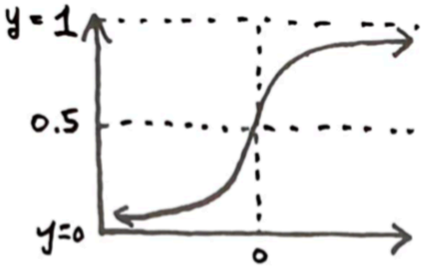
# 5.7 Supervised Learning Algorithms

박달님 03.19.21

# 5.7.1 Probabilisitic Supervised Learning

Sigmoid function




$$f(x) = \frac{1}{1+e^{-x}} \cdot \frac{e^x}{e^x} = \frac{e^x}{1+e^x}$$

$x = -\infty : 0$
$x = 0 : \frac{1}{2}$
$x = \infty : 1$

$$Loss = \sum_i (y_i - f(x_i, w)^2)$$

$$Loss = \sum_i (y_i - f(x_i, w)^{\textcircled{2}})$$ ← Linear이면 제곱이 없음. becomes very non-linear

gradient = 0 로 minimize 가능.

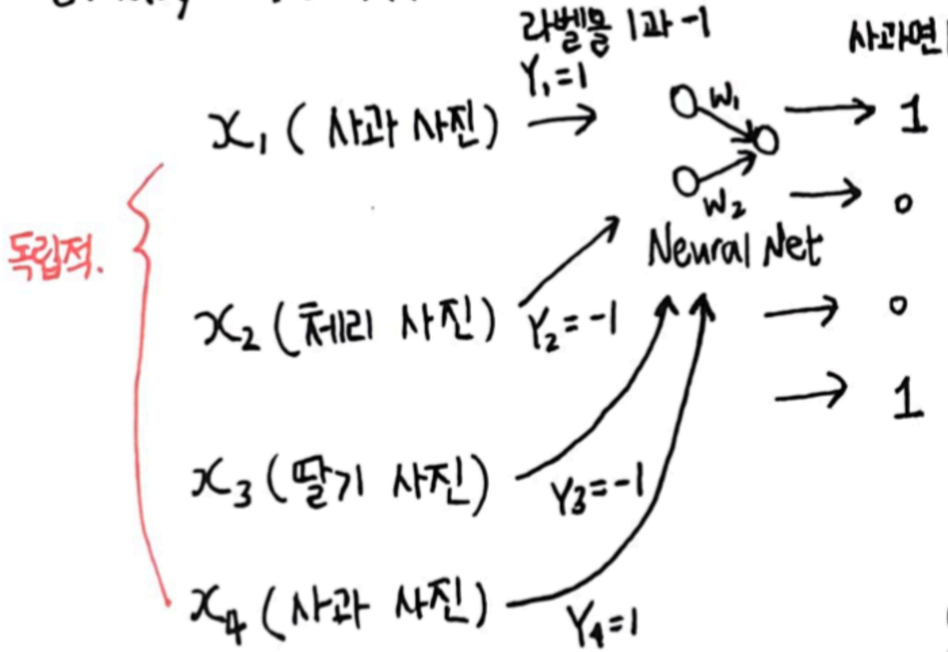
제곱이 있으면 $f(x_i - w)$가 weight에 대응해서 non-linear.

떠버린다. 이미 Gradient가 0으로 update 되어버림.

Sigmoid function의 문제점.

∴ Loss function은 $y = \frac{e^x}{1+e^x}$에 맞는 likelihood를 가지고 loss function을 세운다.

# Binary classification




0~1 사이의 값 → 사과 사진일 확률

조건부확률: $P(Y_i \mid X_i)$

$Y_i$: 라벨을 어떻게할지. / $X_i$: 어떤 사진이 주어짐.

출력값은 확률.

Sigmoid func.에 통과시킴으로써 조건부 확률을 얻는다

$$= \begin{cases} Sig(f(x_i, w)) & Y_i = 1 \\ 1 - Sig(f(x_i, w)) & Y_i = -1 \end{cases}$$

f가 neural net. 사과 사진이라고 판단할 확률.

sigmoid를 통과한 값이 사과 사진일 확률 2가지

사과 사진이 아니라고 판단할 확률.

사과 사진일때는 사과가 출력될 확률을 높이고 아니면 아니라고 출력될 확률을 높여야한다.

확률이 곱으로 표현되려면 독립이면 된다. ∴ 가능.

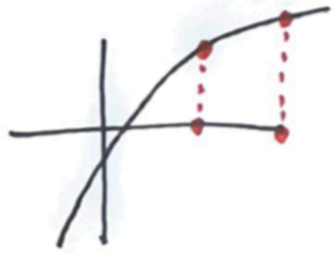
$$\text{Loss} = -\prod_{i=1}^{N} P(Y_i \mid X_i)$$

이 부분을 Maximize. 조건부 확률

Loss ↓

$P(Y_i \mid X_i)$ 부분이 알아서 1이면 $sig(f(x_i, w))$로 아니면 $1-sig(f(x_i, w))$로 값을 계속 곱해줌.

곱하기는 더하기로 바꿀 수 있다 → log를 씌우기.



log를 씌우나마나 minimize하는 w를 찾는것은 같다.

양수에 대해서 단조증가함수이기 때문에 (Monotonic)

log를 씌워도 된다.

Likelihood loss

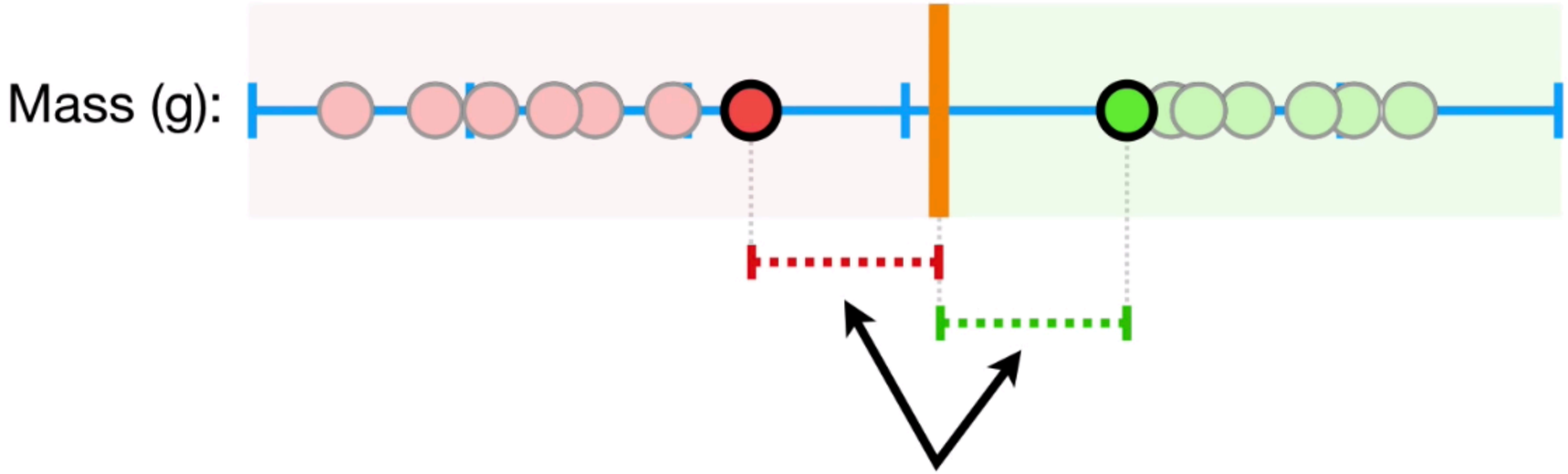
logistic regression에서 말하는 cost function.

$$\text{Loss} = -\sum_{i} \log P(Y_i \mid X_i)$$

maximizing the log-likelihood.

