What enabled AI?

Philosophers (going back to 400 B.C.) made AI conceivable by considering the ideas that the mind is in some ways like a machine, that it operates on knowledge encoded in some internal language, and that thought can be used to choose what actions to take.

Mathematicians provided the tools to manipulate statements of logical certainty as well as uncertain, probabilistic statements. They also set the groundwork for understanding computation and reasoning about algorithms.

Economists formalized the problem of making decisions that maximize the expected outcome to the decision maker.

Neuroscientists discovered some facts about how the brain works and the ways in which it is similar to and different from computers.

Psychologists adopted the idea that humans and animals can be considered information processing machines. Linguists showed that language use fits into this model.

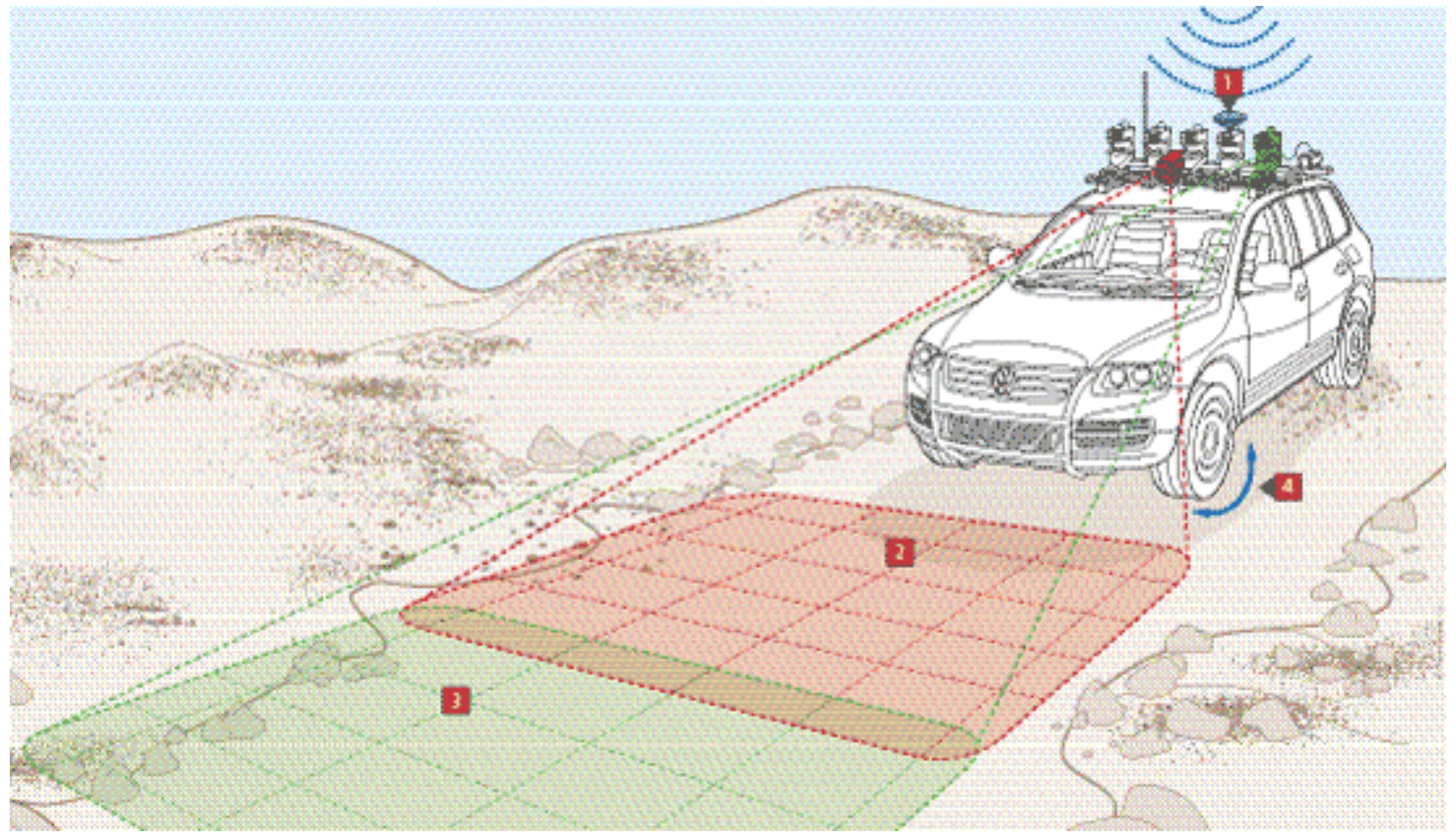
Computer engineers provided the ever-more-powerful machines that make AI applications possible.

Control theory deals with designing devices that act optimally on the basis of feedback from the environment. Initially, the mathematical tools of control theory were quite different from AI, but the fields are coming closer together.

The history of AI has had cycles of success, misplaced optimism, and resulting cutbacks in enthusiasm and funding. There have also been cycles of introducing new creative approaches and systematically refining the best ones.

AI has advanced more rapidly in the past decade because of greater use of the scientific method in experimenting with and comparing approaches. Recent progress in understanding the theoretical basis for intelligence has gone hand in hand with improvements in the capabilities of real systems. The subfields of AI have become more integrated, and AI has found common ground with other disciplines.

Robotic vehicles



2005 Won DARPA Grand Challenge - Cameras, Radar, & Laser rangefinders

2006 CMU Boss won Urban Challenge

MAPGEN plans the daily operations for NASA’s Mars Exploration Rovers

Alphago

208,168,199,381,979,984,699,478,633,344,862,770,286,522,453,884,
530,548,425,639,456,820,927,419,612,738,015,378,525,648,451,698,
519,643,907,259,916,015,628,128,546,089,888,314,427,
129,715,319,317,557,736,620,397,247,064,840, 935.

How many possible moves there are in Go.

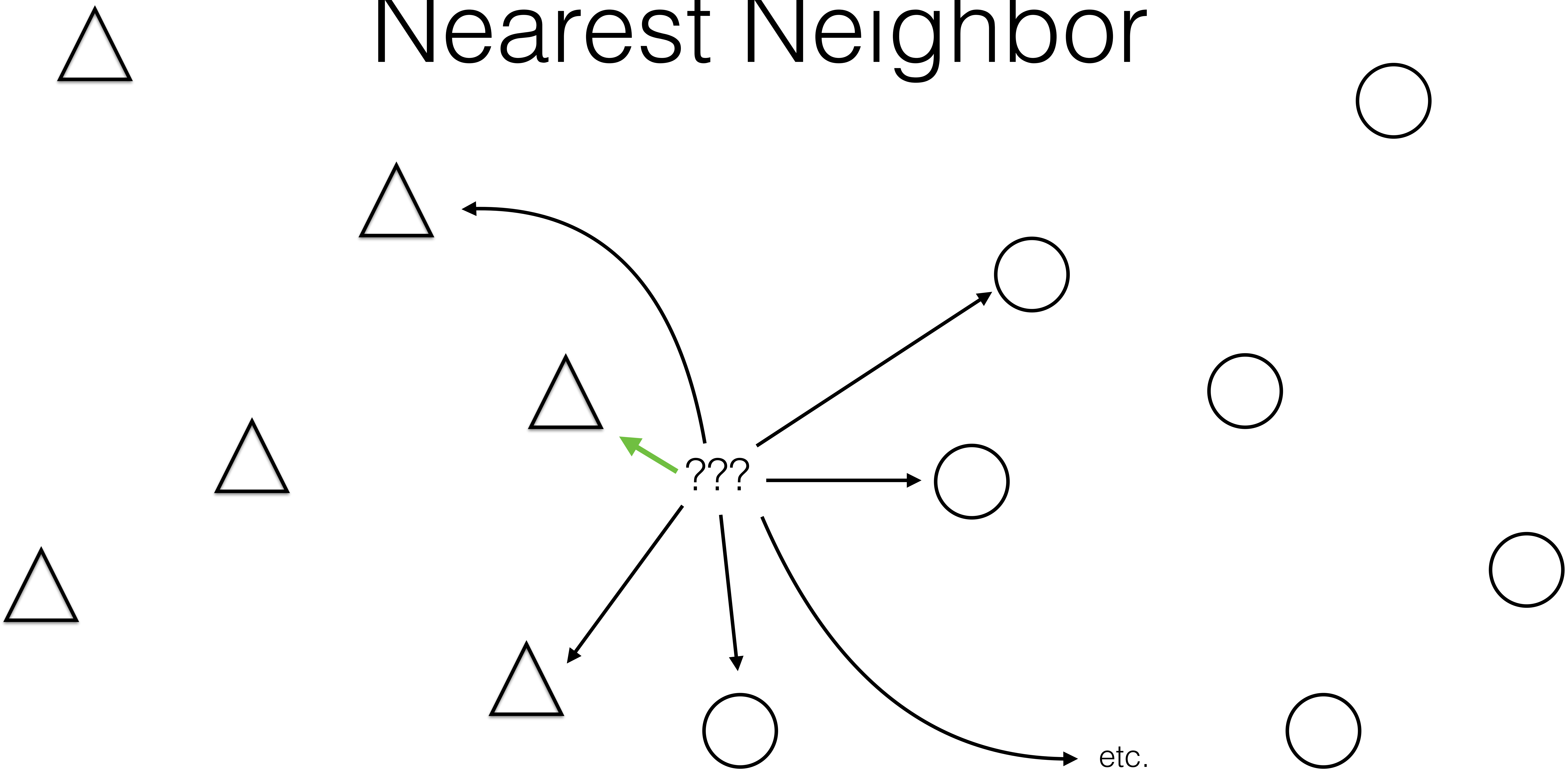

Dynamic Analysis and Replanning Tool (DART)


Dynamic Analysis and Replanning Tool (DART)

