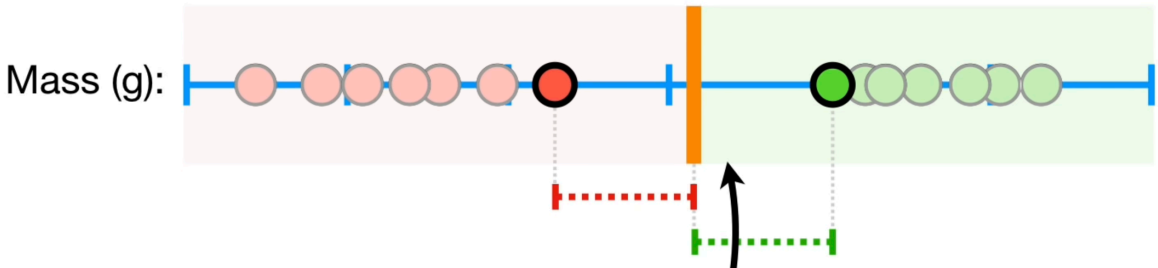


The shortest distance between the observations and the threshold is called the **margin**.

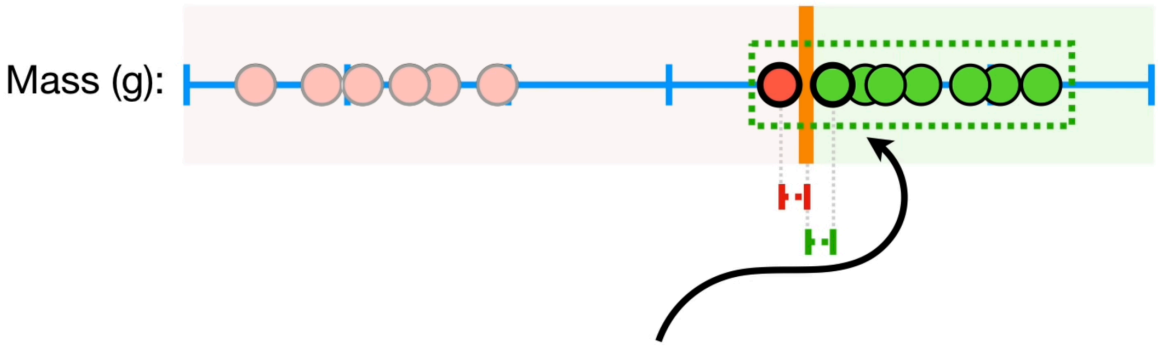


…the distances between the observations and the threshold are the same and both reflect the **margin**.


Mass (g):
…and again, the **margin**
would be smaller.

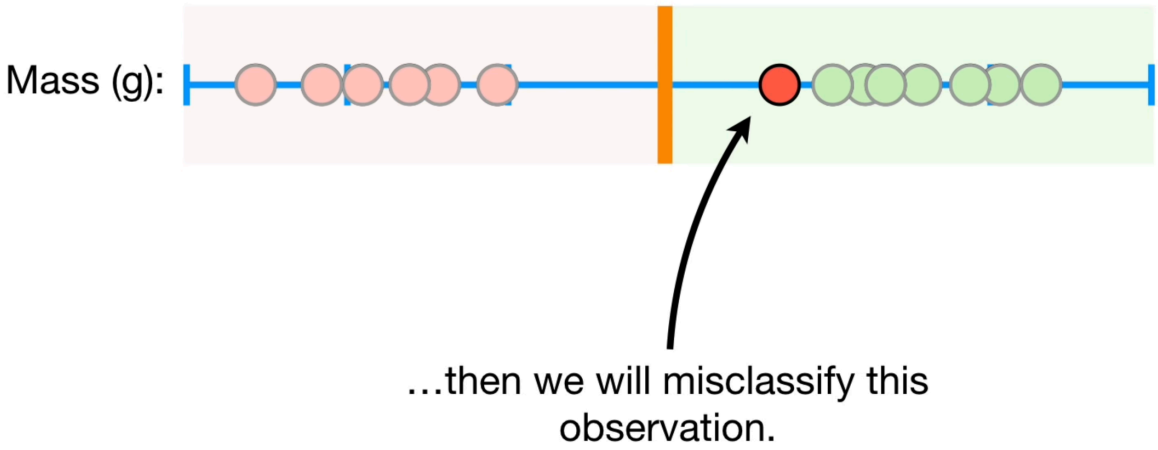


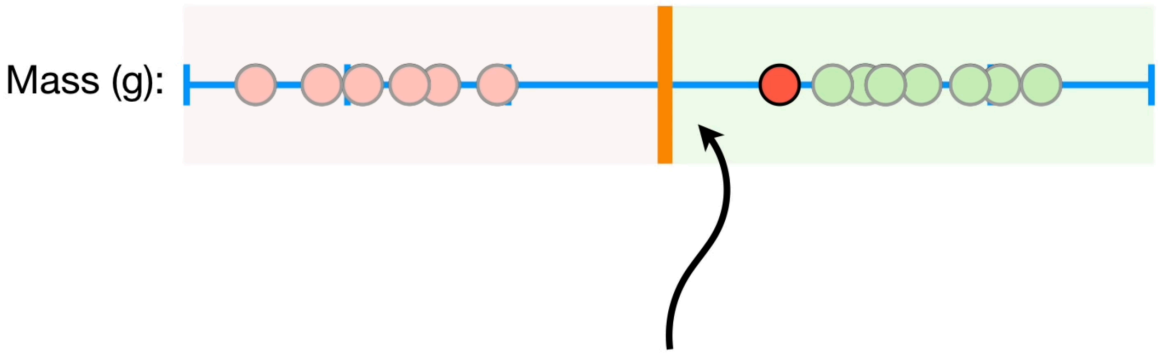
Maximal Margin Classifiers
seem pretty cool…



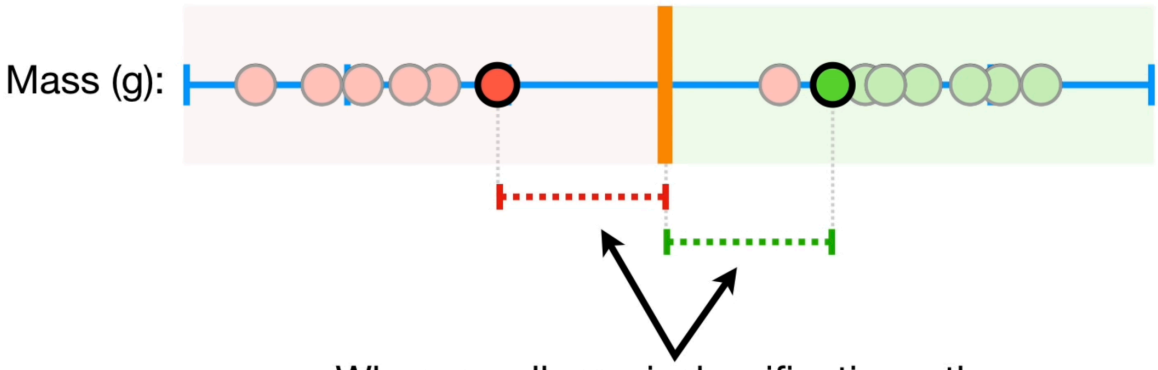
In this case, the **Maximum Margin Classifier** would be super close to the ***obese*** observations…

To make a threshold that is not so sensitive to outliers we must **allow misclassifications**.

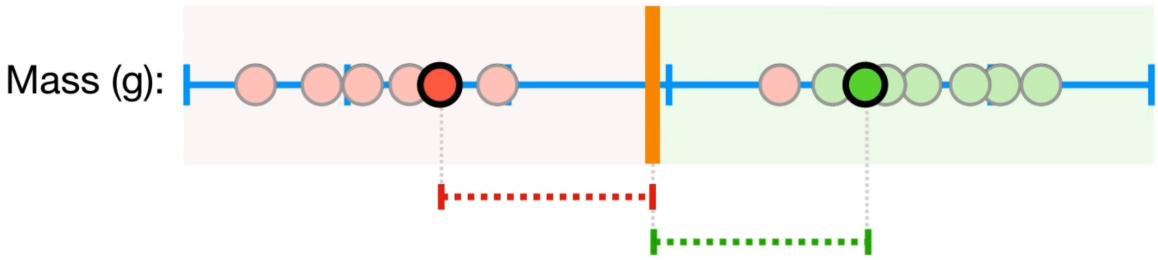

Mass (g):
…then we will misclassify this observation.



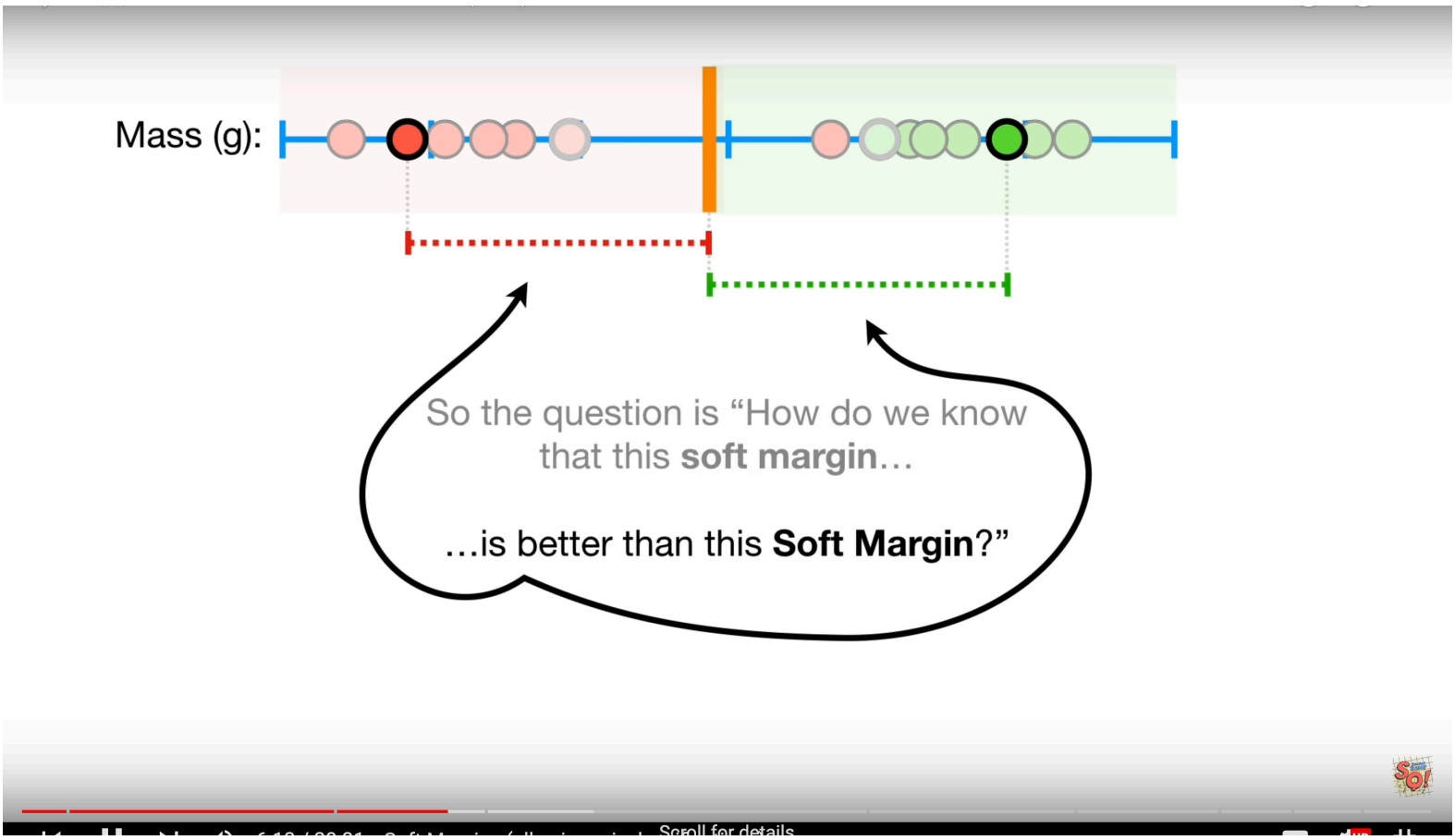
Choosing a threshold that allows misclassifications is an example of the **Bias/Variance Tradeoff** that plagues all of machine learning.

