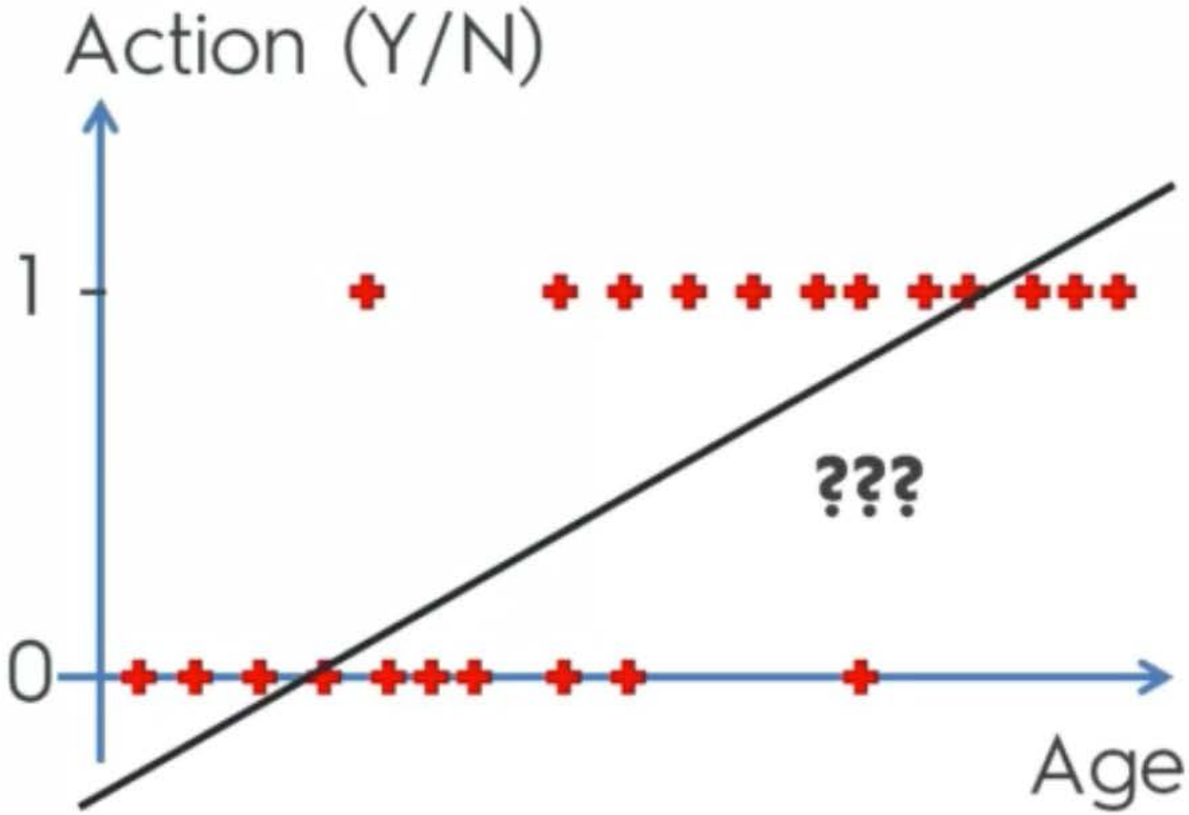
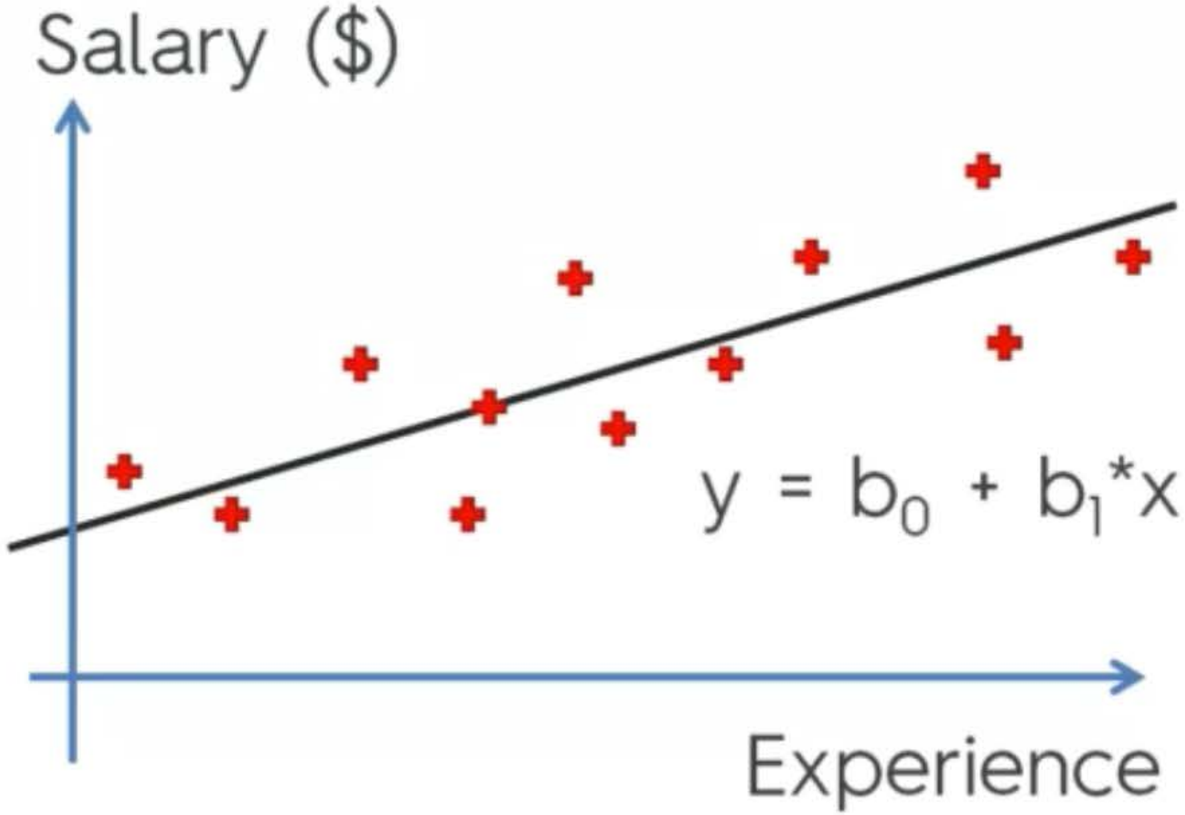
Logistic Regression Intuition

Logistic Regression

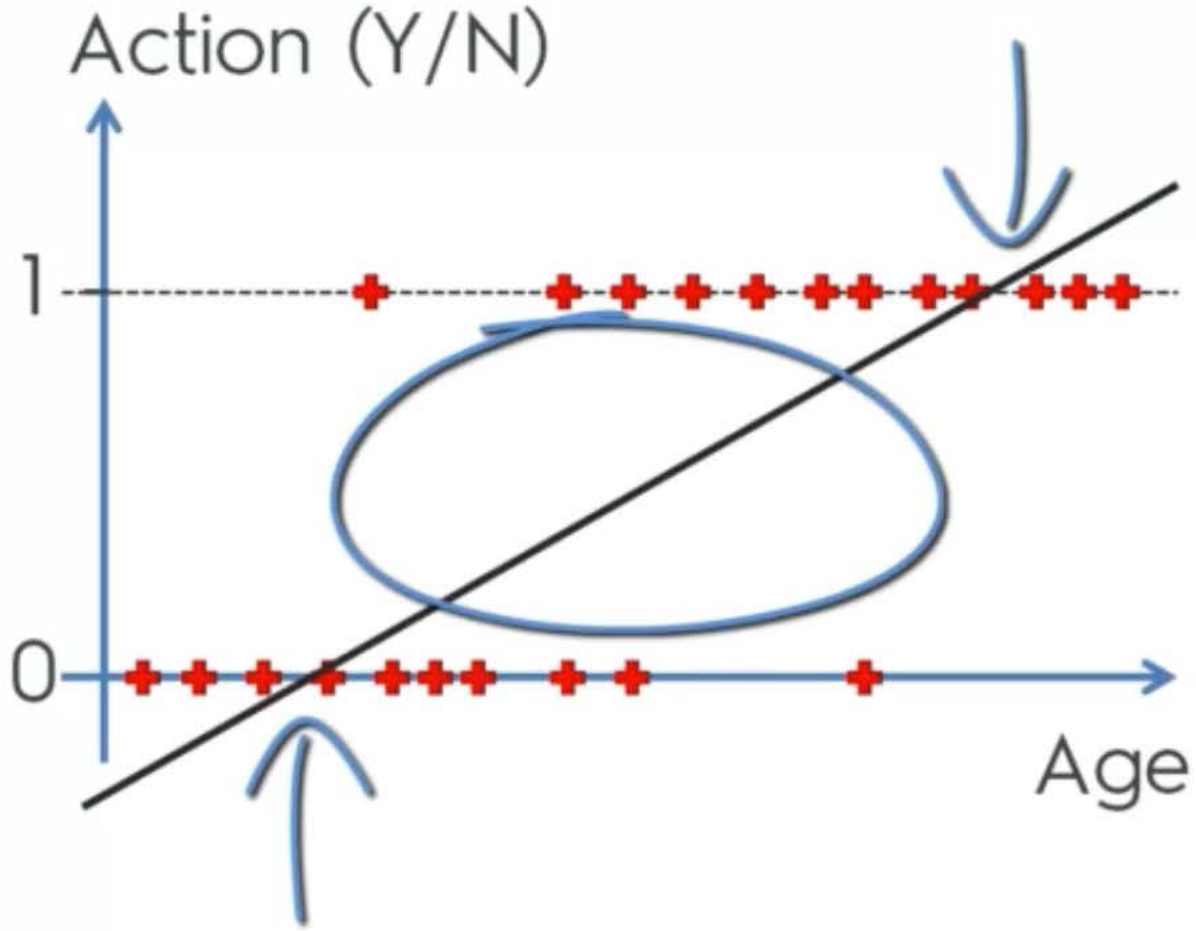
This is new:



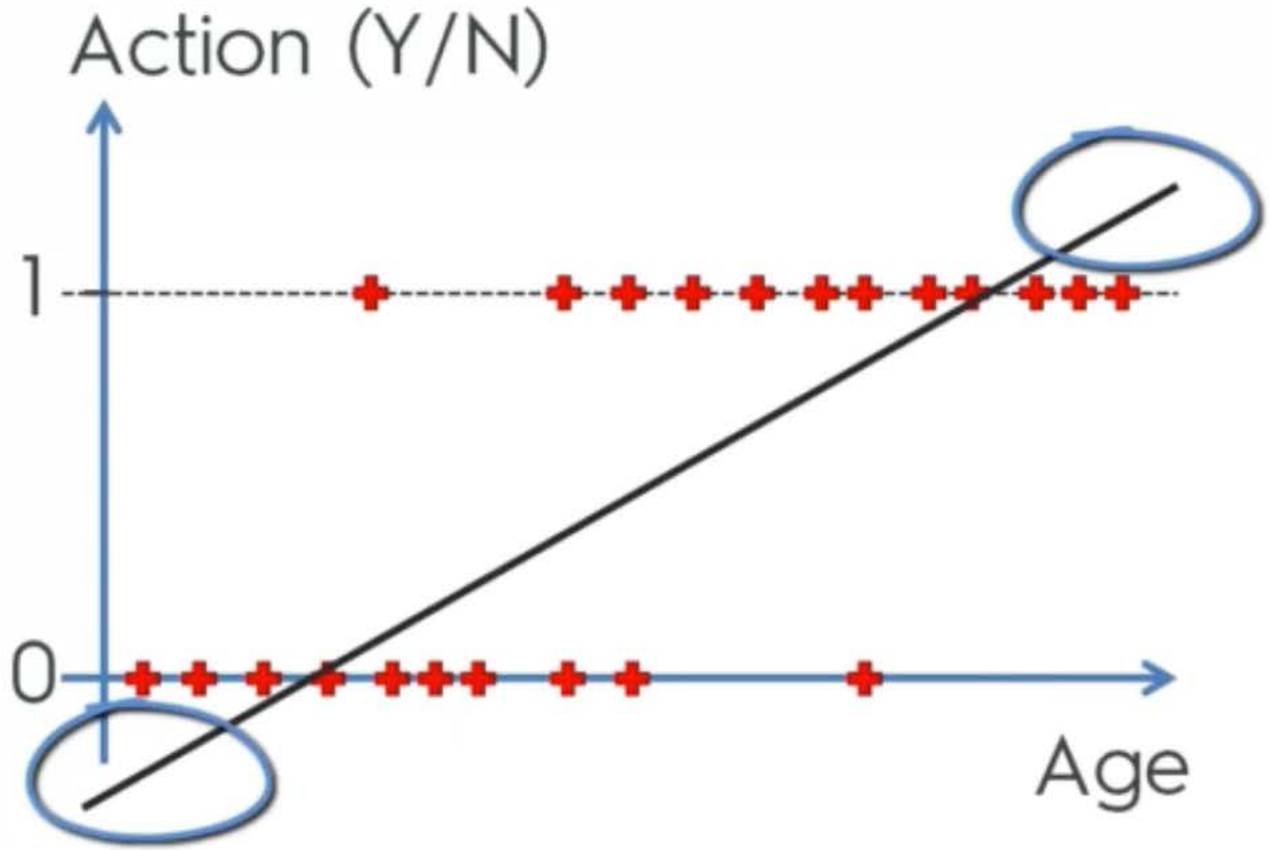
We know this:



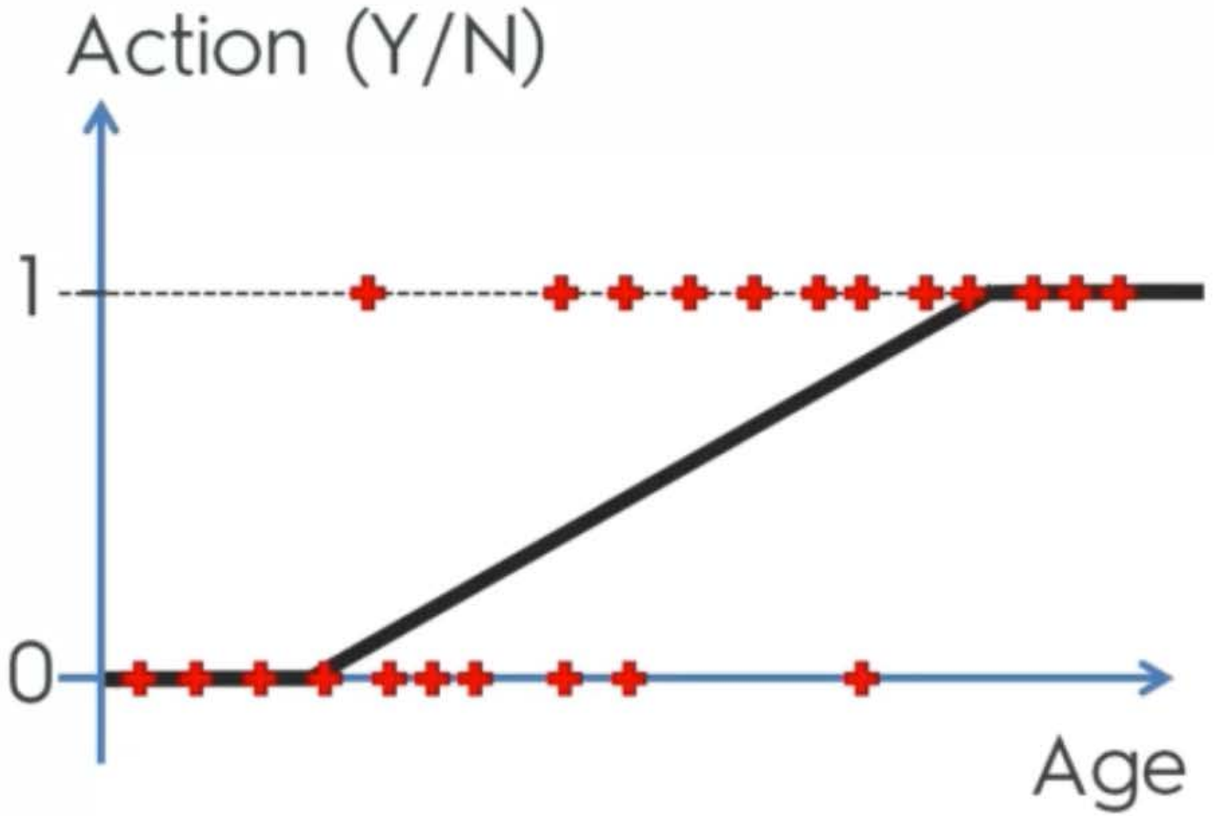
Logistic Regression



Logistic Regression



Logistic Regression



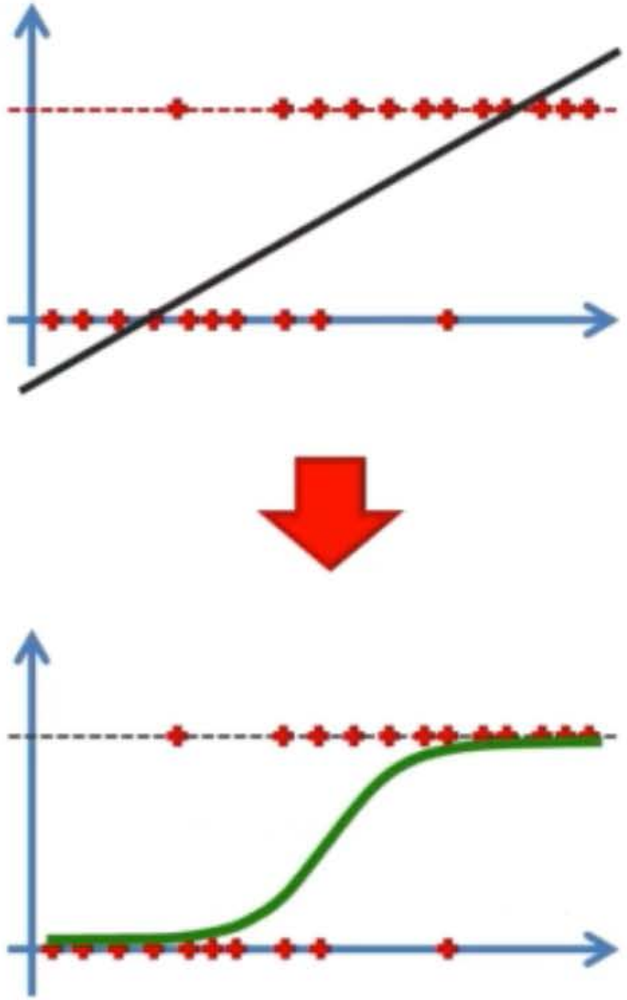
Logistic Regression

$$y = b_0 + b_1 * x$$

Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

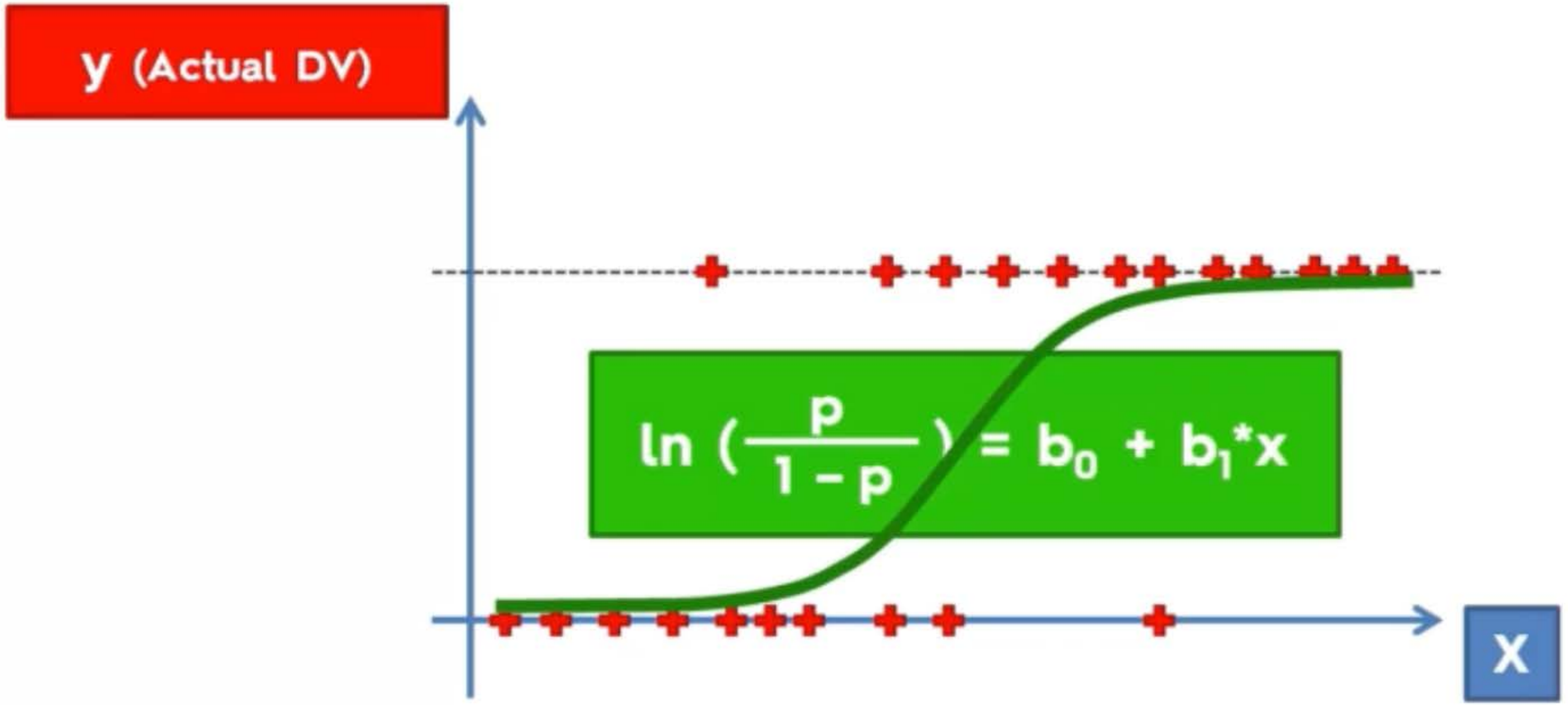
$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$



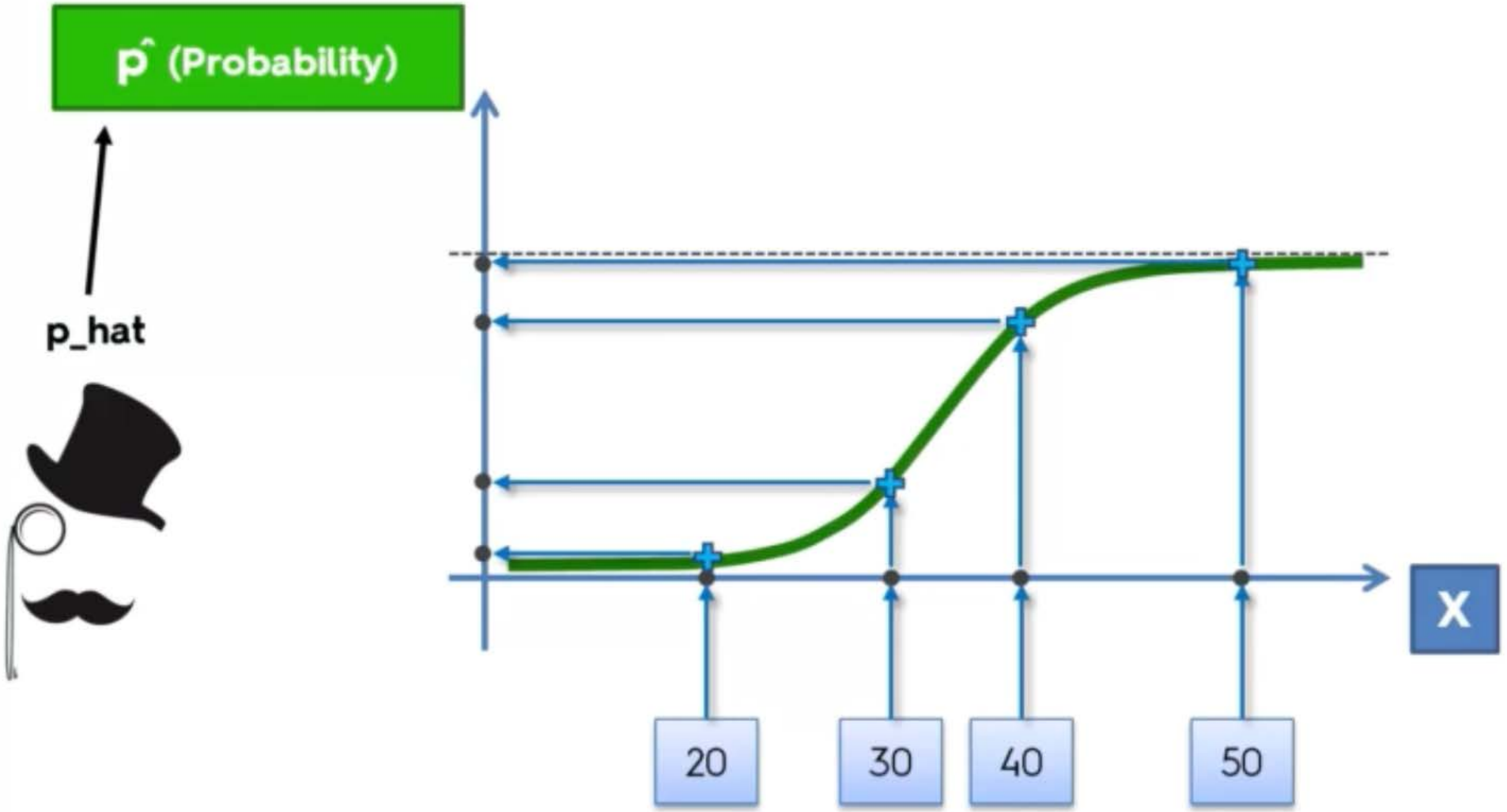
Logistic Regression

WHAT JUST HAPPENED ???