iRobot Roomba & Packbot

Wekinator & other examples folders

Nearest Neighbor

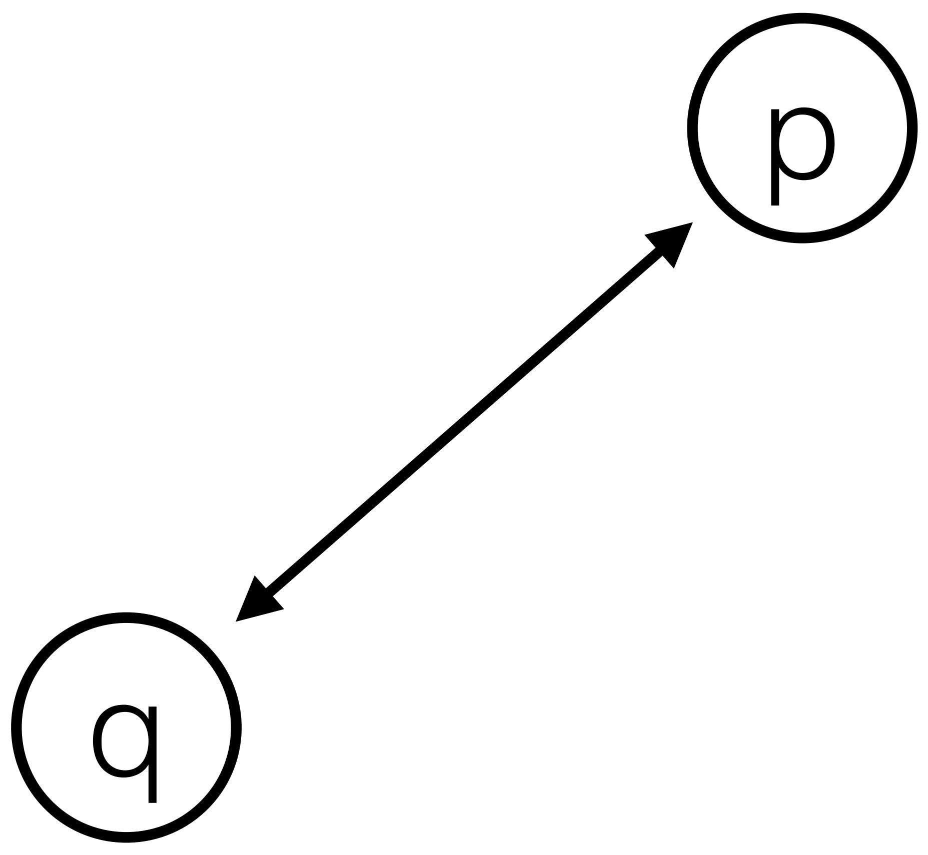


Nearest Neighbor

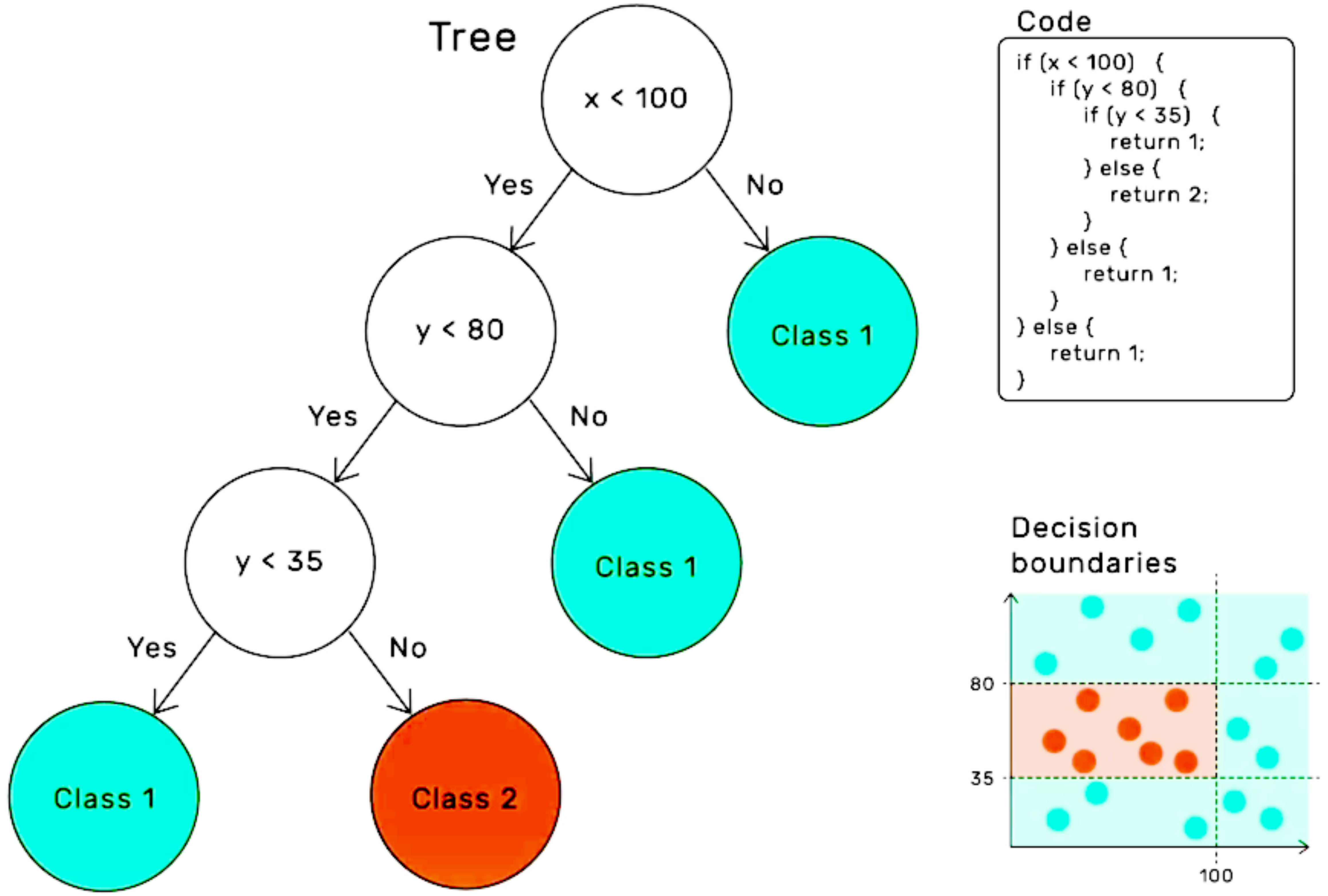
Euclidean distance

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

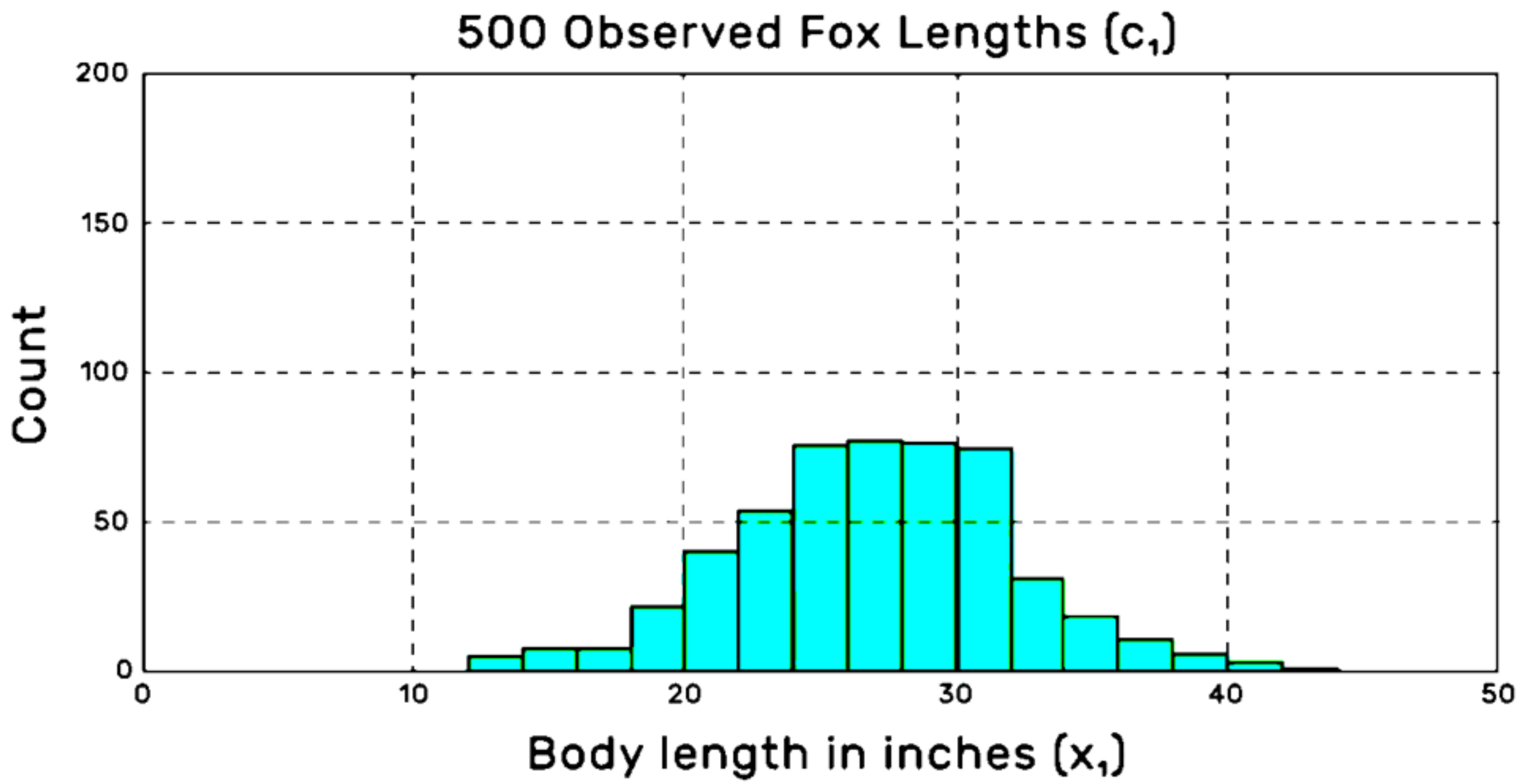


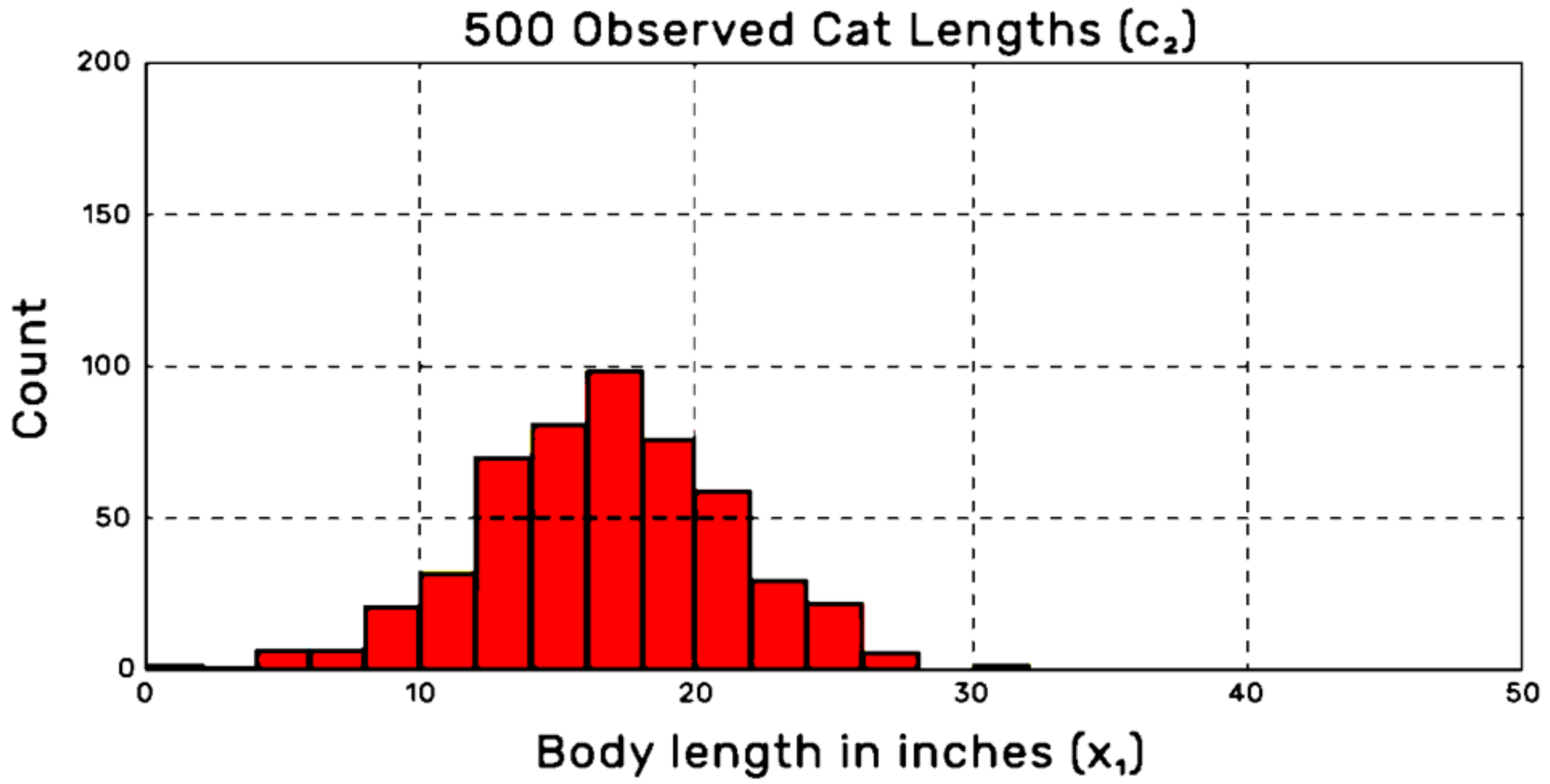
Decision Tree

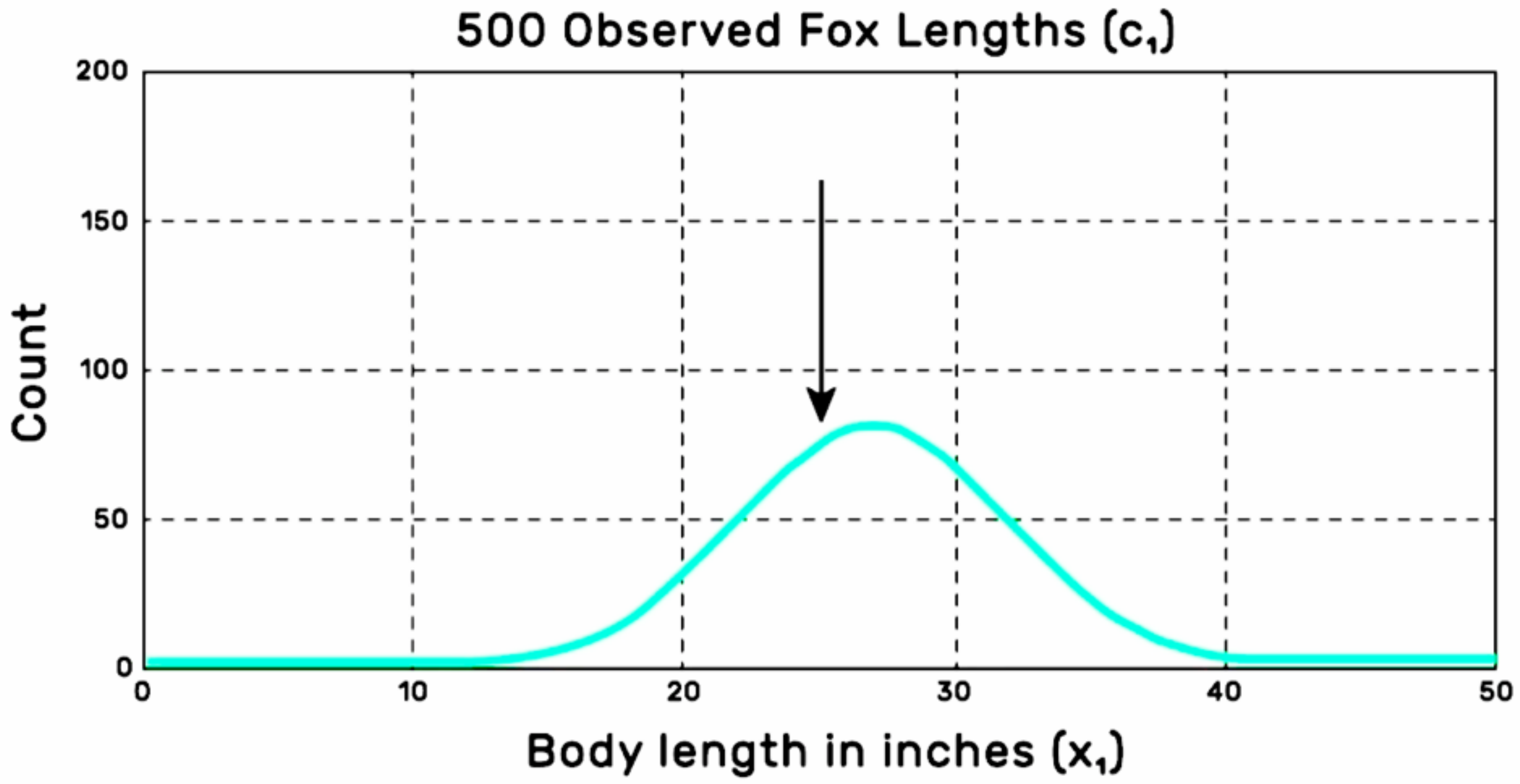


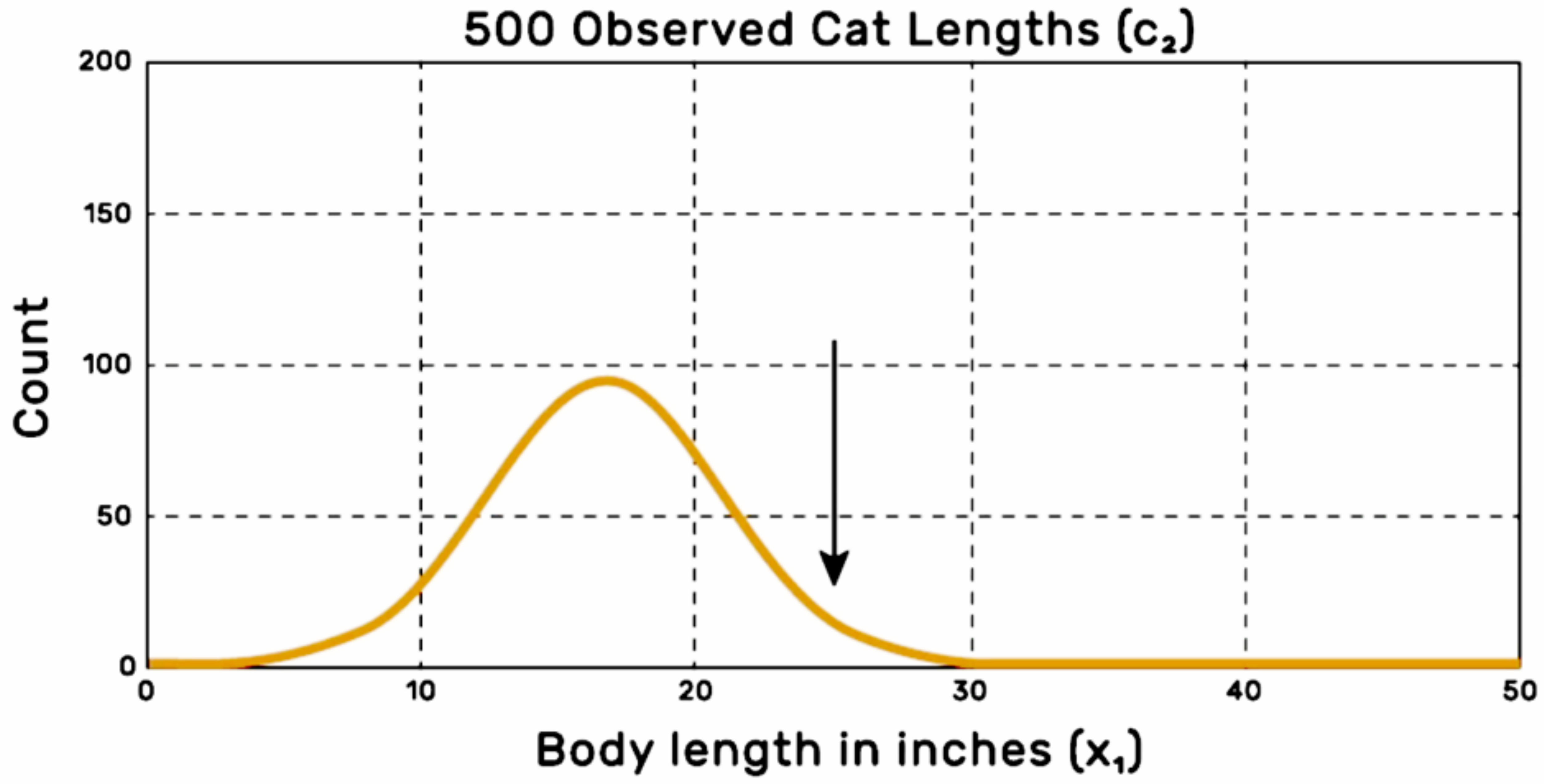
Naive Bayes

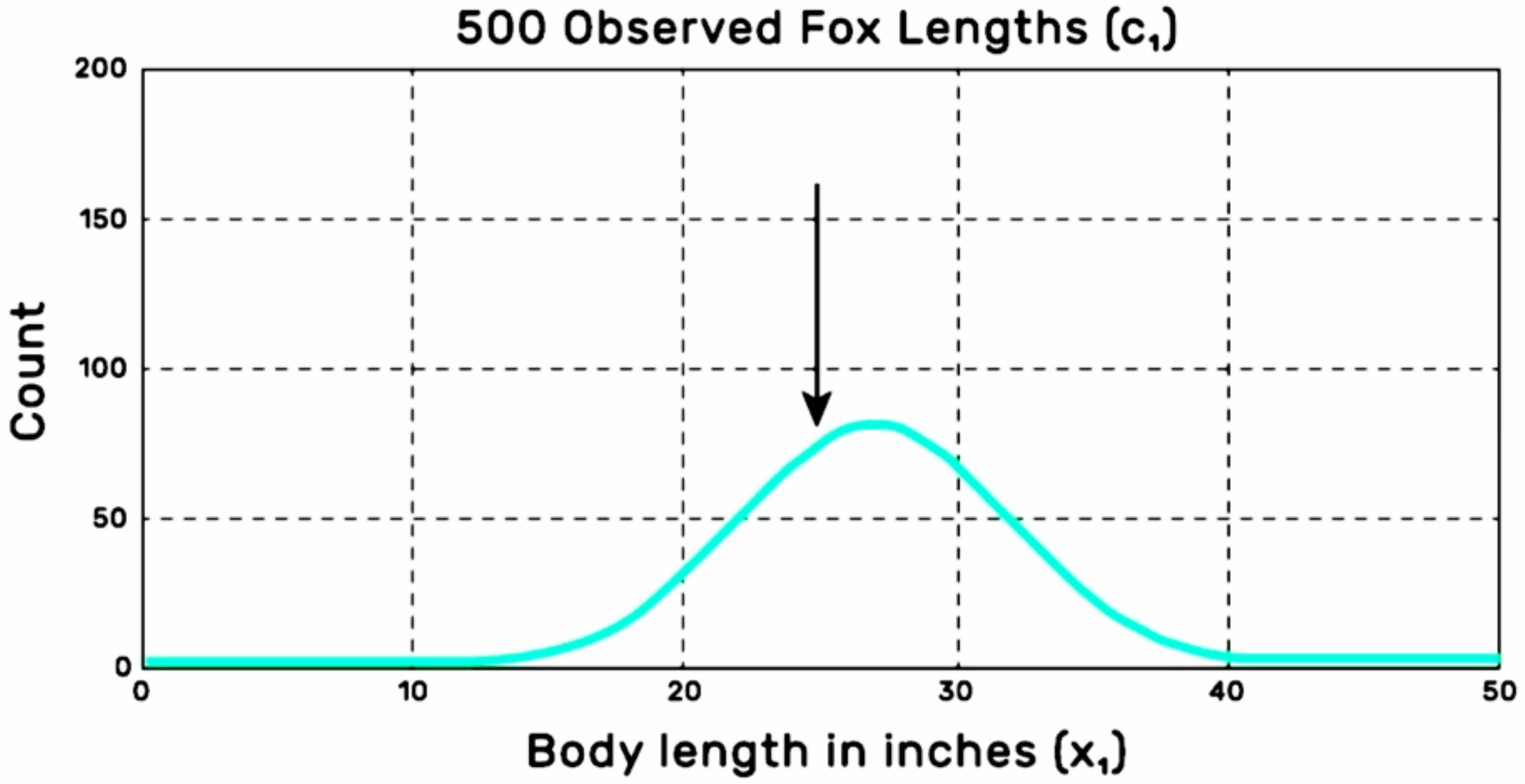
Probability: from class 1, from class 2, from class 3, etc.