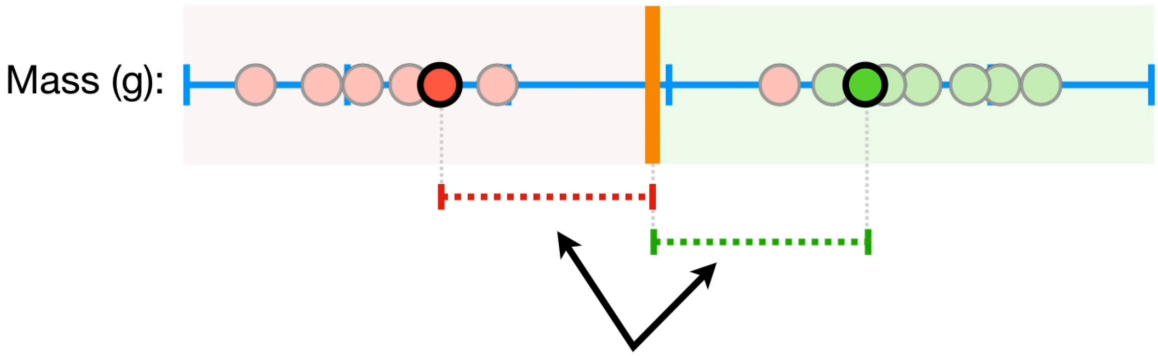


When we allow misclassifications, the distance between the observations and the threshold is called a **Soft Margin**.

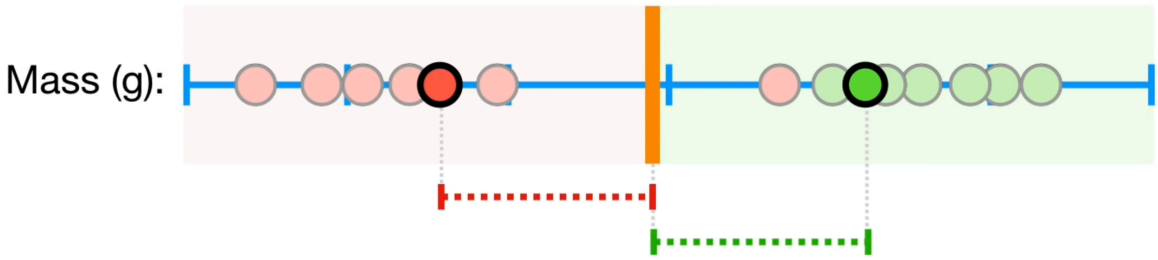


The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.

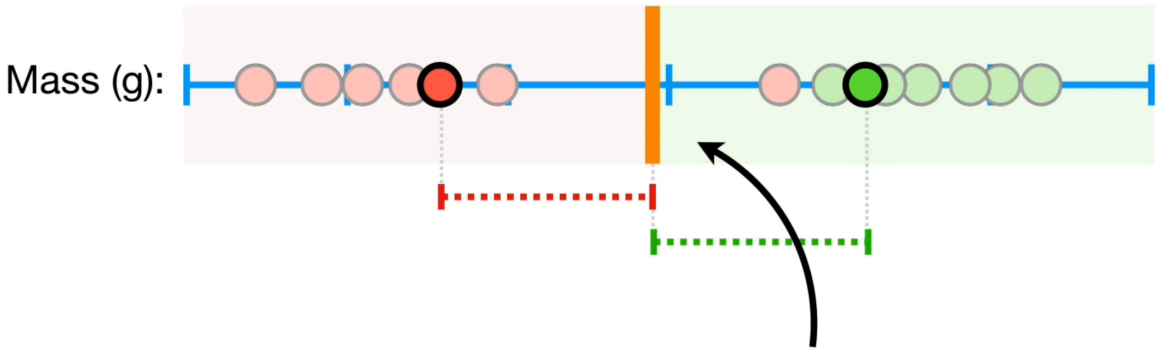

Mass (g):
So the question is “How do we know that this **soft margin**…
…is better than this **Soft Margin**?”



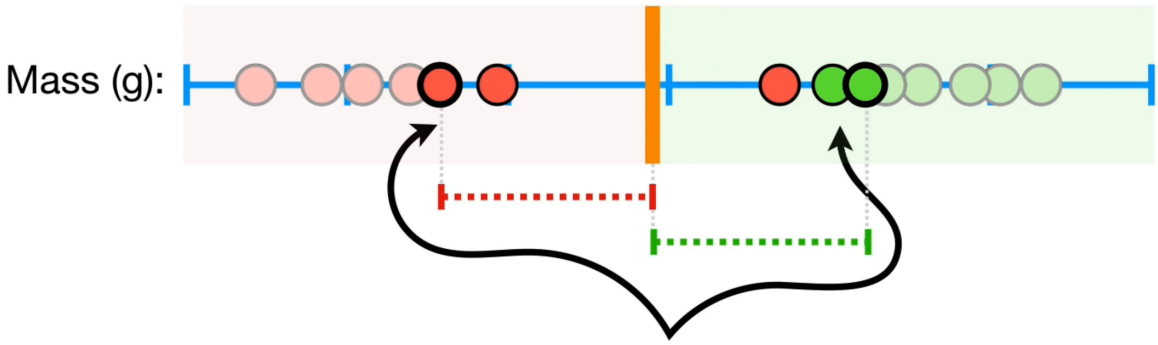
For example, if **Cross Validation** determined that this was the best **Soft Margin**…



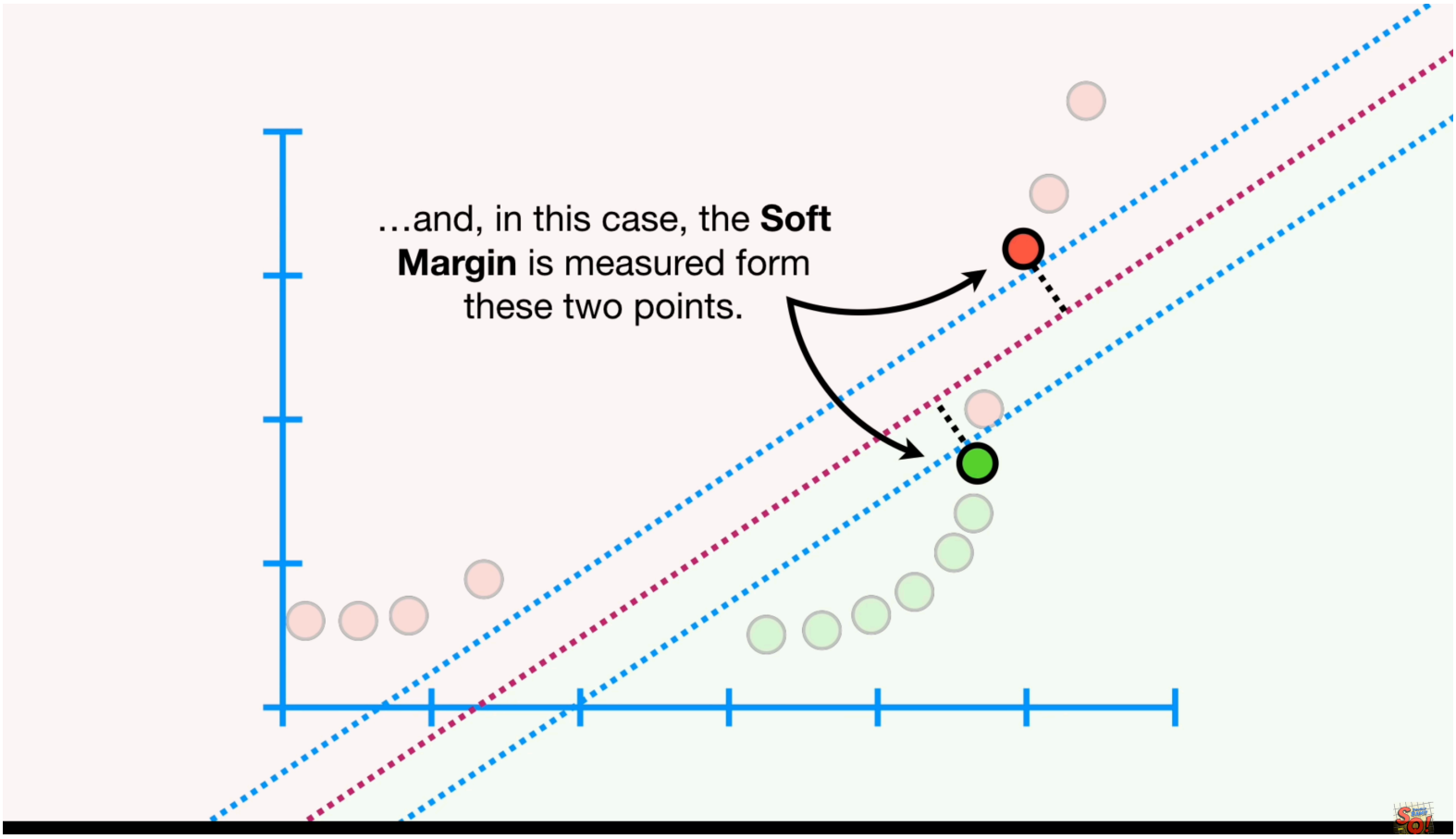
When we use a **Soft Margin** to determine the location of a threshold…