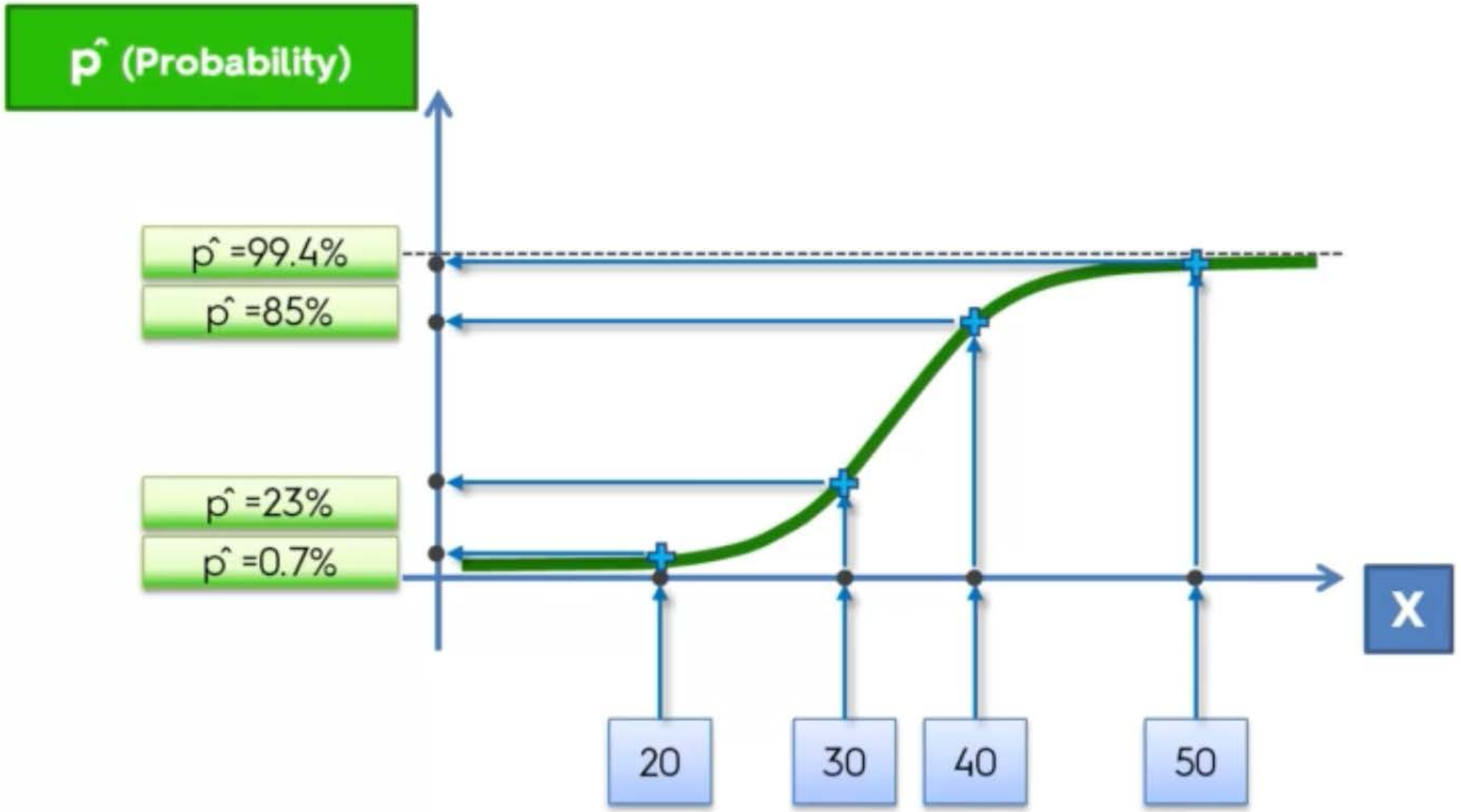
Logistic Regression



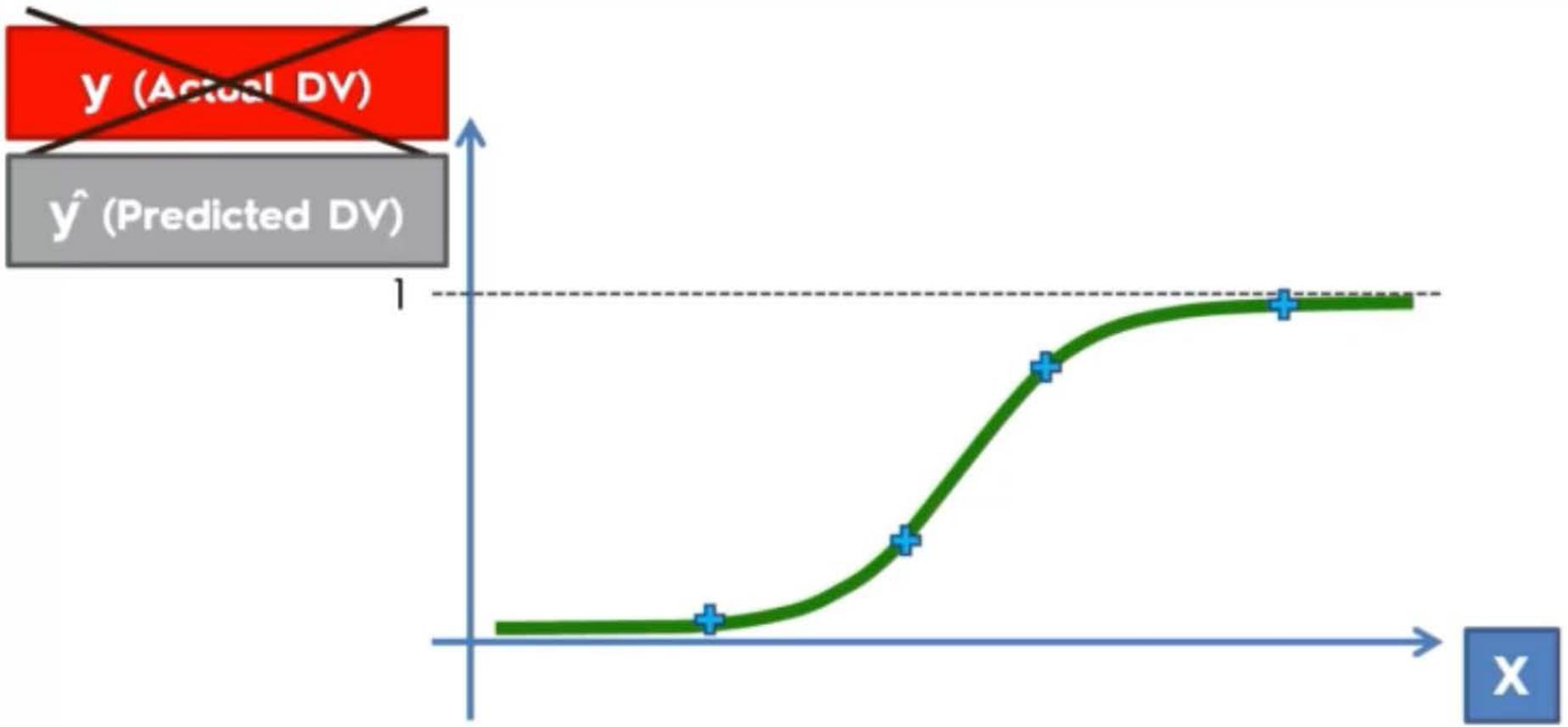
Logistic Regression



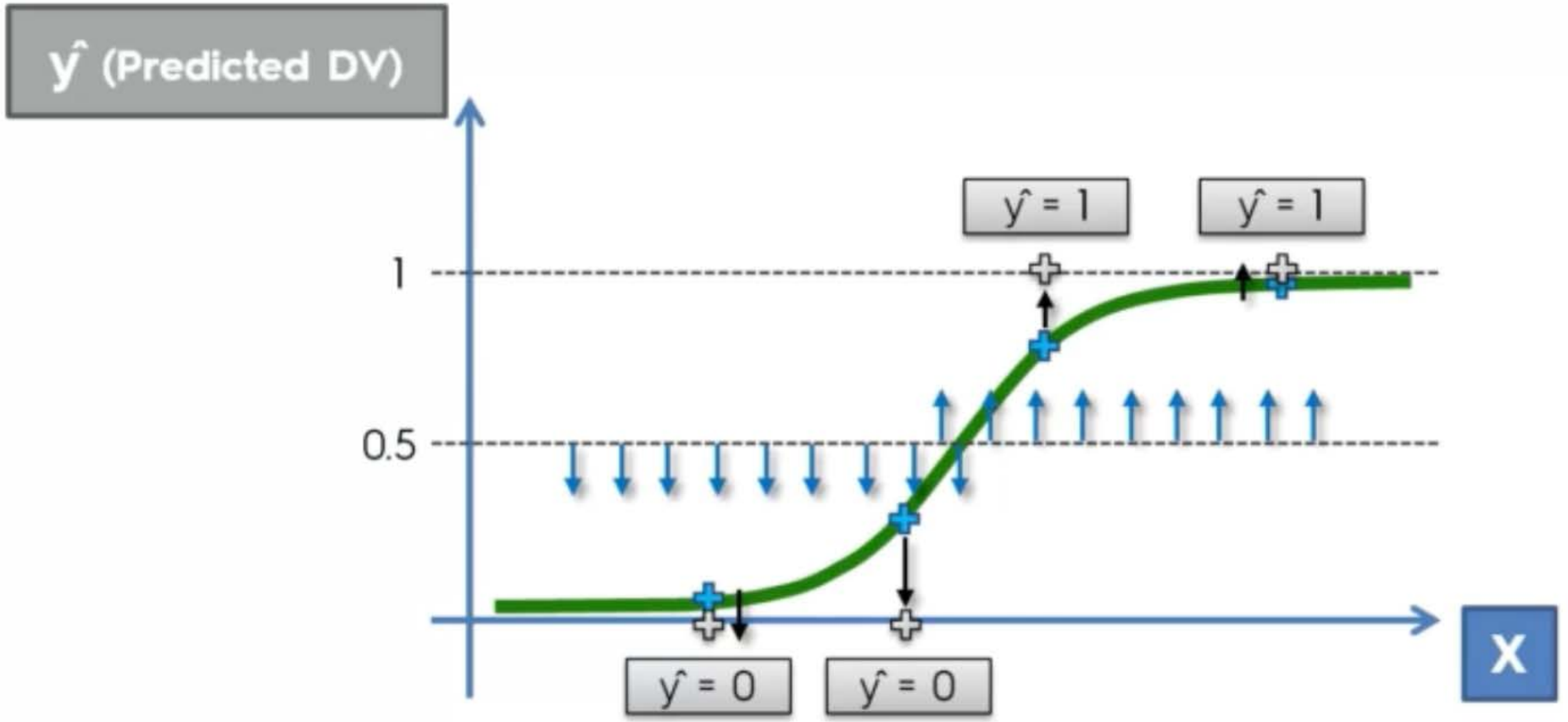
Logistic Regression



Logistic Regression

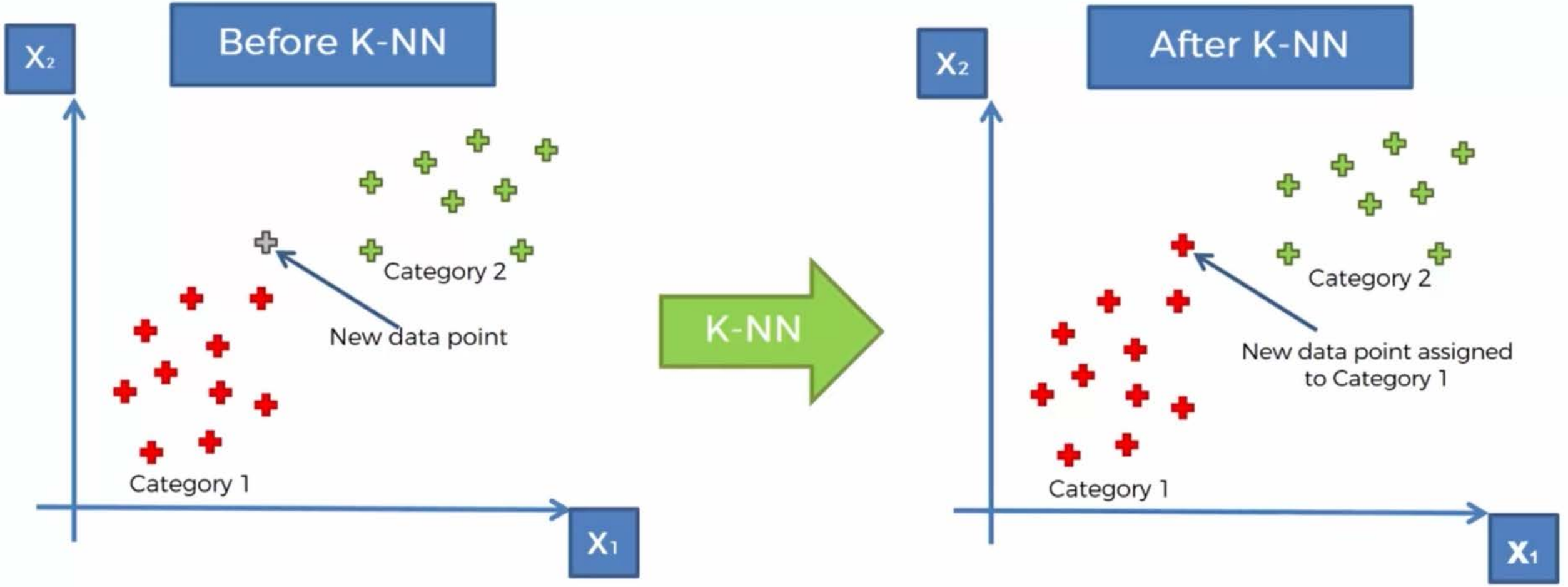


Logistic Regression



K-NN Intuition

What K-NN does for you



How did it do that ?

STEP 1: Choose the number K of neighbors

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

STEP 3: Among these K neighbors, count the number of data points in each category

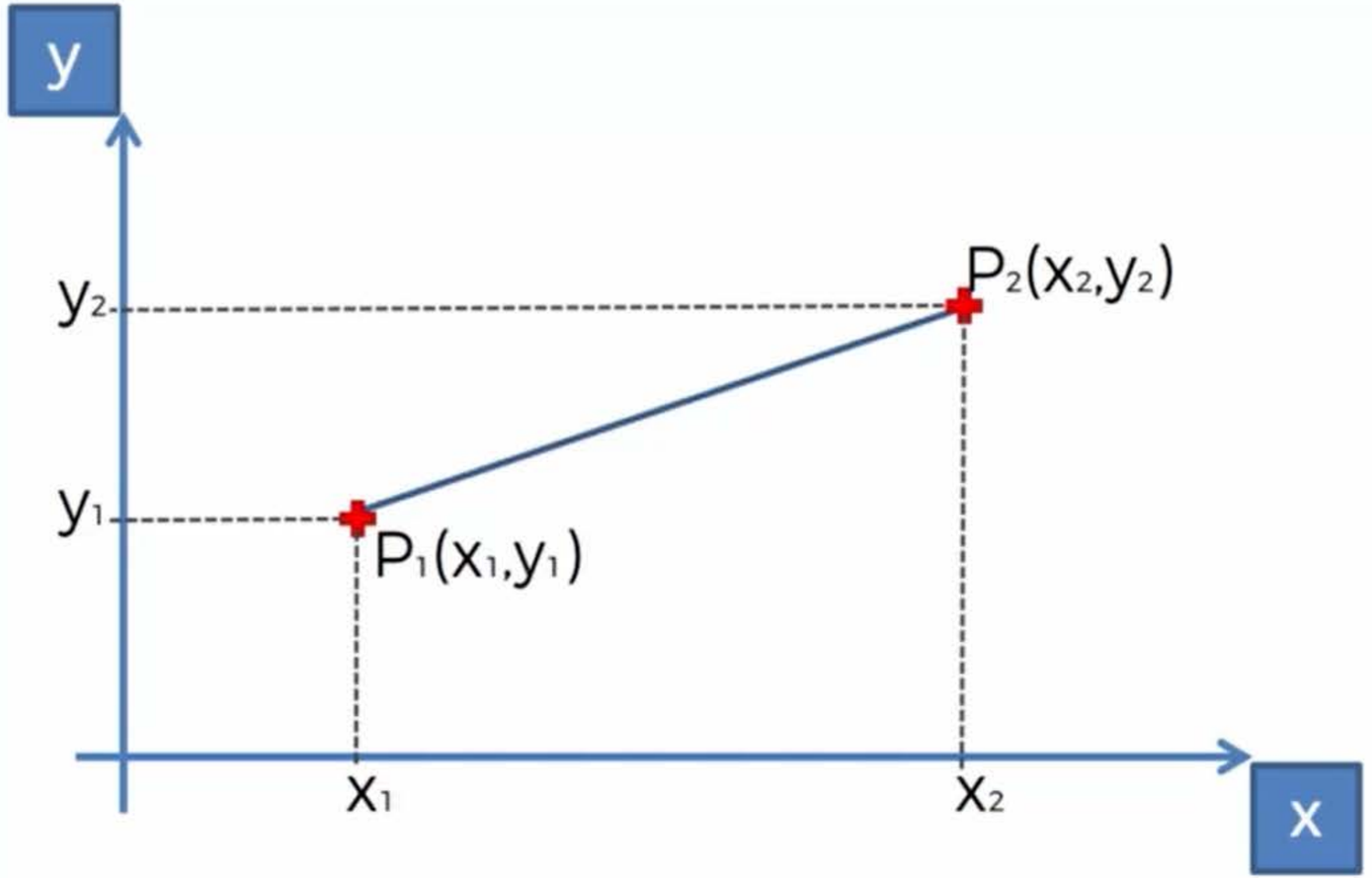
STEP 4: Assign the new data point to the category where you counted the most neighbors

Your Model is Ready

K-NN algorithm

STEP 1: Choose the number K of neighbors: K = 5

Euclidean Distance



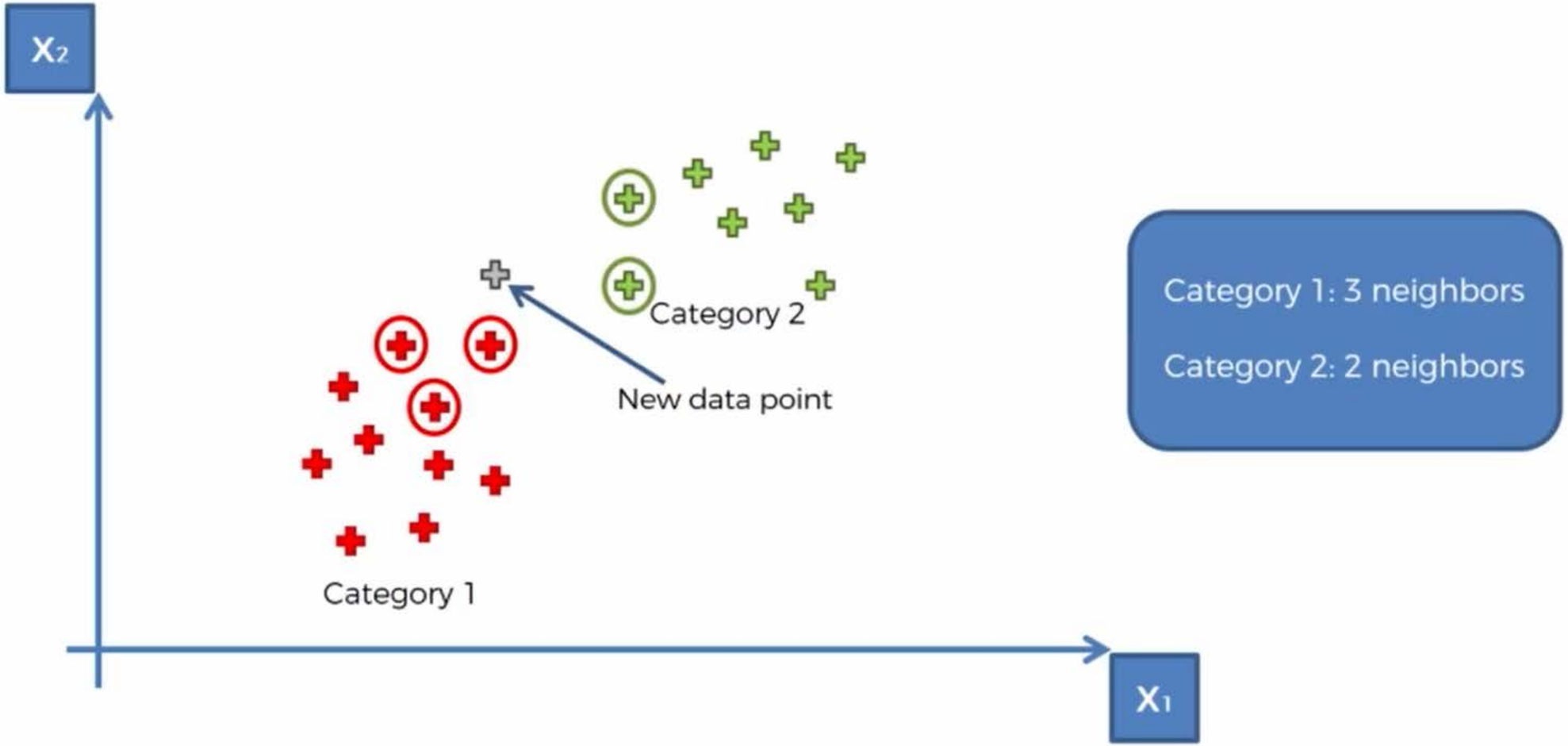
$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-NN algorithm