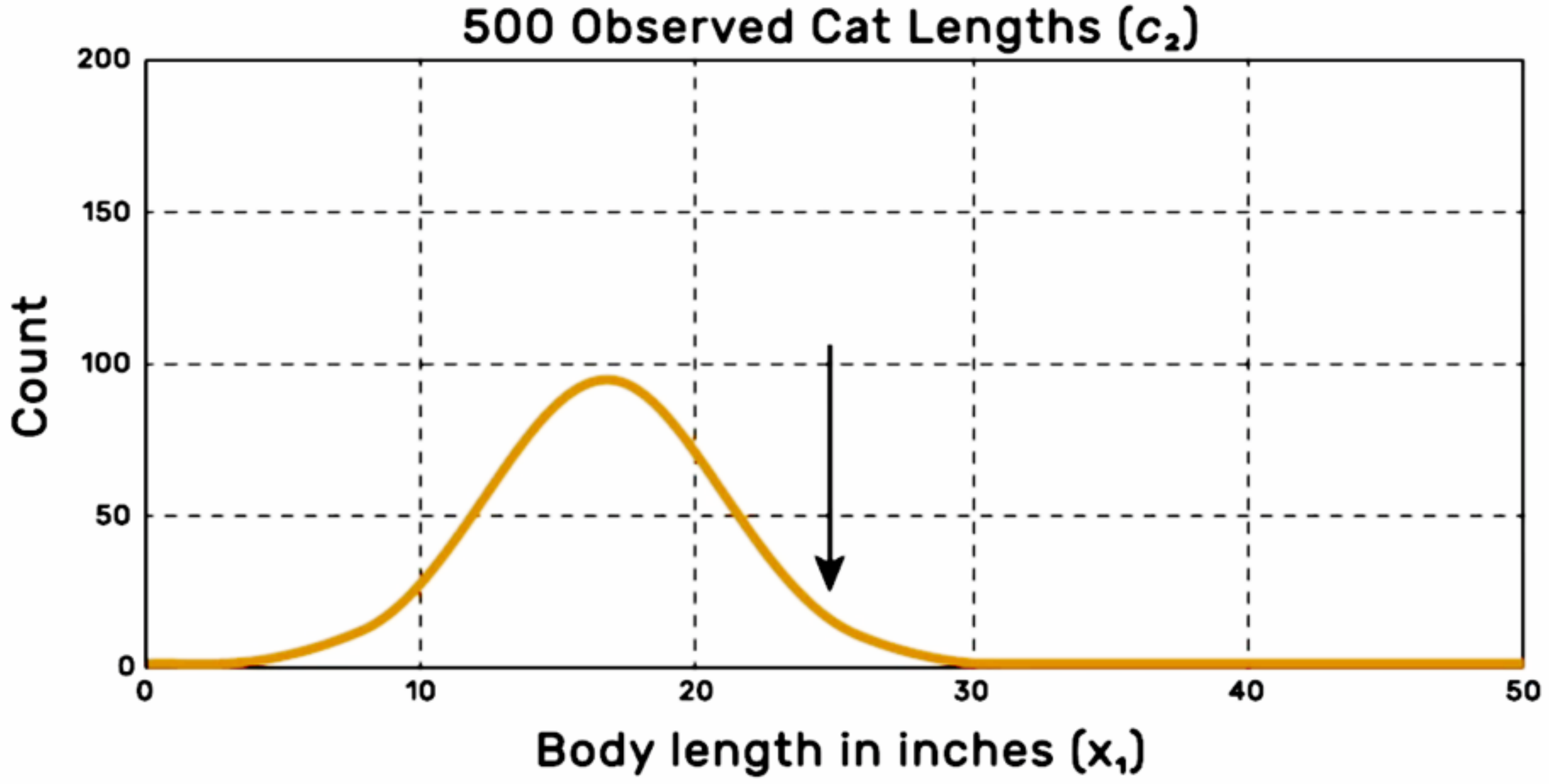

500 Observed Fox Lengths (c_1)
Count
200
150
100
50
0
0
10
20
30
40
50
Body length in inches (x_1)


500 Observed Cat Lengths (c_2)
Count
200
150
100
50
0
0
10
20
30
40
50
Body length in inches (x_1)


500 Observed Fox Lengths (c_1)
Count
200
150
100
50
0
0
10
20
30
40
50
Body length in inches (x_1)


500 Observed Cat Lengths (c_2)
Count
200
150
100
50
0
0
10
20
30
40
50
Body length in inches (x_1)





- 25-inch value of x_1: much more likely under **fox curve** than under **cat curve**
- But, need to take into consideration relative numbers of foxes and housecats! (90% are cats)
- Then combine this knowledge with likelihoods to make a classification

Adaboost

Adaboost

Take a bunch of weak classifiers, and then pool all their answers together, getting a much better answer.

Adaboost

Example

I’m good at this but not good at that, but you’re good at this, and not good at the other thing, and then she’s good at the other thing but not good at the further thing, etc…

Adaboost

1. It helps you choose the training set for each new classifier that you train based on the results of the previous classifier.

2. It determines how much weight should be given to each classifier's proposed answer when combining the results.

Adaboost

- Takes random sub-samples from the training data.
- These are not separate groups, but data can overlap.
- Each training example is given a weight, which determines the probability it will be chosen for training. Sample groups are influenced by earlier classifiers. After training, data that was misclassified will have higher probability to be chosen again.

Adaboost

Depending on how well each classifier performed, a weight will be given to that classifier for the final answer (which is a sum of all the weighted classifier answers).

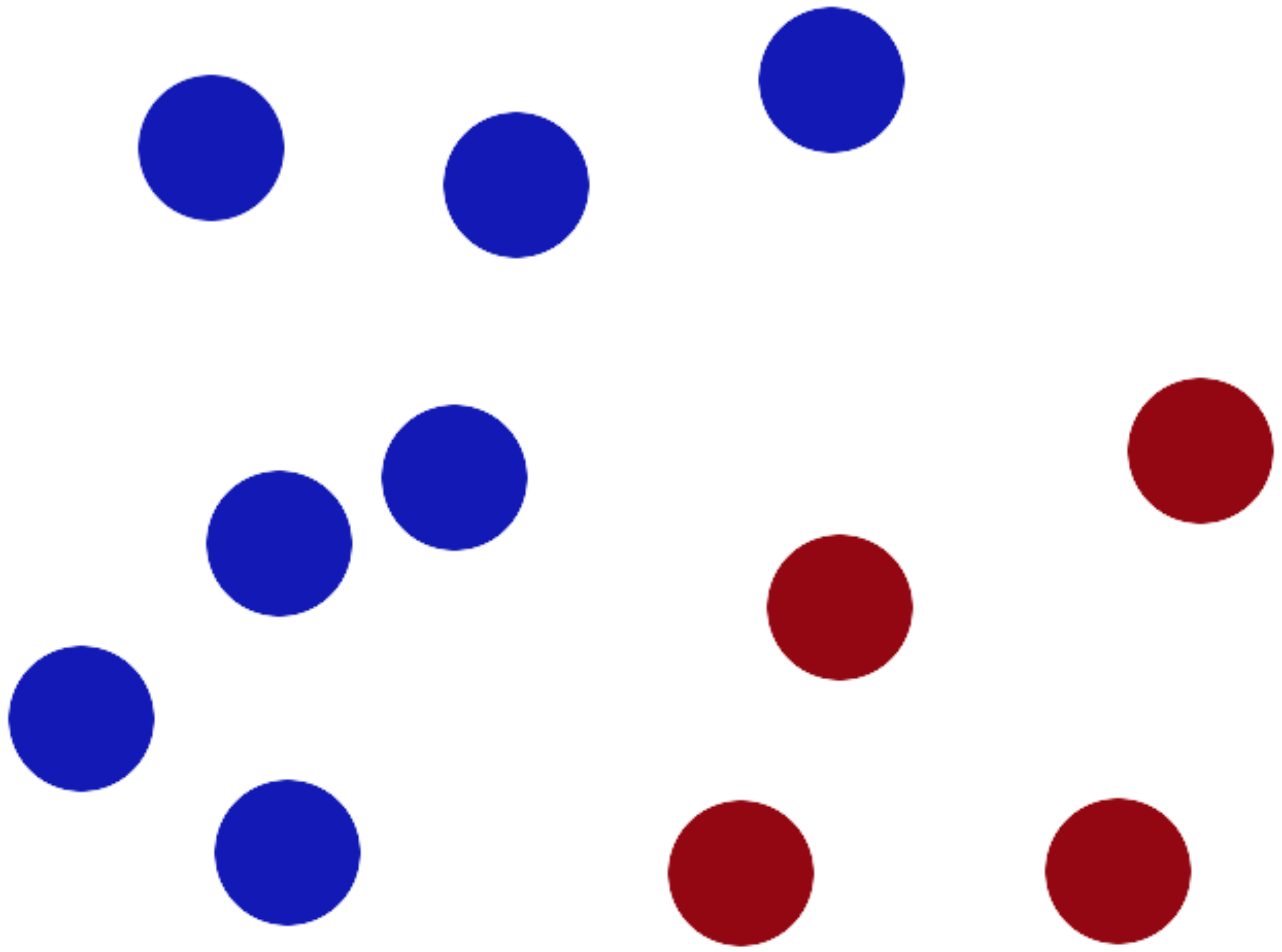
Adaboost

In Wekinator it uses decision trees as classifier algorithm

Adaboost

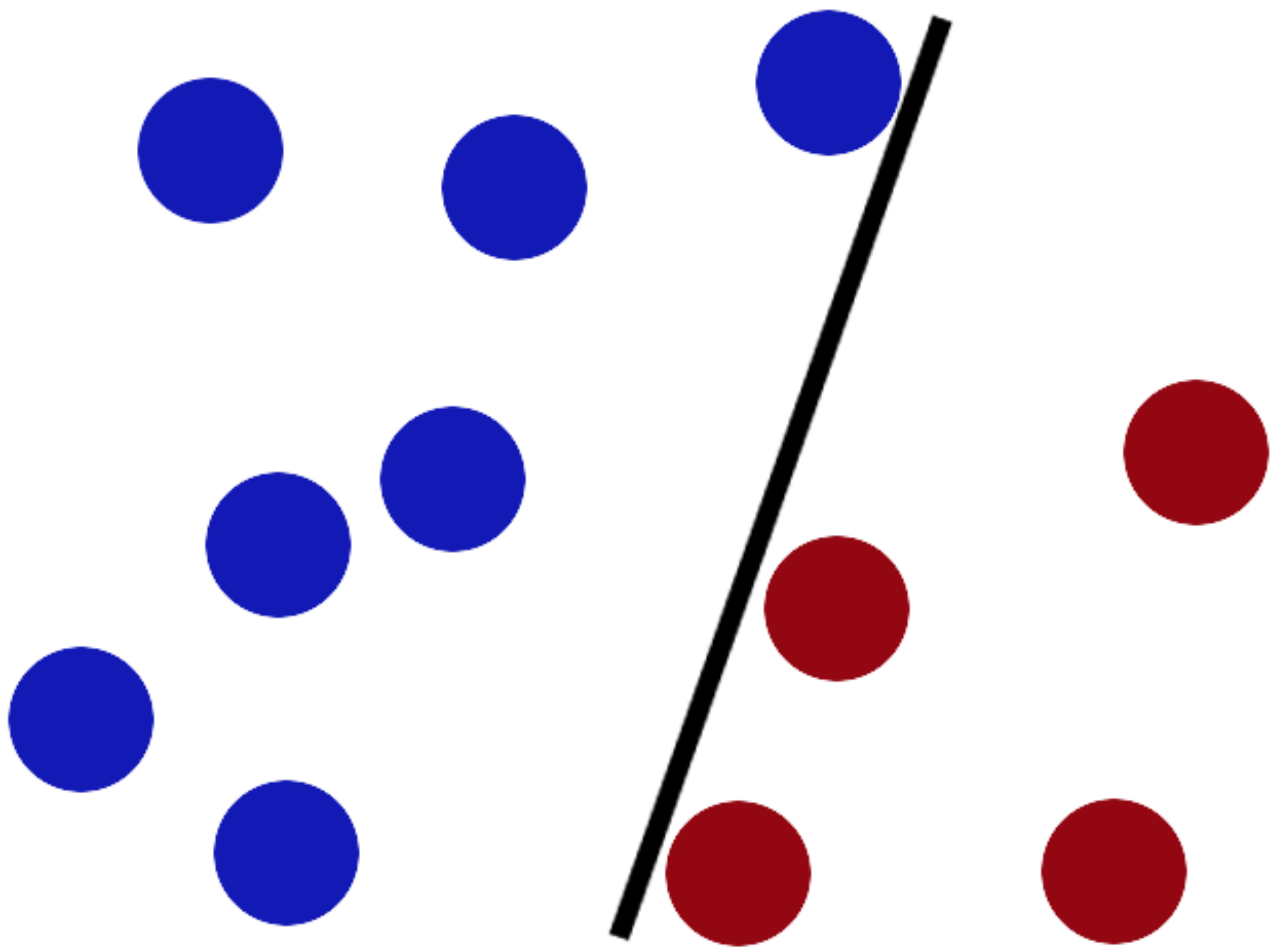
Goto algorithm for more complicated things

Support Vector Machine (SVM)