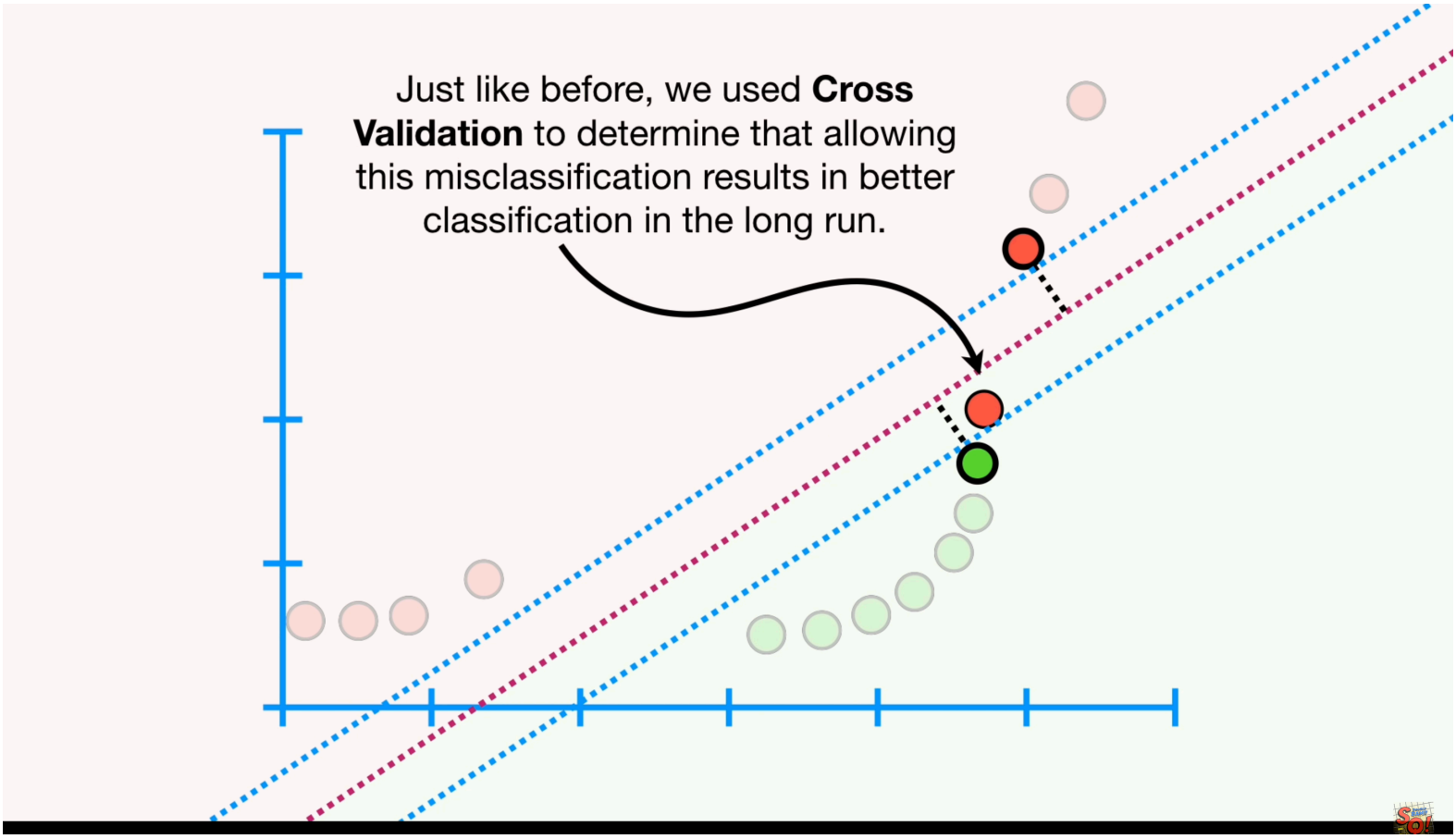


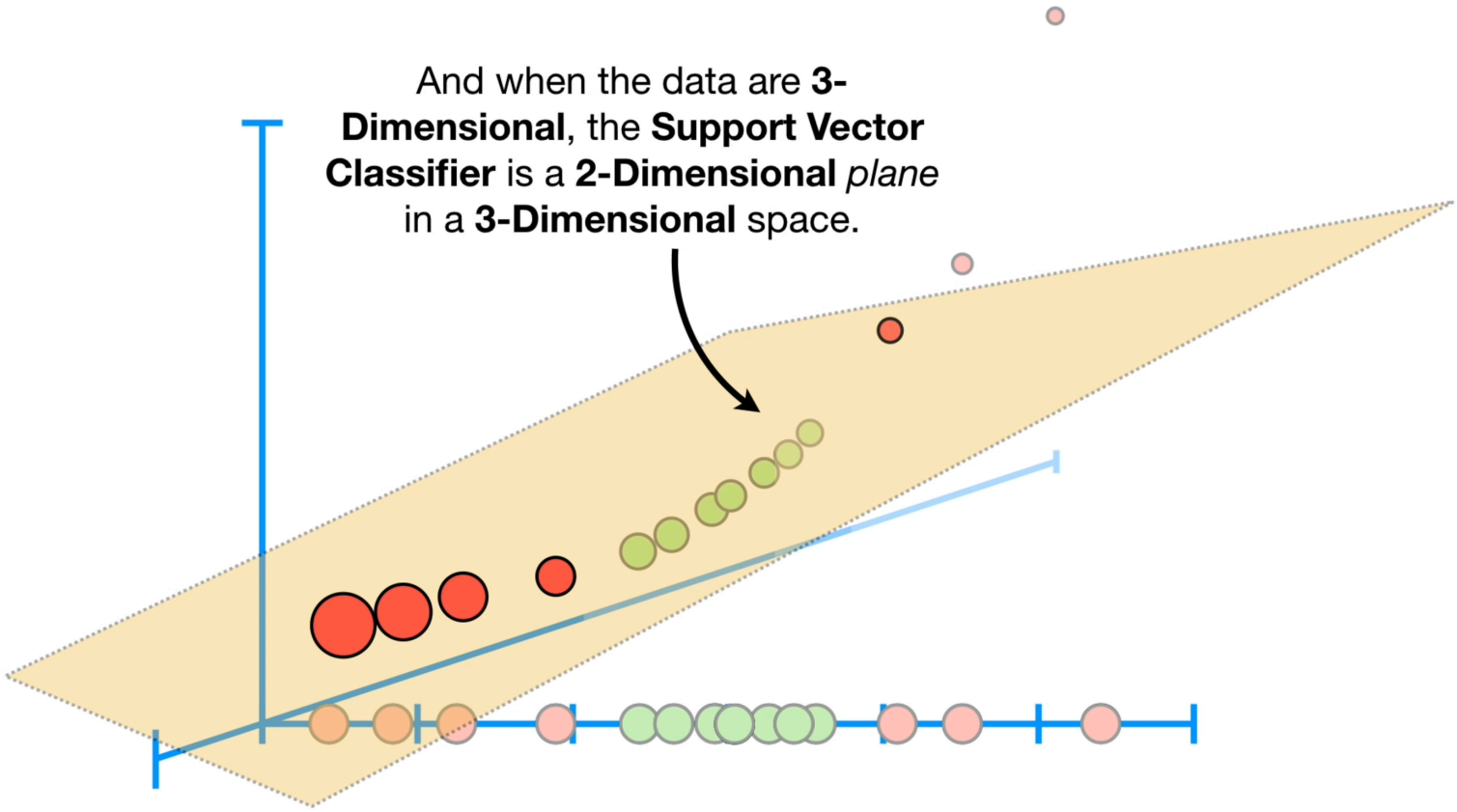
…then we are using a **Soft Margin Classifier** aka a **Support Vector Classifier** to classify observations.

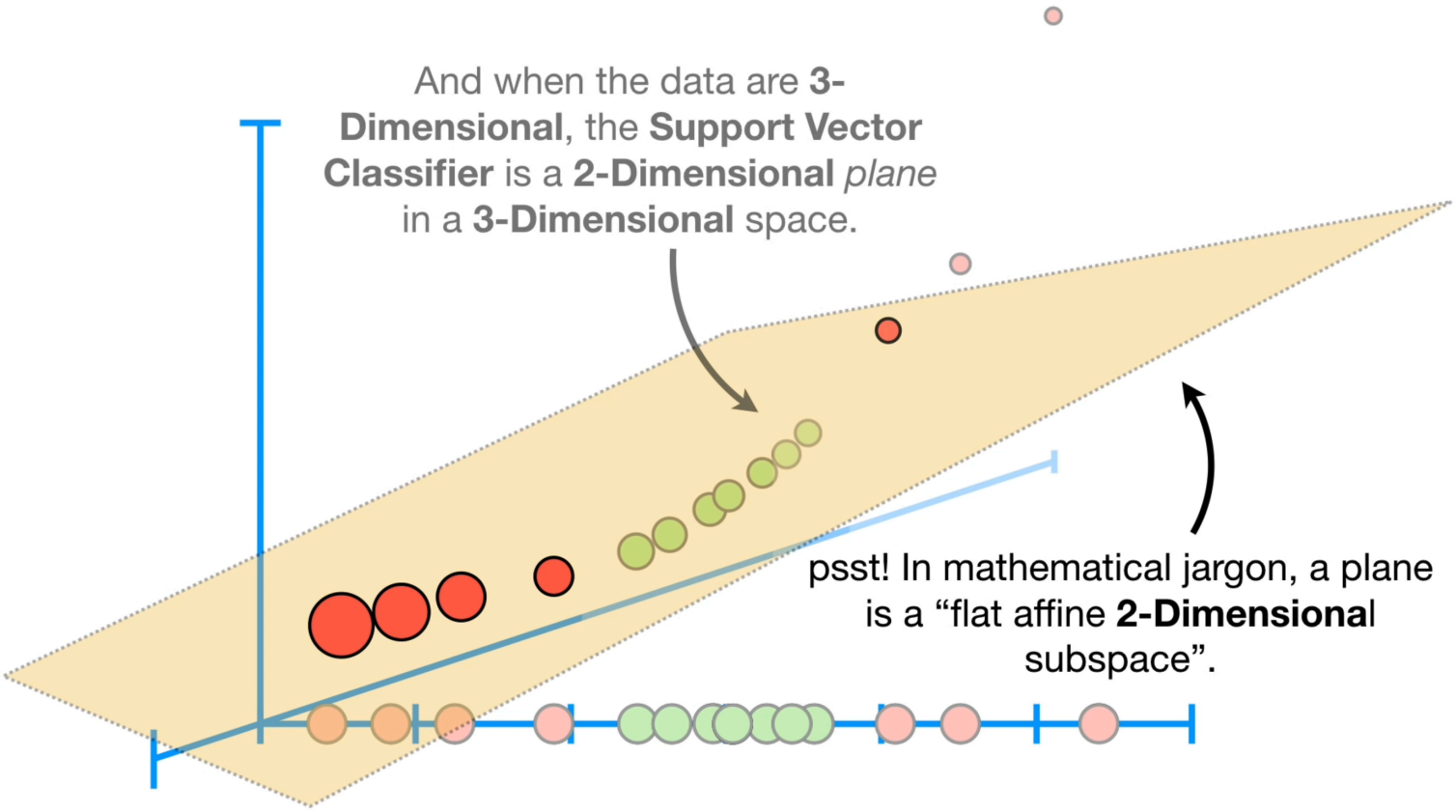


The name **Support Vector Classifier** comes from the fact that the observations on the edge *and within* the **Soft Margin** are called **Support Vectors**.