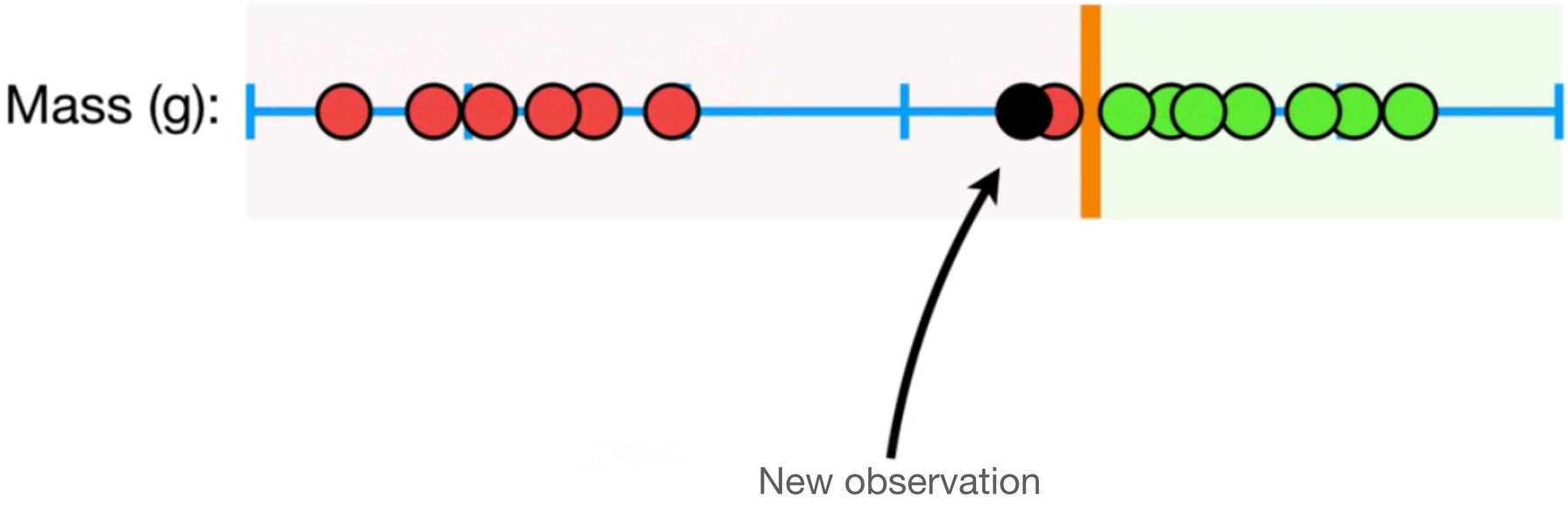
= minimizing the negative log-likelihood (NLL) using gradient descent.