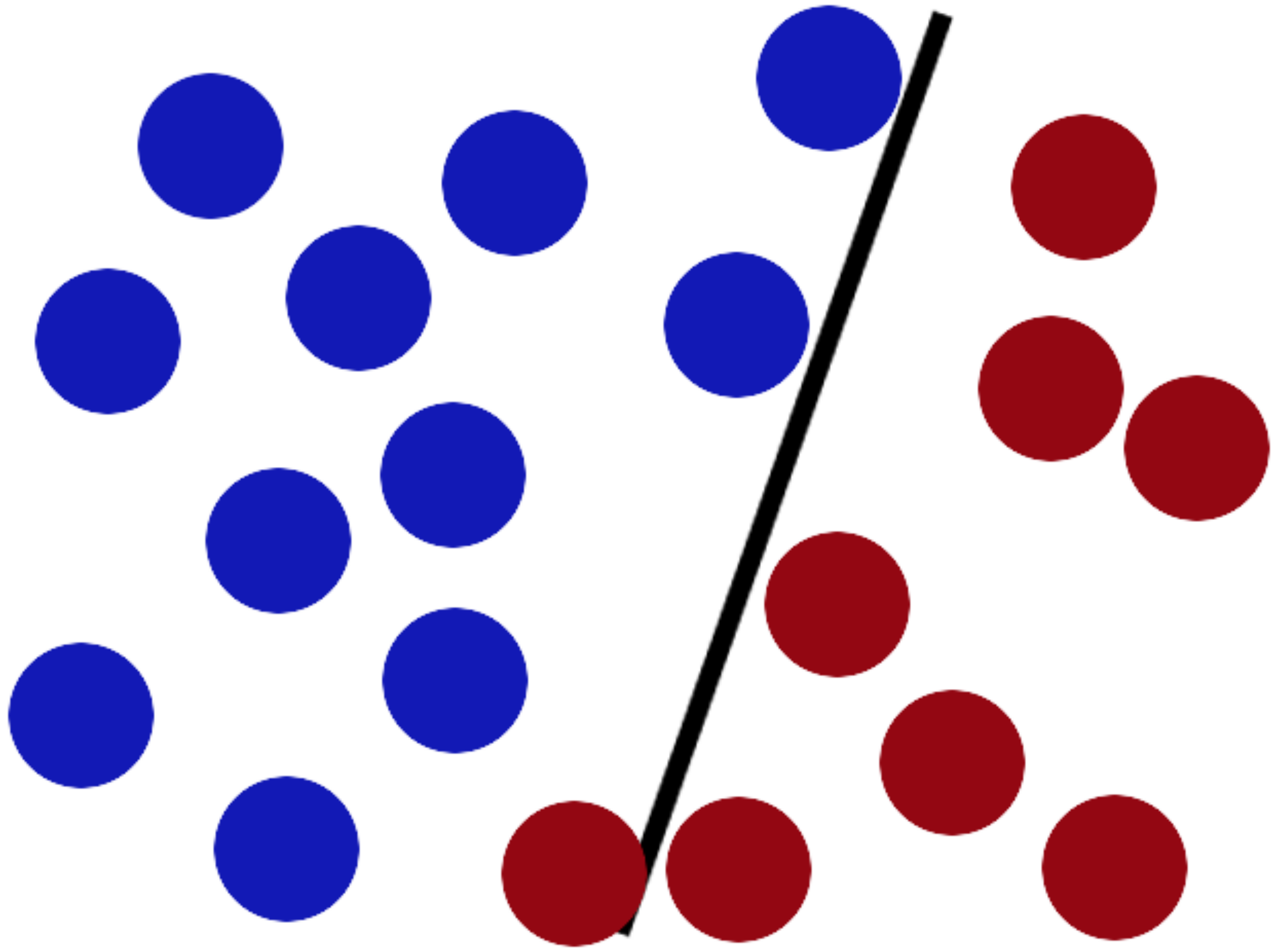
Separate these

Support Vector Machine (SVM)



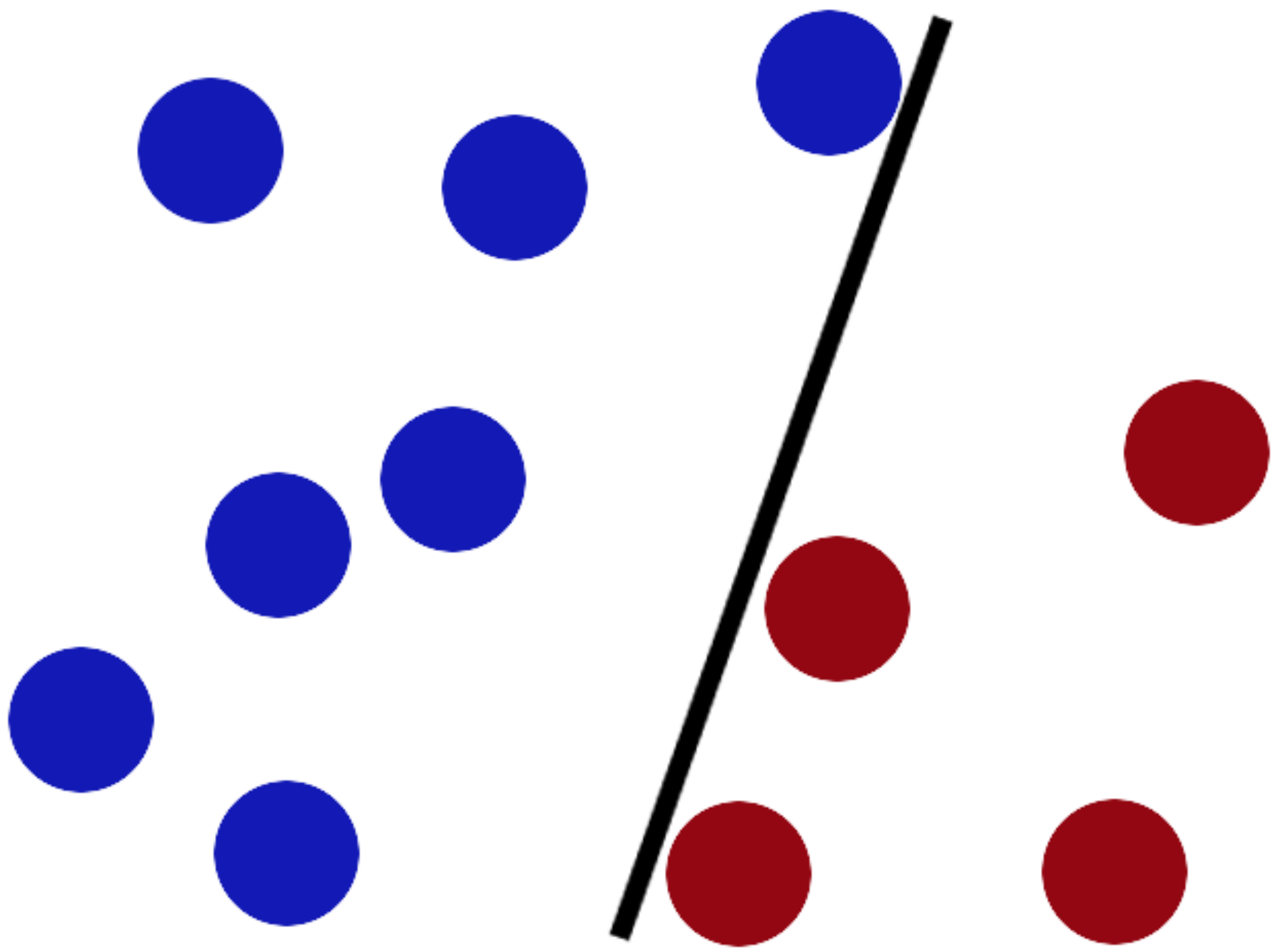
Draw a line

Support Vector Machine (SVM)



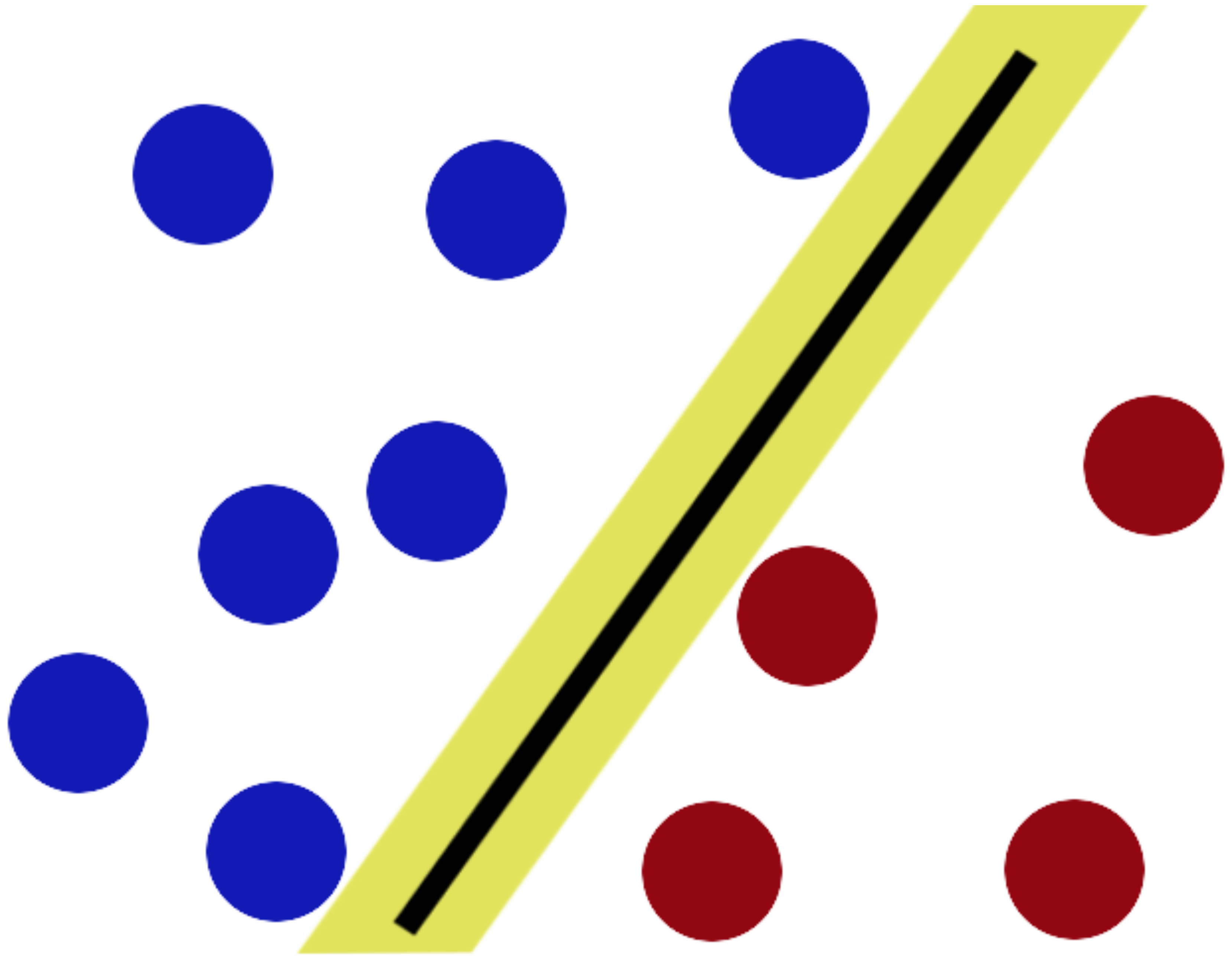
Add some more points, doesn’t work great now

Support Vector Machine (SVM)



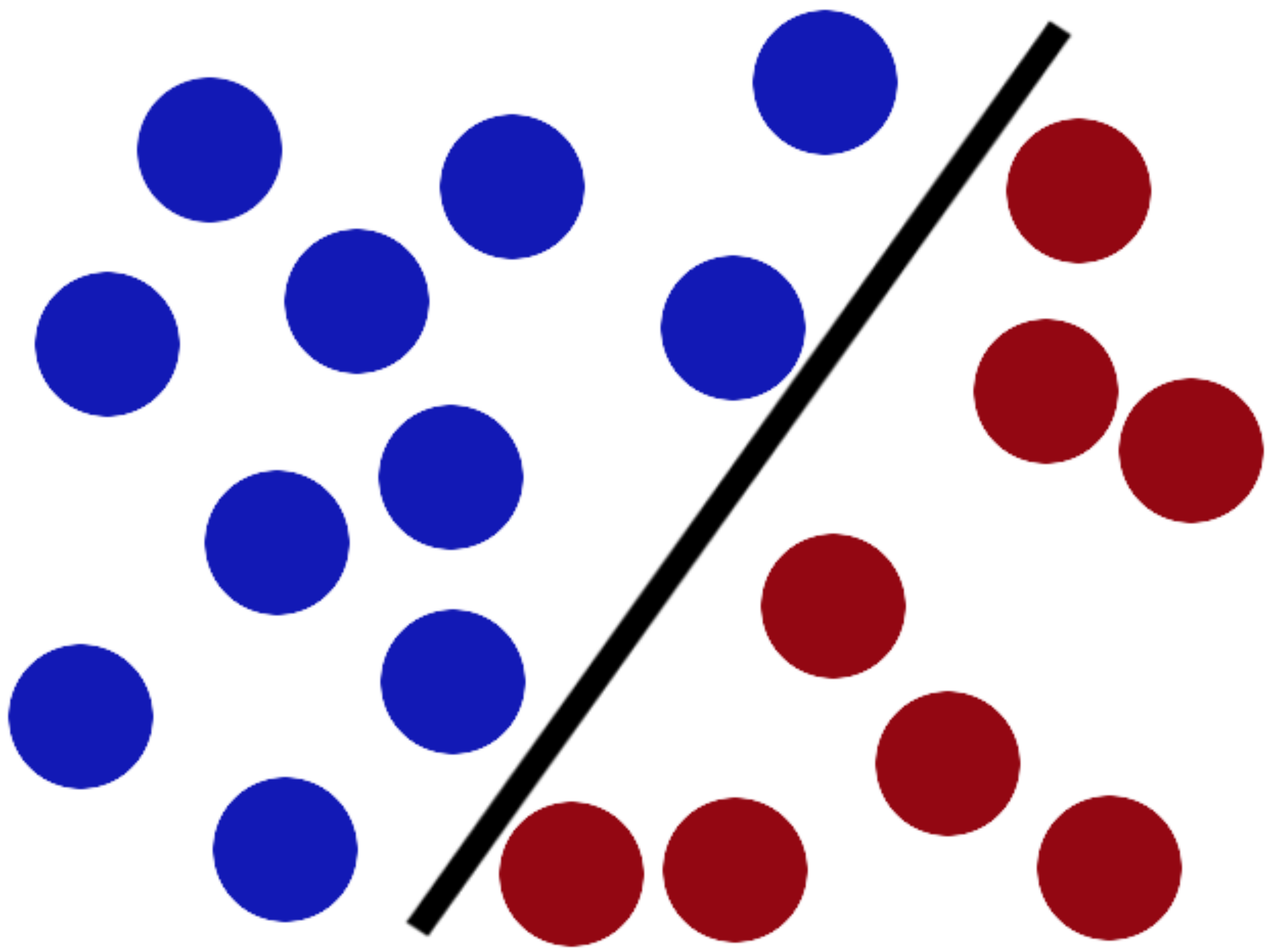
Original line

Support Vector Machine (SVM)



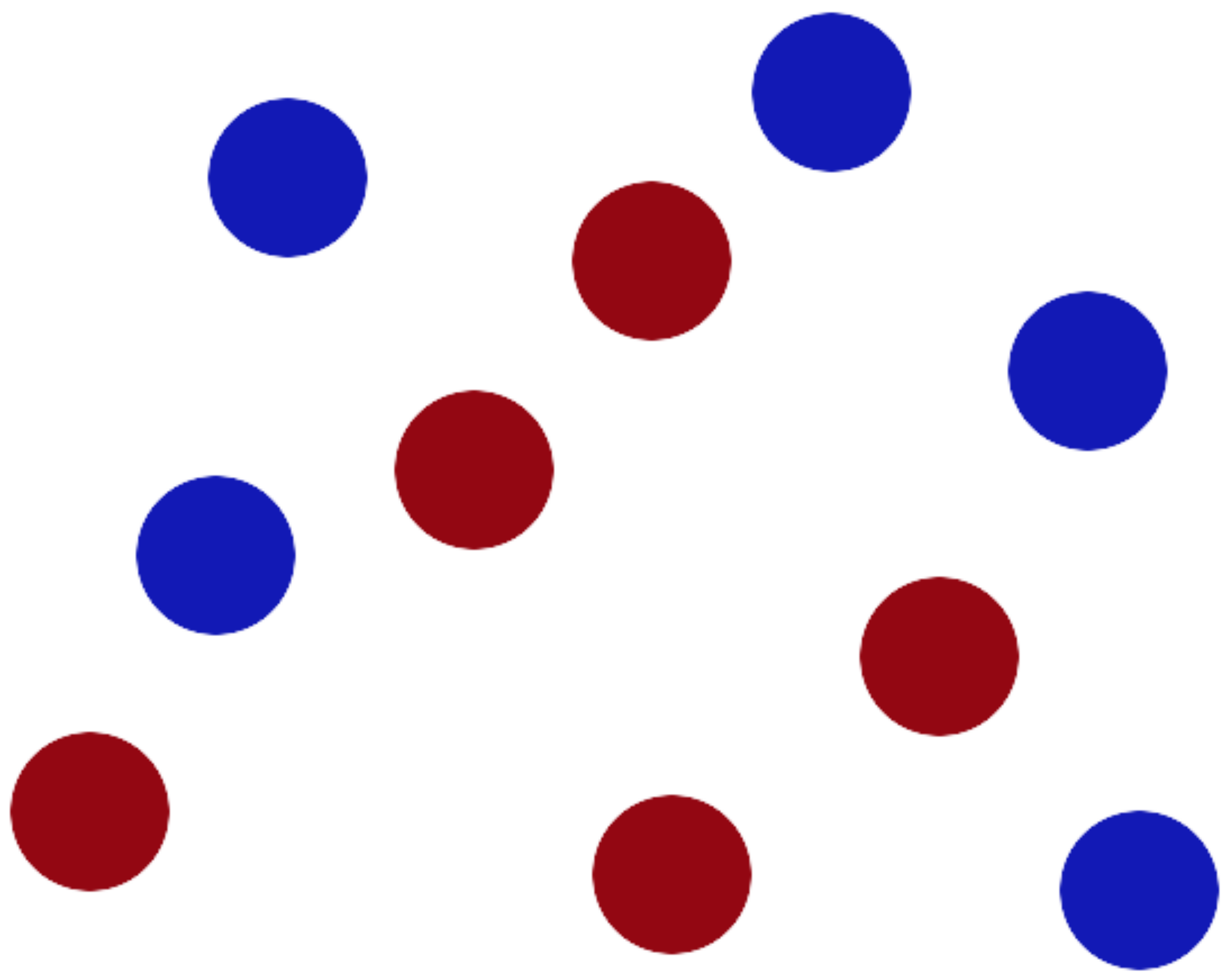
New line, try to have as large margins around the line as possible