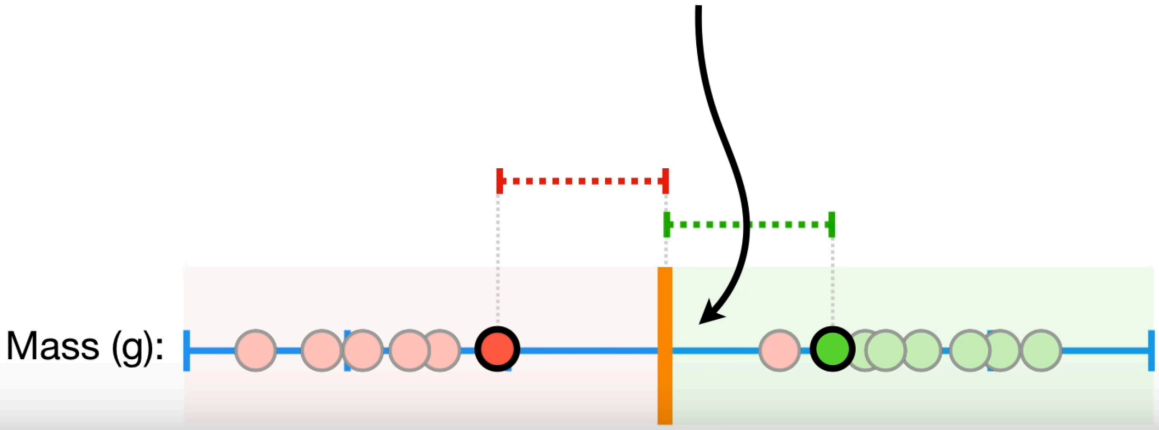

…and, in this case, the **Soft Margin** is measured form these two points.

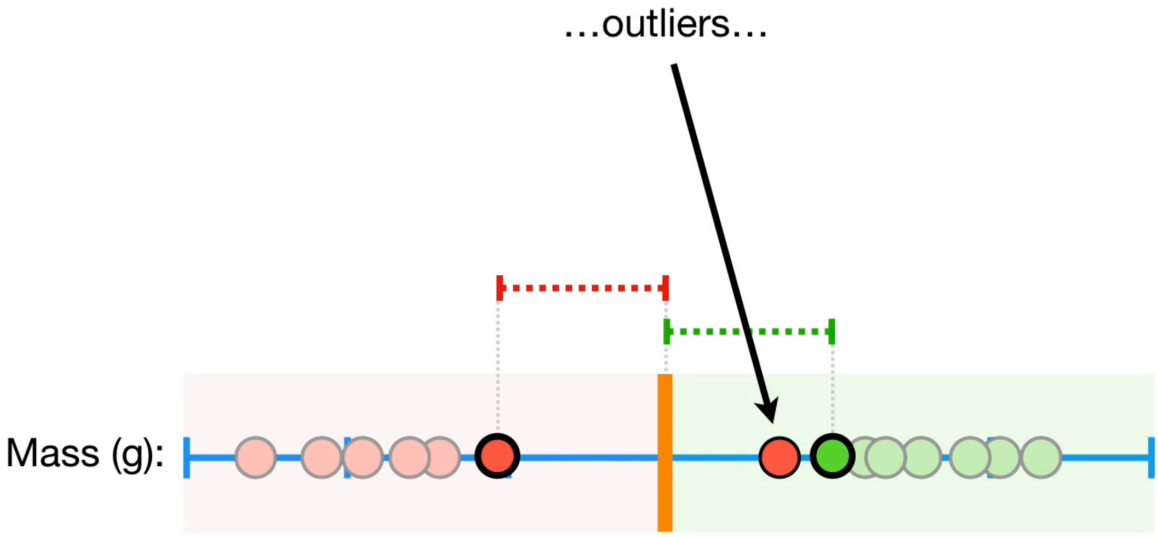

Just like before, we used **Cross Validation** to determine that allowing this misclassification results in better classification in the long run.


And when the data are **3-Dimensional**, the **Support Vector Classifier** is a **2-Dimensional** *plane* in a **3-Dimensional** space.

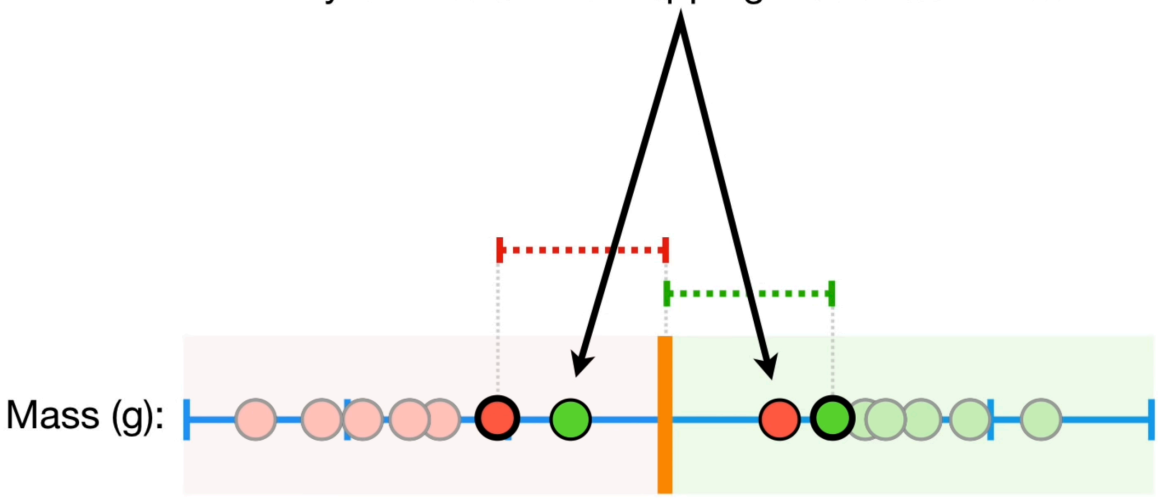

And when the data are **3-Dimensional**, the **Support Vector Classifier** is a **2-Dimensional** *plane* in a **3-Dimensional** space.
psst! In mathematical jargon, a plane is a “flat affine **2-Dimensional** subspace”.

Support Vector Classifiers seem pretty cool because they can handle…

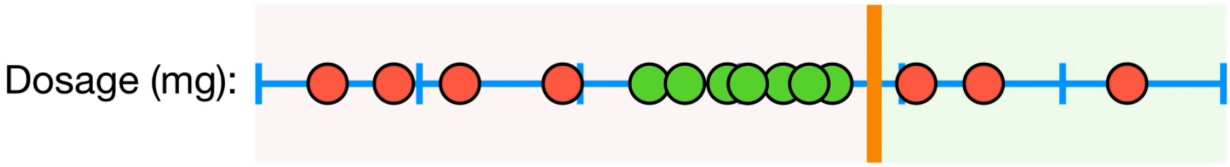



…outliers…
Mass (g):

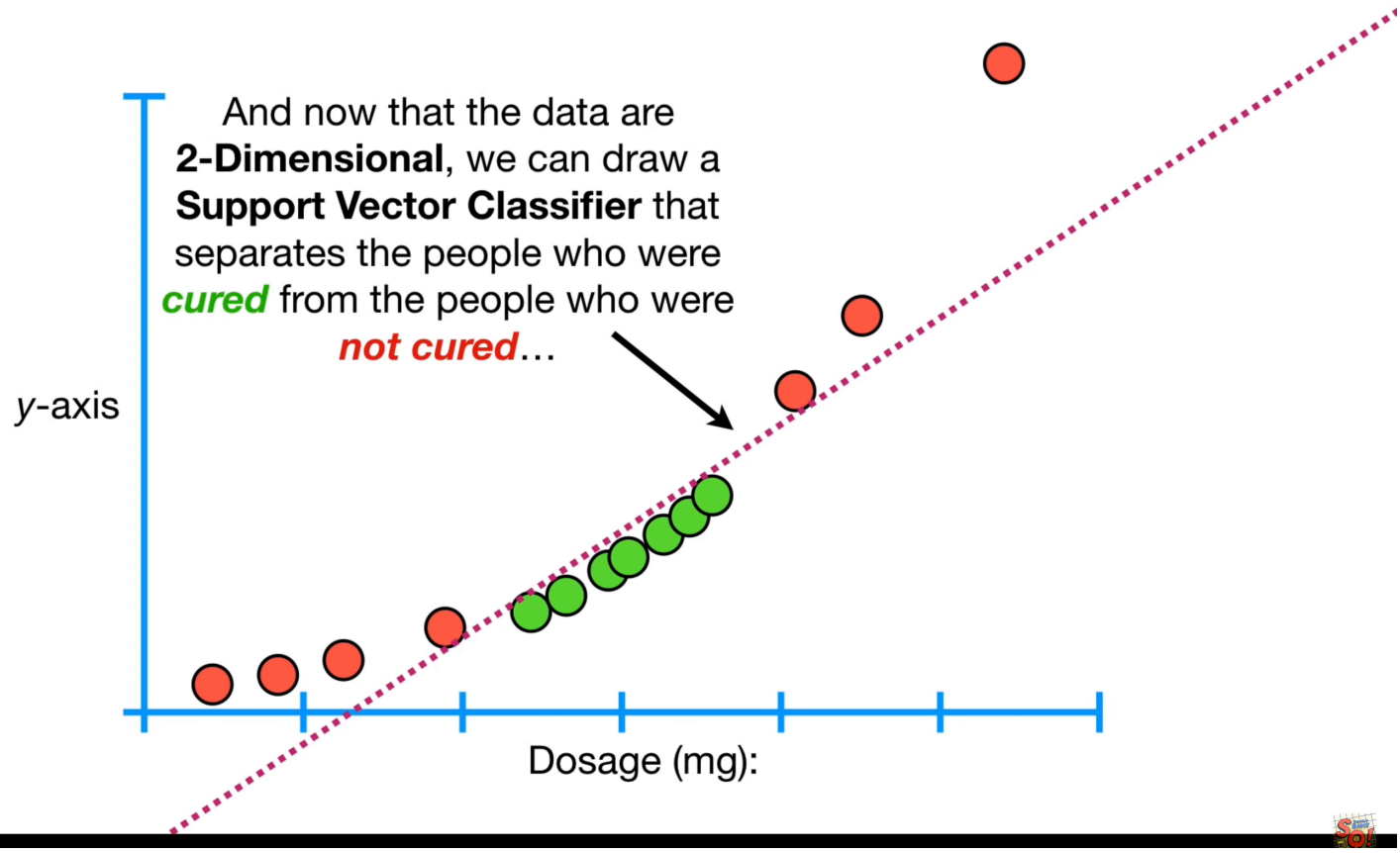
…and, because they allow misclassifications,
they can handle overlapping classifications…