# 5.7.2
# Support Vector Machine



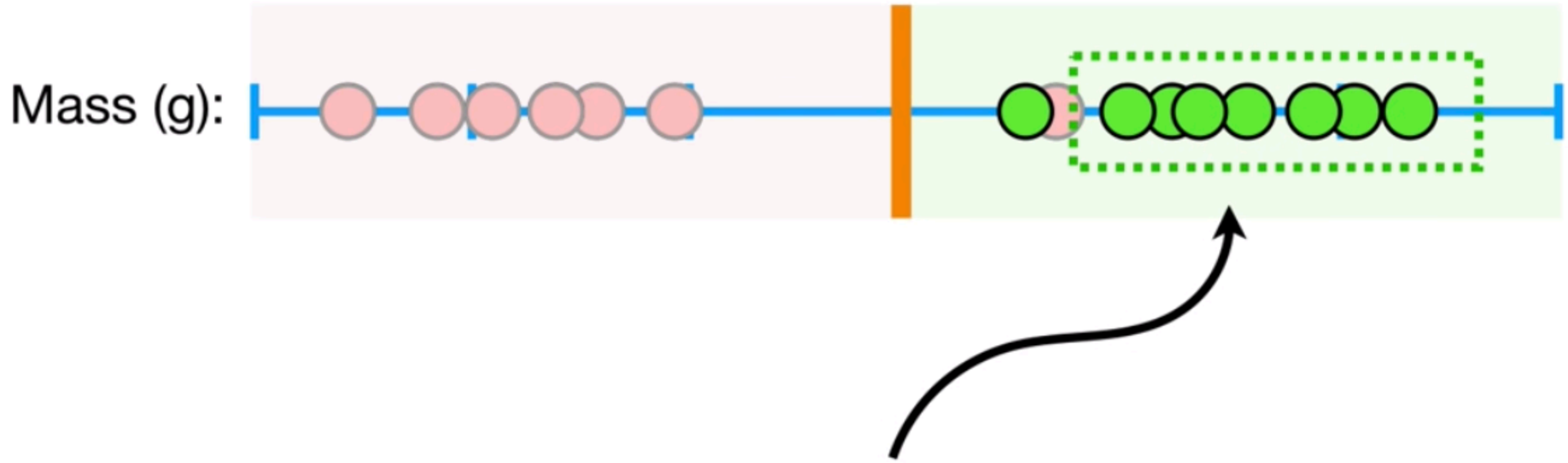

**Margin**

largest margin to make classifications:
**Maximal Margin Classifiers**

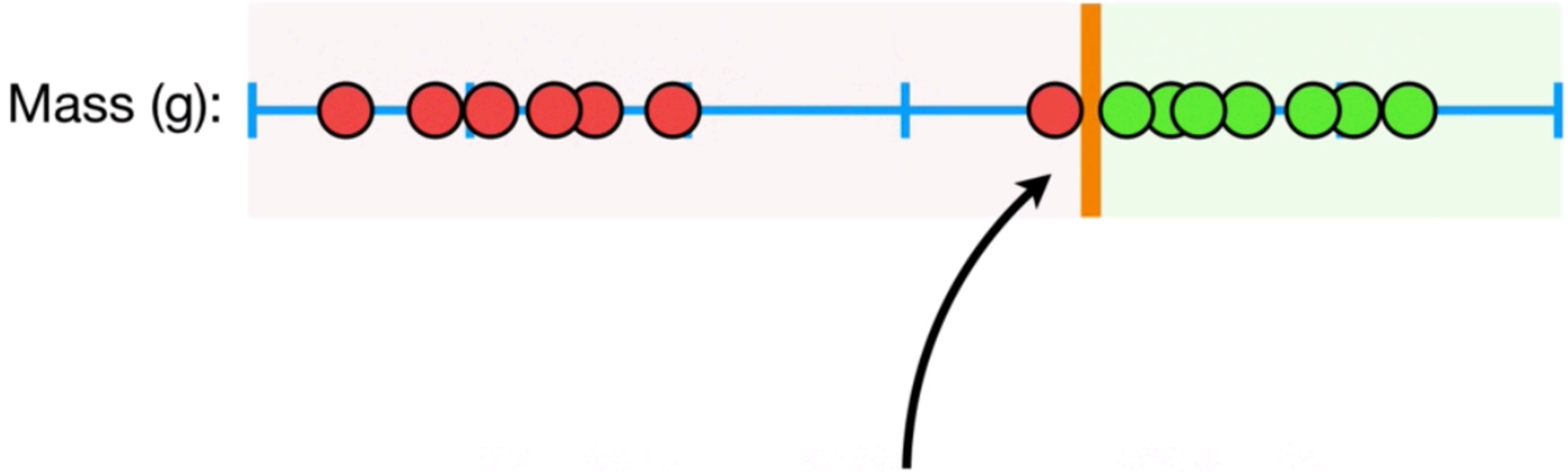



Maximal Margin Classifiers super **sensitive** to outliers in the training data

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

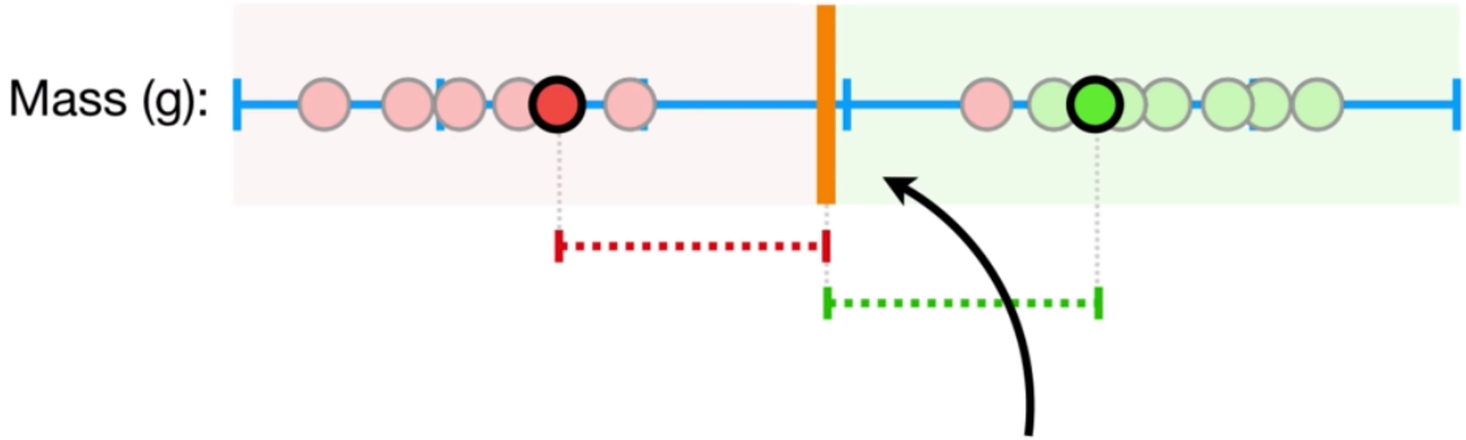



Choosing a threshold that allows misclassifications is an example of the **Bias/Variance Tradeoff** that plagues all of machine learning

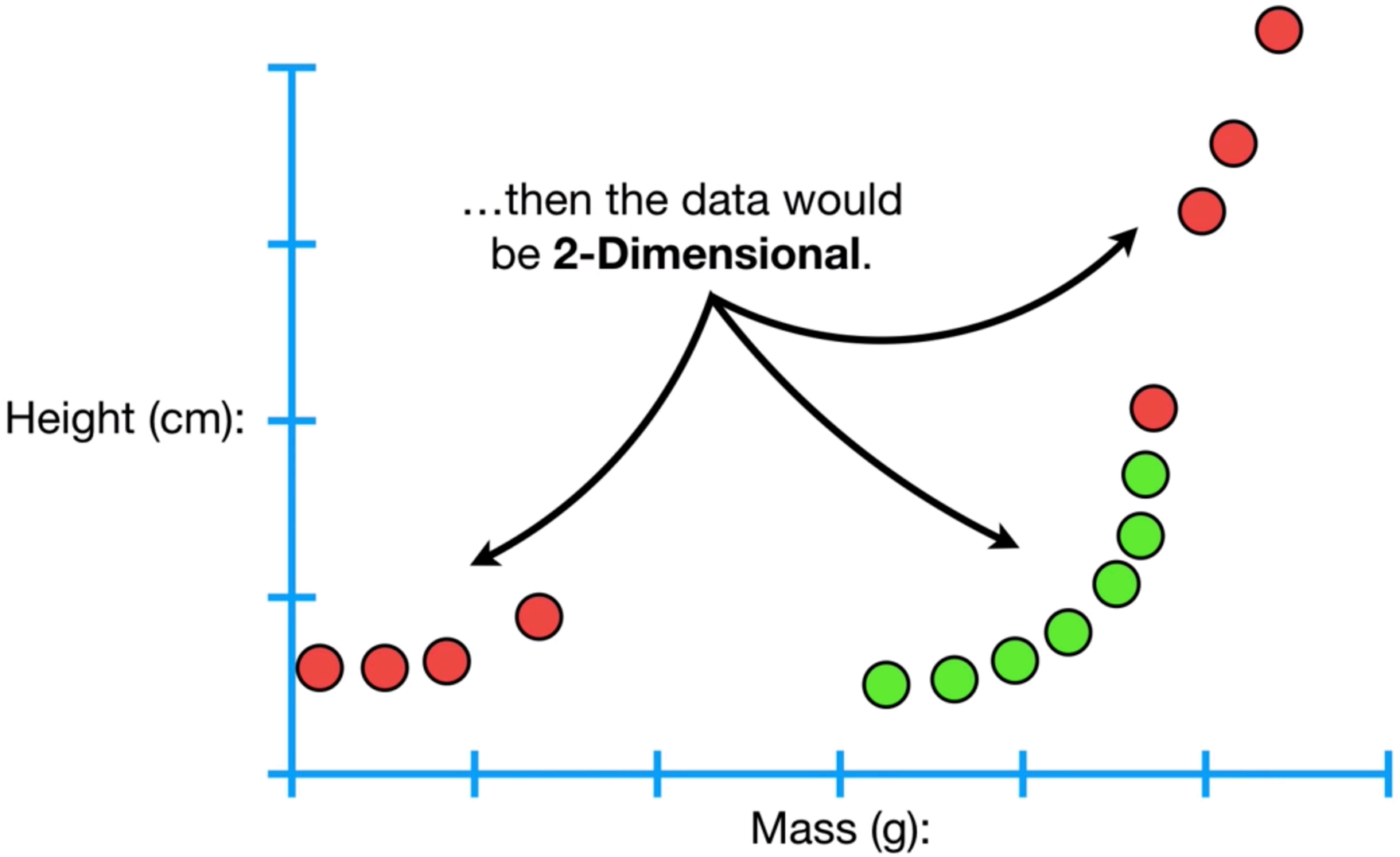


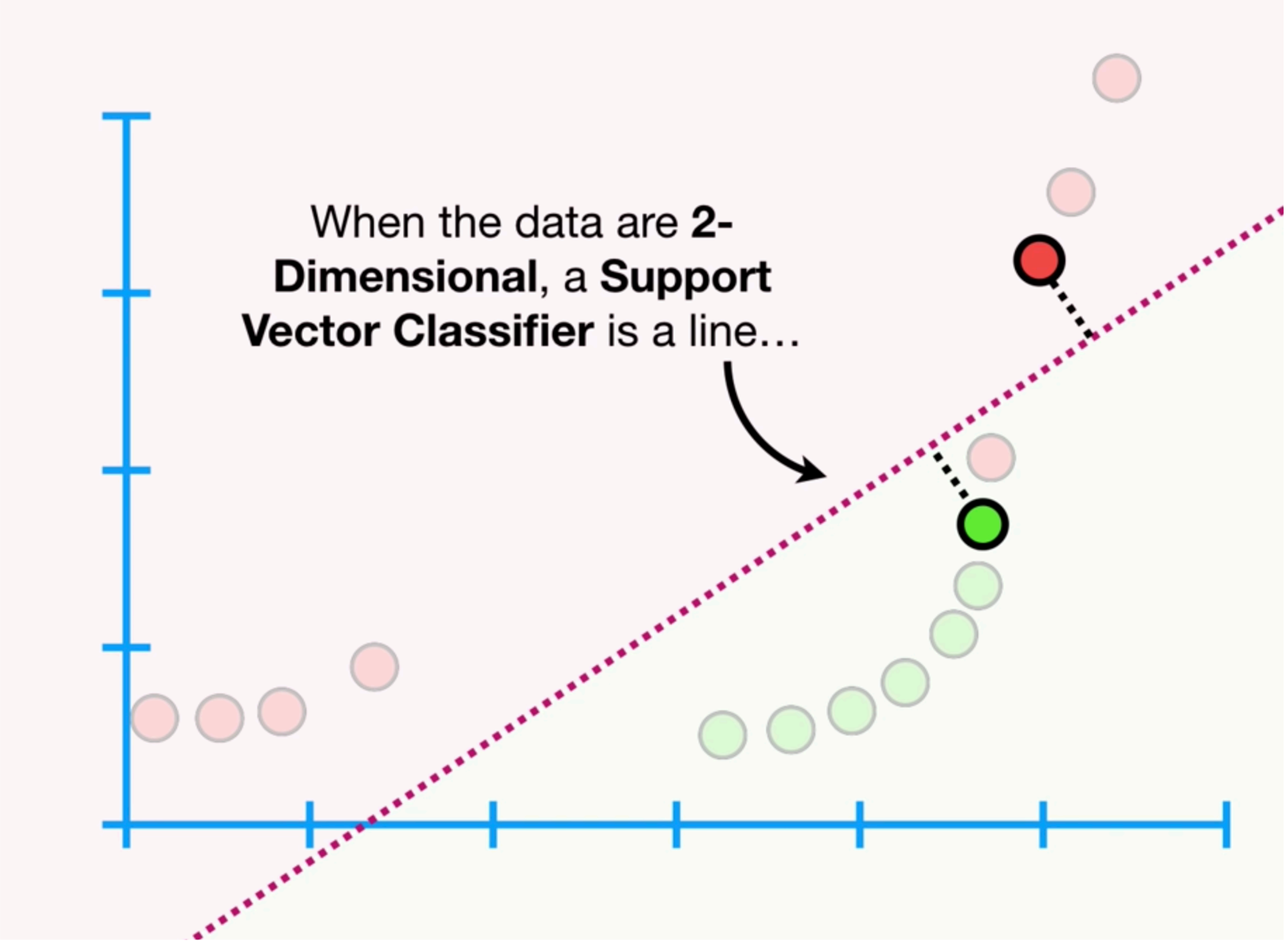

When we allow **misclassfications**, the distance between observations and the threshold is called a **Soft Margin**