Support Vector Machine (SVM)



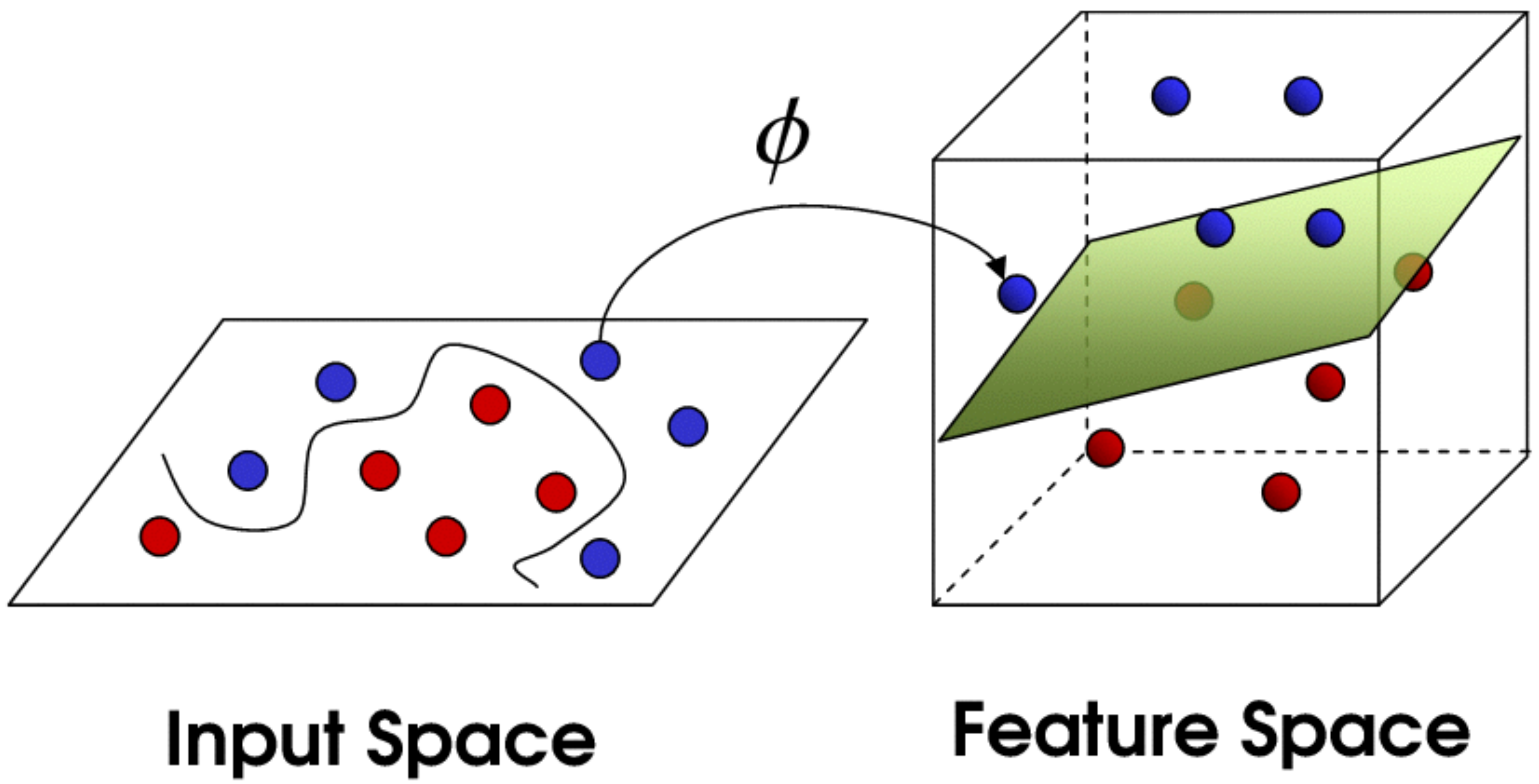
Works better now

Support Vector Machine (SVM)



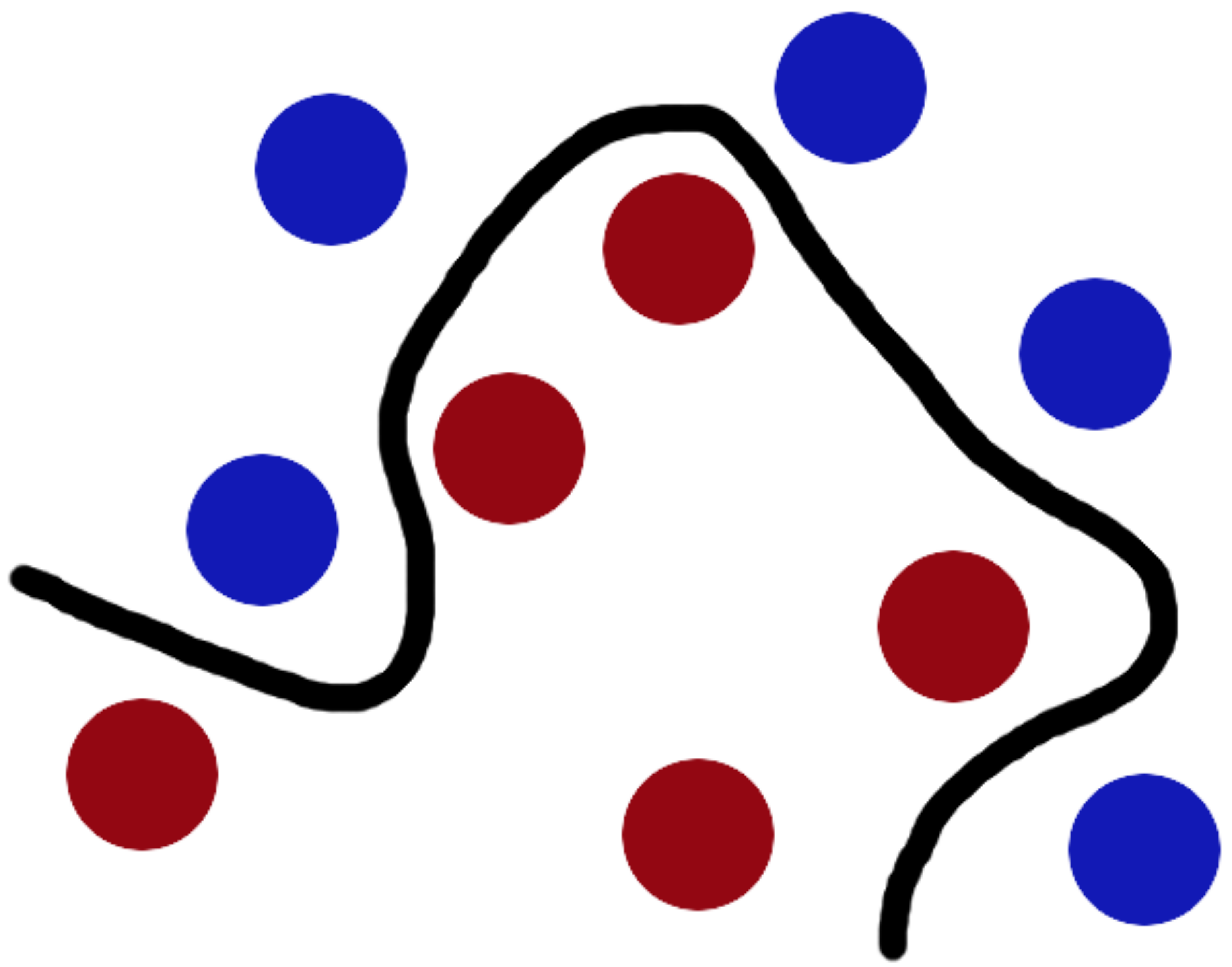
How to draw line now?

Support Vector Machine (SVM)



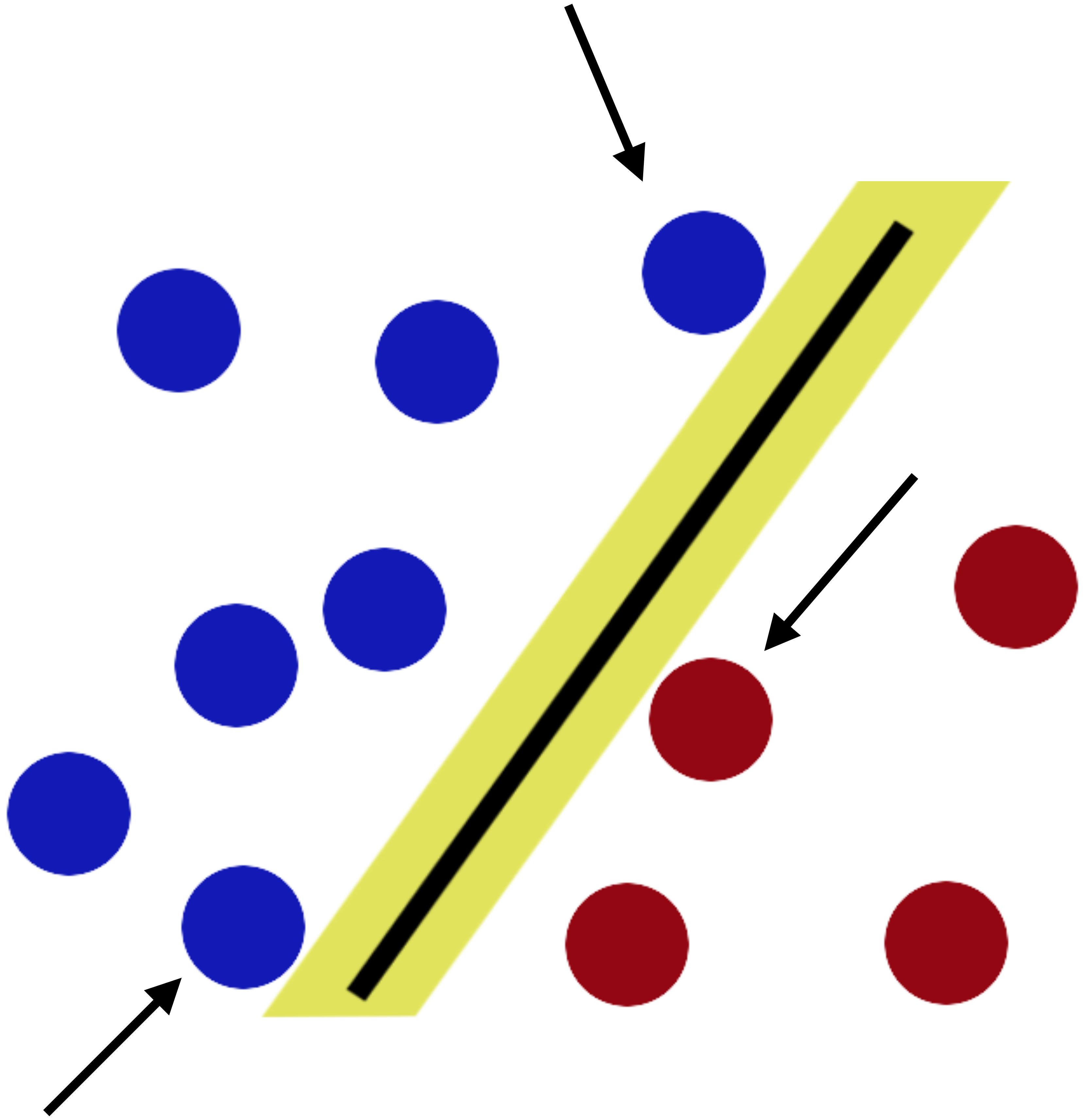
Separate them out into another dimension, then hyperplane can divide

Support Vector Machine (SVM)



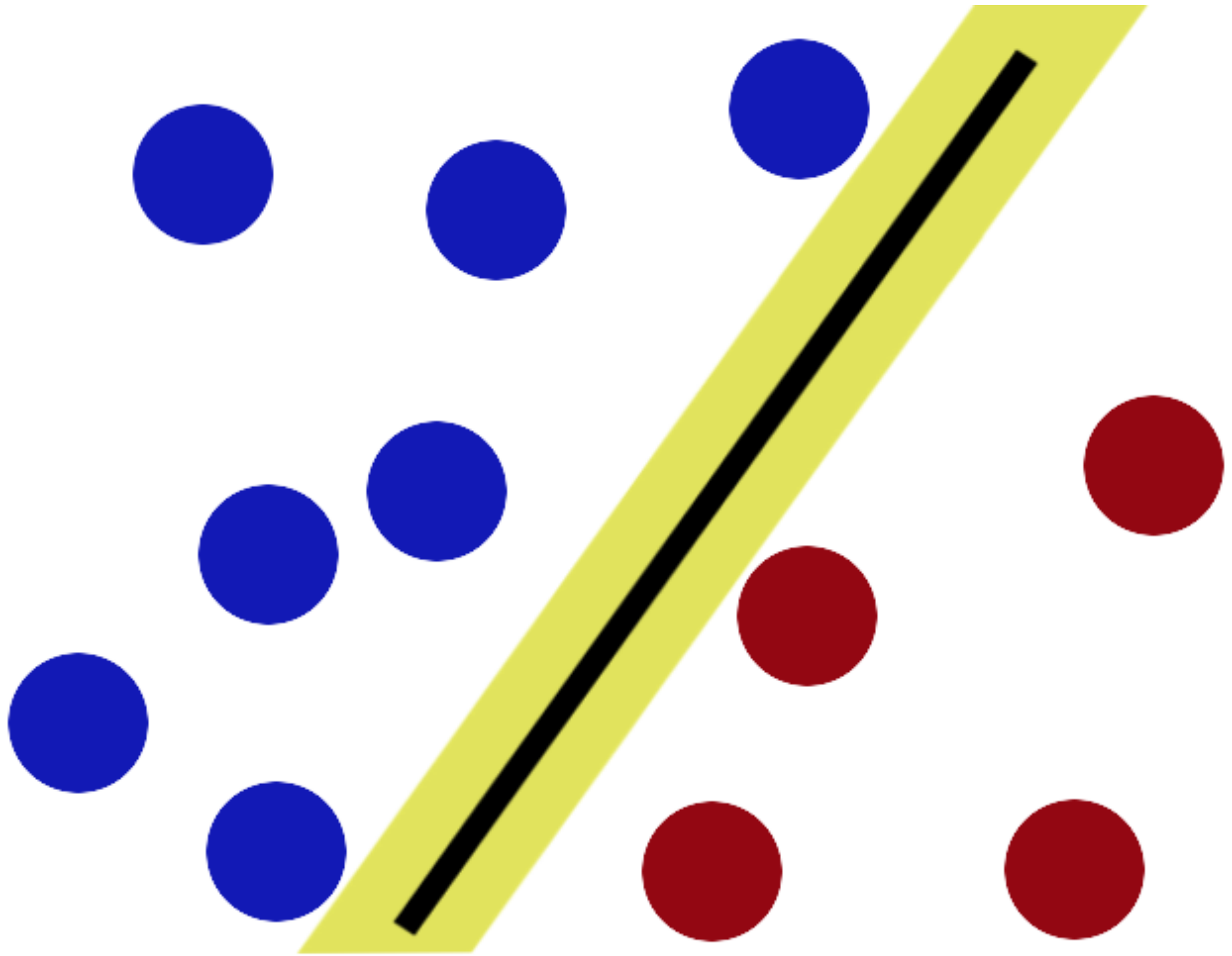
Now we have them separated

Support Vector Machine (SVM)