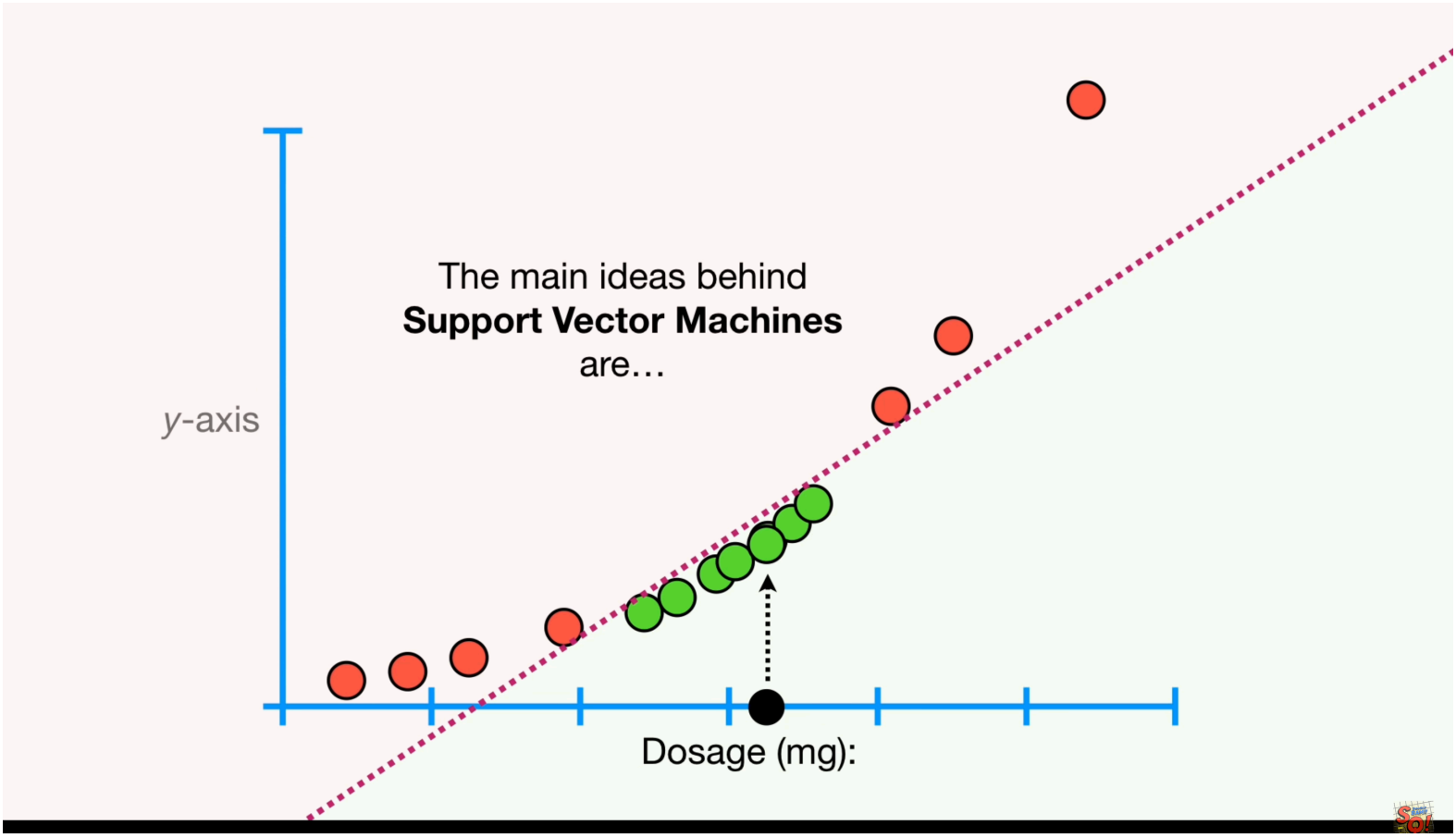


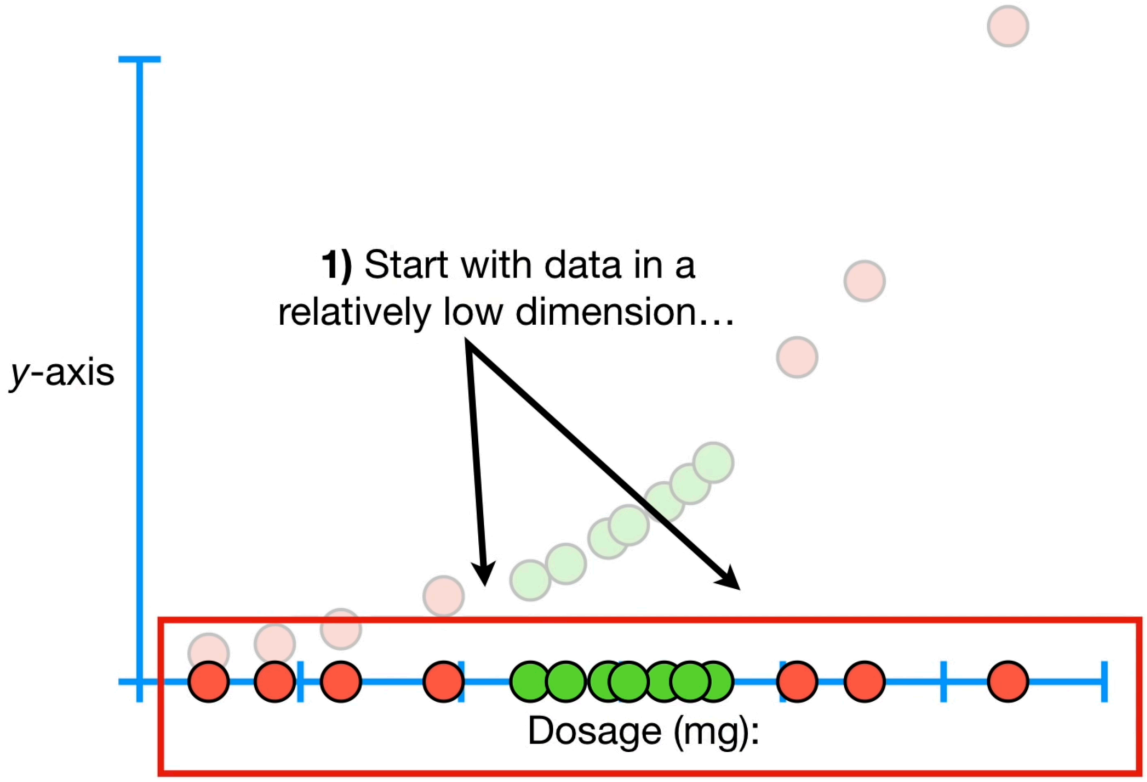
Since **Maximal Margin Classifiers** and **Support Vector Classifiers** can't handle this data, it's high time we talked about…