STEP 2: Take the K = 5 nearest neighbors of the new data point, according to the Euclidean distance

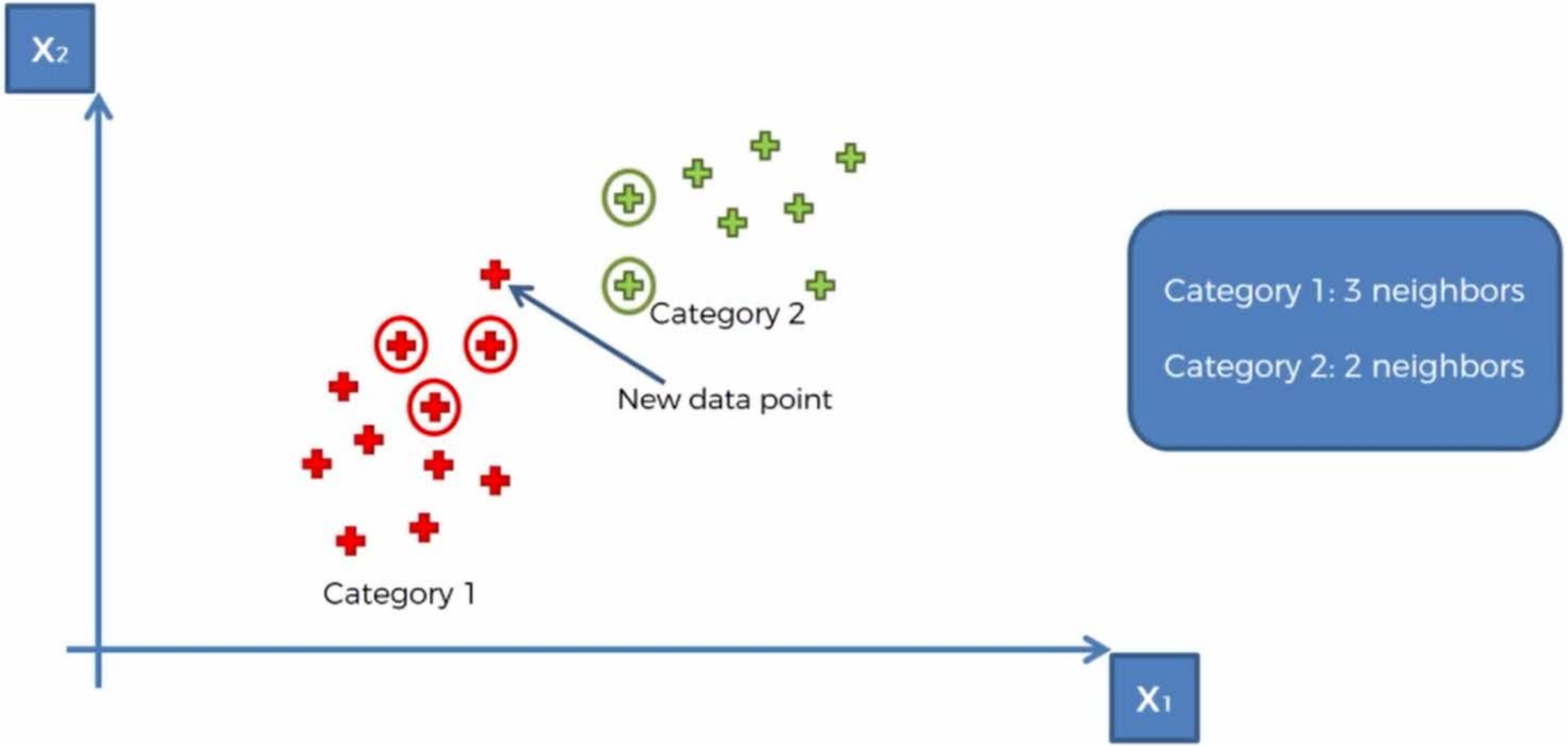
K-NN algorithm

STEP 3: Among these K neighbors, count the number of data points in each category



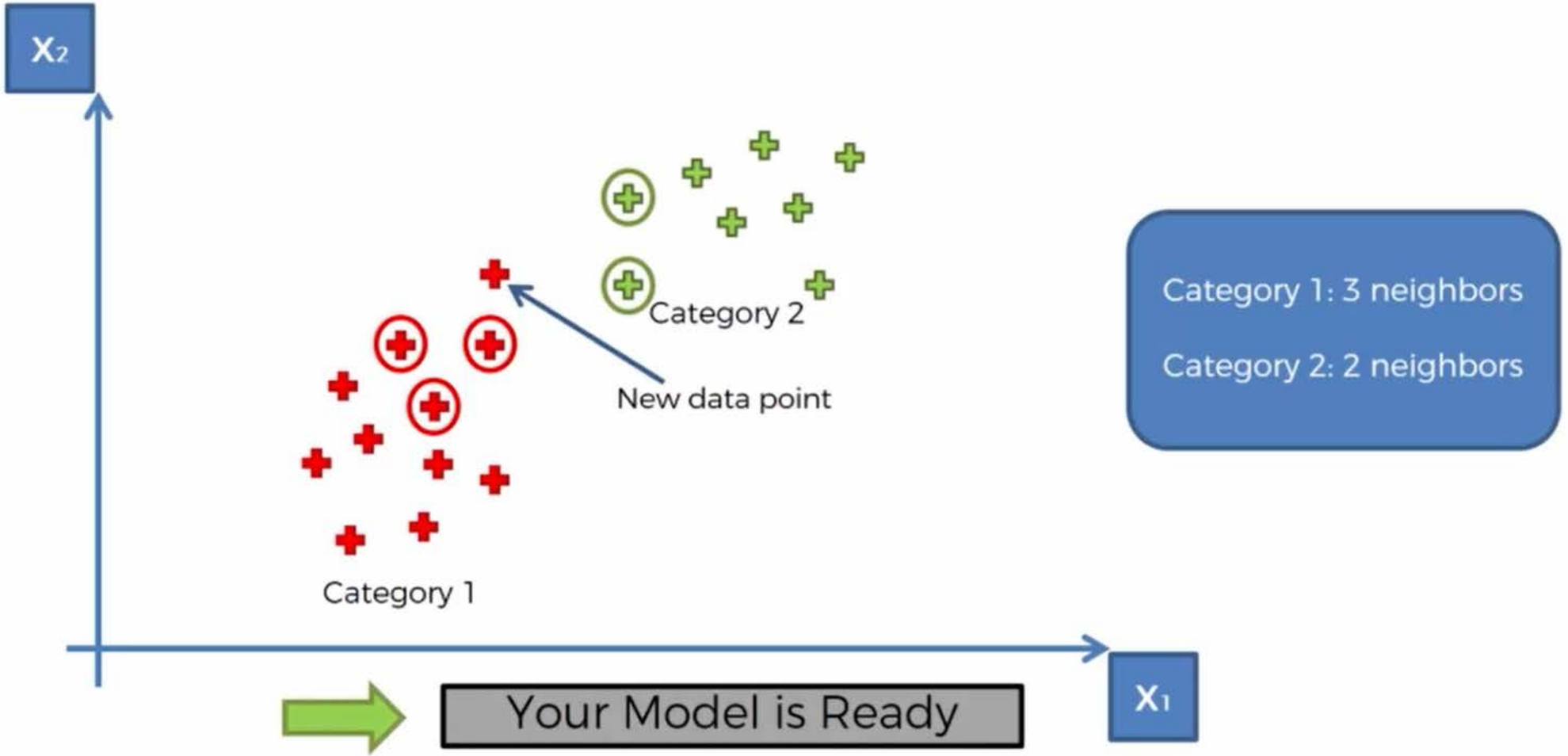
K-NN algorithm

STEP 4: Assign the new data point to the category where you counted the most neighbors



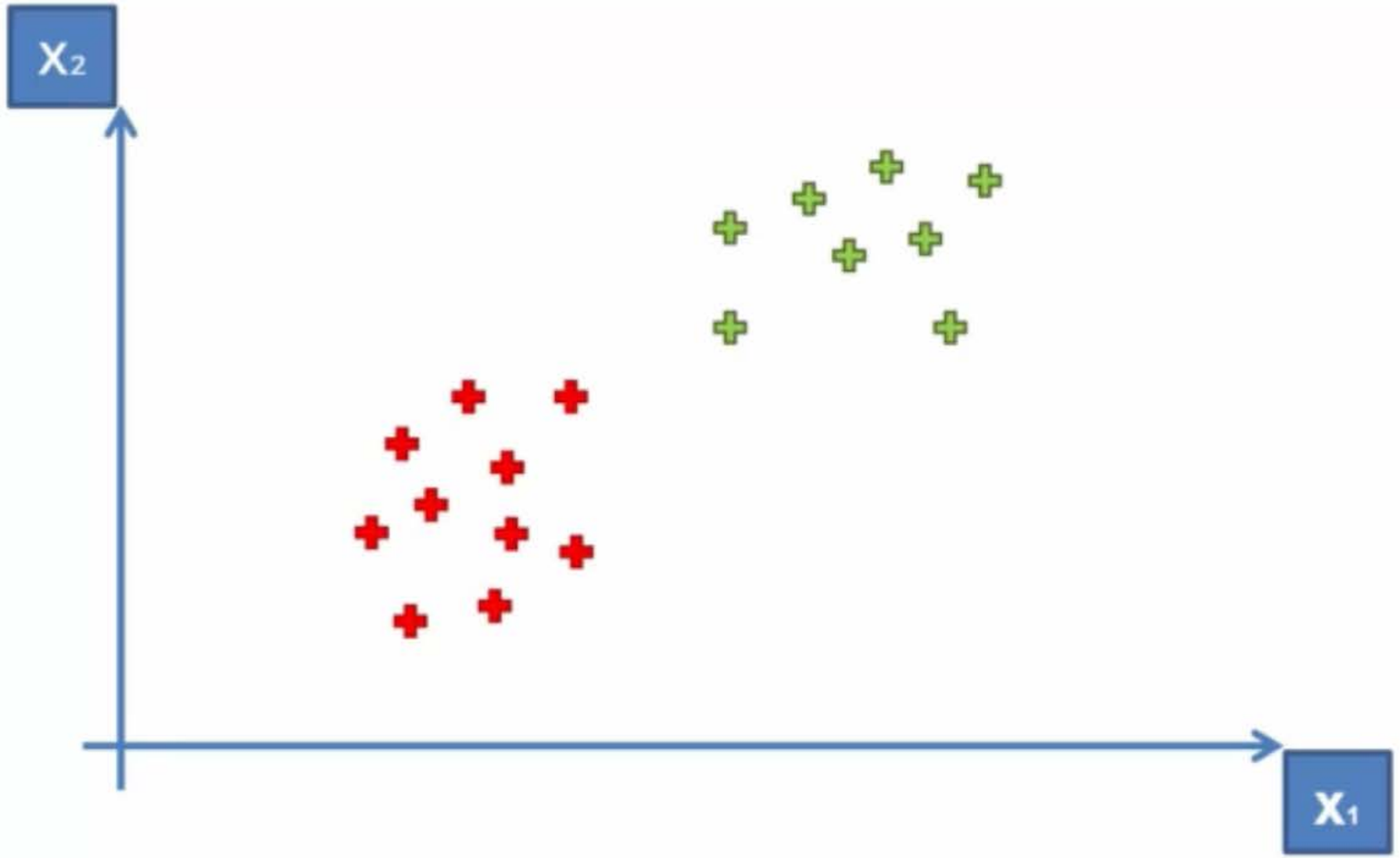
K-NN algorithm

STEP 4: Assign the new data point to the category where you counted the most neighbors

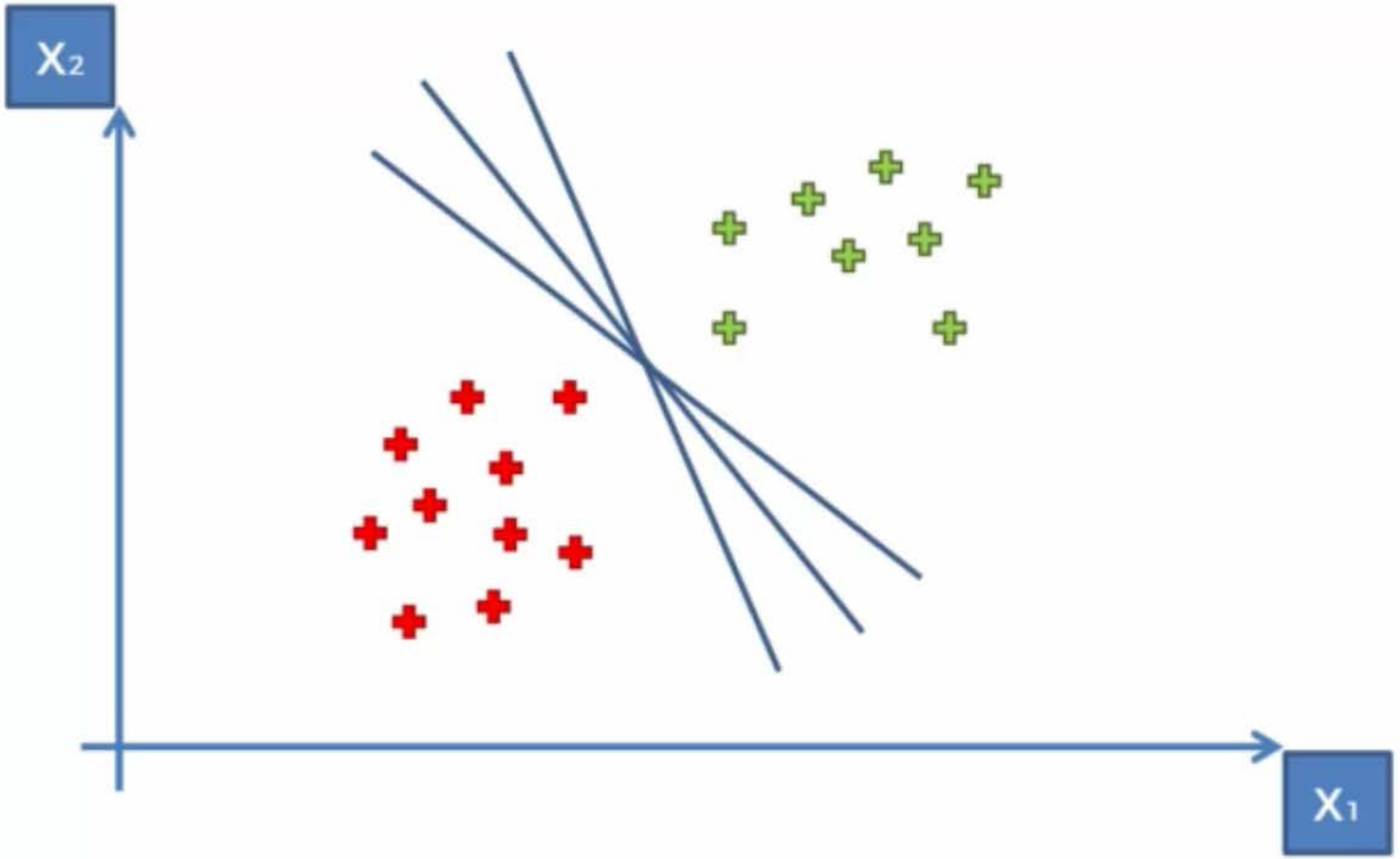


SVM Intuition

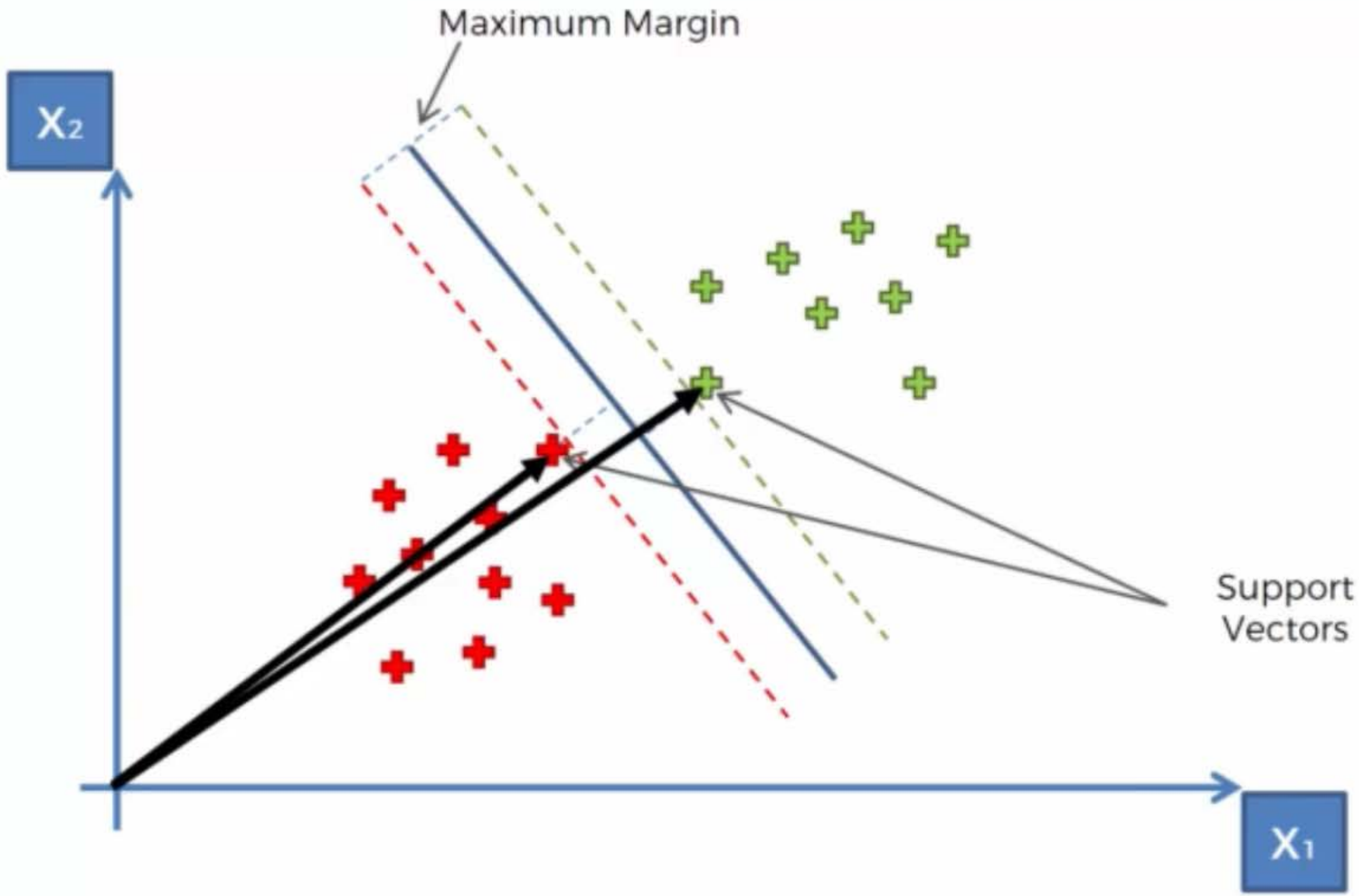
How to separate these points ?