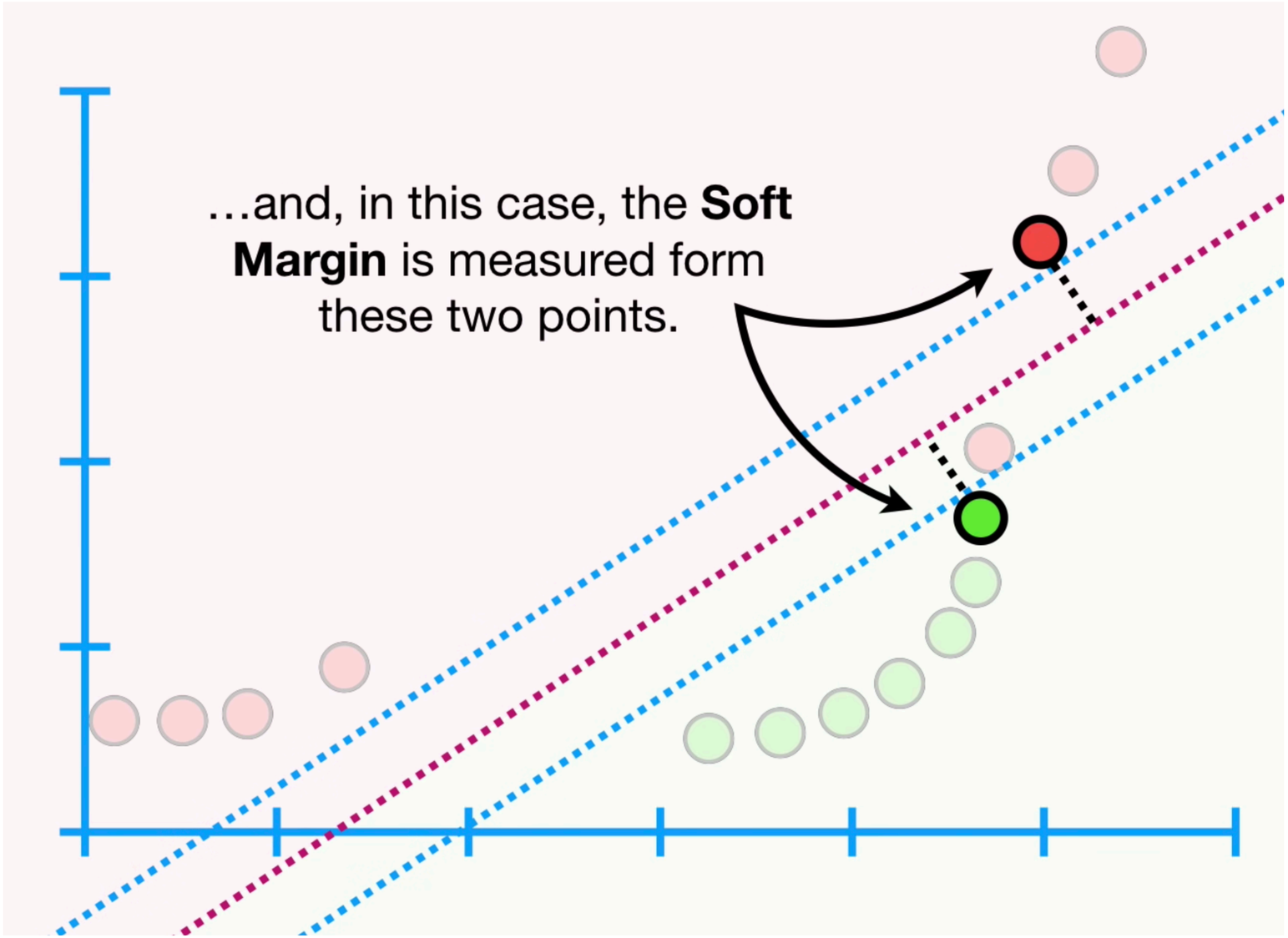
Decide which soft margin is better than the other by using **cross validation** to determine how many misclassifications and observations to allow inside of the soft margin to get the best classification