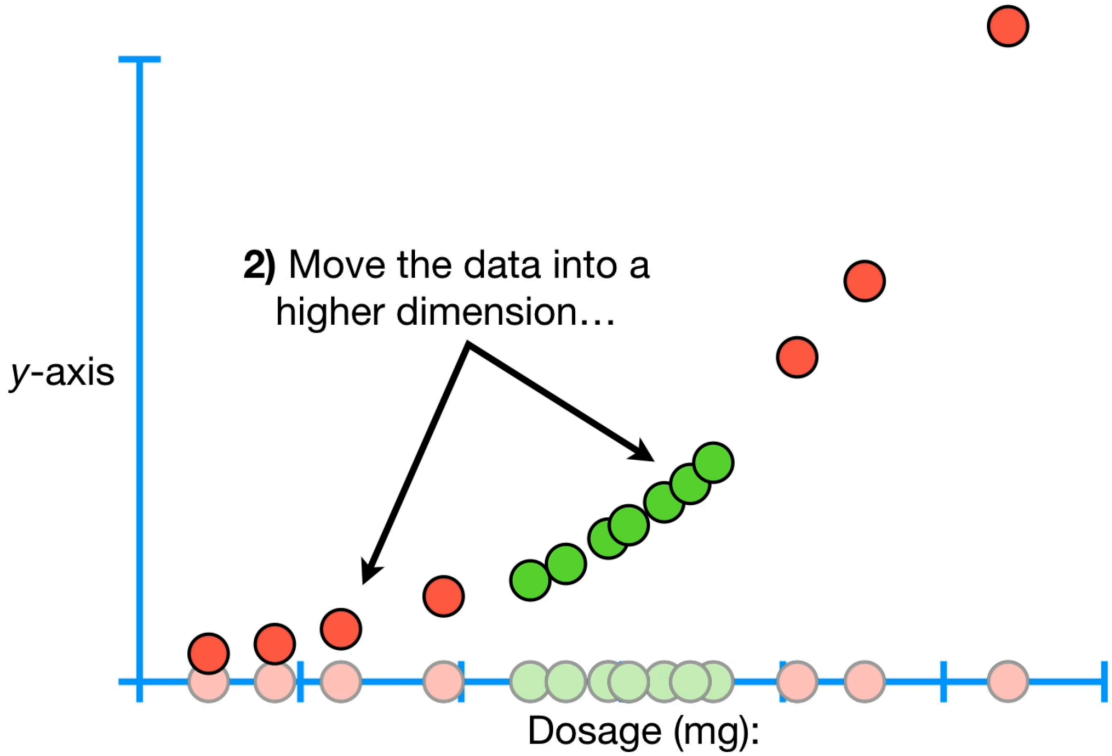


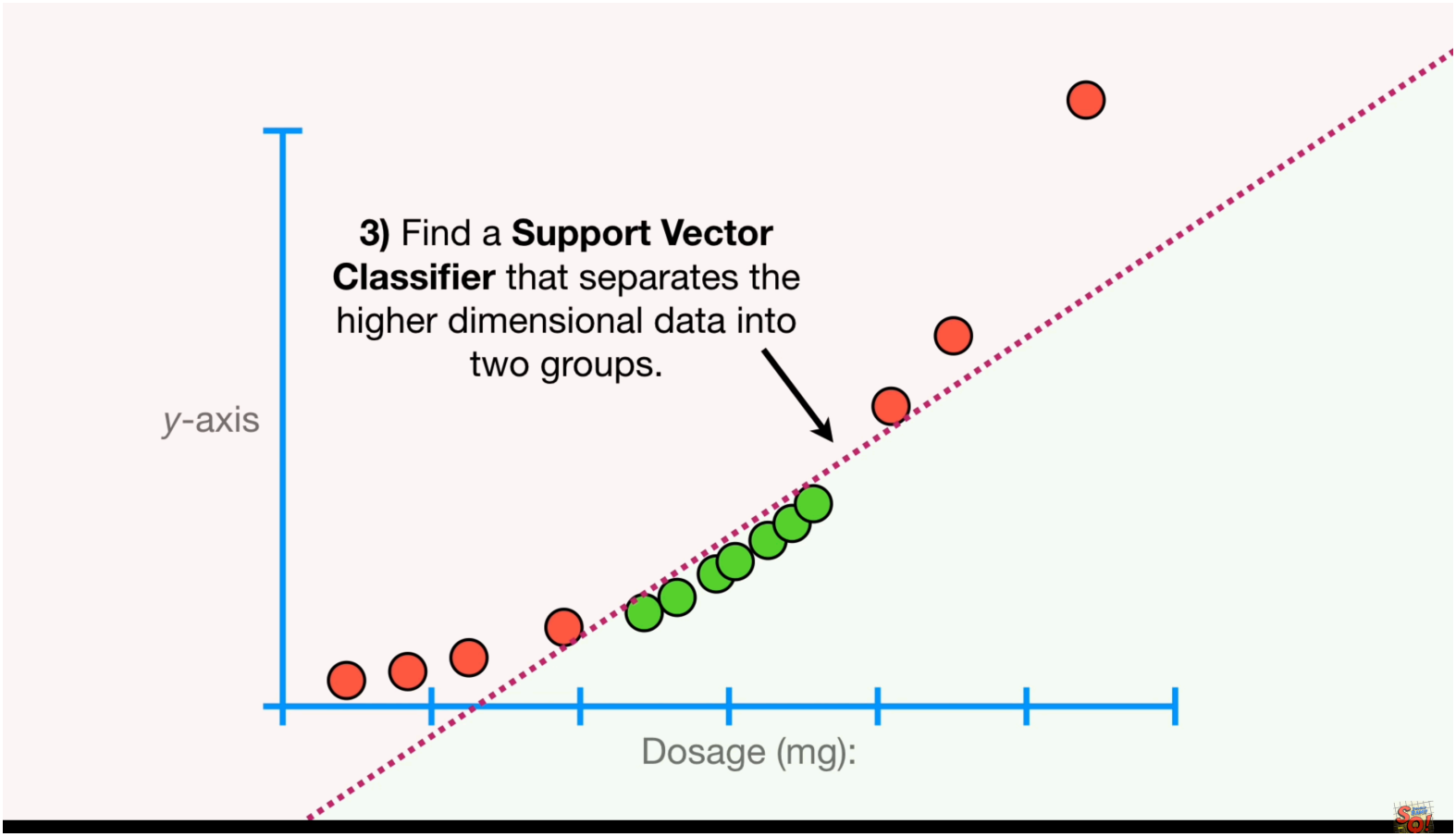
Support Vector Machines!!!

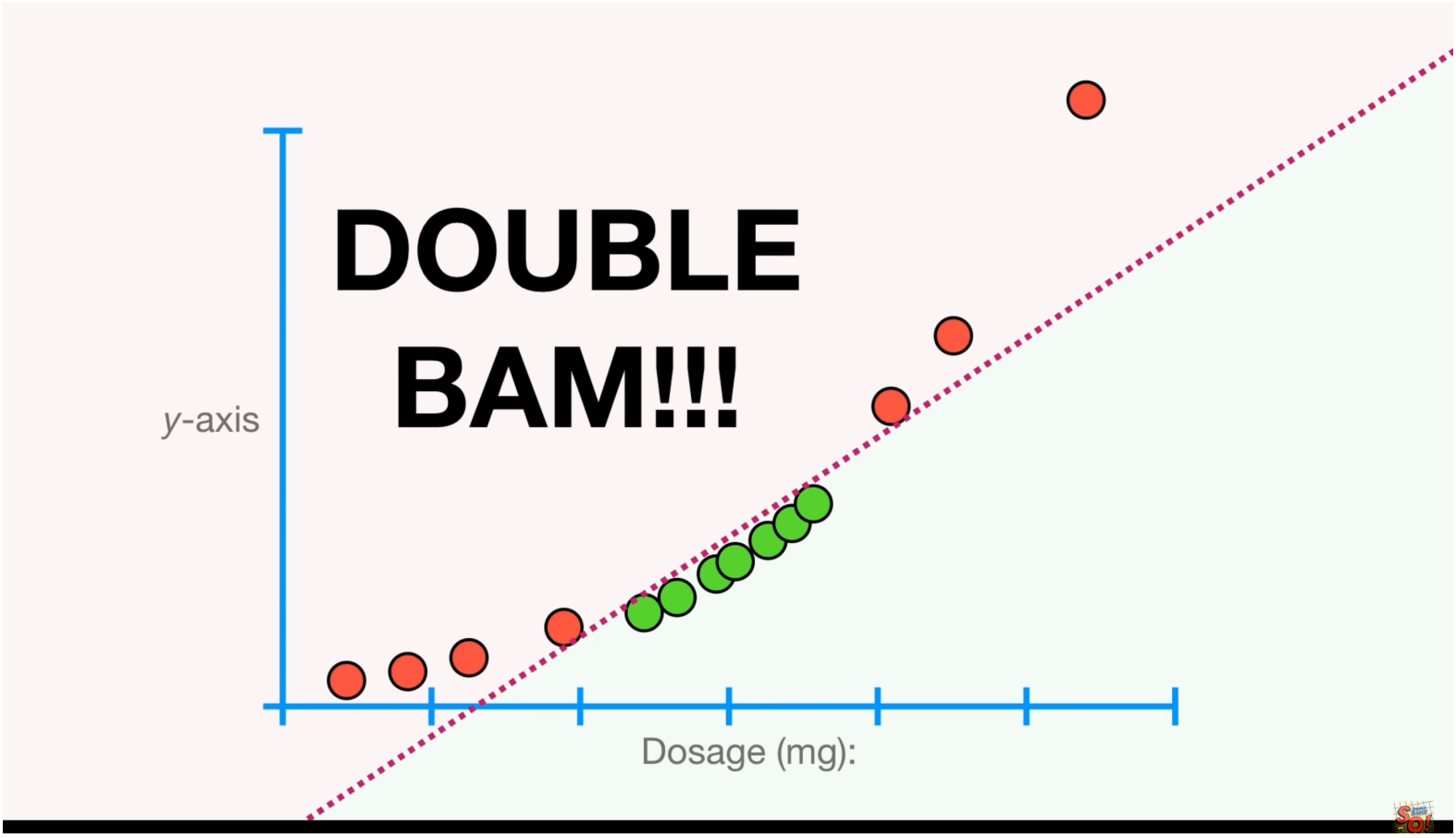

And now that the data are
2-Dimensional, we can draw a
Support Vector Classifier that
separates the people who were
cured from the people who were
not cured…
y-axis
Dosage (mg):