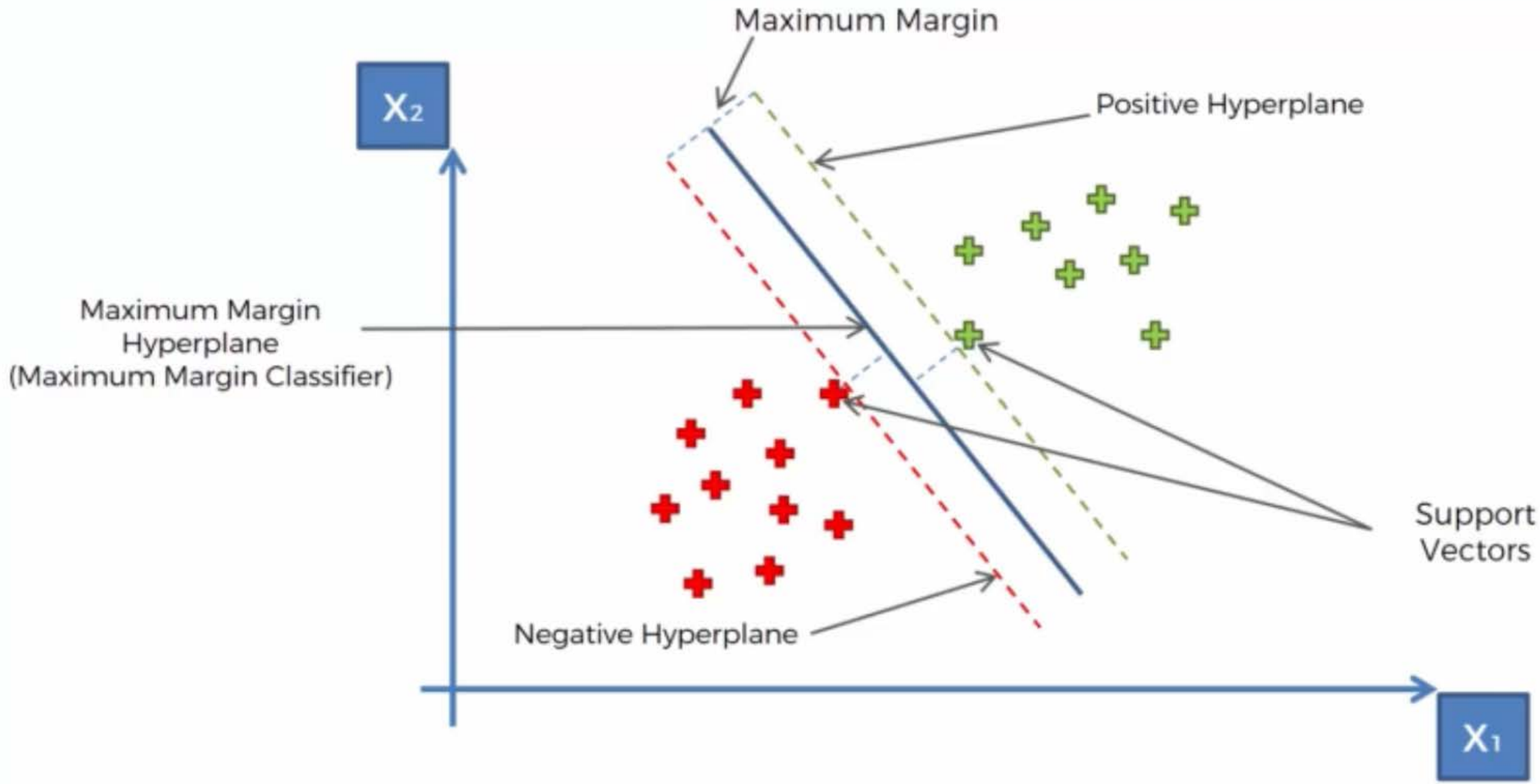
SVM



Support Vectors



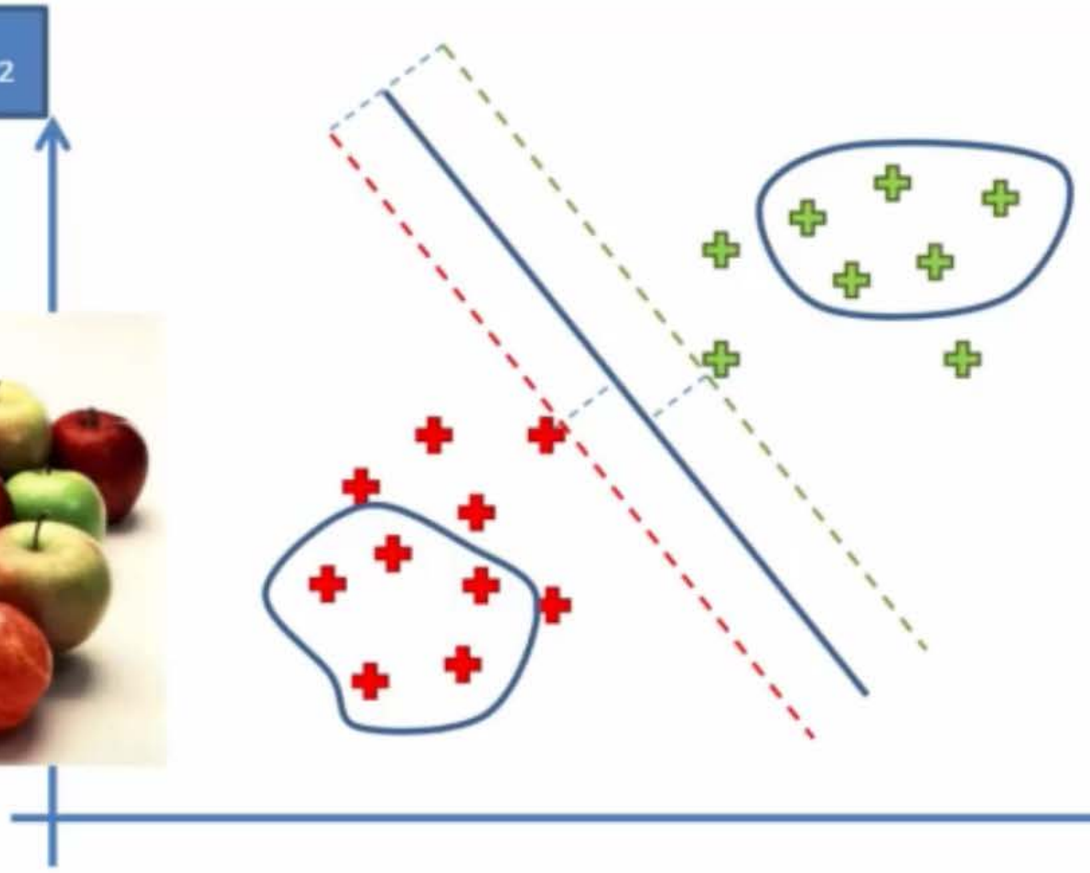
Hyperplanes



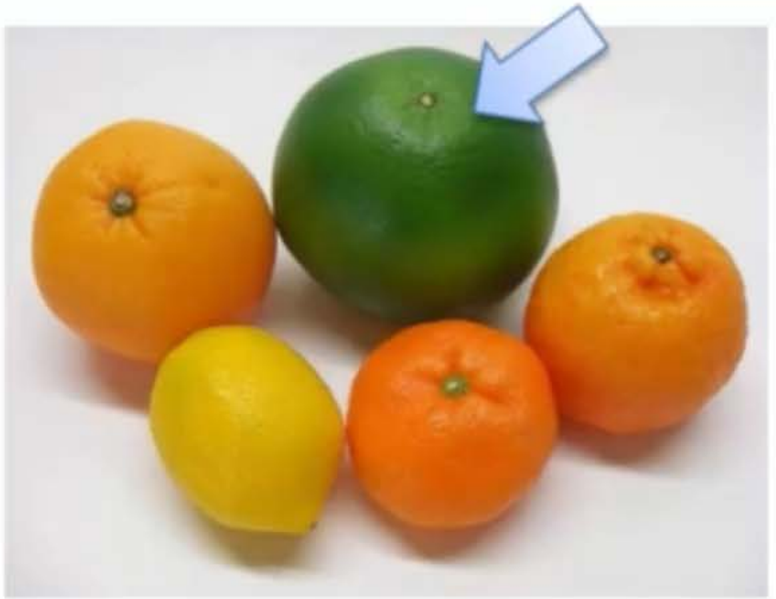
What's So Special About SVMs?

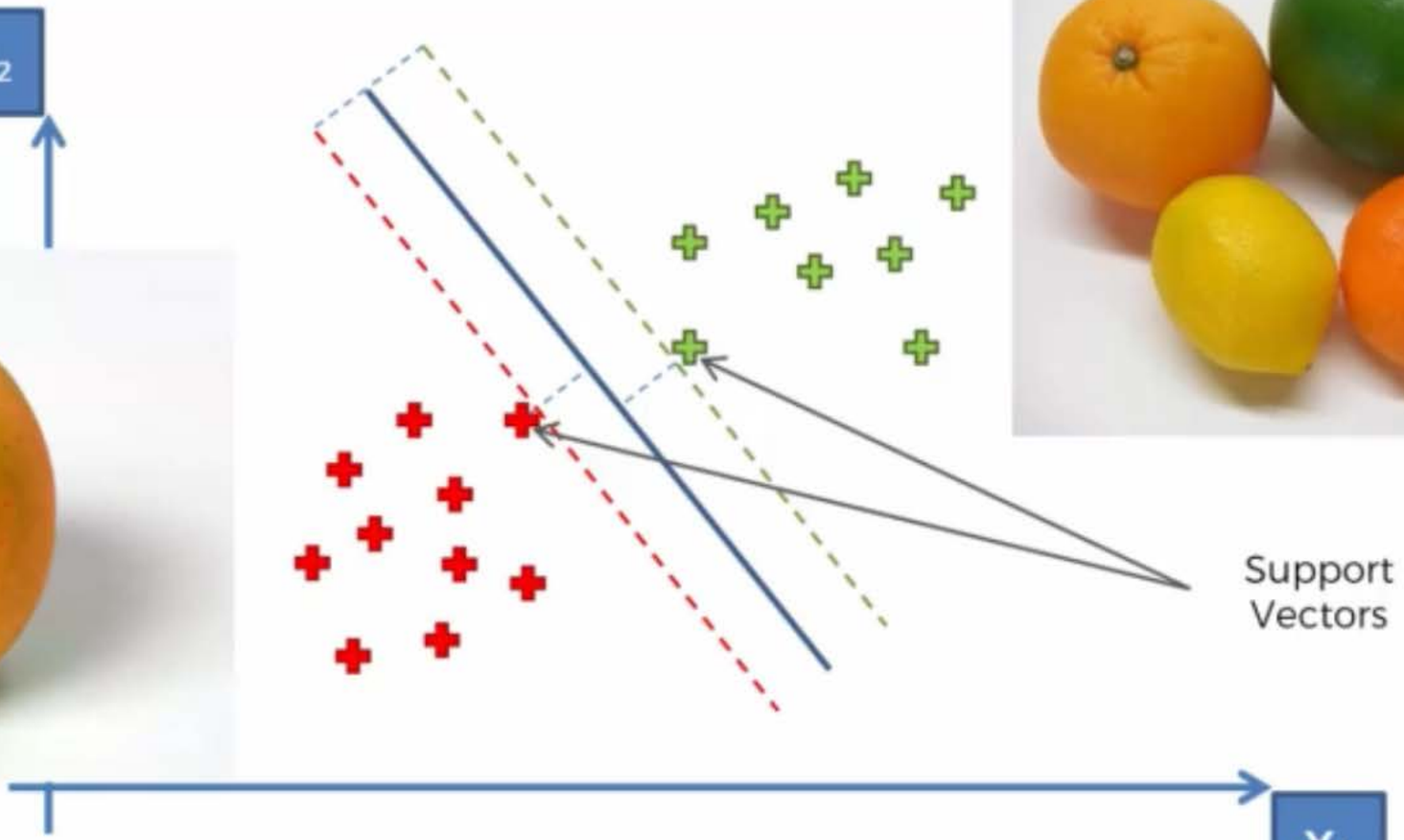
What's So Special About SVMs?

What's So Special About SVMs?



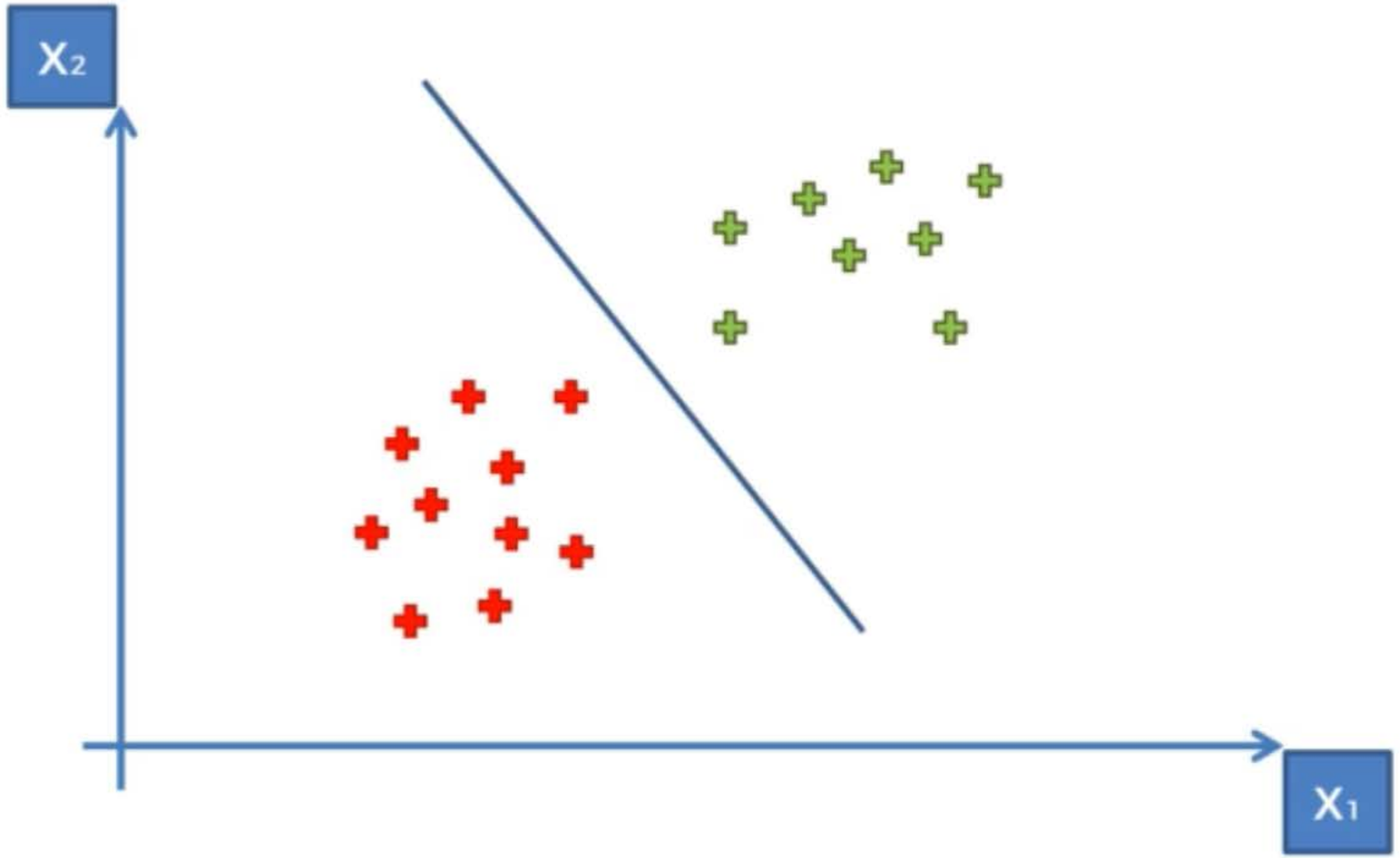
What's So Special About SVMs?



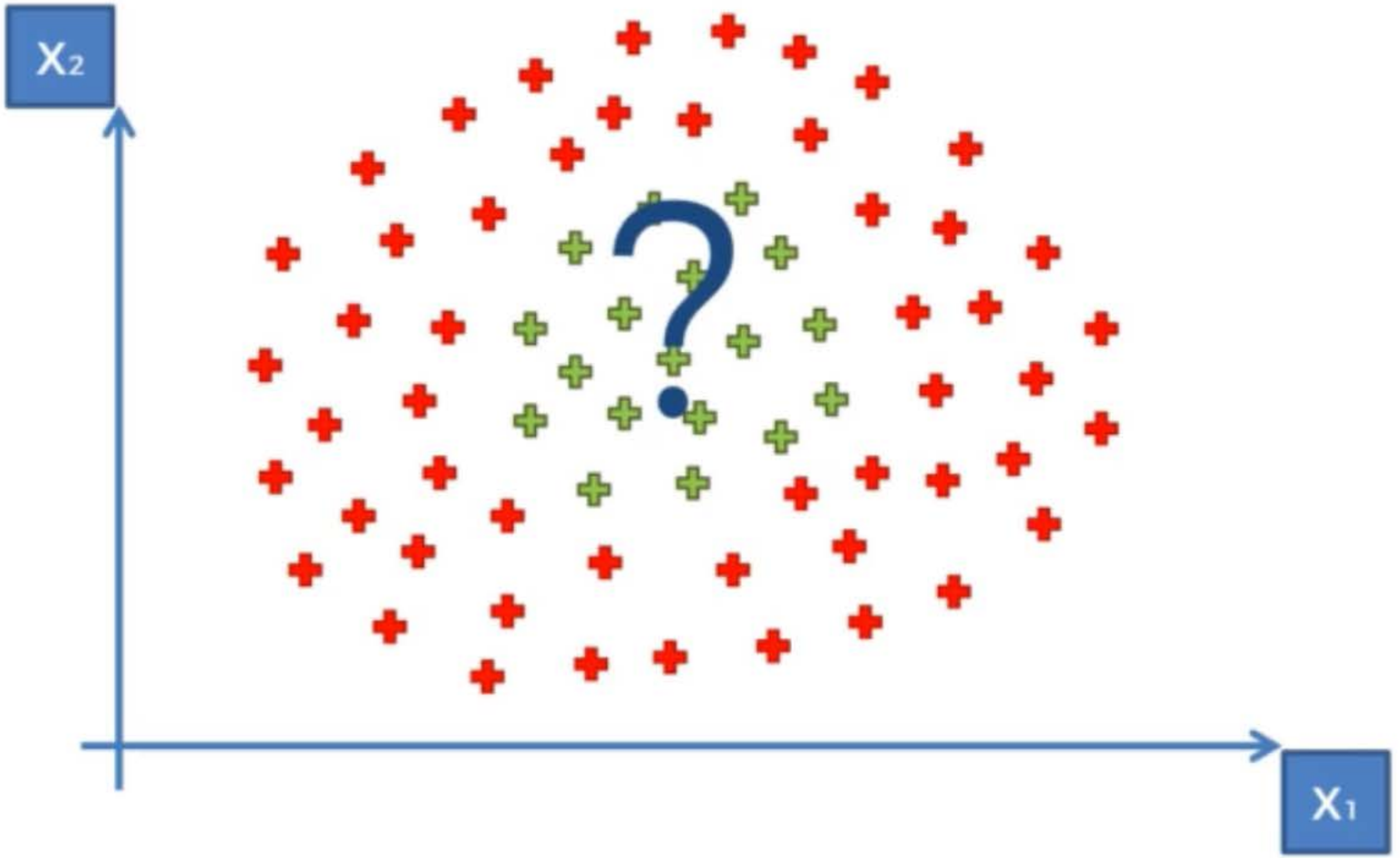


Kernel SVM Intuition

SVM separates well these points



What about these points ?

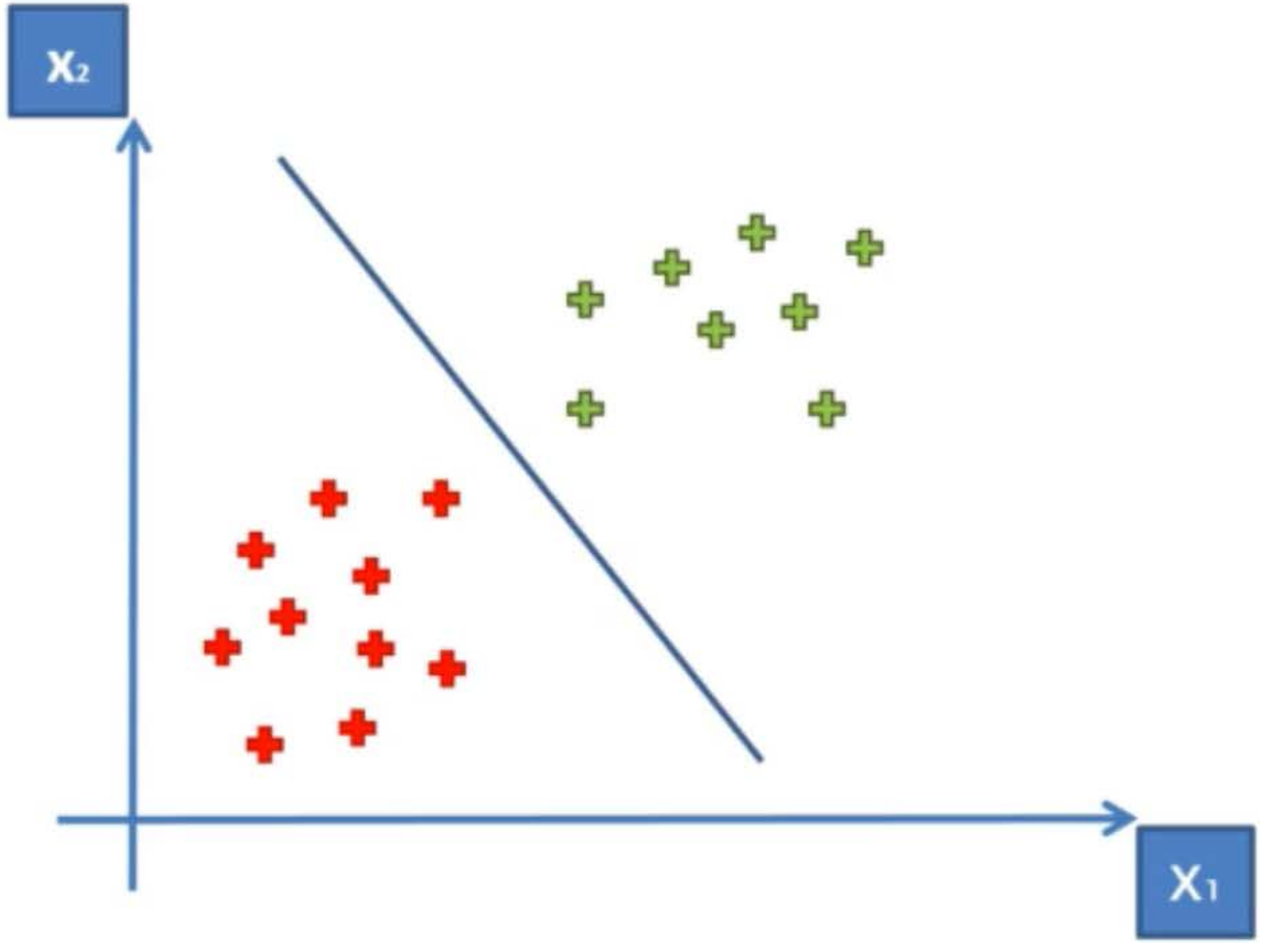


Why ?

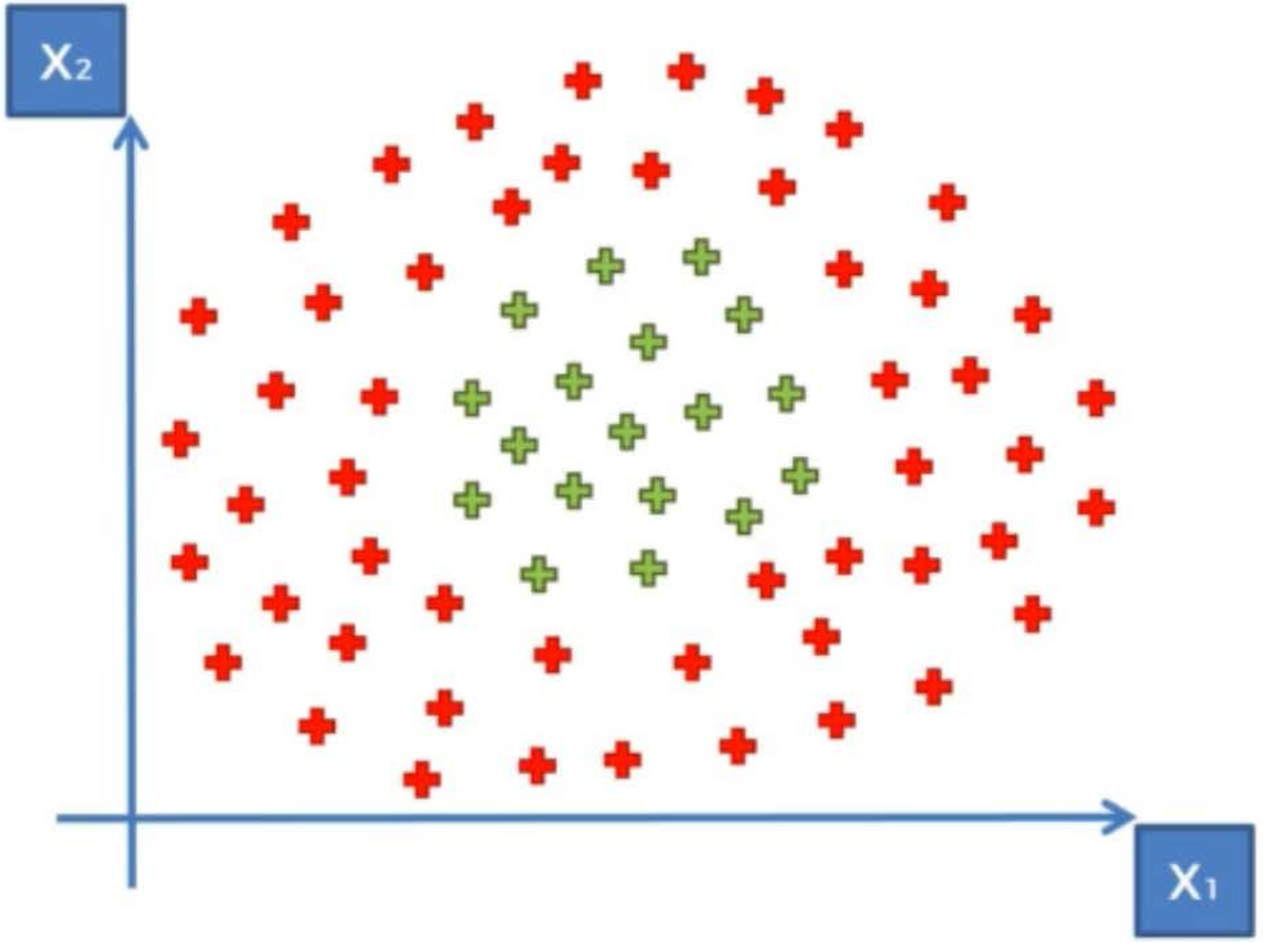
Because the data points are not LINEARLY SEPARABLE