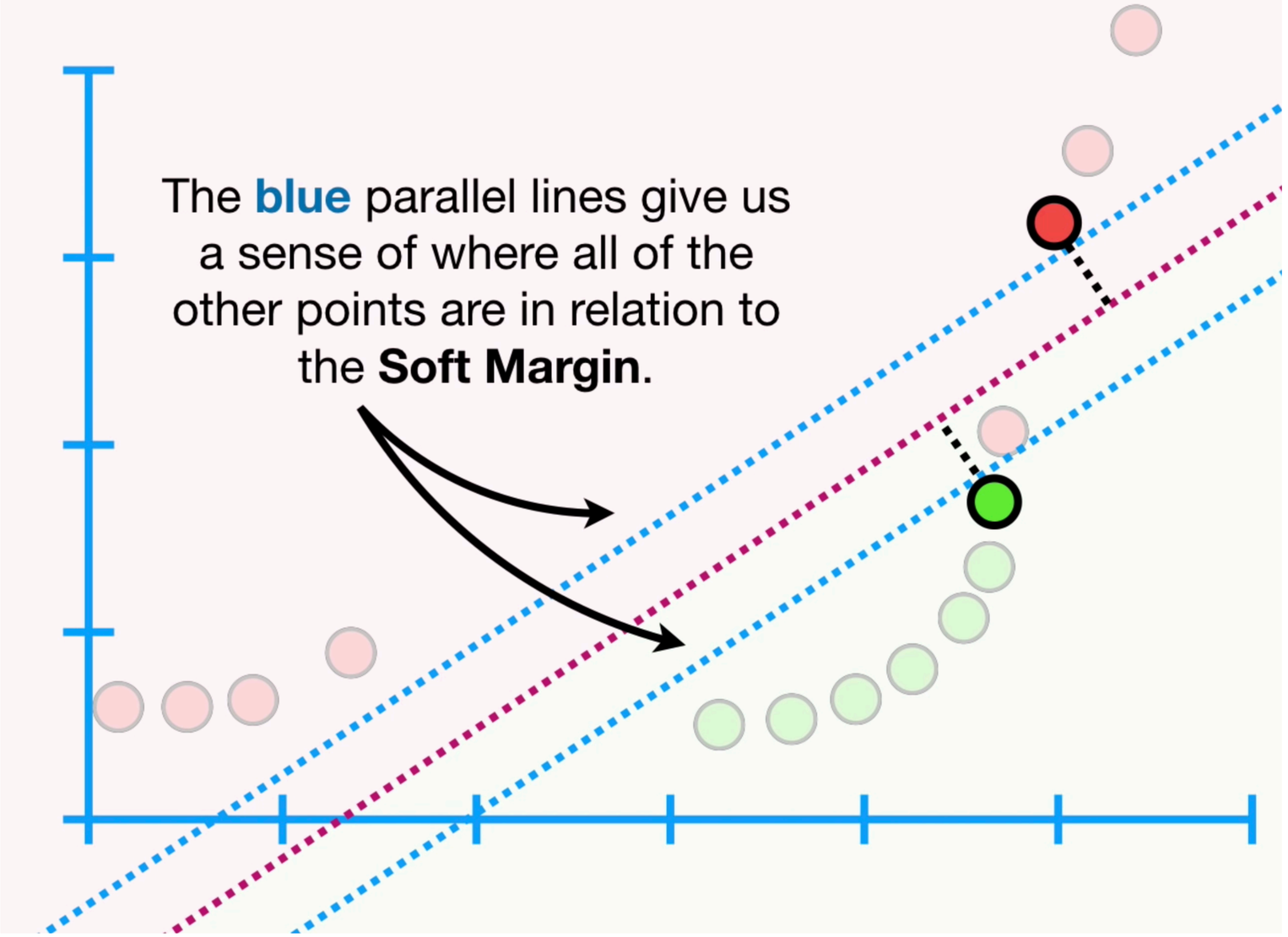


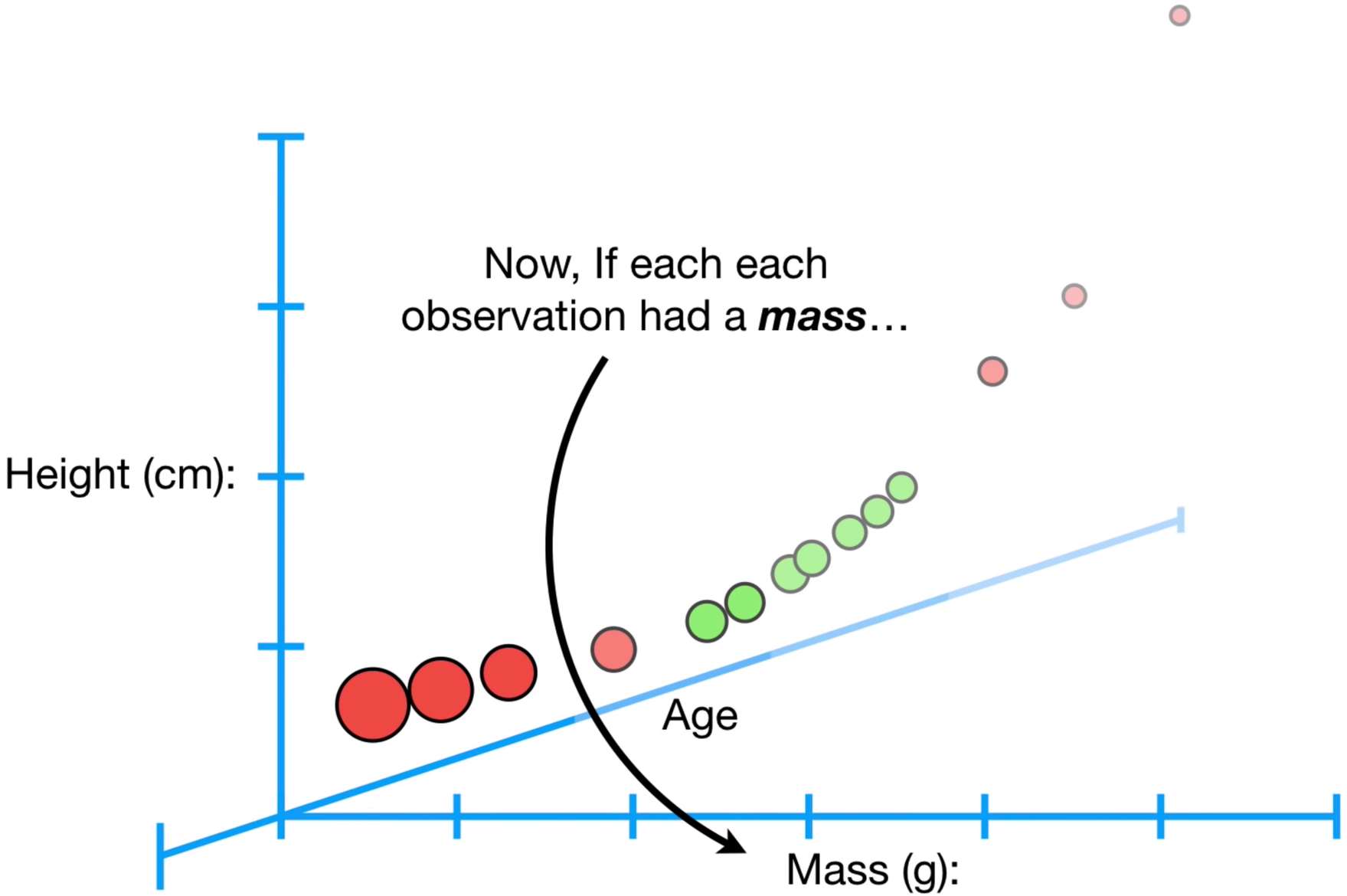

When we use a soft margin to determine the location of a threshold, then we are using a **Soft Margin classifier** aka a **Support Vector Classifier** to classify observations.

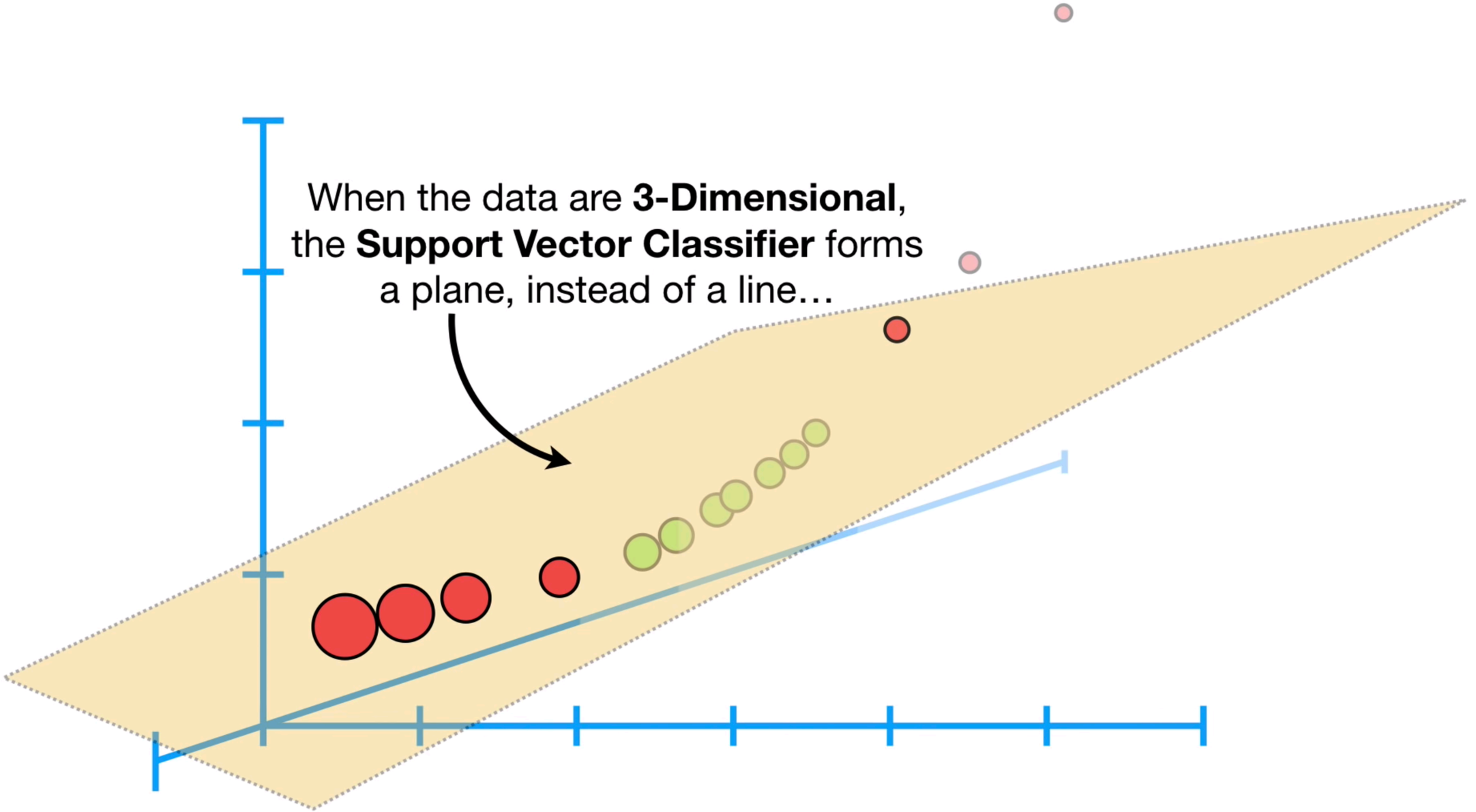

…then the data would
be **2-Dimensional**.
Height (cm):
Mass (g):