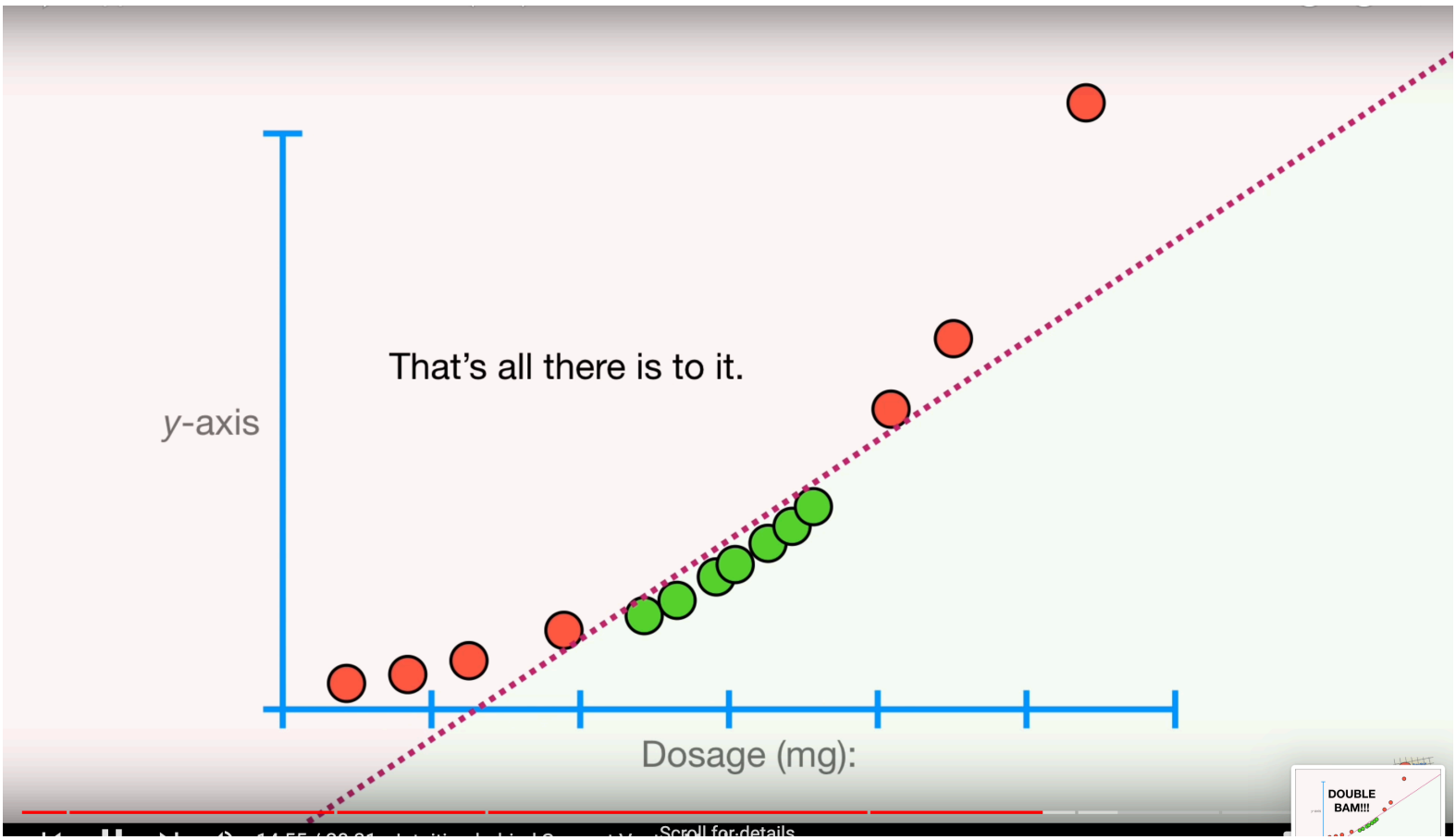

The main ideas behind
Support Vector Machines
are…
y-axis
Dosage (mg):

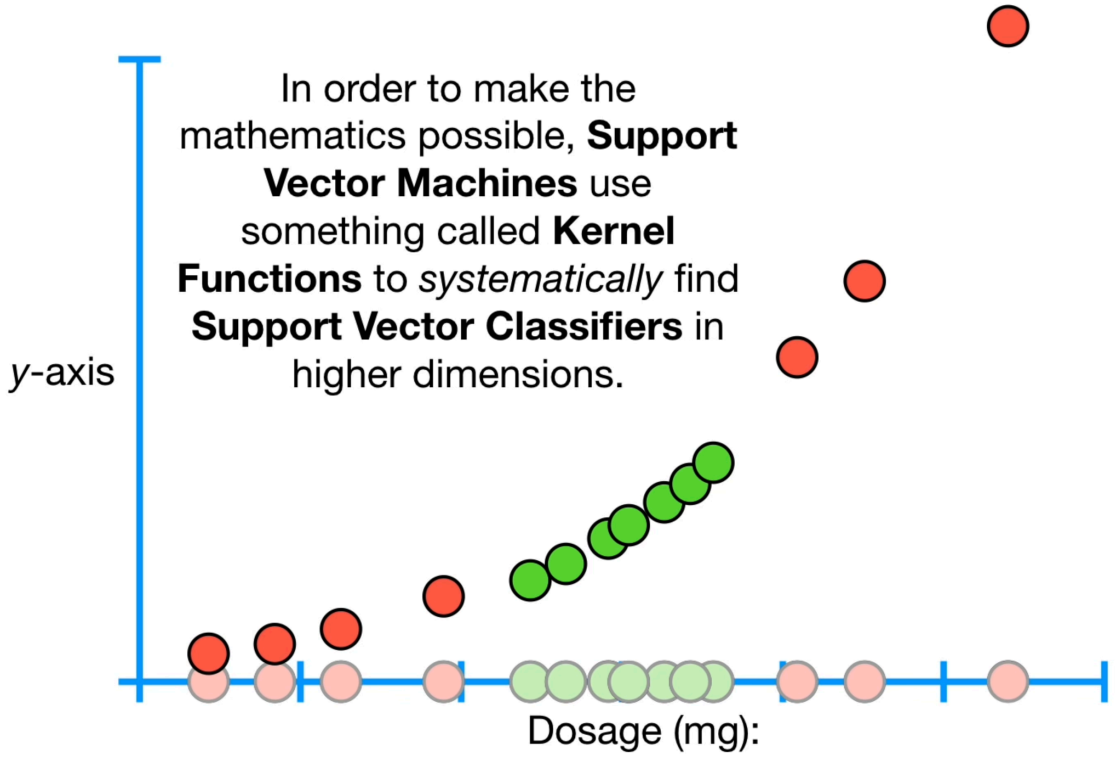

1) Start with data in a relatively low dimension…
y-axis
Dosage (mg):

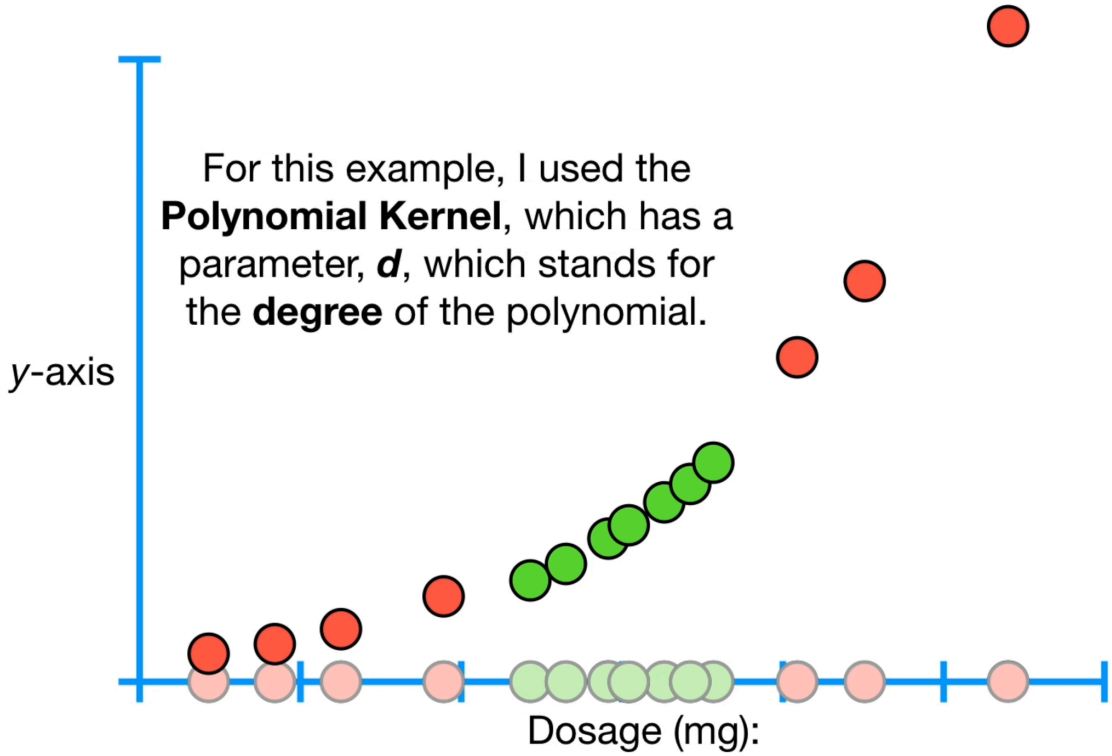

2) Move the data into a higher dimension…
y-axis
Dosage (mg):


3) Find a **Support Vector Classifier** that separates the higher dimensional data into two groups.
y-axis
Dosage (mg):


DOUBLE
BAM!!!
y-axis
Dosage (mg):


That's all there is to it.
y-axis
Dosage (mg):
DOUBLE BAM!!!


In order to make the mathematics possible, **Support Vector Machines** use something called **Kernel Functions** to *systematically* find **Support Vector Classifiers** in higher dimensions.
y-axis
Dosage (mg):

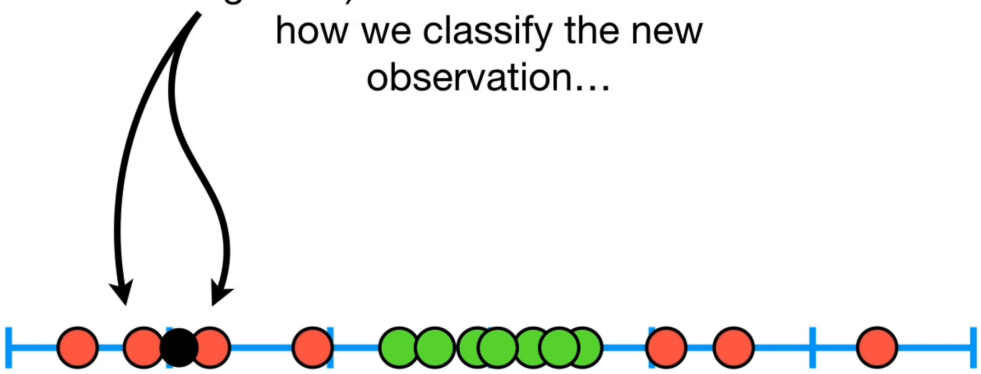

For this example, I used the **Polynomial Kernel**, which has a parameter, ***d***, which stands for the **degree** of the polynomial.
y-axis
Dosage (mg):

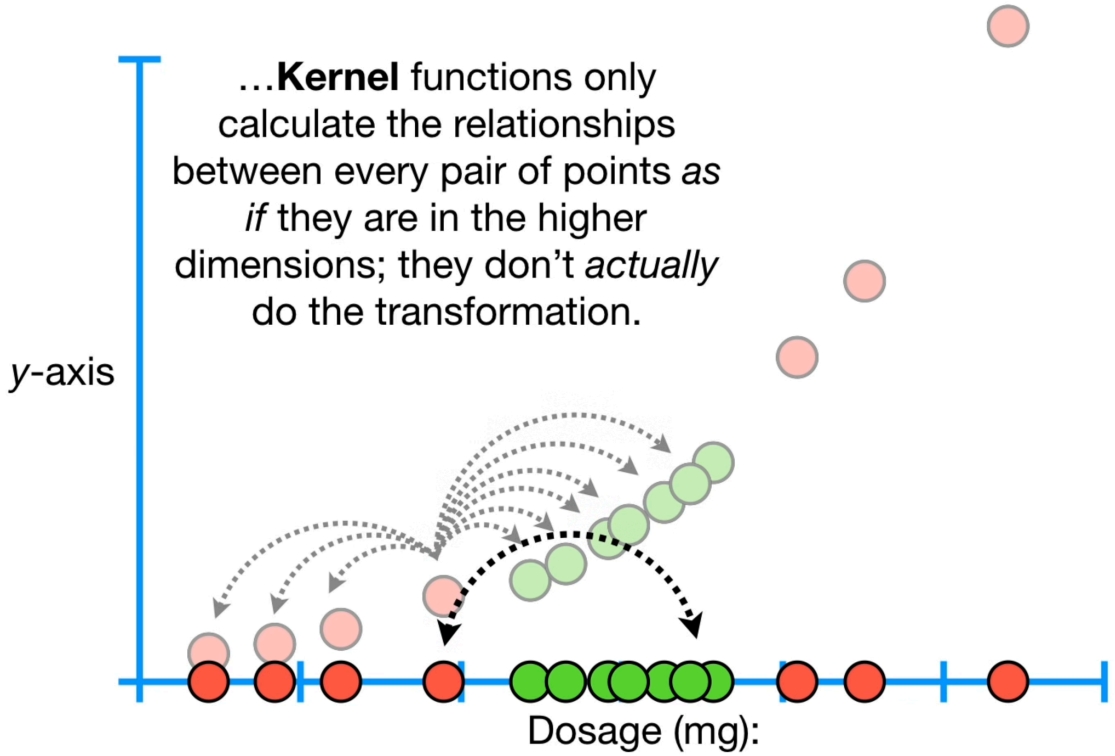
Another very commonly used **Kernel** is the **Radial Kernel**, also known as the **Radial Basis Function (RBF) Kernel**.