Linear Separability

Linearly Separable



Not Linearly Separable



A Higher-Dimensional Space

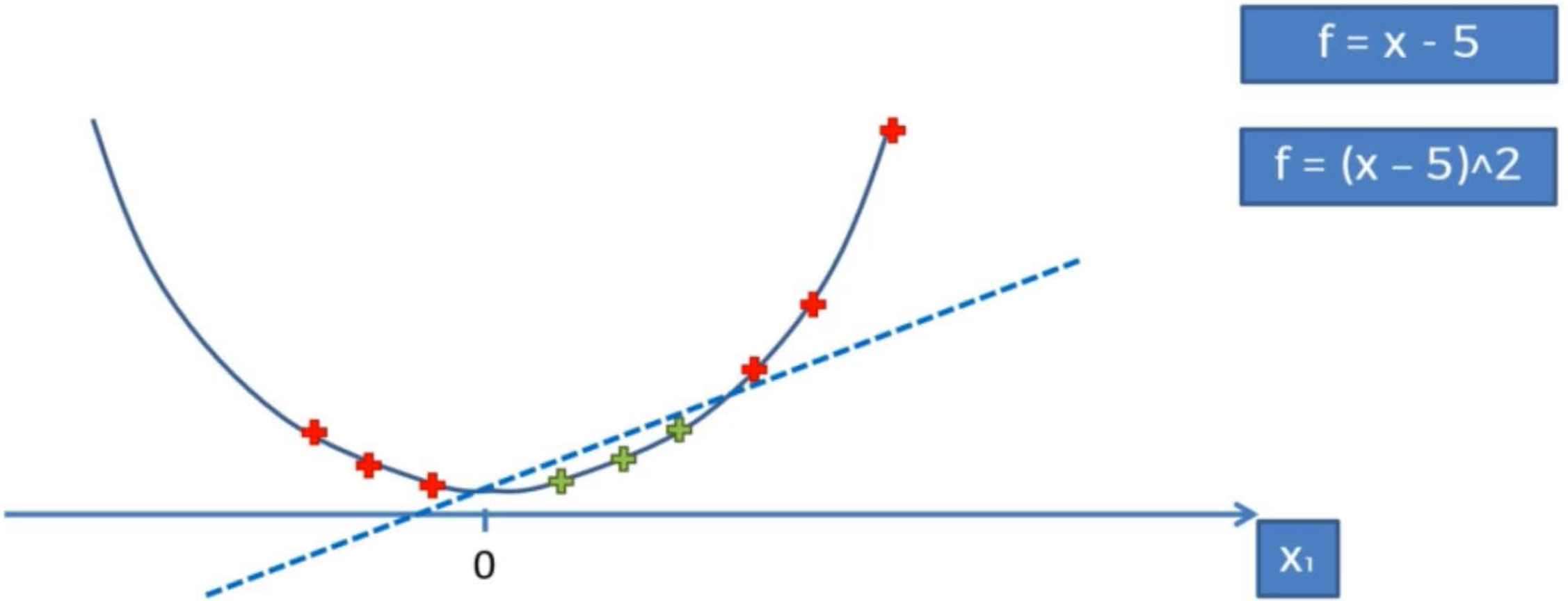
Mapping to a Higher Dimension

$f = x - 5$

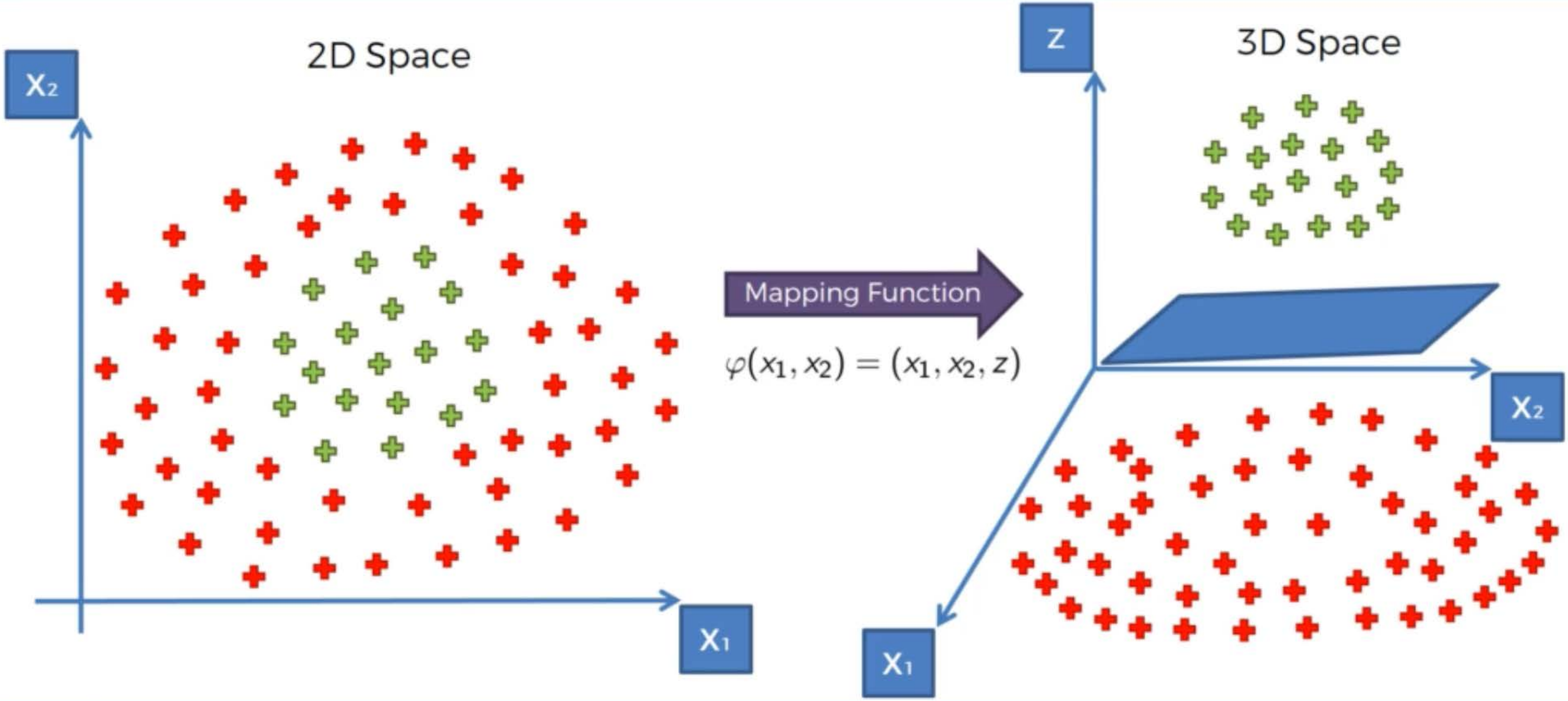
Mapping to a Higher Dimension

$f = x - 5$

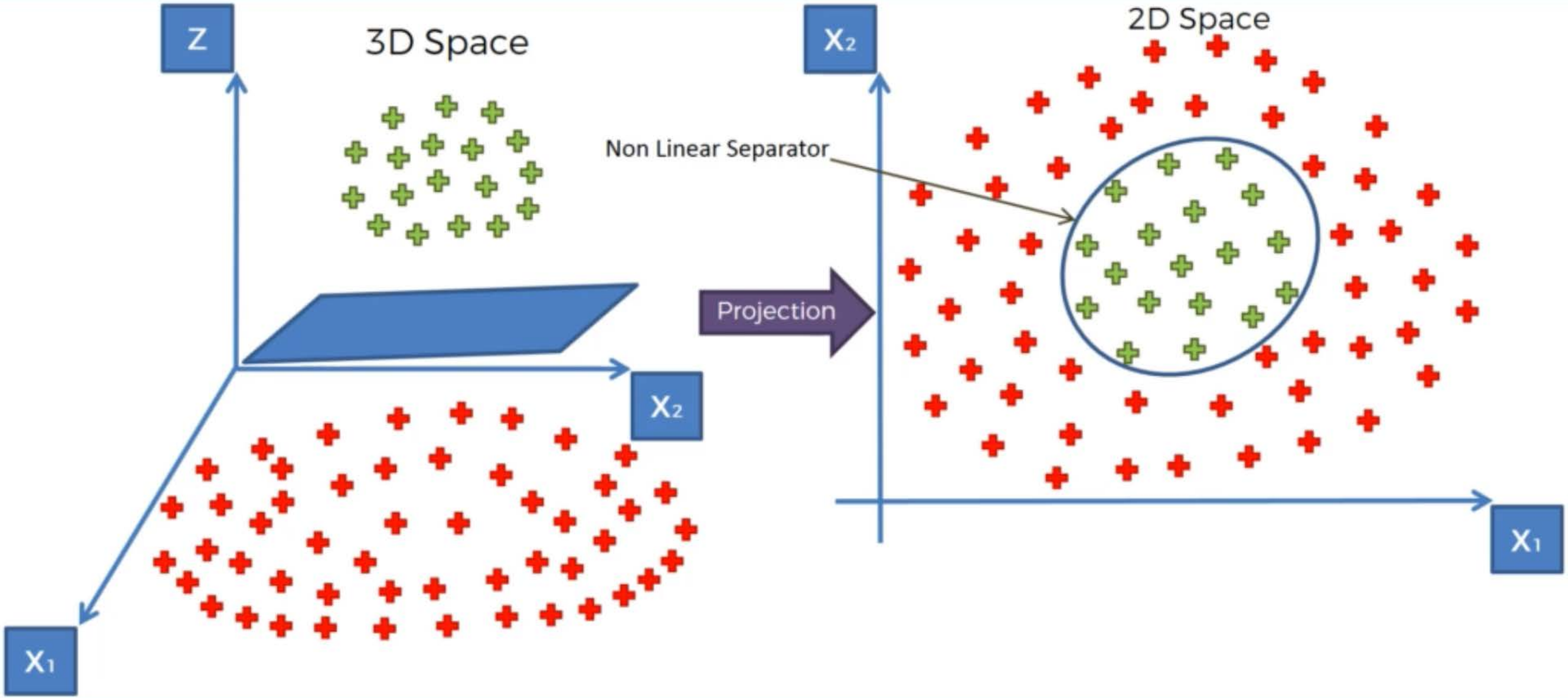
Mapping to a Higher Dimension



Mapping to a Higher Dimension



Projecting back to 2D Space



But there is a catch...

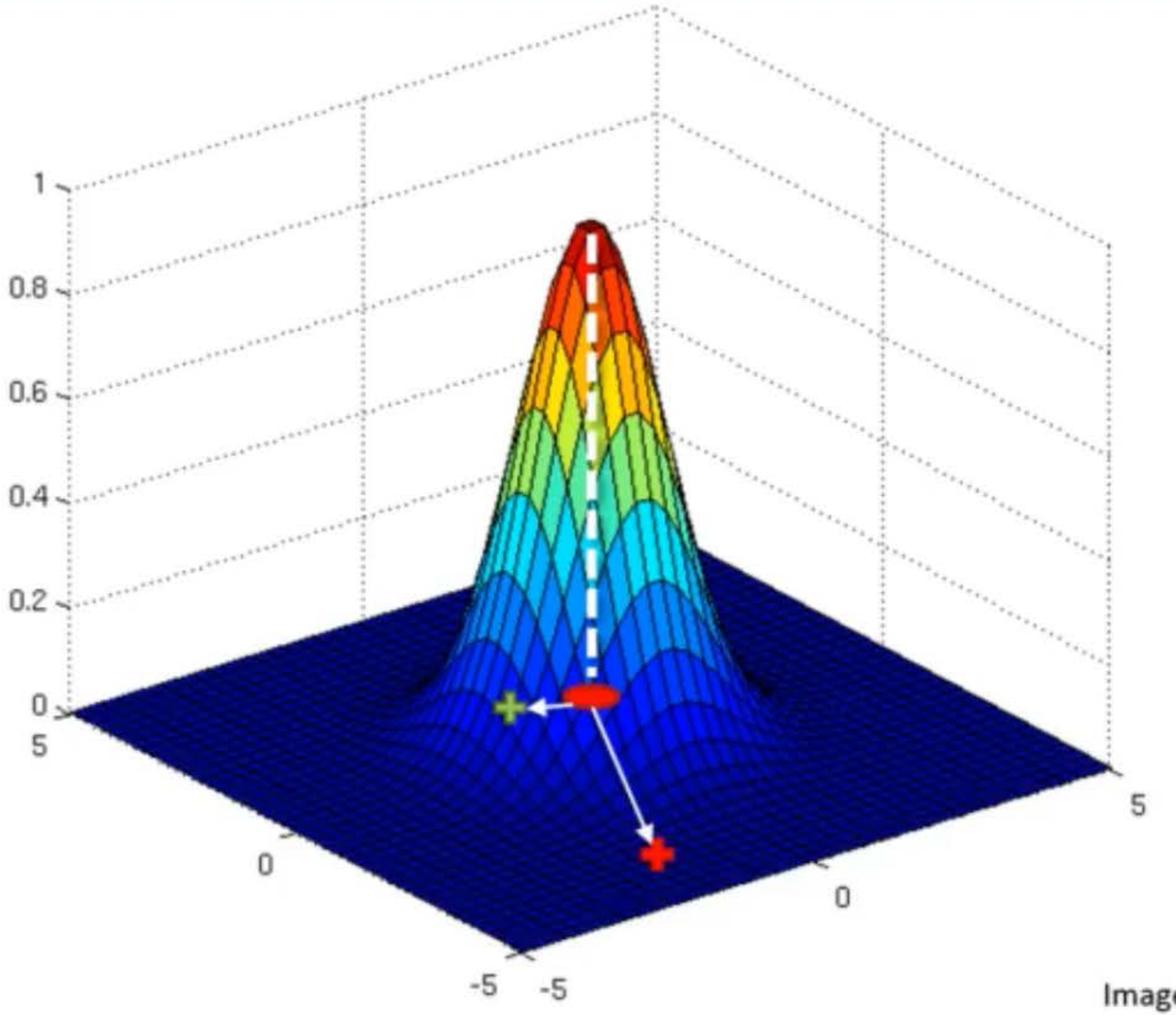
Mapping to a Higher Dimensional Space
can be highly compute-intensive

The Kernel Trick

The Gaussian RBF Kernel

$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$

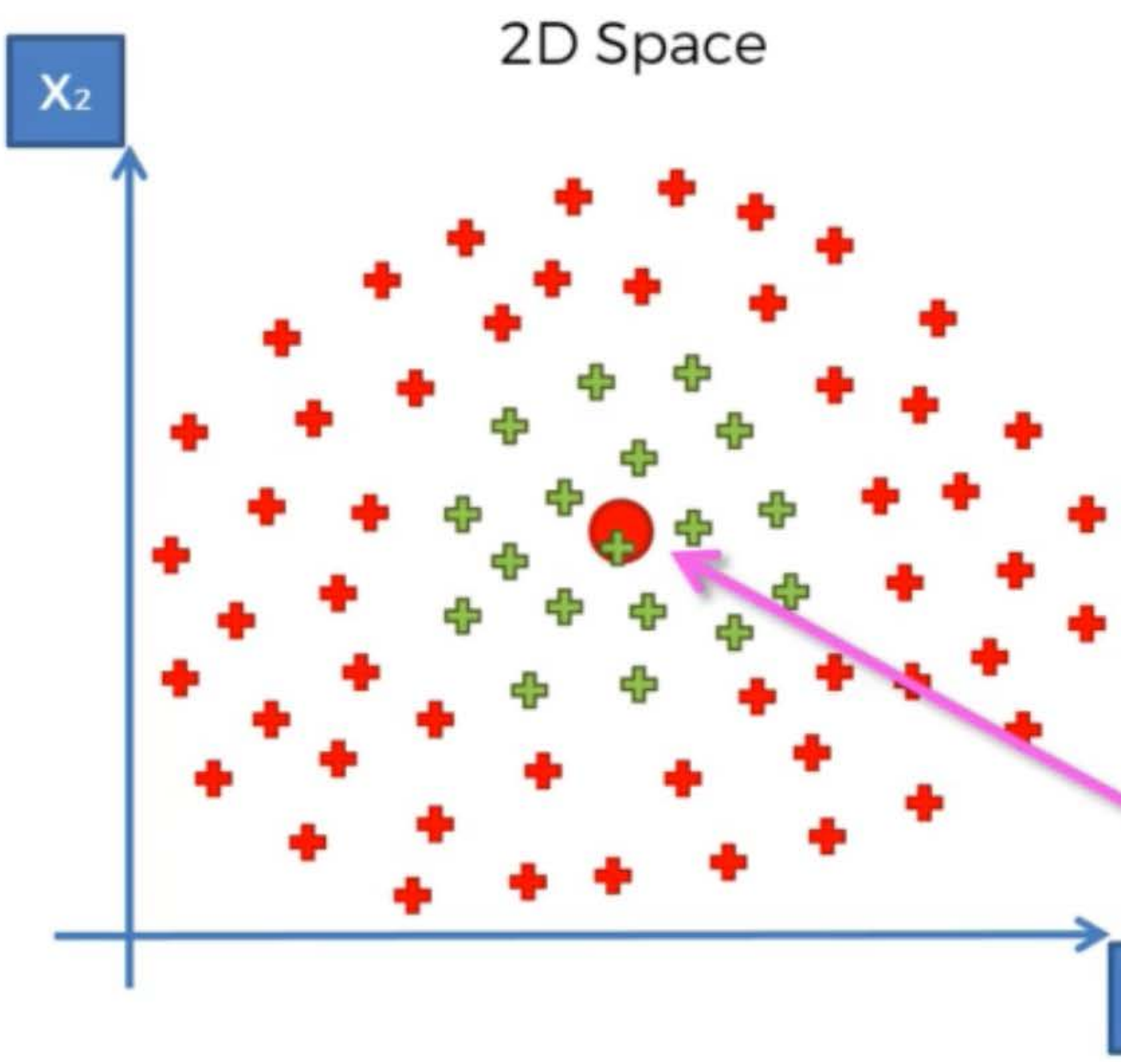
The Gaussian RBF Kernel

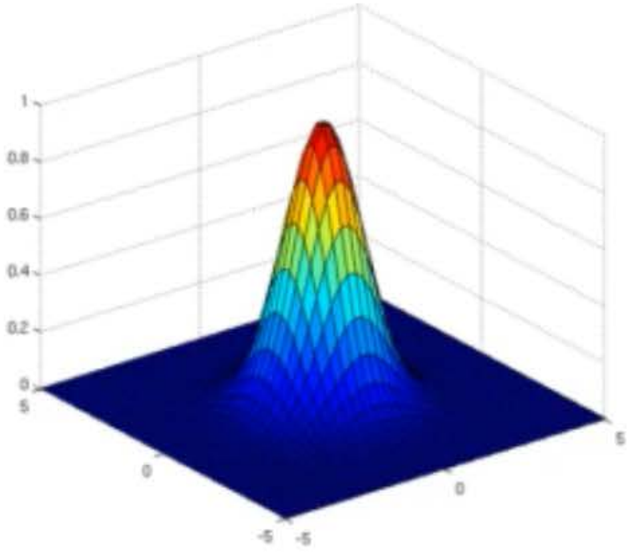


$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^{2}}{2\sigma^{2}}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