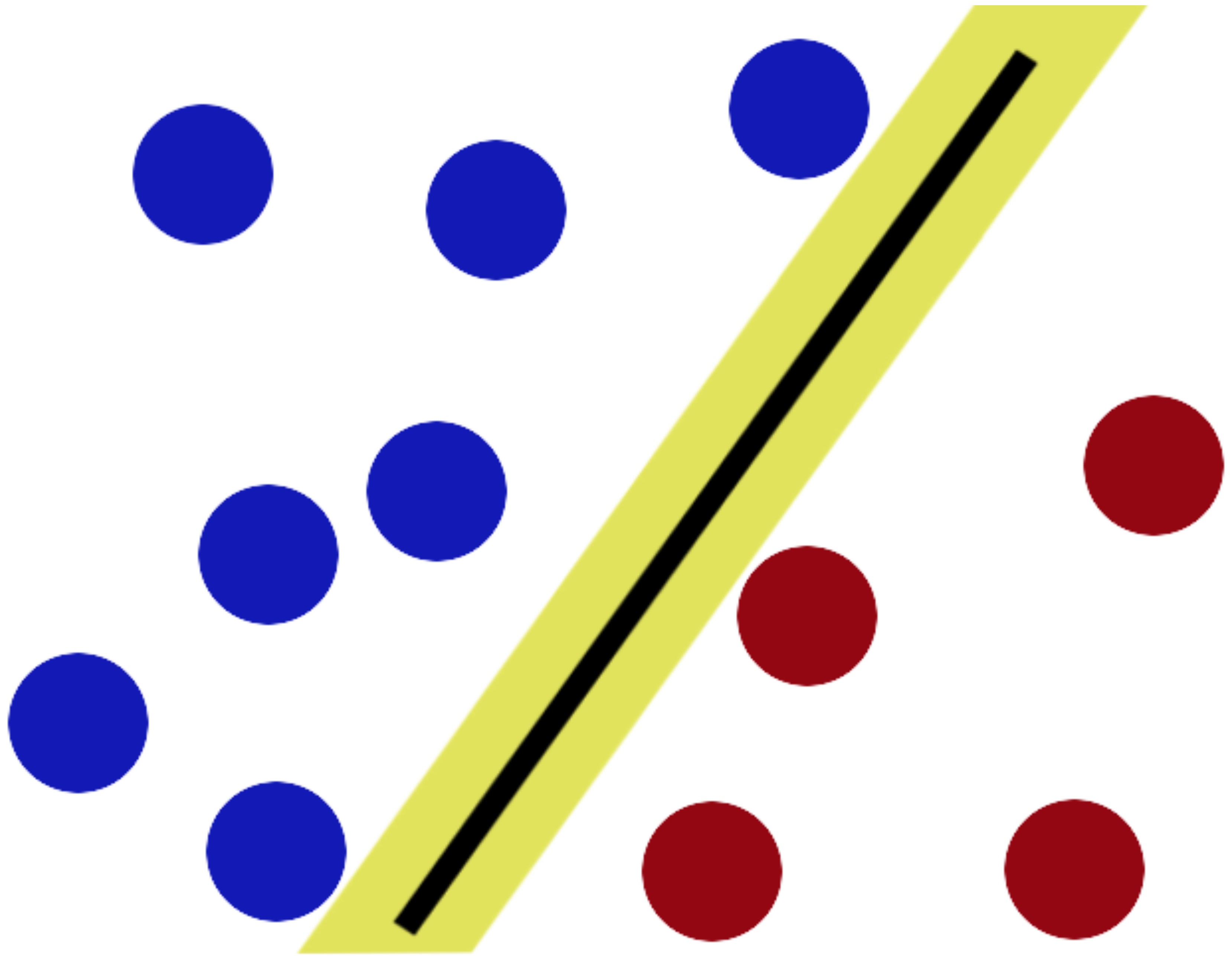
Support vectors are the points closest to margins, determining the margins

Support Vector Machine (SVM)



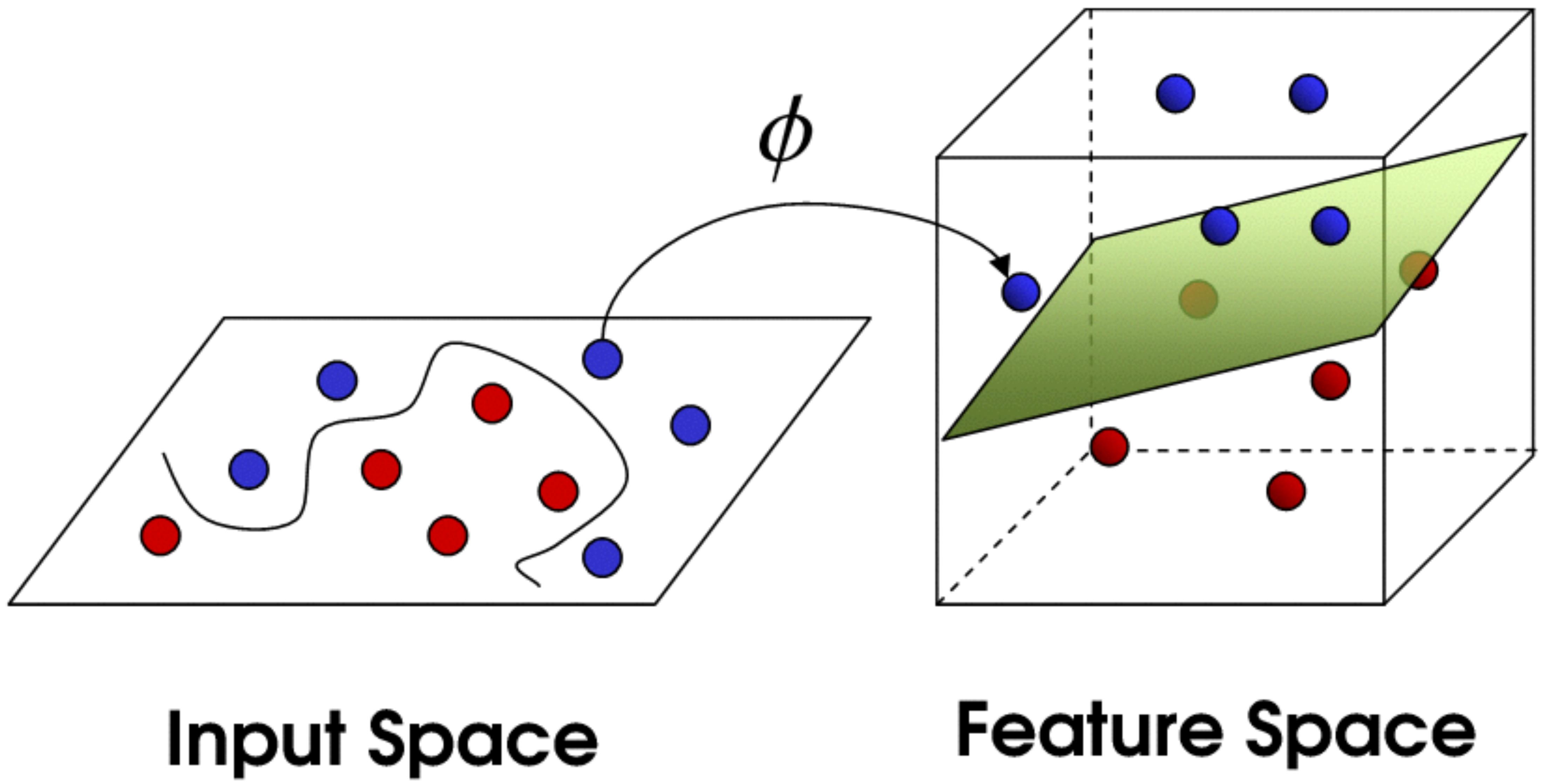
Hyperplane goes down the middle, and is straight thing, n-1 dimensions

Support Vector Machine (SVM)



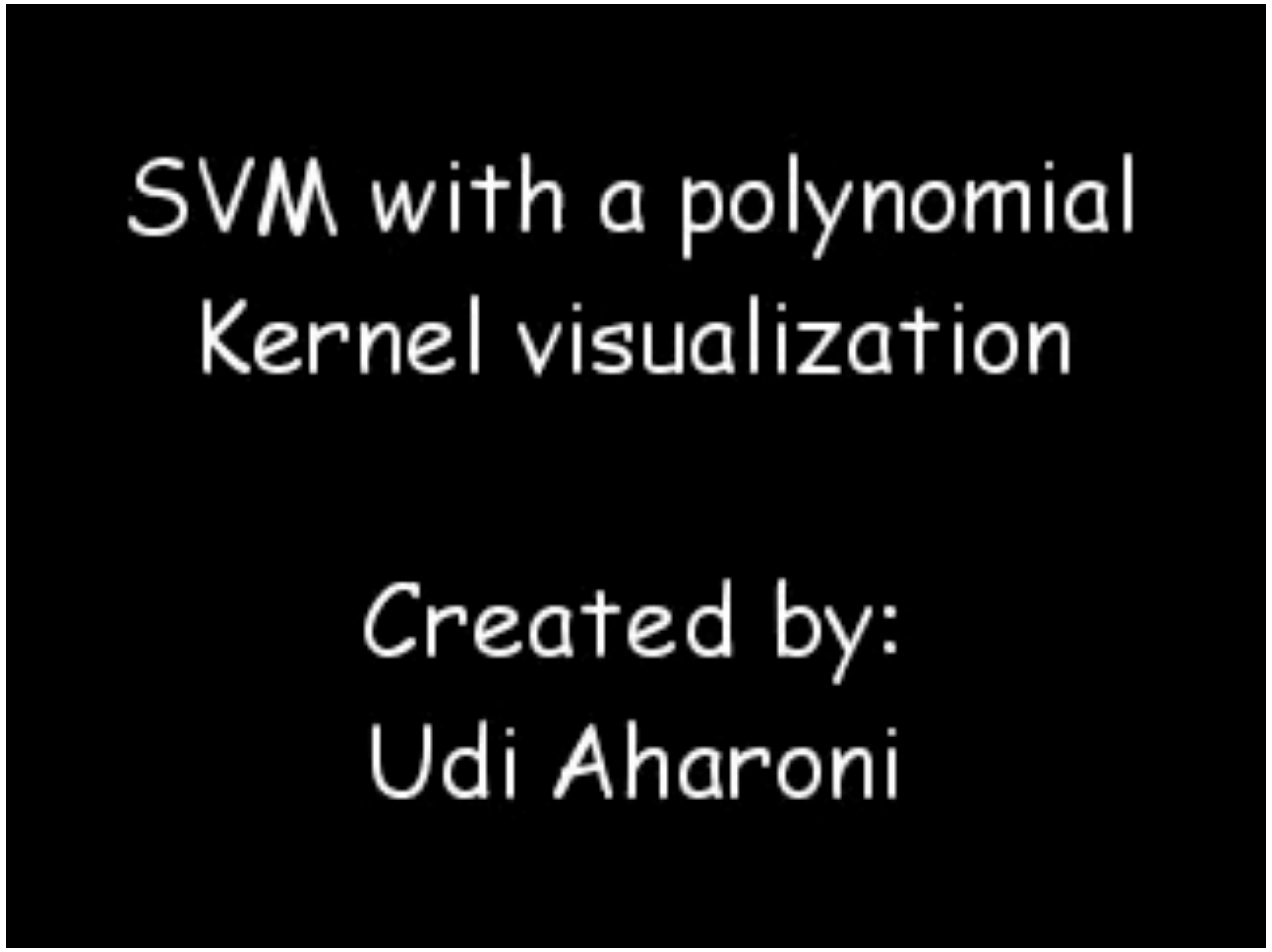
If using 2 dimensions only then hyperplane would be a line (n-1 is 1 dimension)

Support Vector Machine (SVM)



If using 3 dimensions, hyperplane would be 2 dimensions

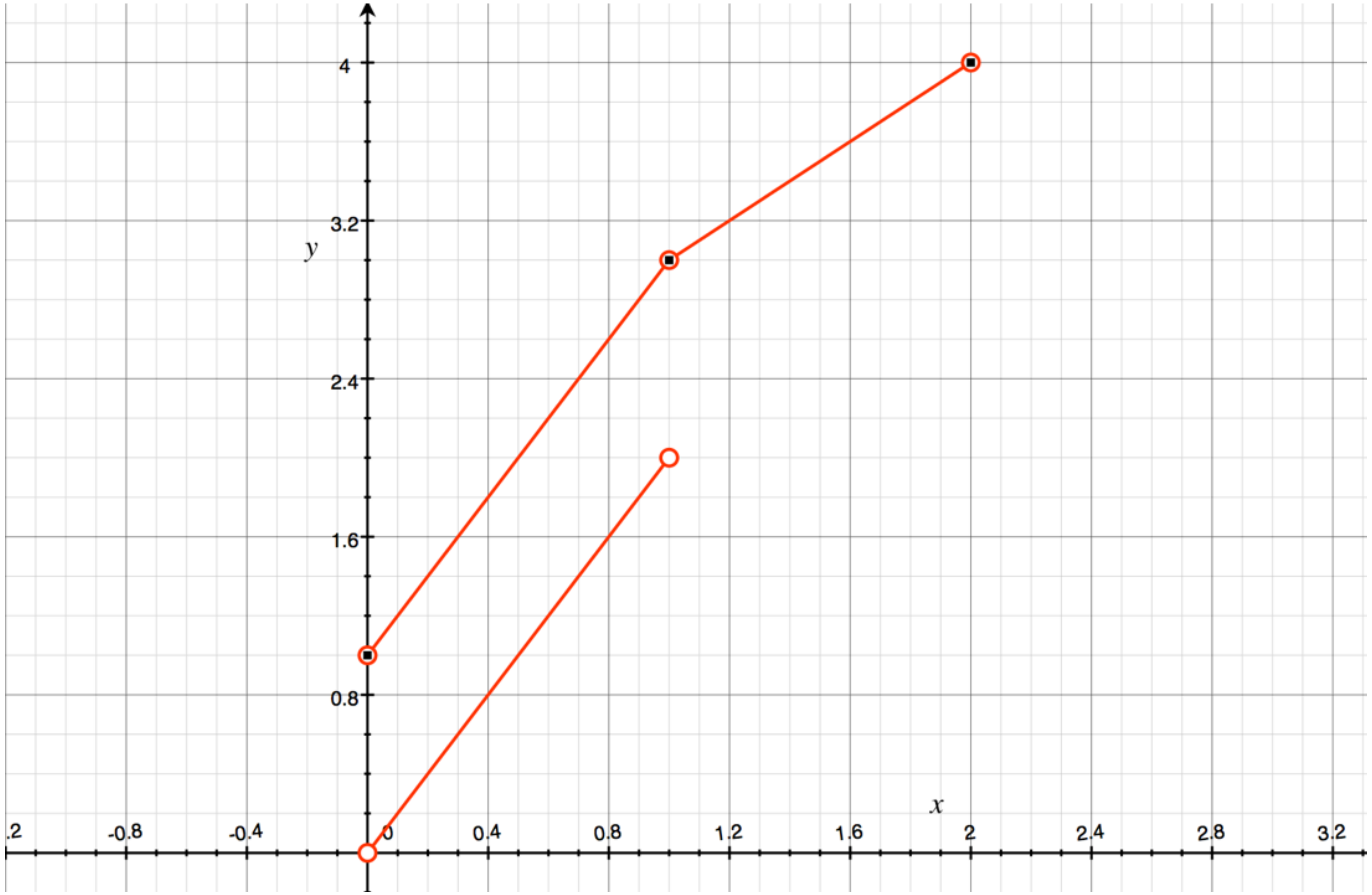
Support Vector Machine (SVM)



How does this work?