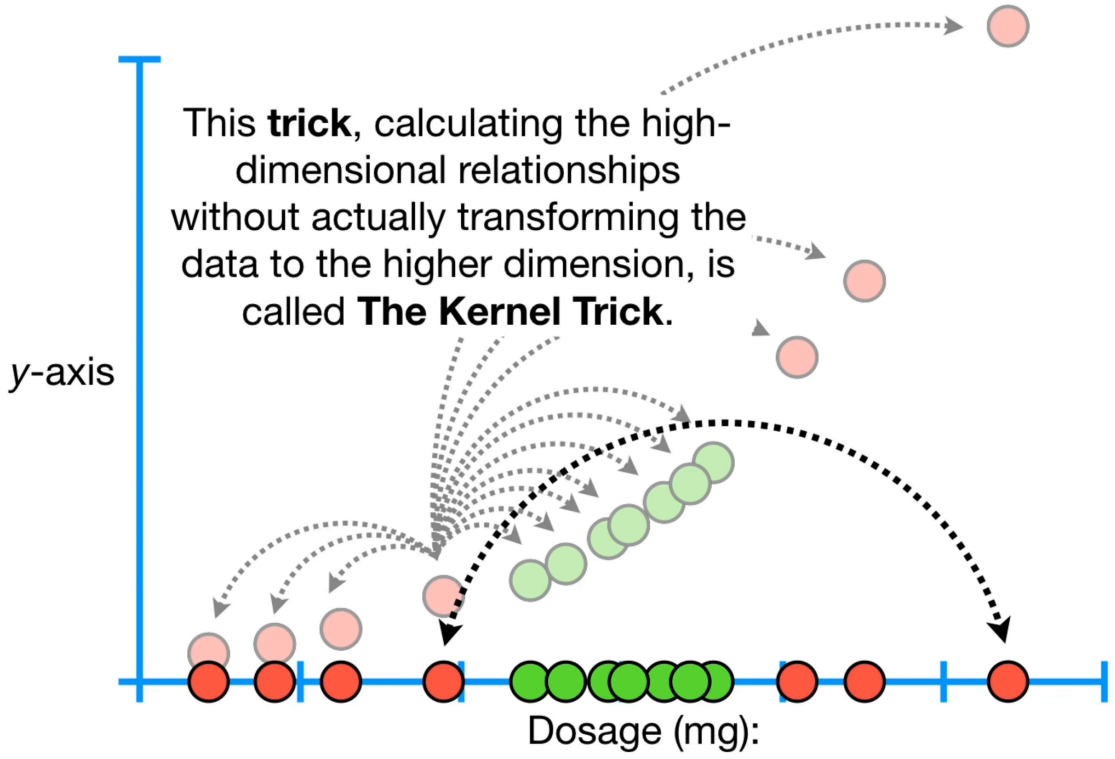
…the **Radial Kernel** behaves like a
Weighted Nearest Neighbor model.

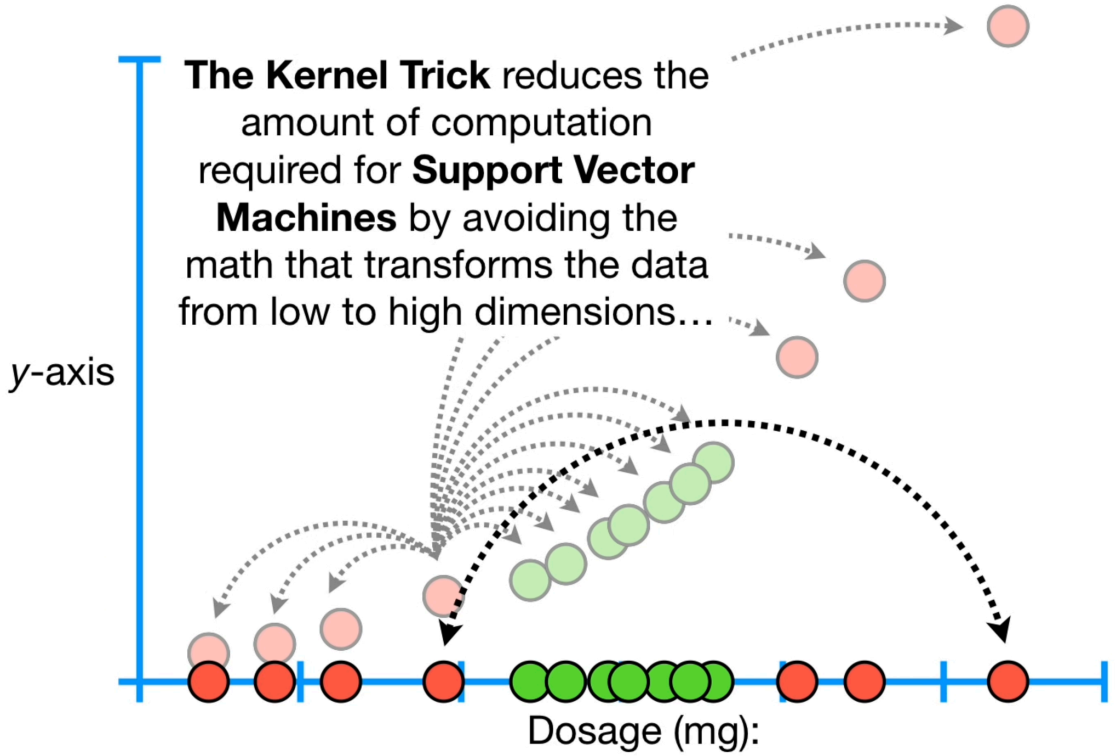
Dosage (mg):

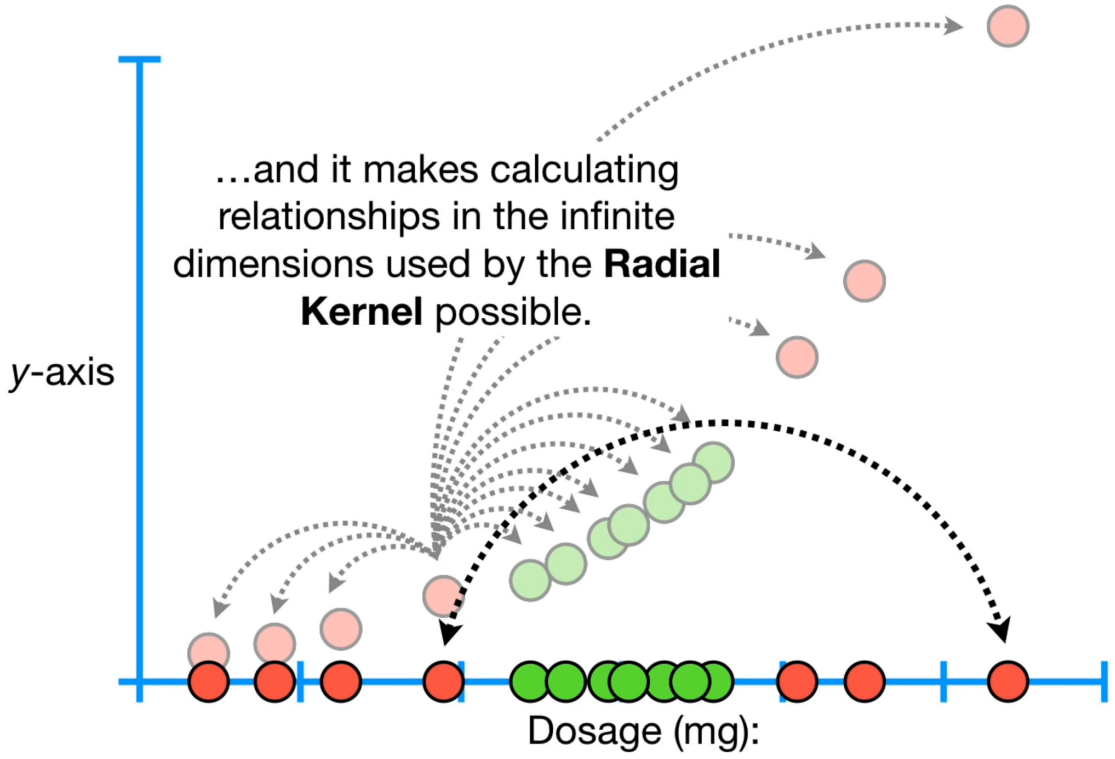
In other words, the closest observations (aka the nearest neighbors) have a lot of influence on how we classify the new observation…




…**Kernel** functions only calculate the relationships between every pair of points *as if* they are in the higher dimensions; they don't *actually* do the transformation.
y-axis
Dosage (mg):


This **trick**, calculating the high-dimensional relationships without actually transforming the data to the higher dimension, is called **The Kernel Trick**.
y-axis
Dosage (mg):


The Kernel Trick reduces the amount of computation required for **Support Vector Machines** by avoiding the math that transforms the data from low to high dimensions…
y-axis
Dosage (mg):


…and it makes calculating relationships in the infinite dimensions used by the **Radial Kernel** possible.
y-axis
Dosage (mg):