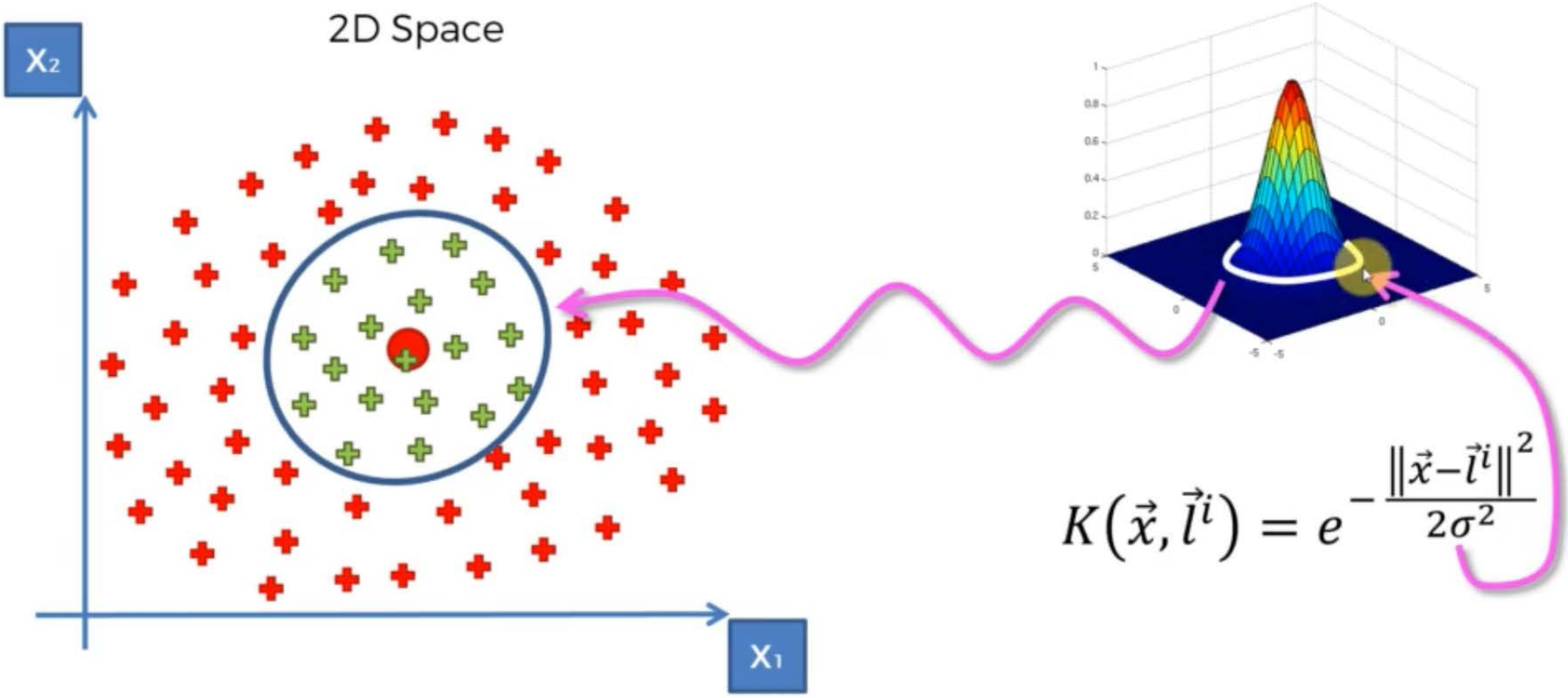
The Gaussian RBF Kernel



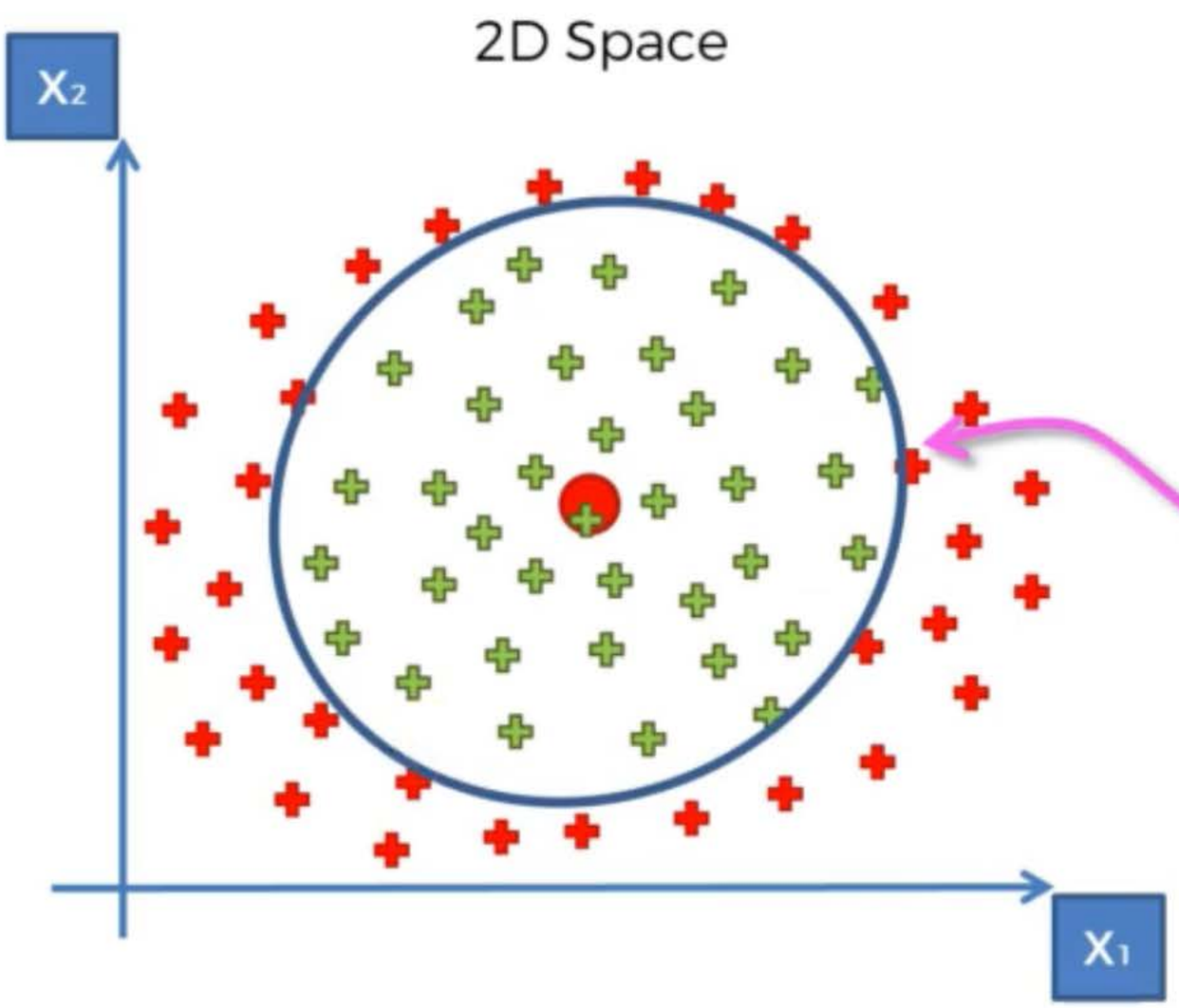


$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^{2}}{2\sigma^{2}}}$$

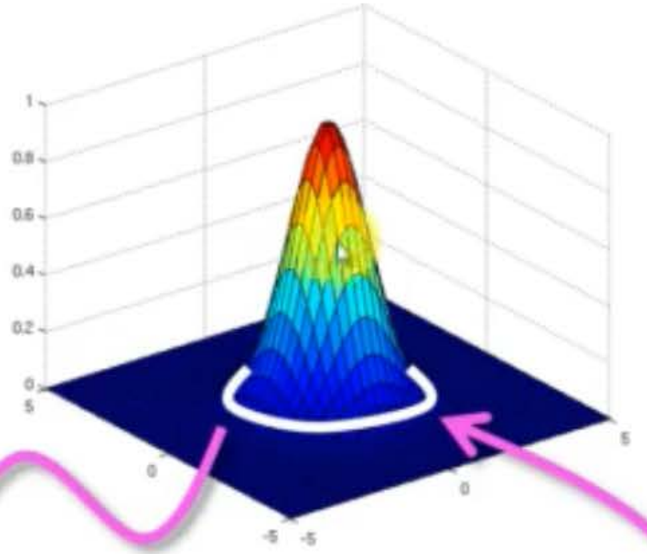
The Gaussian RBF Kernel



The Gaussian RBF Kernel

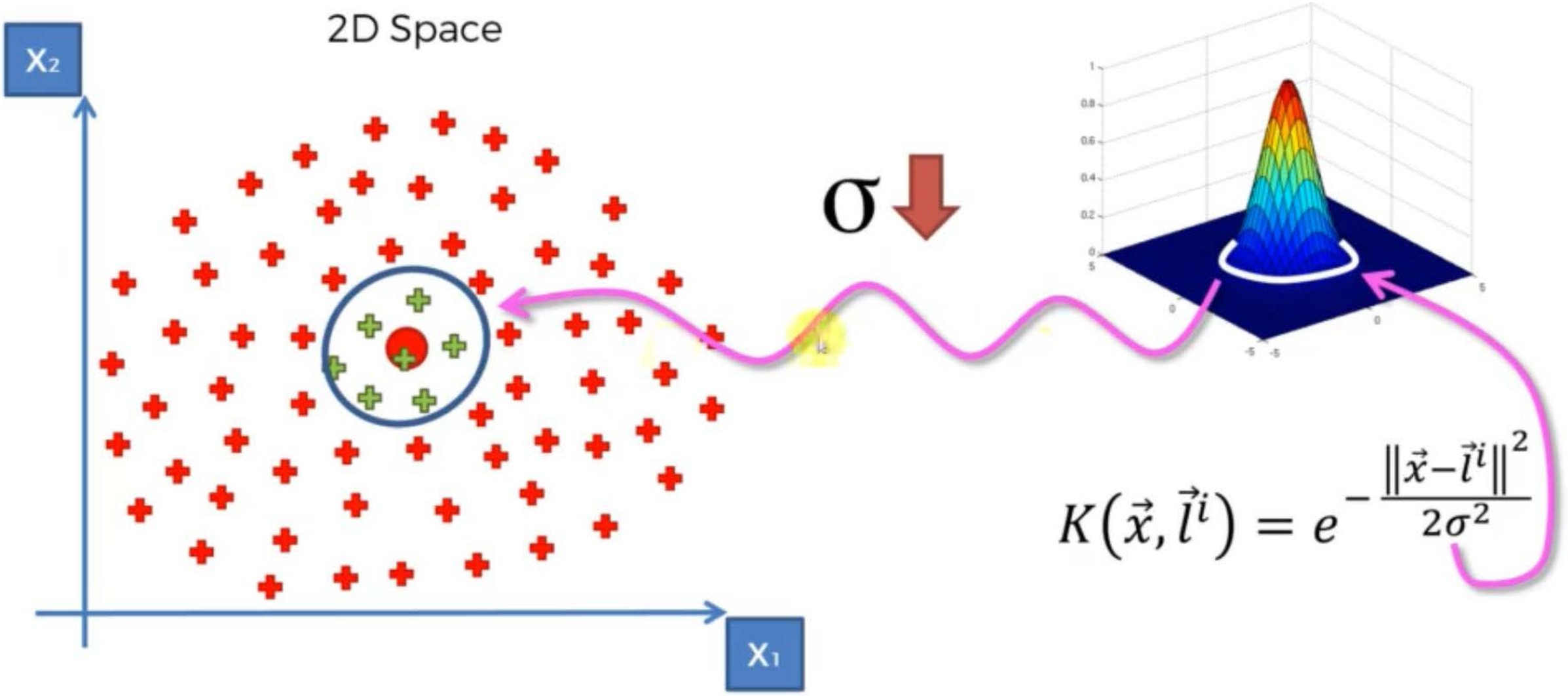


σ

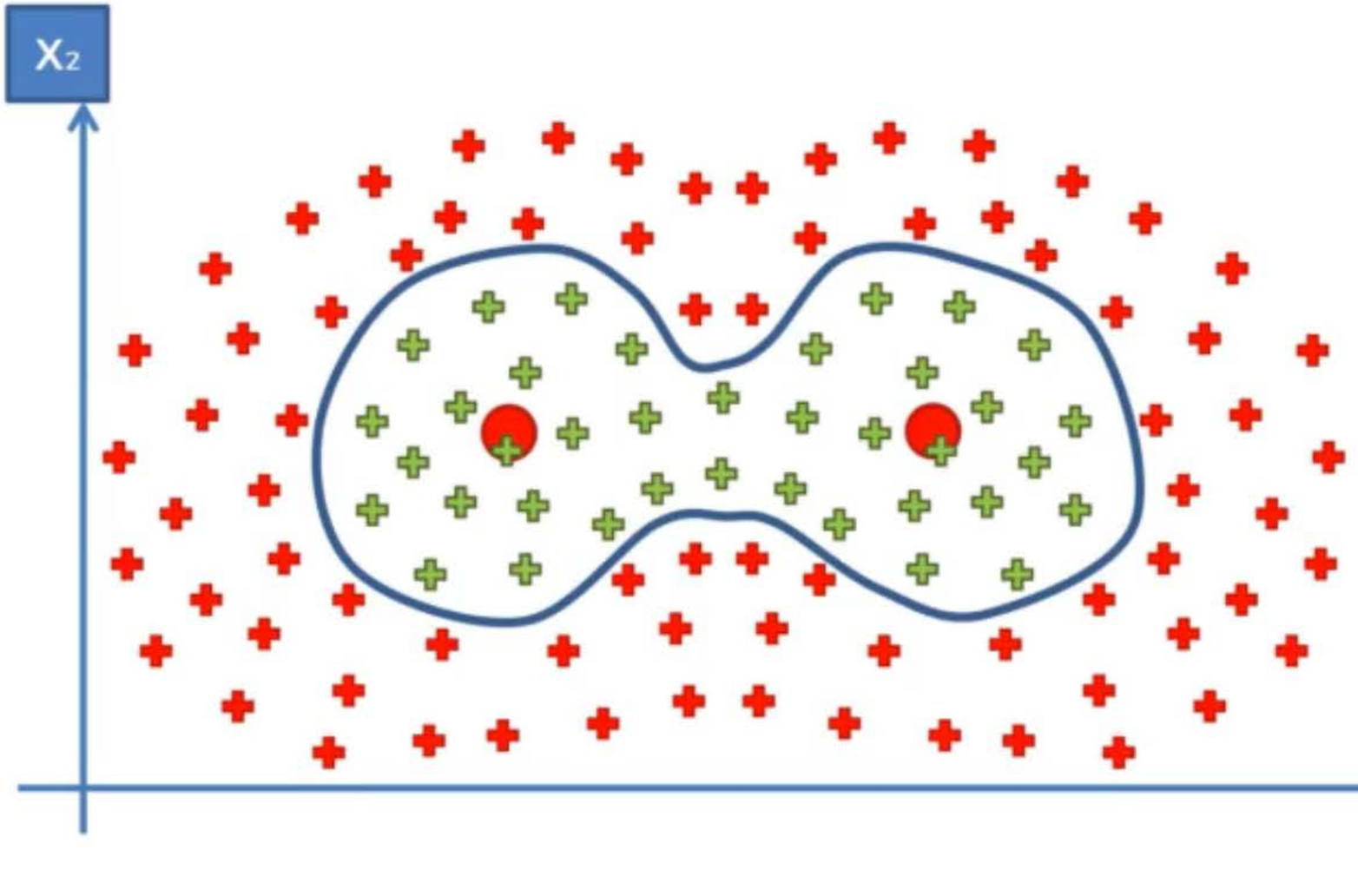


$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^{2}}{2\sigma^{2}}}$$

The Gaussian RBF Kernel



The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

Green when:

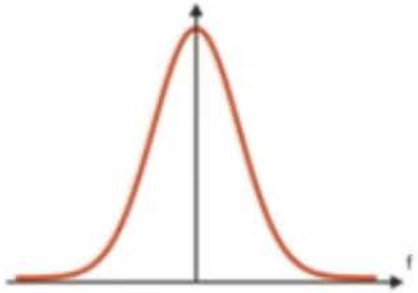
$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$

Red when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) = 0$$

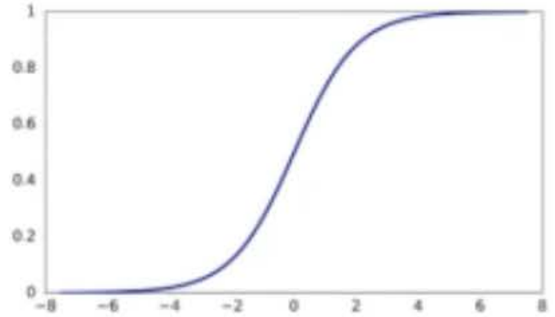
Types of Kernel Functions

Types of Kernel Functions



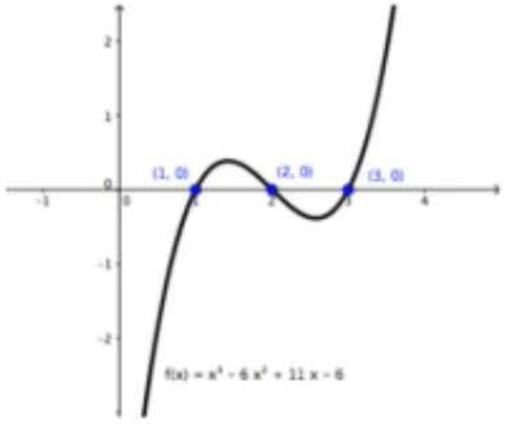
Gaussian RBF Kernel

$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$



Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^{T}Y + r)$$



Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^{T}Y + r)^{d}, \gamma > 0$$

Non-Linear SVR (Advanced)

Heads-up about Non-Linear SVR

Section on SVR:

- SVR Intuition ✓

Section on SVM:

- SVM Intuition ✓

Section on Kernel SVM:

- Kernel SVM Intuition ✓
- Mapping to a higher dimension ✓
- The Kernel Trick ✓
- Types of Kernel Functions ✓
- Non-linear Kernel SVR ←

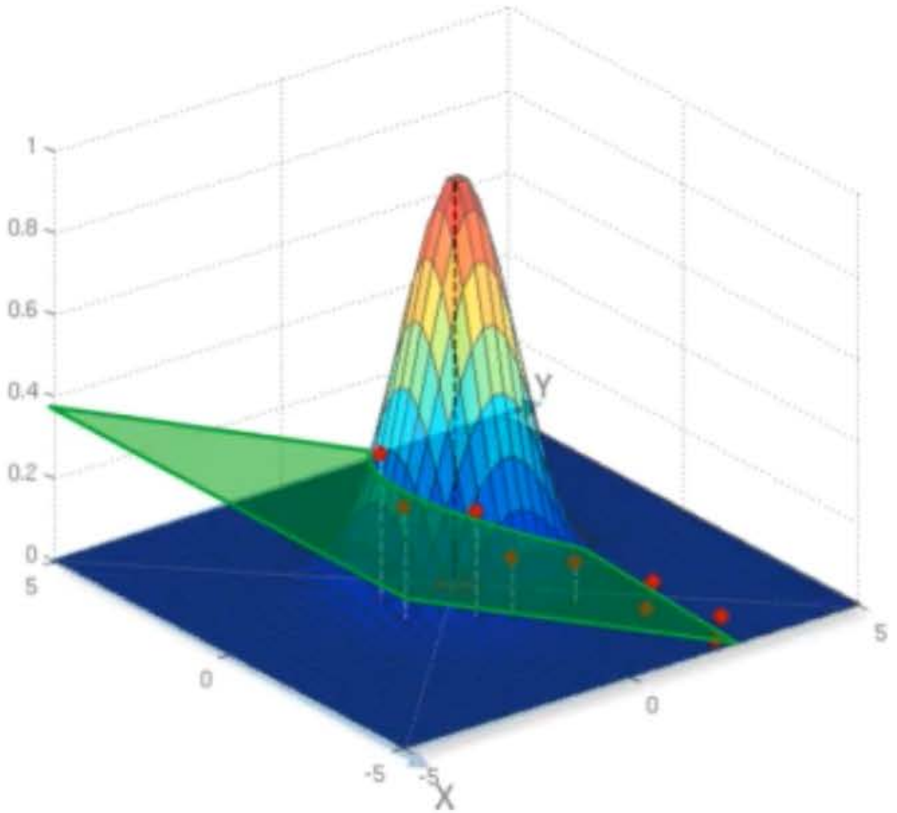
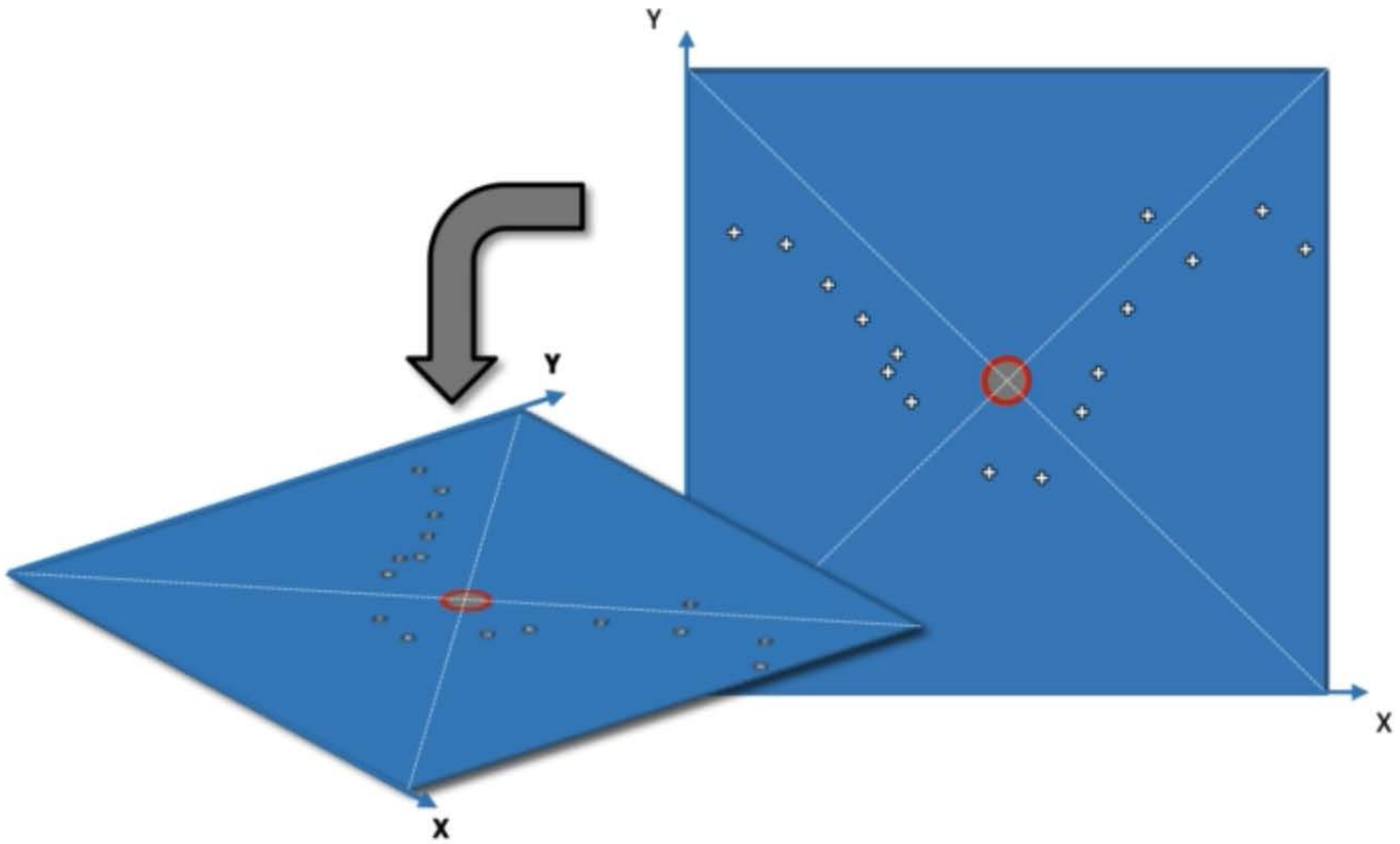


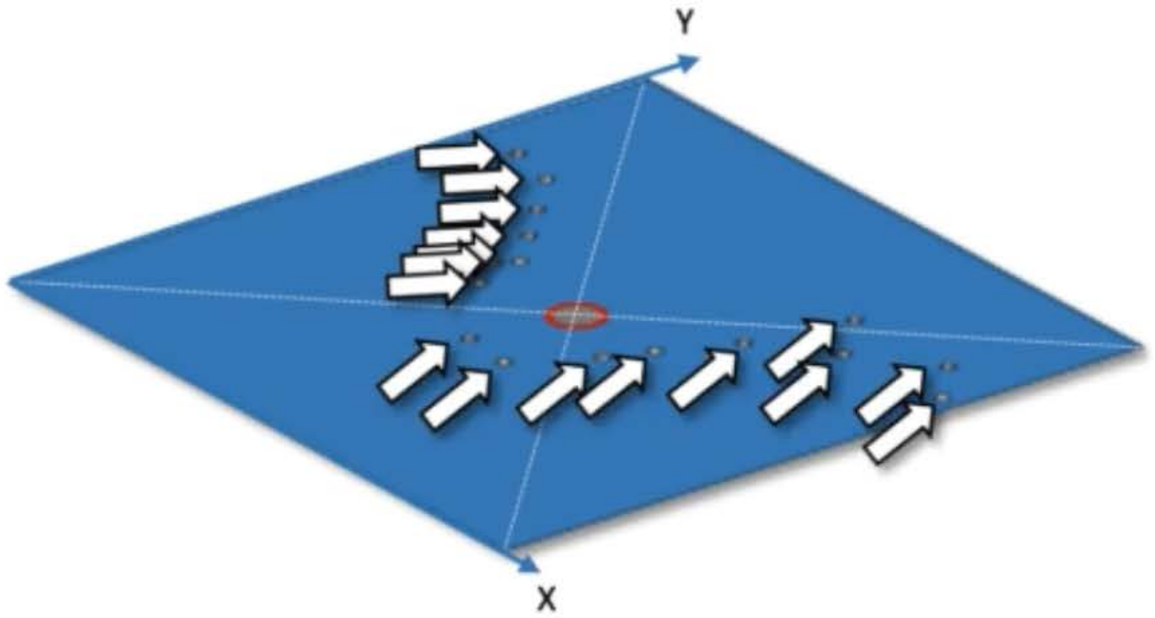
Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