When the data are **2-Dimensional**, a **Support Vector Classifier** is a line…

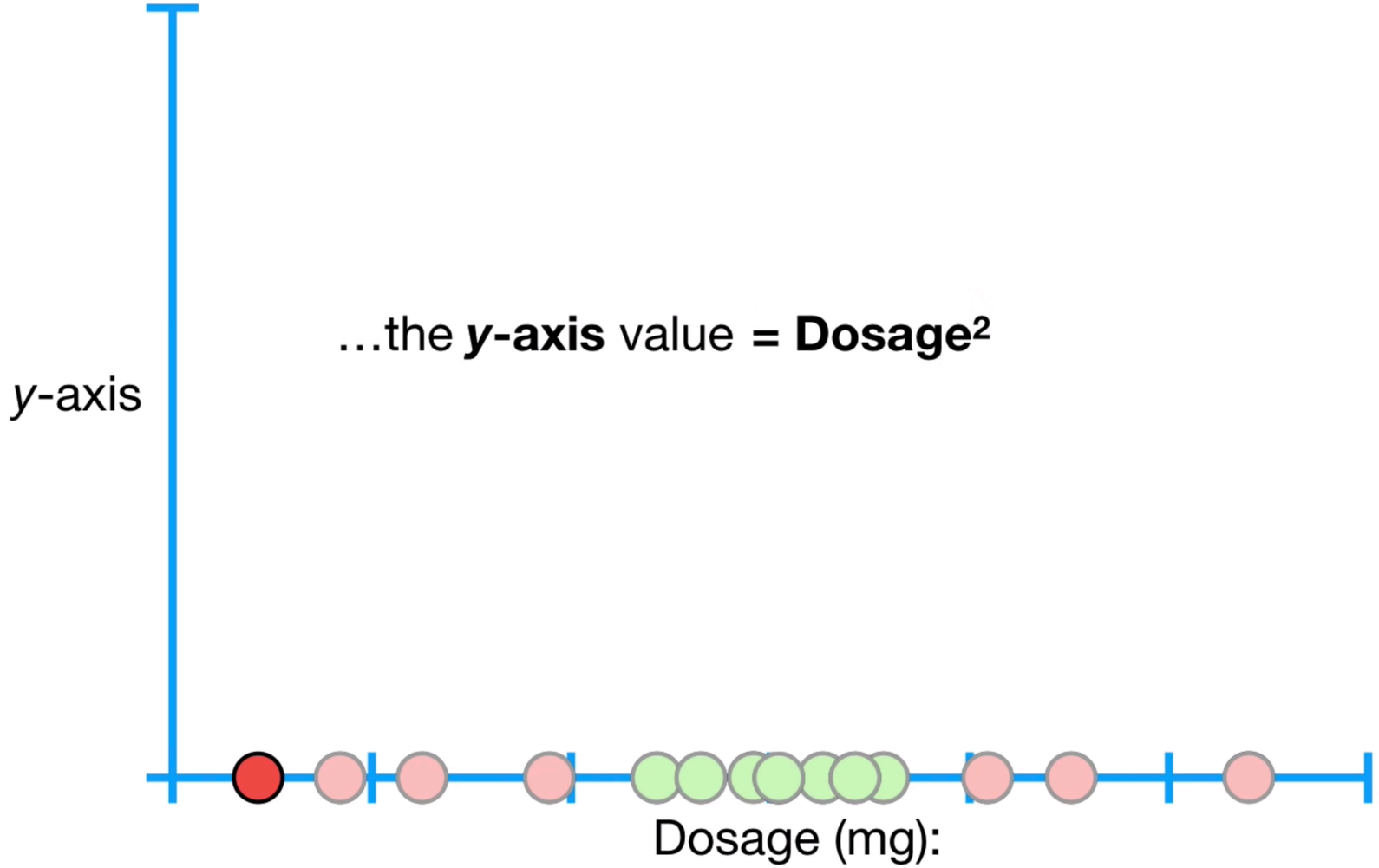

…and, in this case, the **Soft Margin** is measured form these two points.

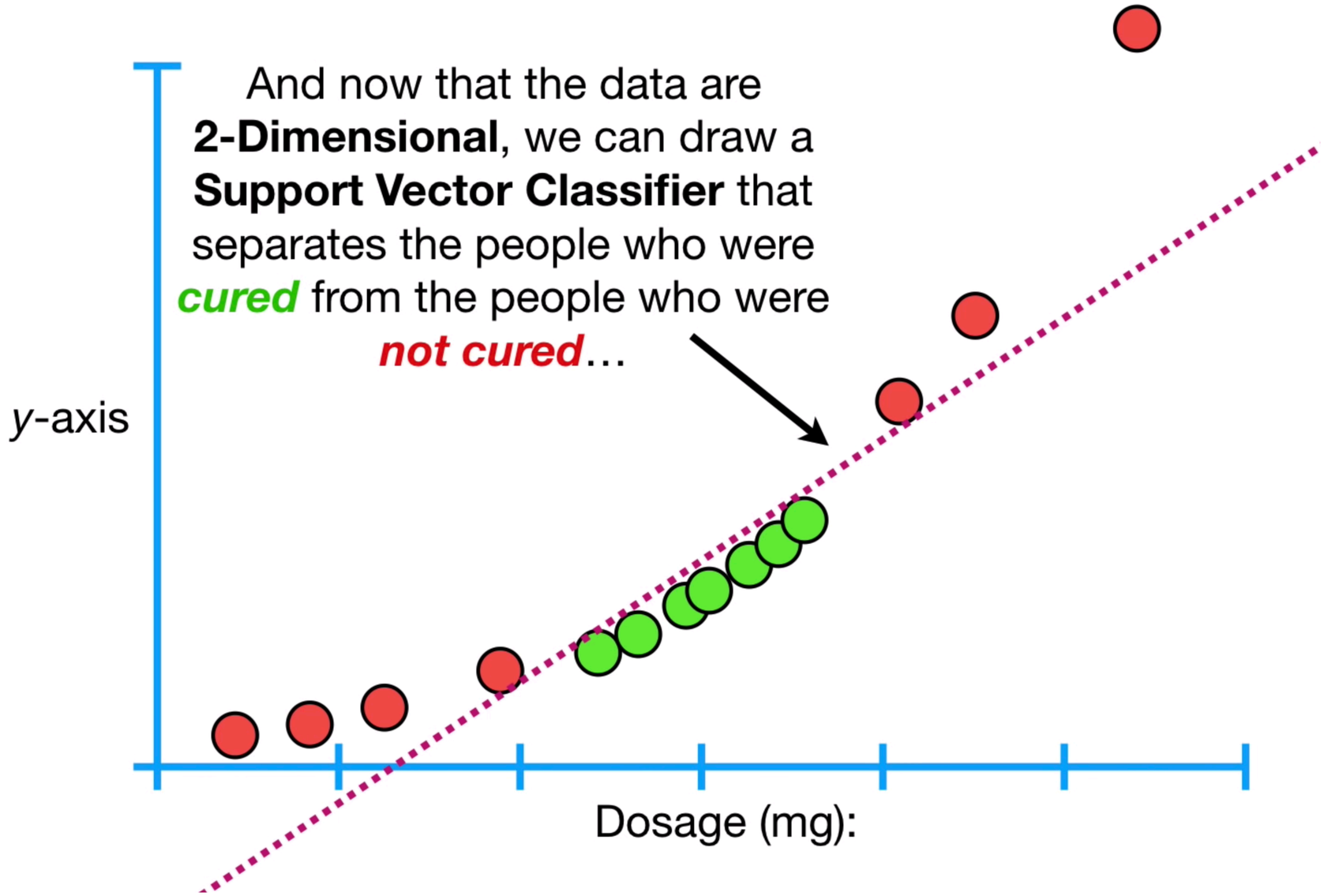

The blue parallel lines give us a sense of where all of the other points are in relation to the Soft Margin.