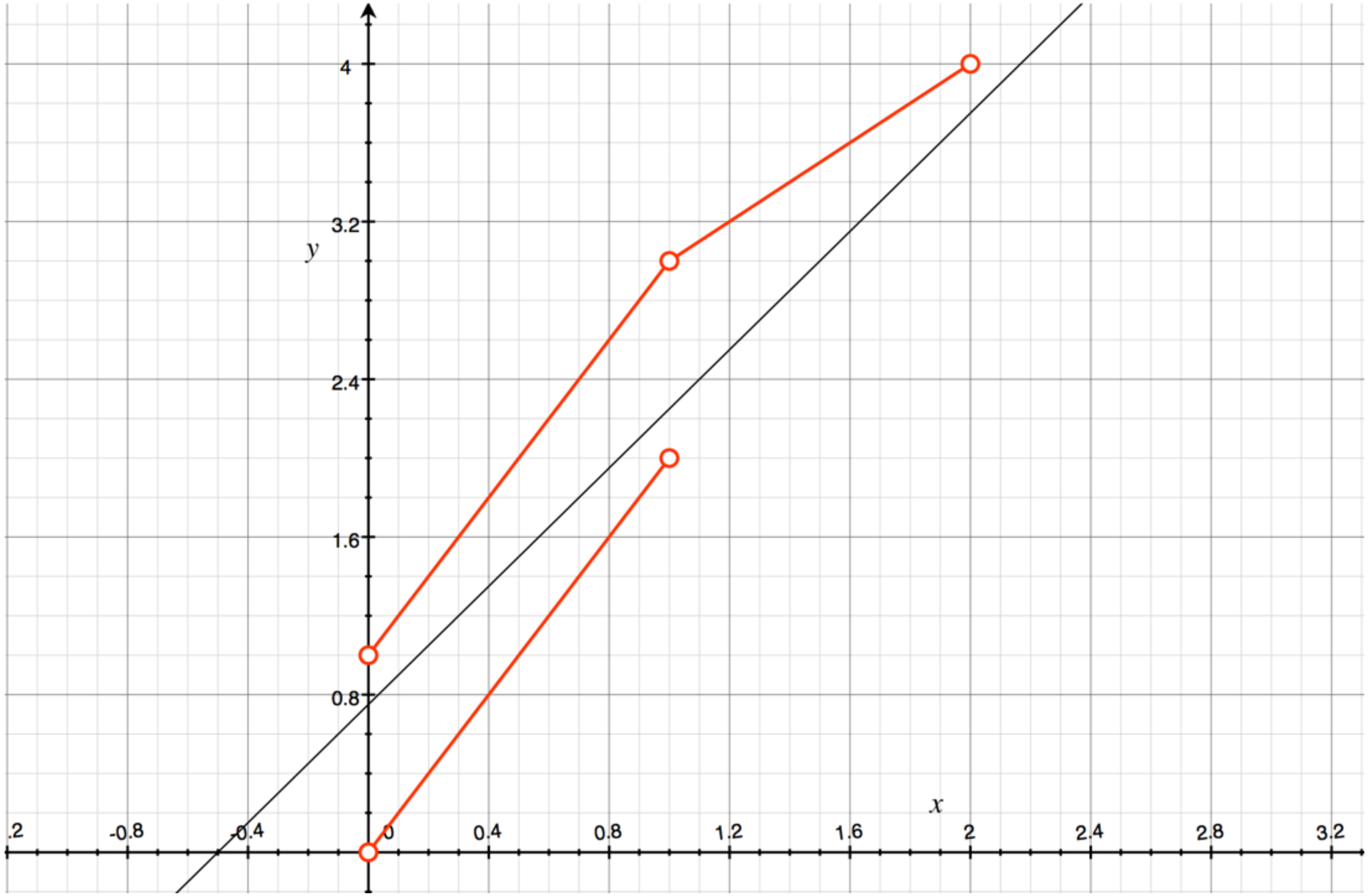
Support Vector Machine (SVM)

Support Vector Machine (SVM)



Transform even in 2D

Support Vector Machine (SVM)



Support Vector Machine (SVM)

Kernels!

Finds similarity between things (very over simplified explanation)

Support Vector Machine (SVM)

Kernel Trick:

If we find the dot product then we don't have to transpose to higher feature space!!!!

Support Vector Machine (SVM)

For example:

Polynomial Kernel:

$$K(\vec{x},\vec{y})=(\vec{x}\cdot\vec{y}+c)^{d}$$

(c is a constant)

Support Vector Machine (SVM)

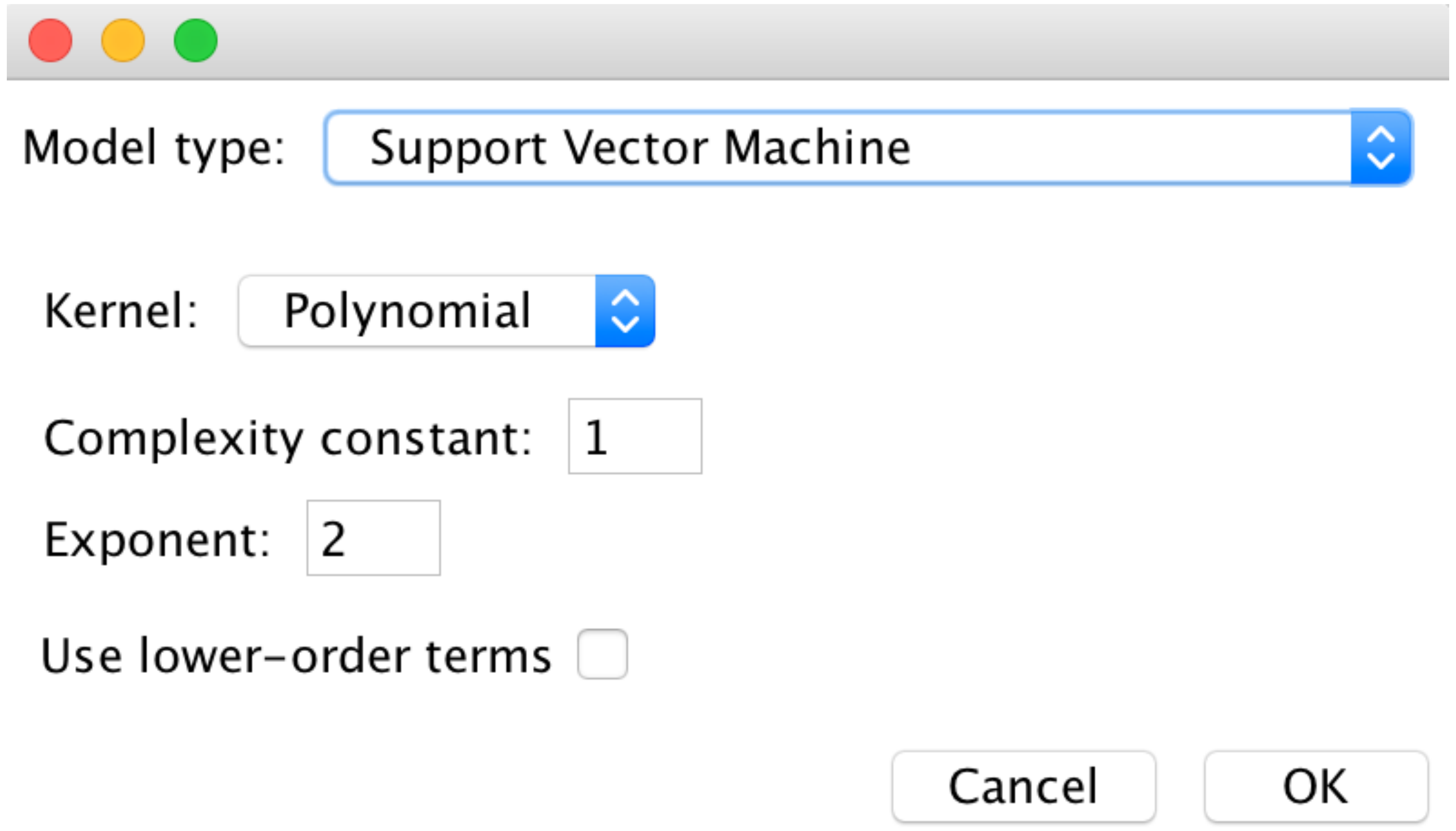
**KERNEL
TRICK!!!**


MAGIC

Support Vector Machine (SVM)

SVM’s can give us very smooth shapes with complex data

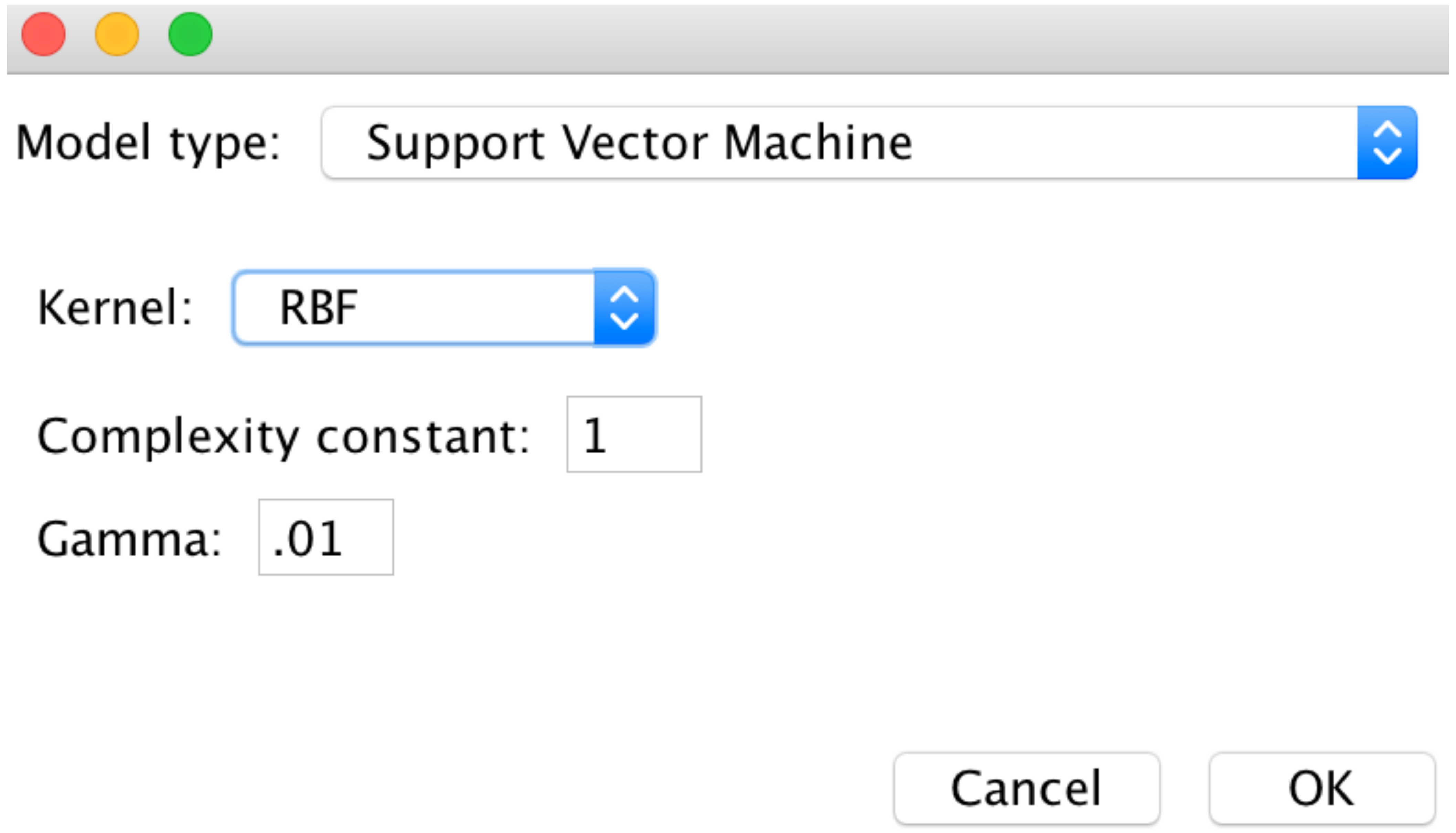
Support Vector Machine (SVM)



Support Vector Machine (SVM)

C is large could overfit, C is small could underfit

Support Vector Machine (SVM)

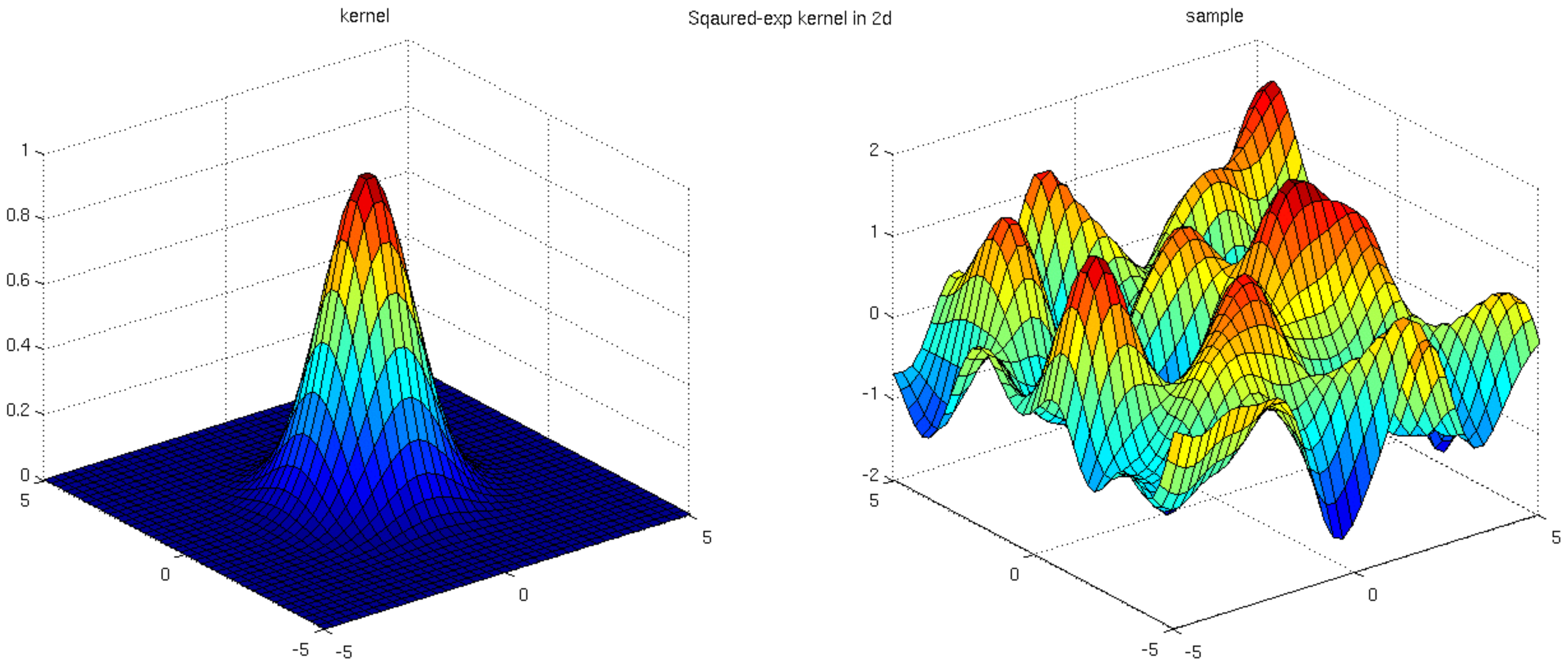


Radial Basis Function (Gaussian distributed distance from anchor point)

Gamma is inverse of how wide radial base is

Low Gamma will underfit, high Gamma will overfit

Support Vector Machine (SVM)