Non-Linear SVR

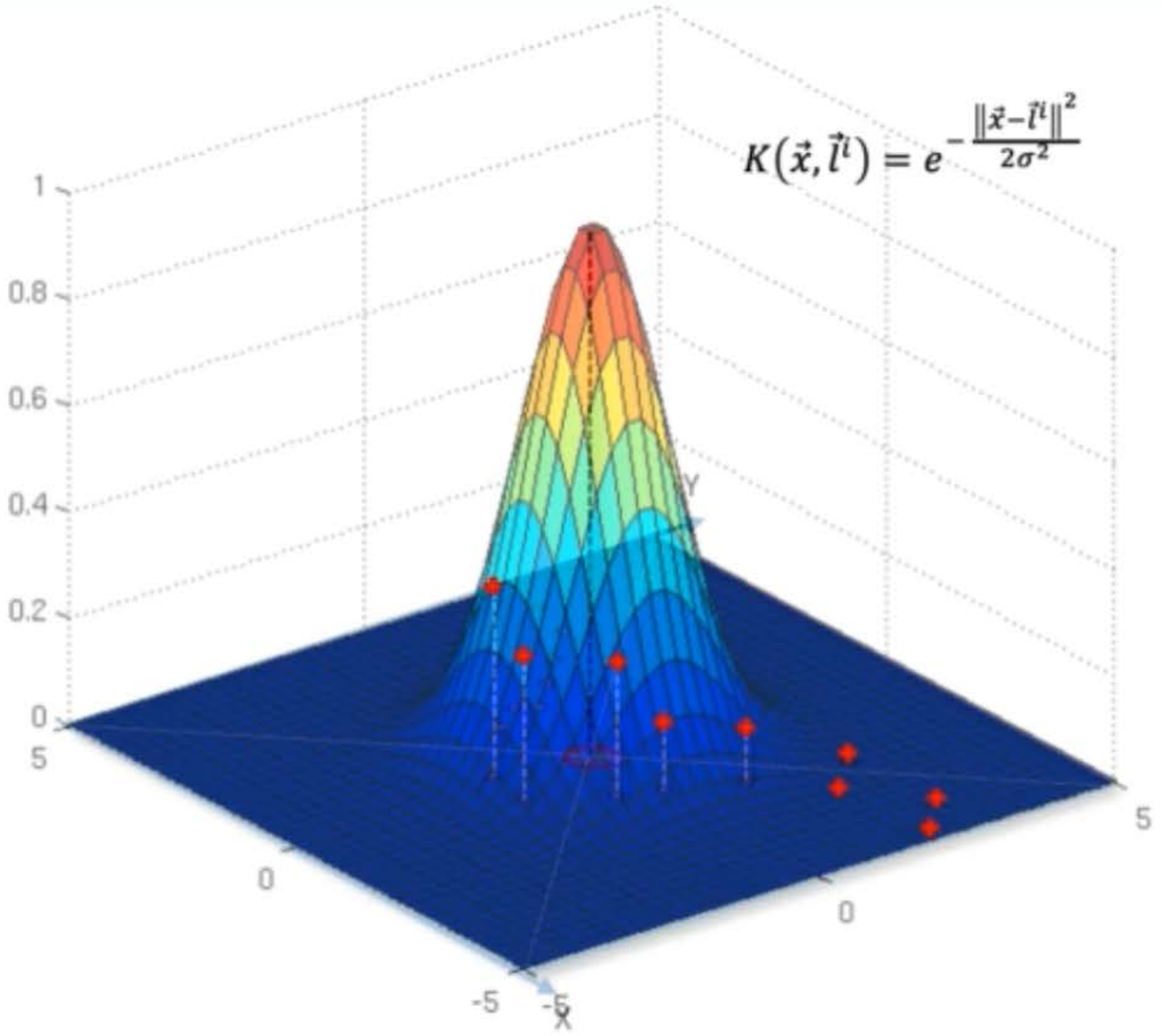
Non-Linear SVR



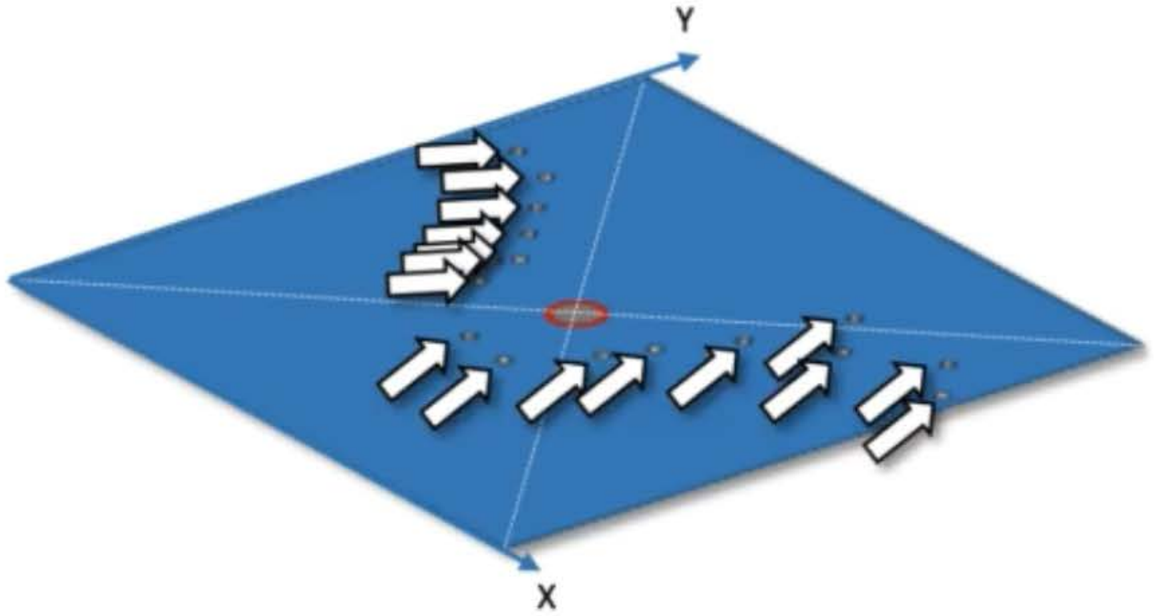
Non-Linear SVR

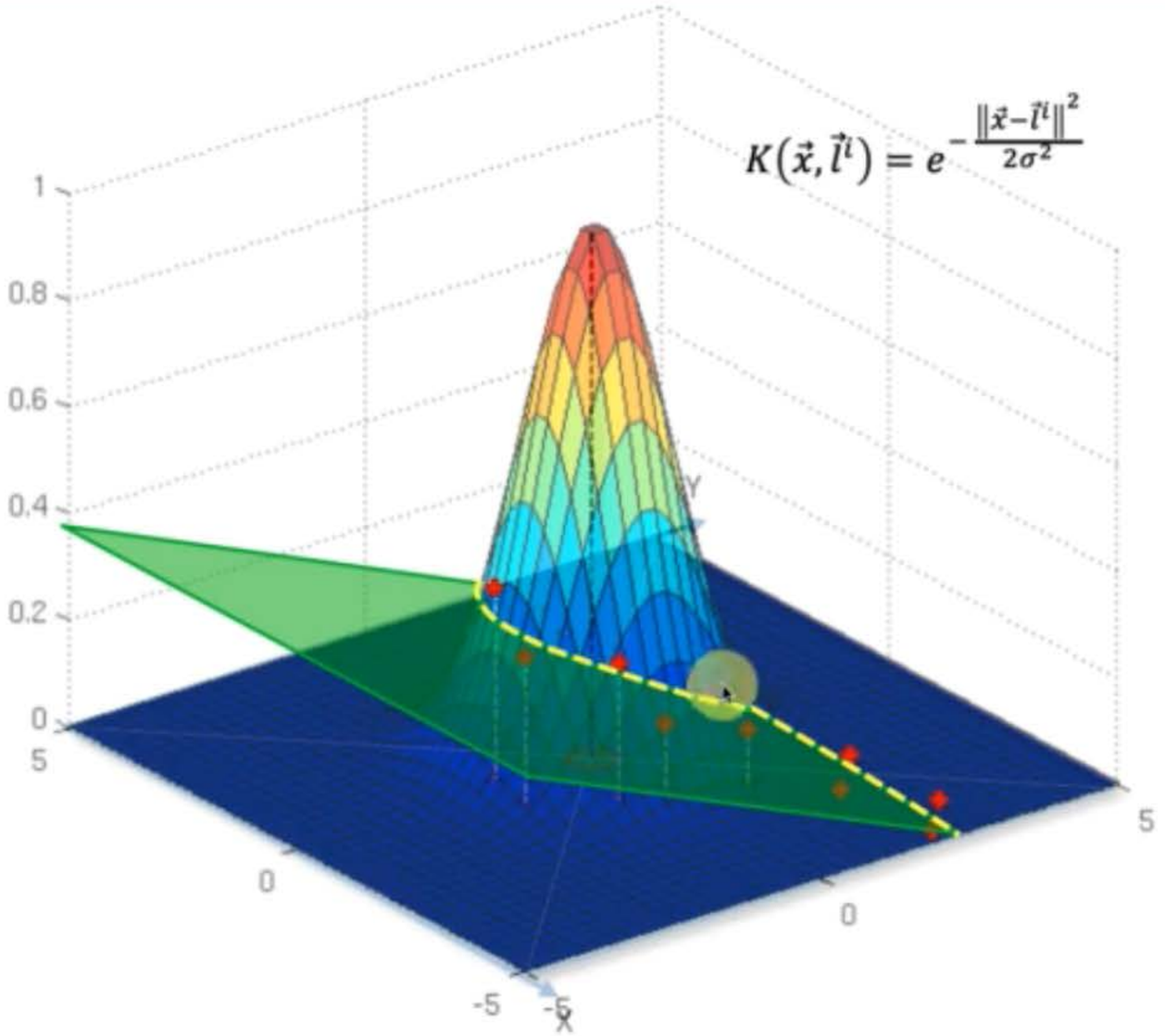


$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$

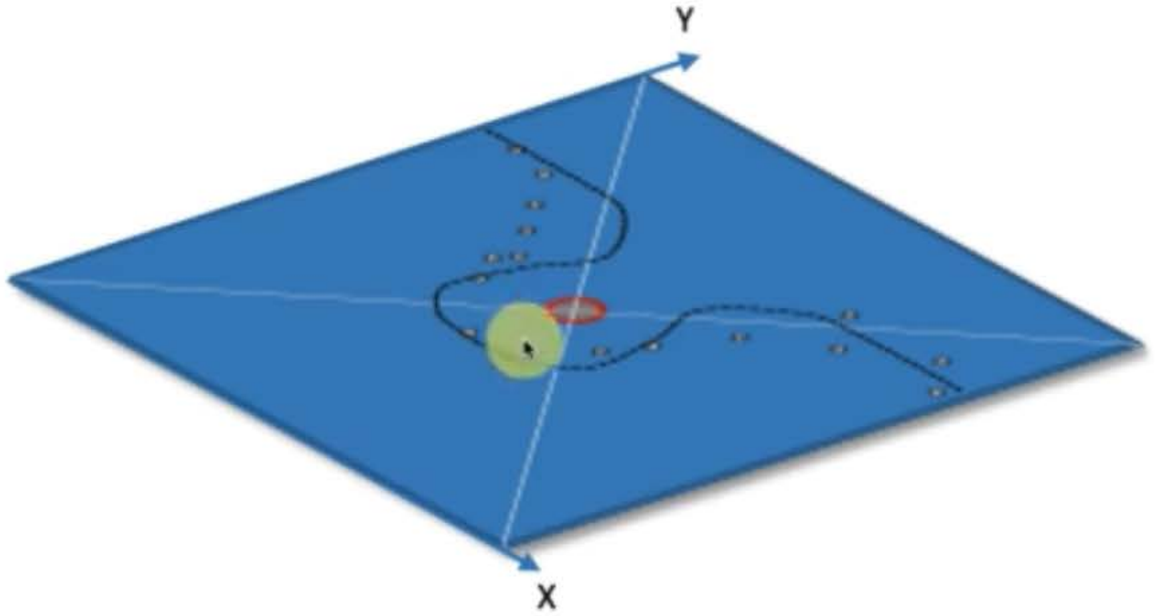


Non-Linear SVR

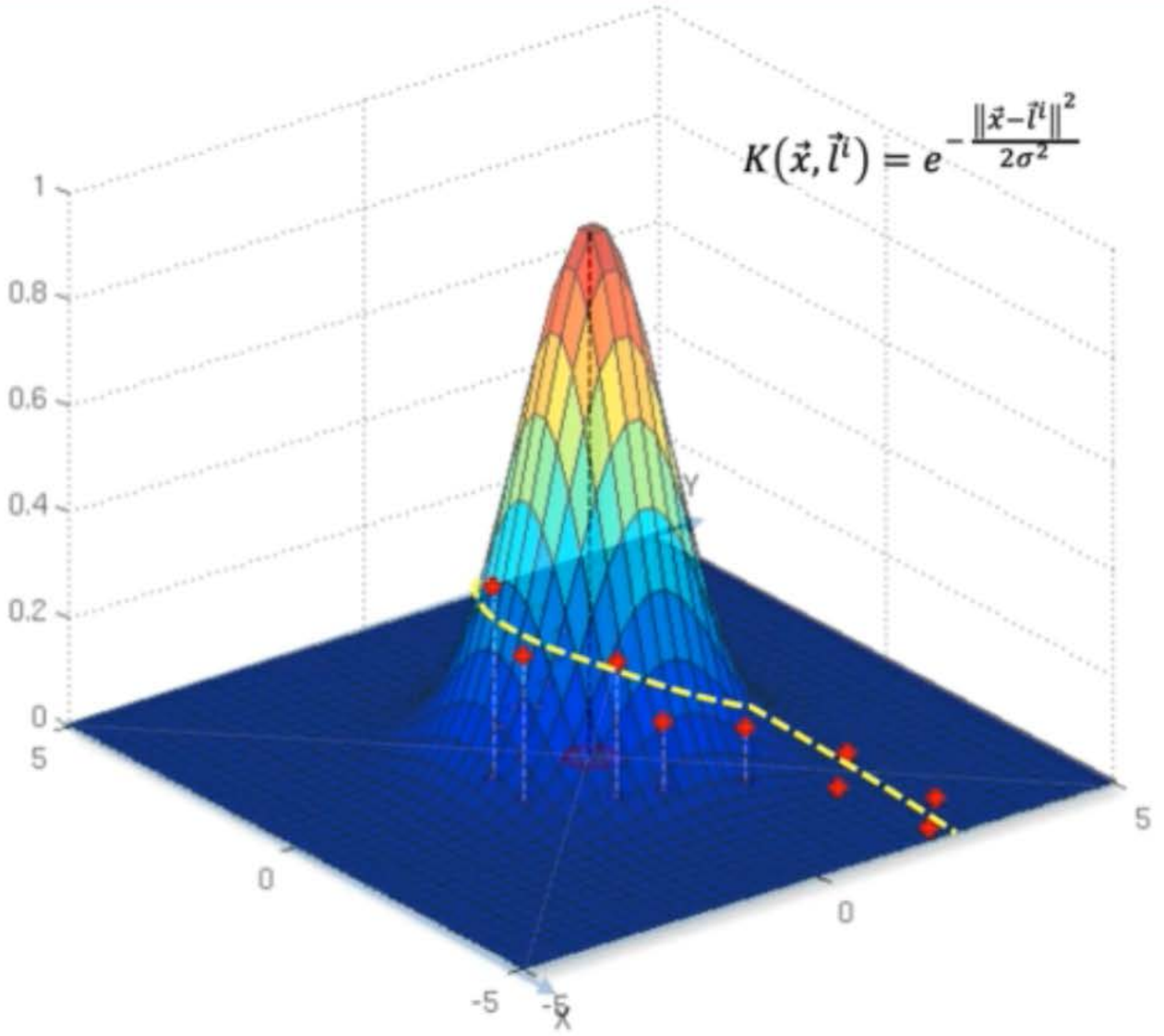




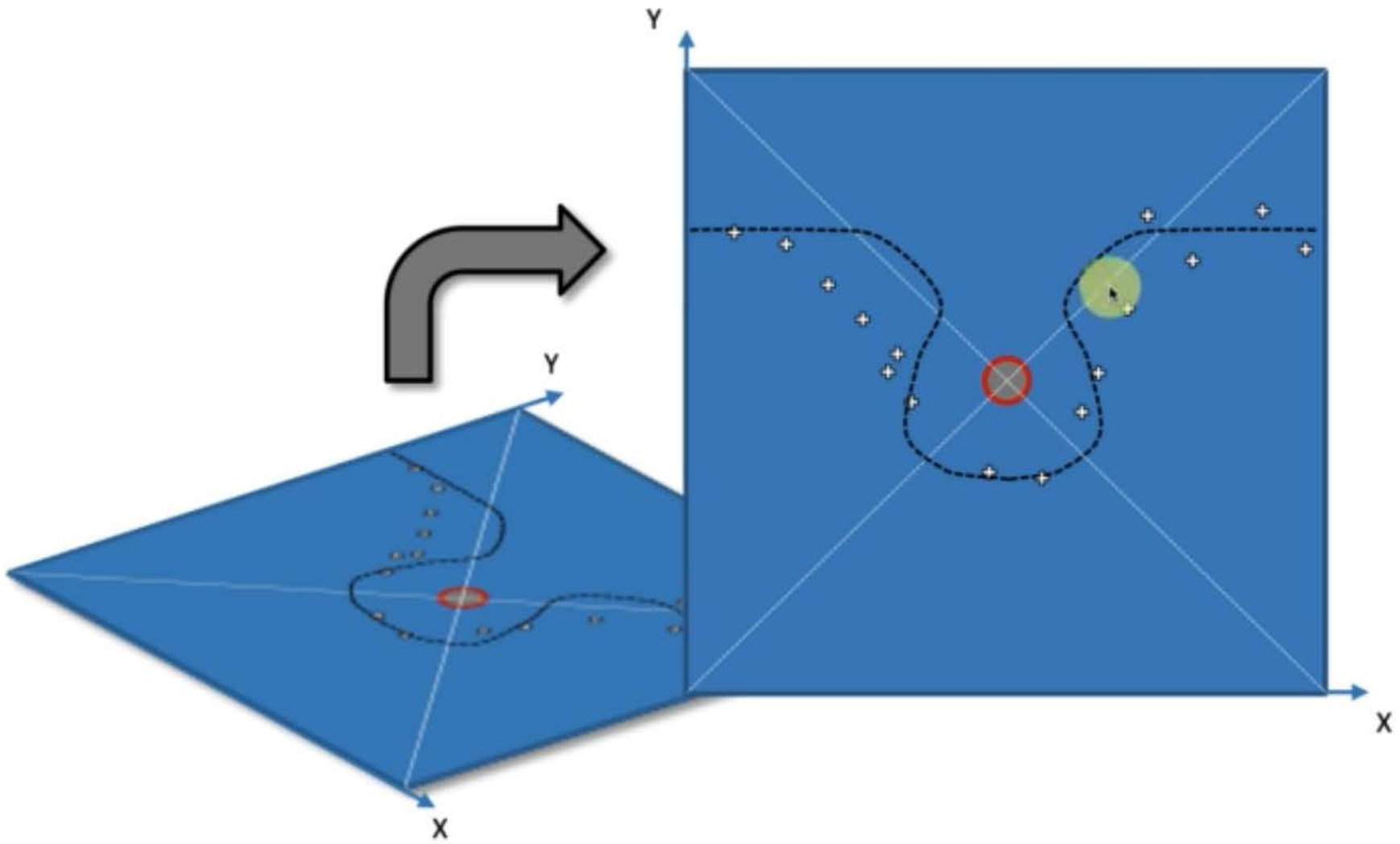
Non-Linear SVR



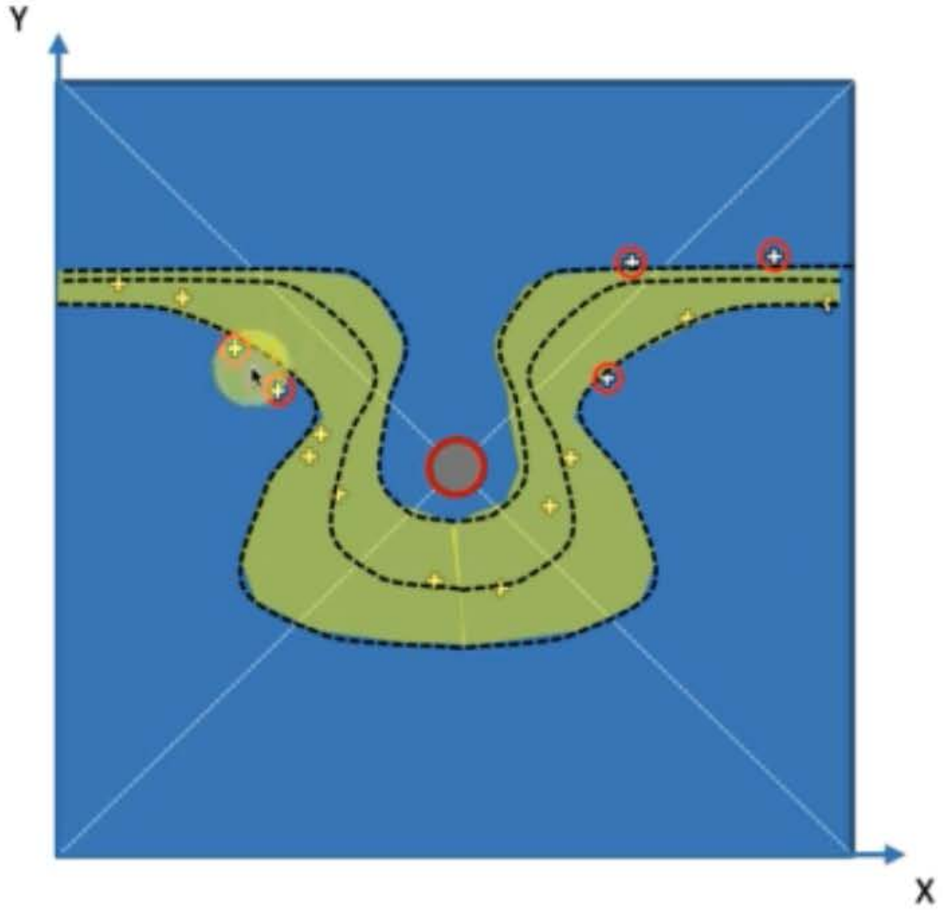
$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$