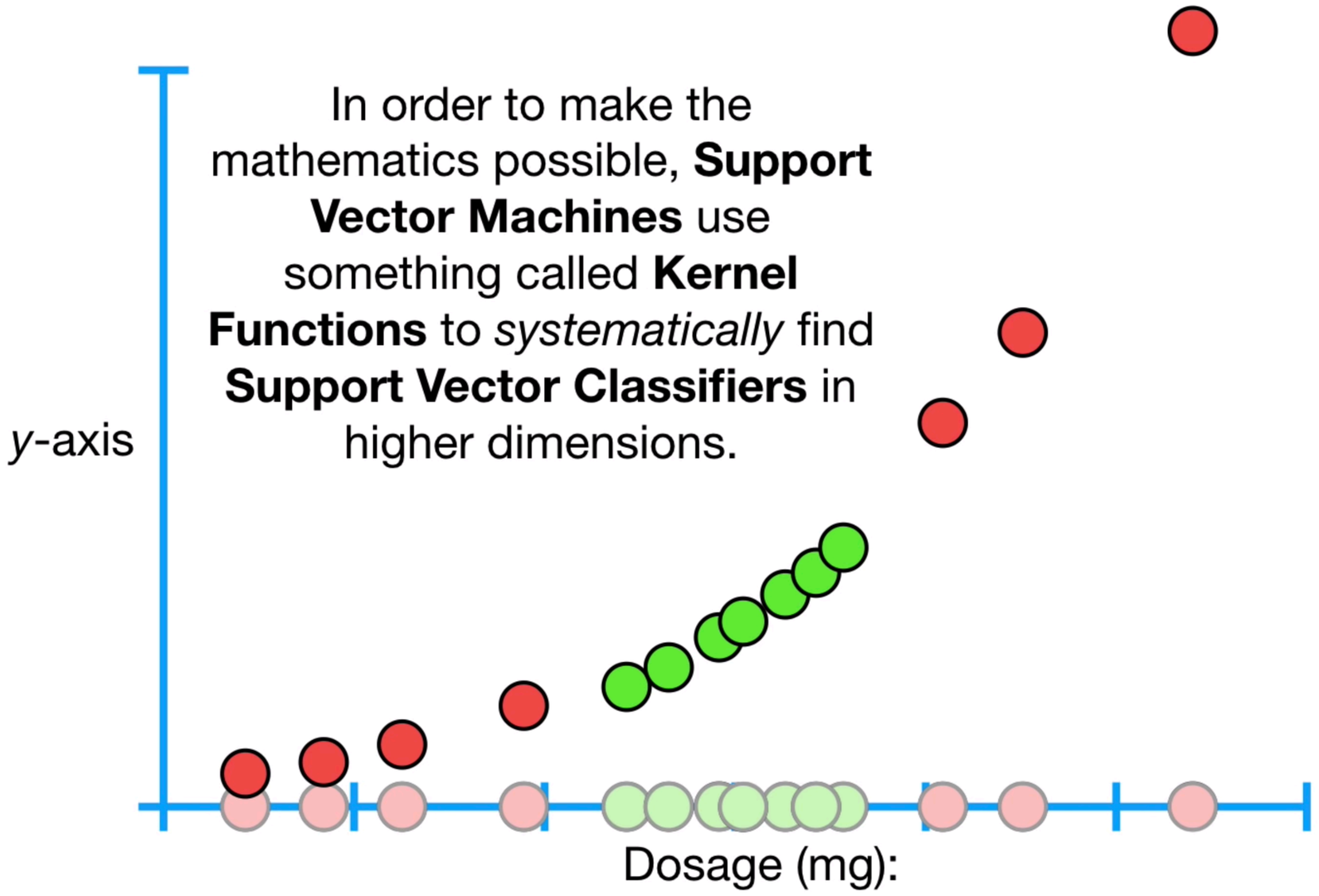

Now, If each each observation had a ***mass***…
Height (cm):
Age
Mass (g):

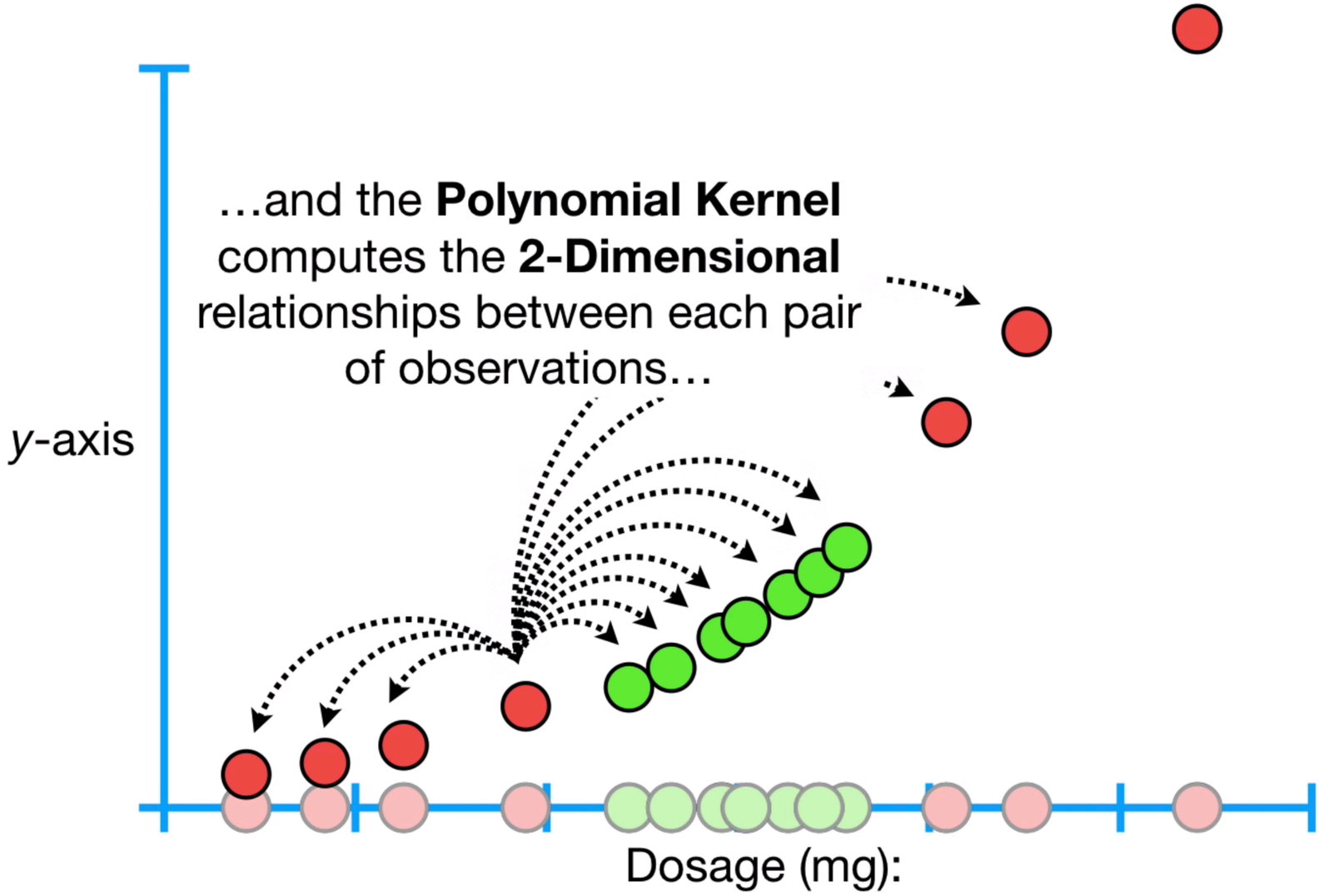

When the data are **3-Dimensional**,
the **Support Vector Classifier** forms
a plane, instead of a line…