Regression

Regression

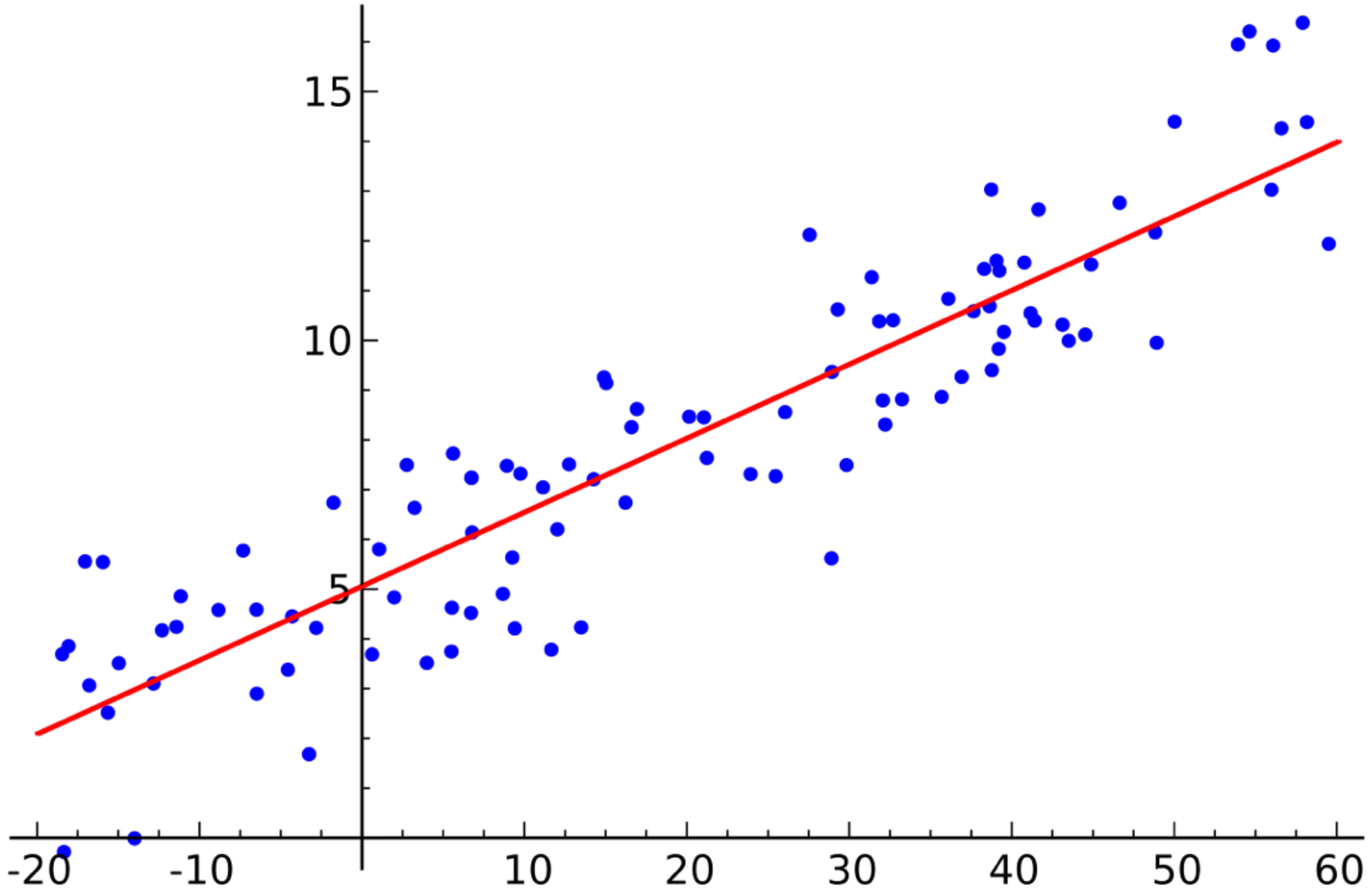
- Classification gives us labels for inputs
- Regression computes a number based on the inputs
- Values will change smoothly as input changes
- Could be values not in the training set

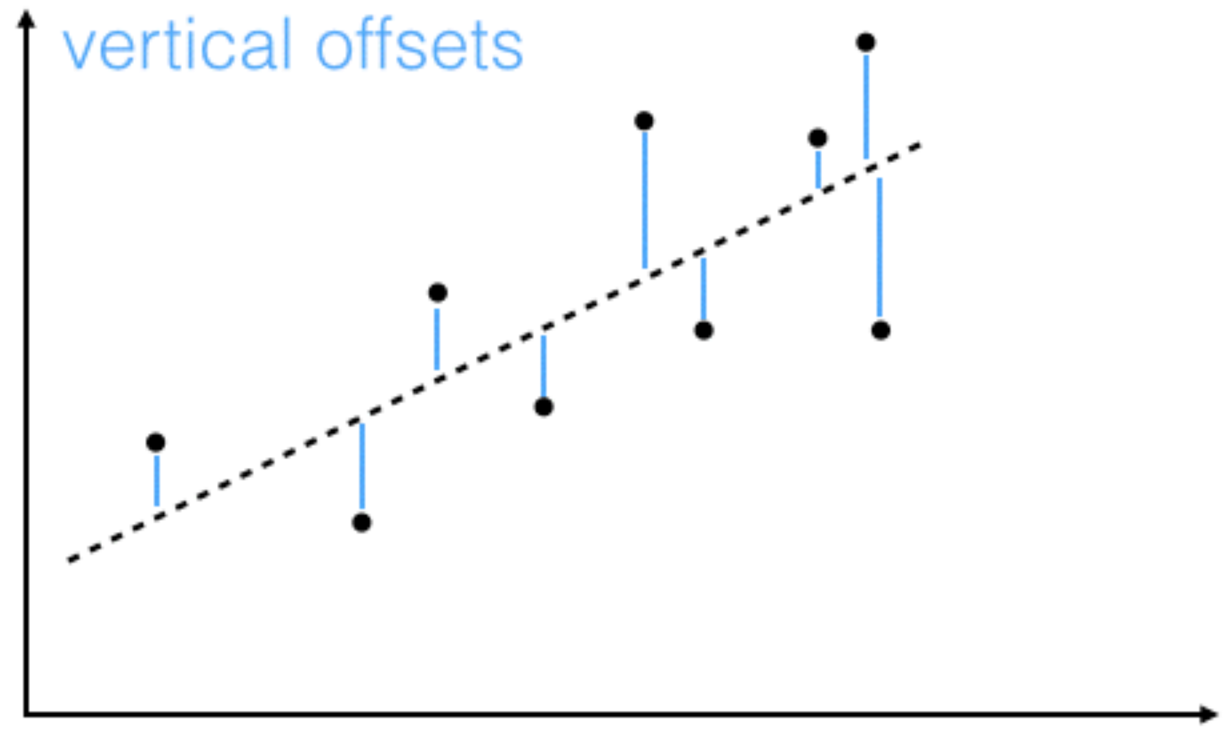
Regression

Linear Regression

find a straight line: $\underset{\text{Output}}{y} = a\underset{\text{Input}}{x} + b$

Least Squares

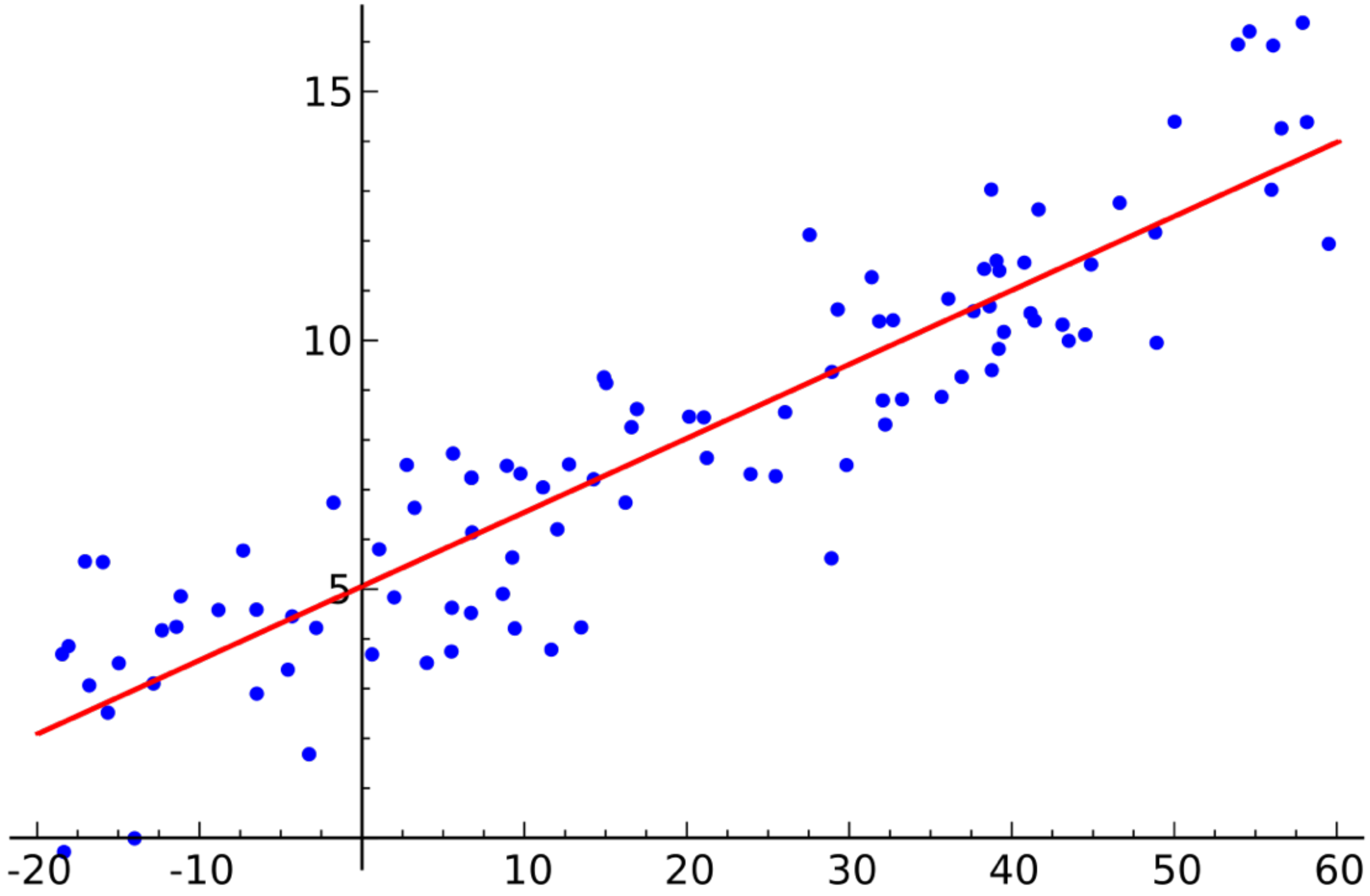


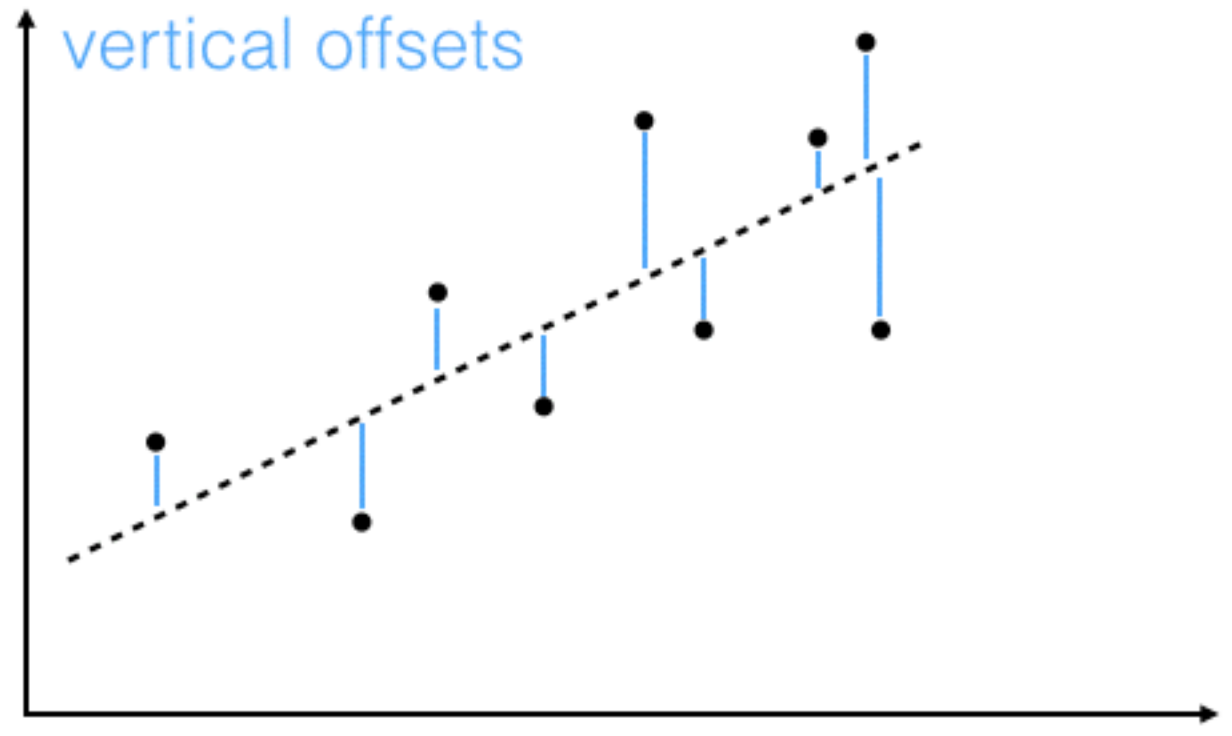


$$S = \sum_{i=1}^{n} {r_i}^2$$

Get differences of each training point and points on line, square the difference, sum everything to get total error: makes larger differences even larger

Linear Regression





$$\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\alpha} = \bar{y} - \hat{\beta}\,\bar{x}$$

Regression

Polynomial Regression

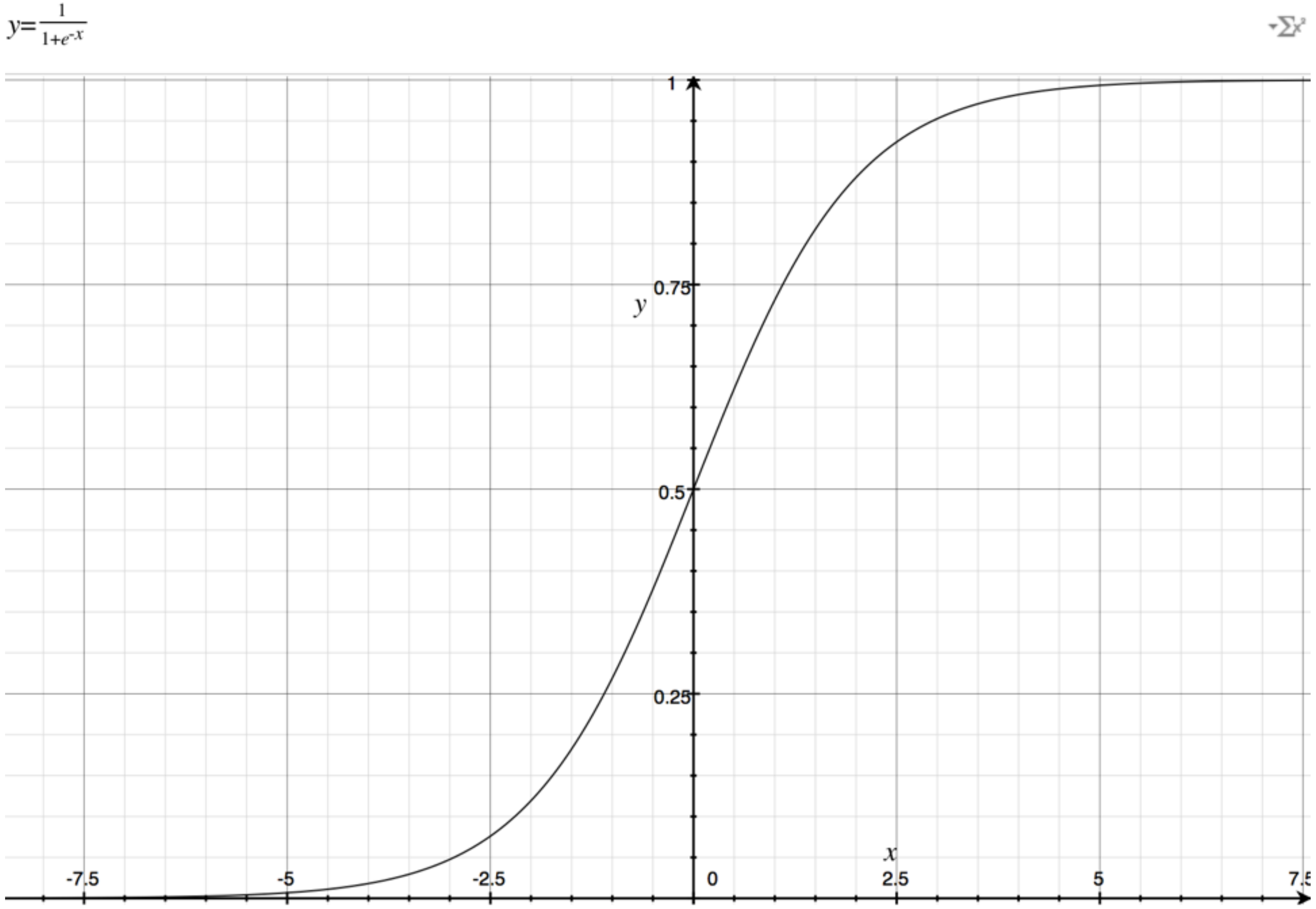
parabola: $y = ax^2 + bx + c$

Regression

Higher orders allow for more complexity

3rd order: $y = ax^3 + bx^2 + cx + d$

Neural Network



Neuron Activation Function: Sigmoid

Interactive linear regression

https://kwichmann.github.io/ml_sandbox/linear_regression_diagnostics/