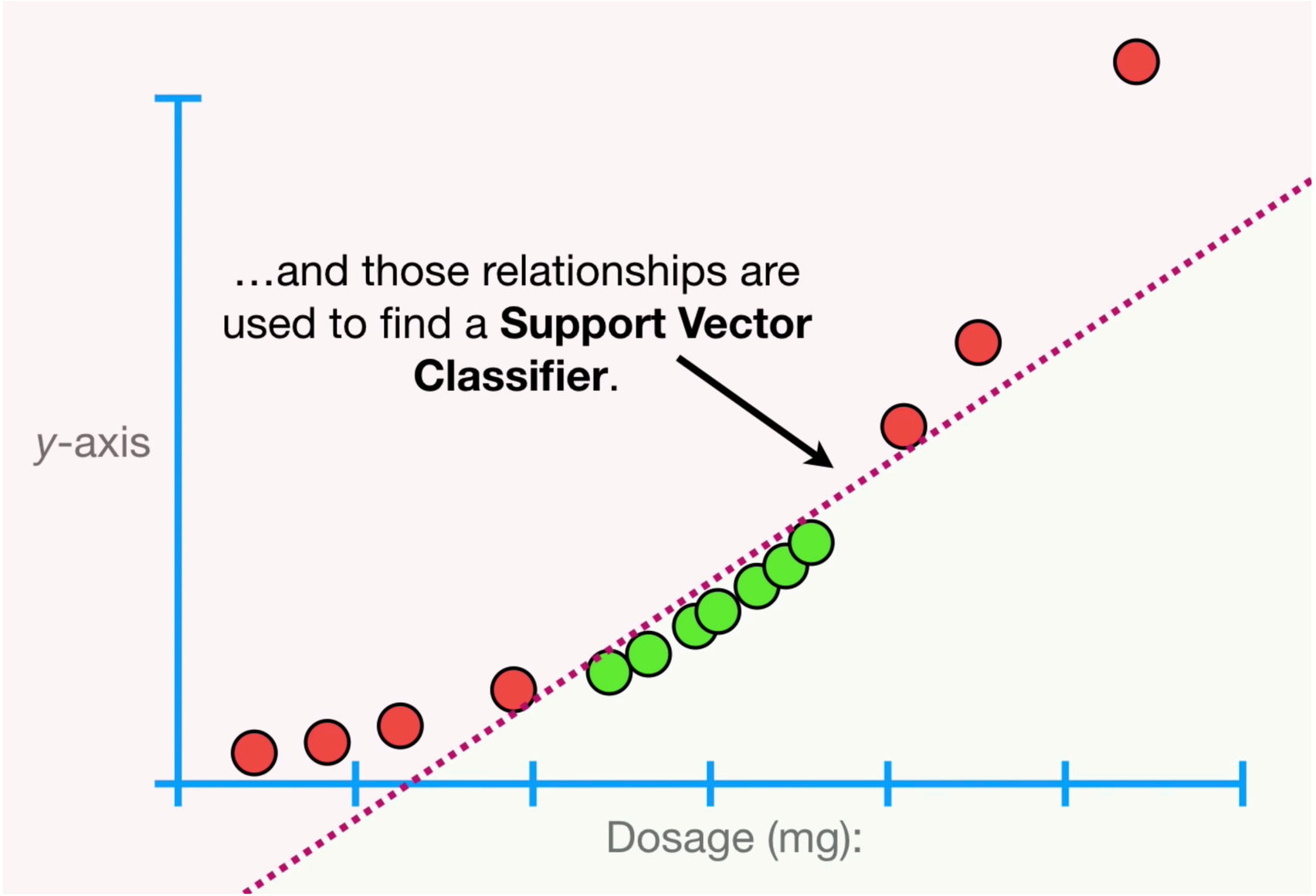
In this new example, with tons of overlap, we are now looking at **Drug Dosages**…

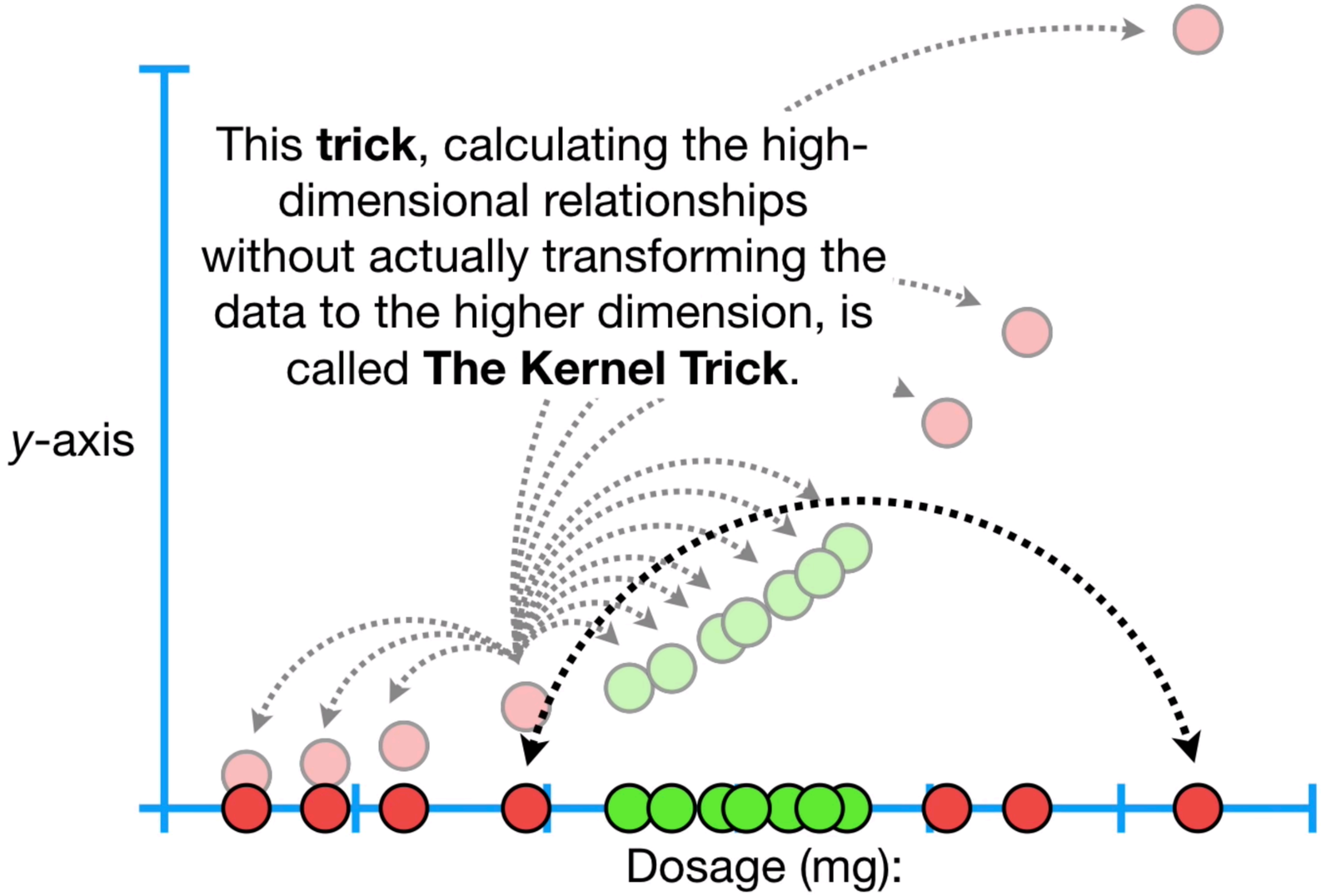

…the **y-axis** value = **Dosage²**
*y*-axis
Dosage (mg):


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):


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):


…and the **Polynomial Kernel** computes the **2-Dimensional** relationships between each pair of observations…
*y*-axis
Dosage (mg):


…and those relationships are used to find a **Support Vector Classifier**.
*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):