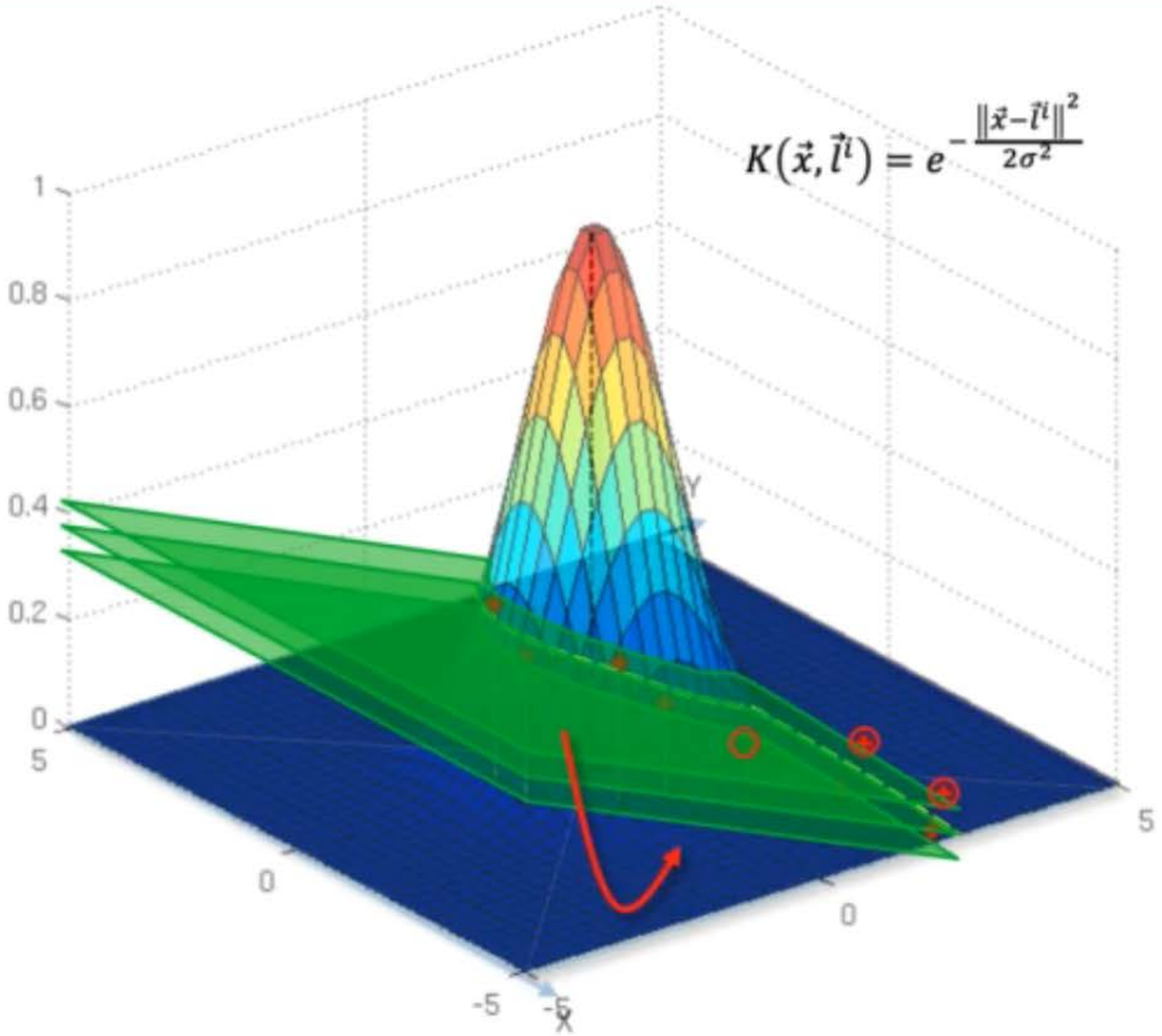


Non-Linear SVR



Non-Linear SVR

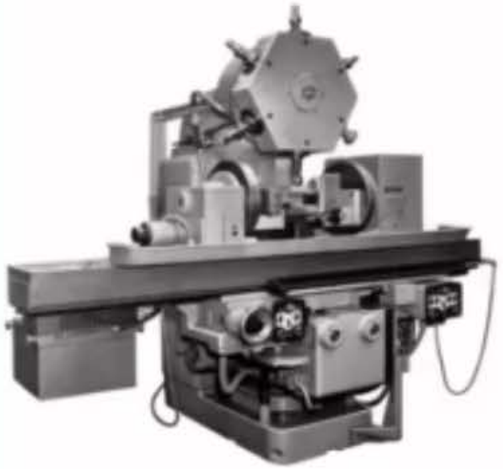




Bayes' Theorem

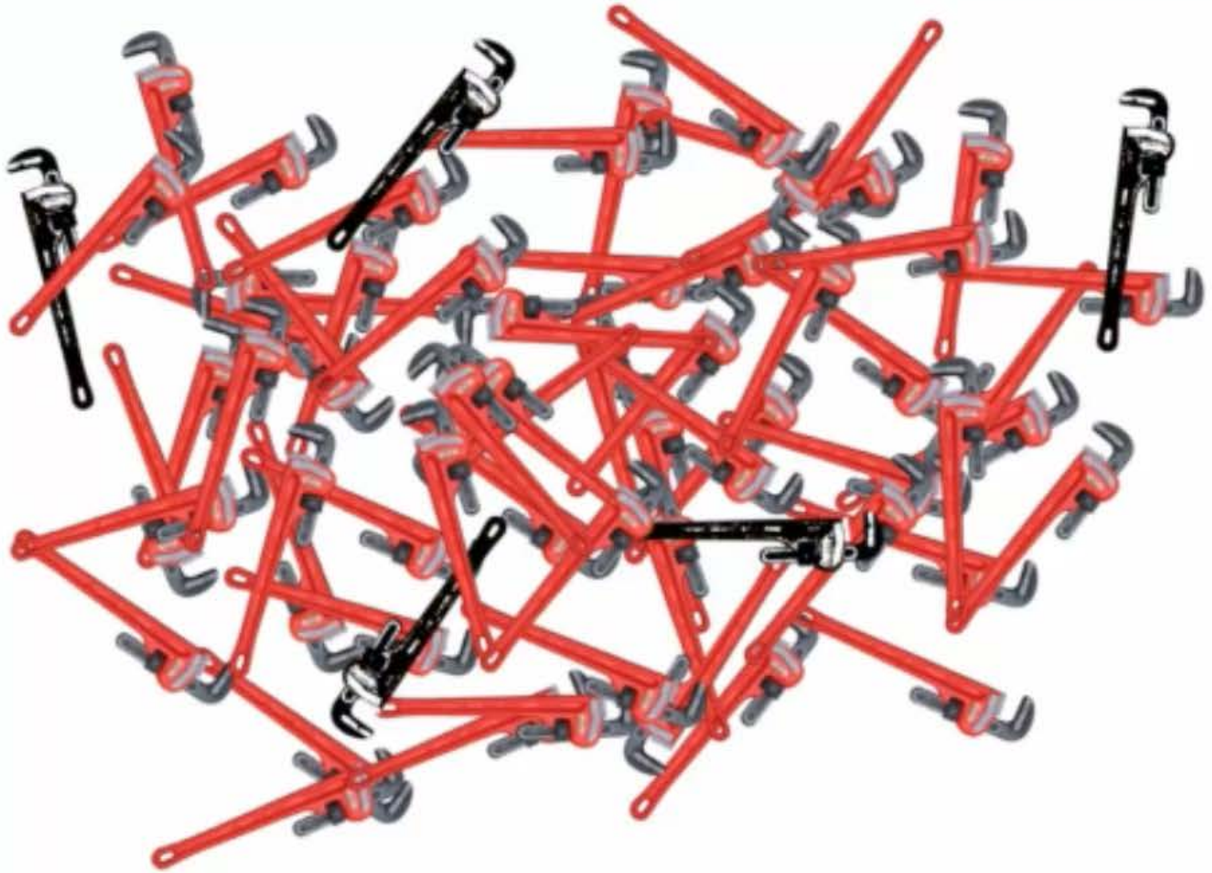
Bayes Theorem

m1 m1 m1 m1 m1 m1 m1 m1 m1 m1 m1 m1 m1



m2 m2 m2 m2 m2 m2 m2 m2 m2 m2

Bayes Theorem



Bayes Theorem

What's the probability?

m2

Bayes Theorem

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr

-> P(Mach1) = 30/50 = 0.6
-> P(Mach2) = 20/50 = 0.4

Out of all produced parts:
We can SEE that 1% are defective

-> P(Defect) = 1%

Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2

-> P(Mach1 | Defect) = 50%
-> P(Mach2 | Defect) = 50%

Question:
What is the probability that a part produced by mach2 is defective = ?

-> P(Defect | Mach2) = ?

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr
Out of all produced parts:
We can SEE that 1% are defective
Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2
Question:
What is the probability that a part produced by mach2 is defective = ?

-> P(Mach2) = 20/50 = 0.4
-> P(Defect) = 1%
-> P(Mach2 | Defect) = 50%
-> P(Defect | Mach2) = ?

$$P(Defect \mid Mach2) = \frac{P(Mach2 \mid Defect) * P(Defect)}{P(Mach2)}$$

It's intuitive!

$$P(\text{Defect} \mid \text{Mach2}) = \frac{P(\text{Mach2} \mid \text{Defect}) * P(\text{Defect}) \ *1000}{P(\text{Mach2}) \ *1000} = 1.25\%$$

Let's look at an example:

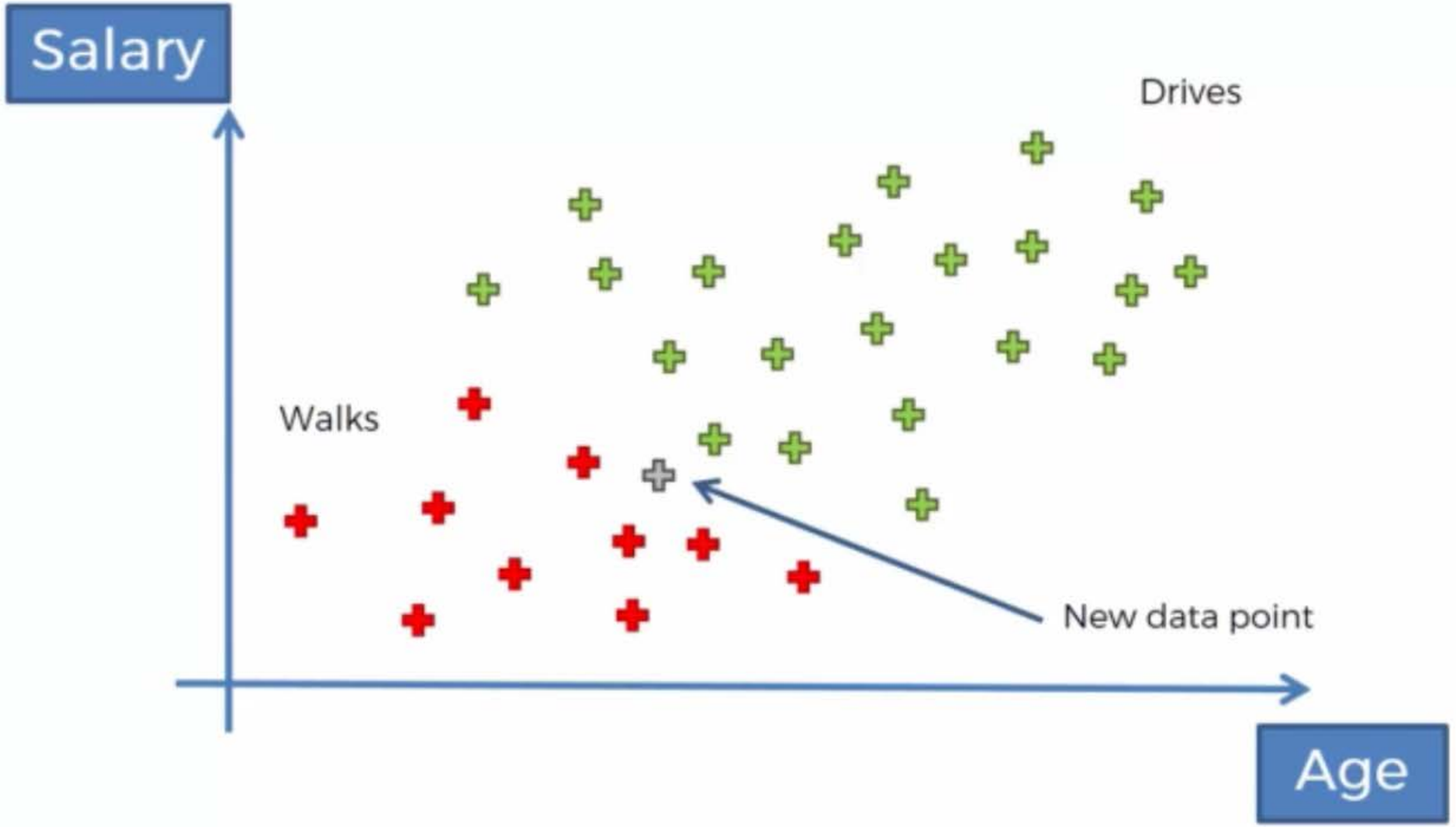
- 1000 wrenches
- 400 came from Mach2
- 1% have a defect = 10
- of them 50% came from Mach2 = 5
- % defective parts from Mach2 = 5/400 = 1.25%

It's intuitive!

Obvious question:
If the items are labeled, why couldn't we just count the number of defective wrenches that came from Mach2 and divide by the total number that came from Mach2?

Naïve Bayes Classifier Intuition

Naïve Bayes



Plan of Attack

Step 1

#4 Posterior Probability

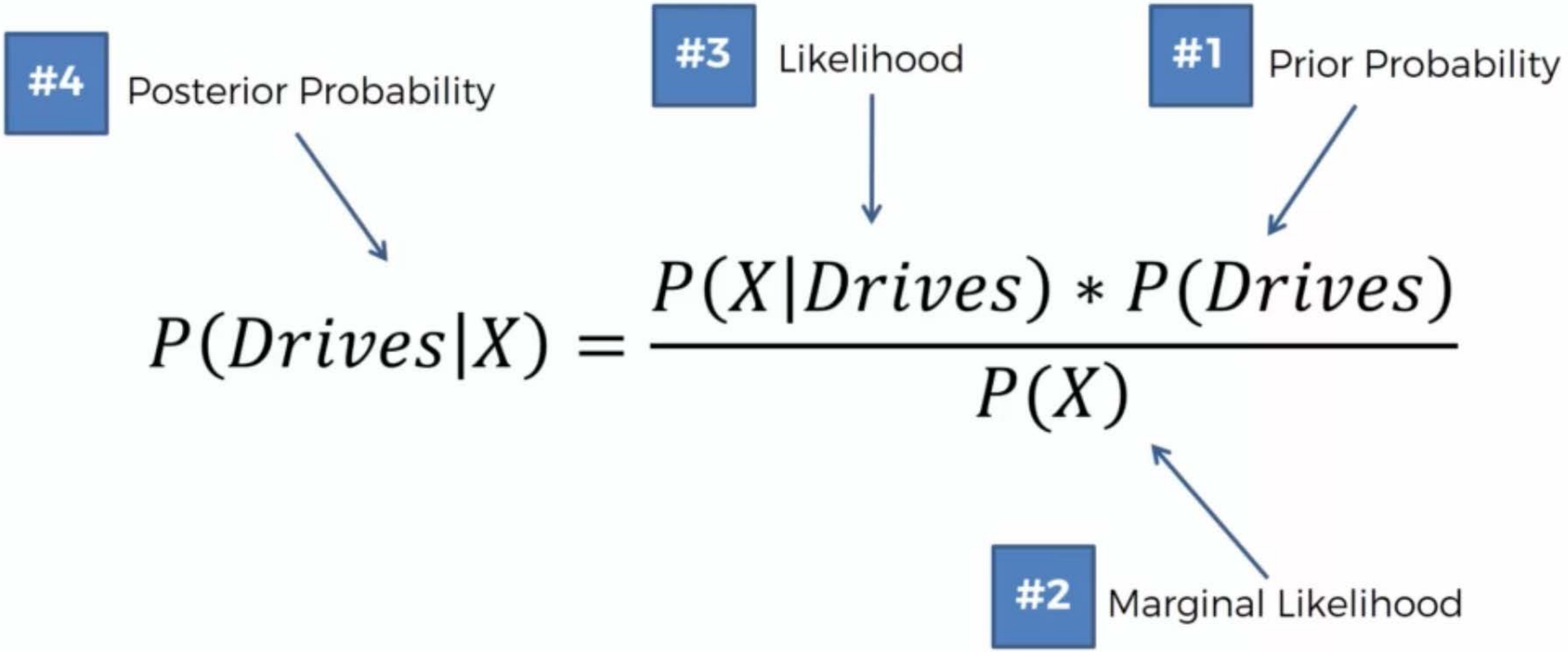
#3 Likelihood

#1 Prior Probability

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

#2 Marginal Likelihood

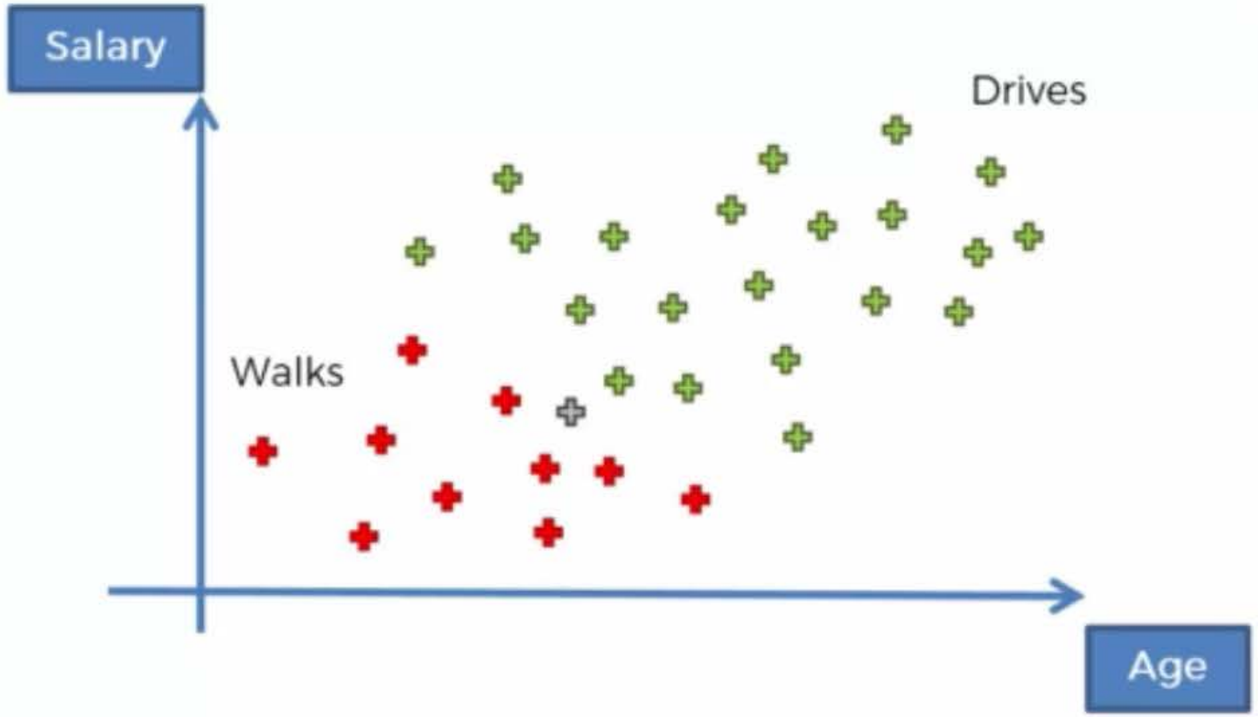
Step 2



Step 3

$$P(Walks|X) \; v.s. \; P(Drives|X)$$

Naïve Bayes: Step 1

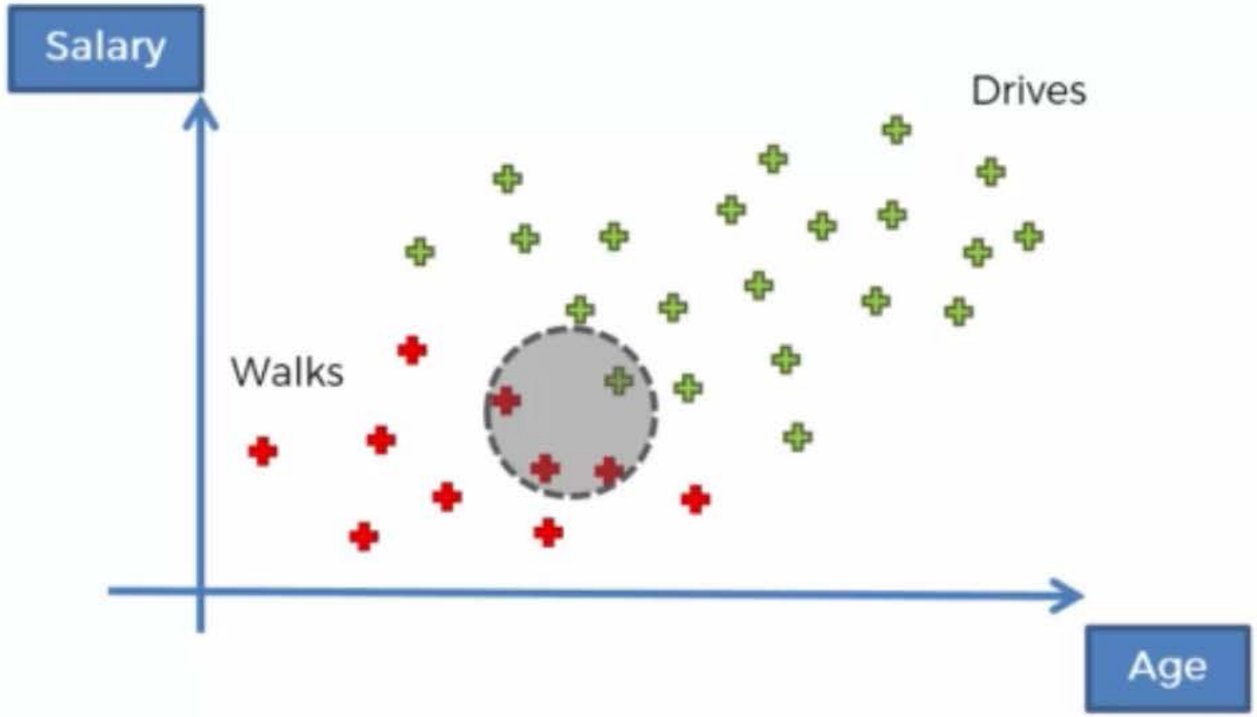


#1. P(Walks)

$$P(Walks) = \frac{Number\ of\ Walkers}{Total\ Observations}$$

$$P(Walks) = \frac{10}{30}$$

Naïve Bayes: Step 1

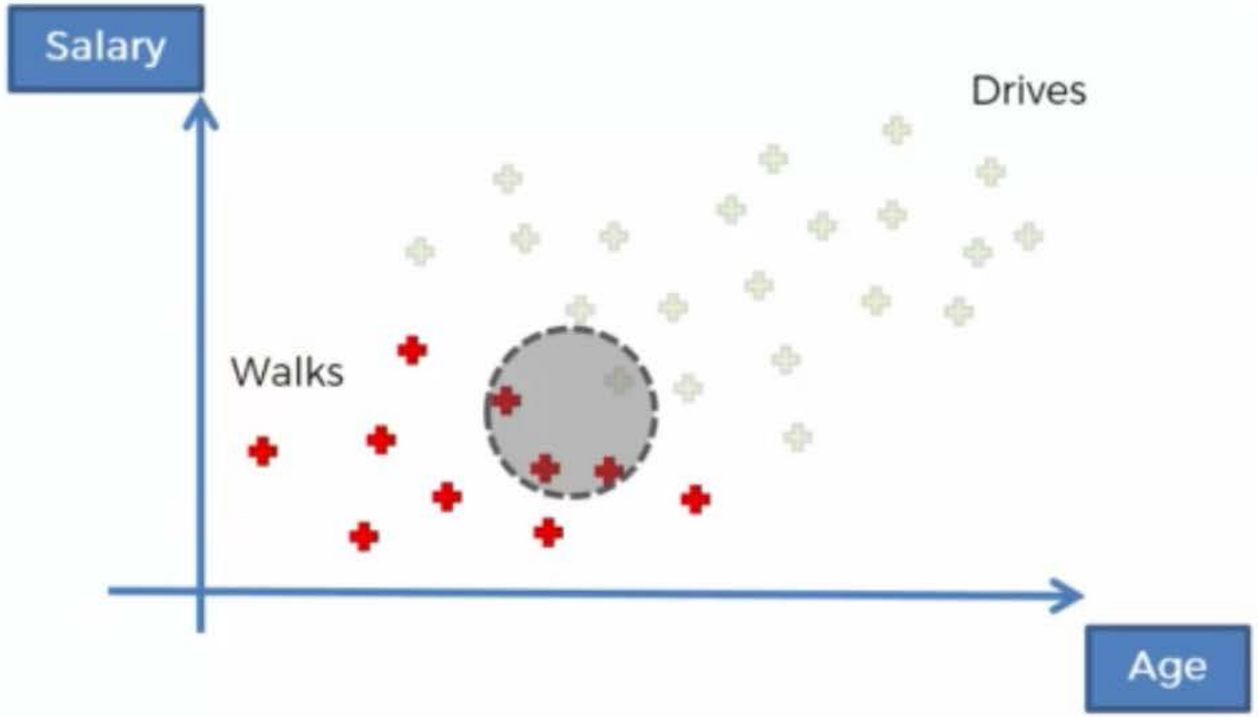


#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

Naïve Bayes: Step 1



#3. P(X|Walks)

$$P(X|Walks) = \frac{Number\ of\ Similar\ Observations\ Among\ those\ who\ Walk}{Total\ number\ of\ Walkers}$$

$$P(X|Walks) = \frac{3}{10}$$

Naïve Bayes: Step 1

#4 Posterior Probability

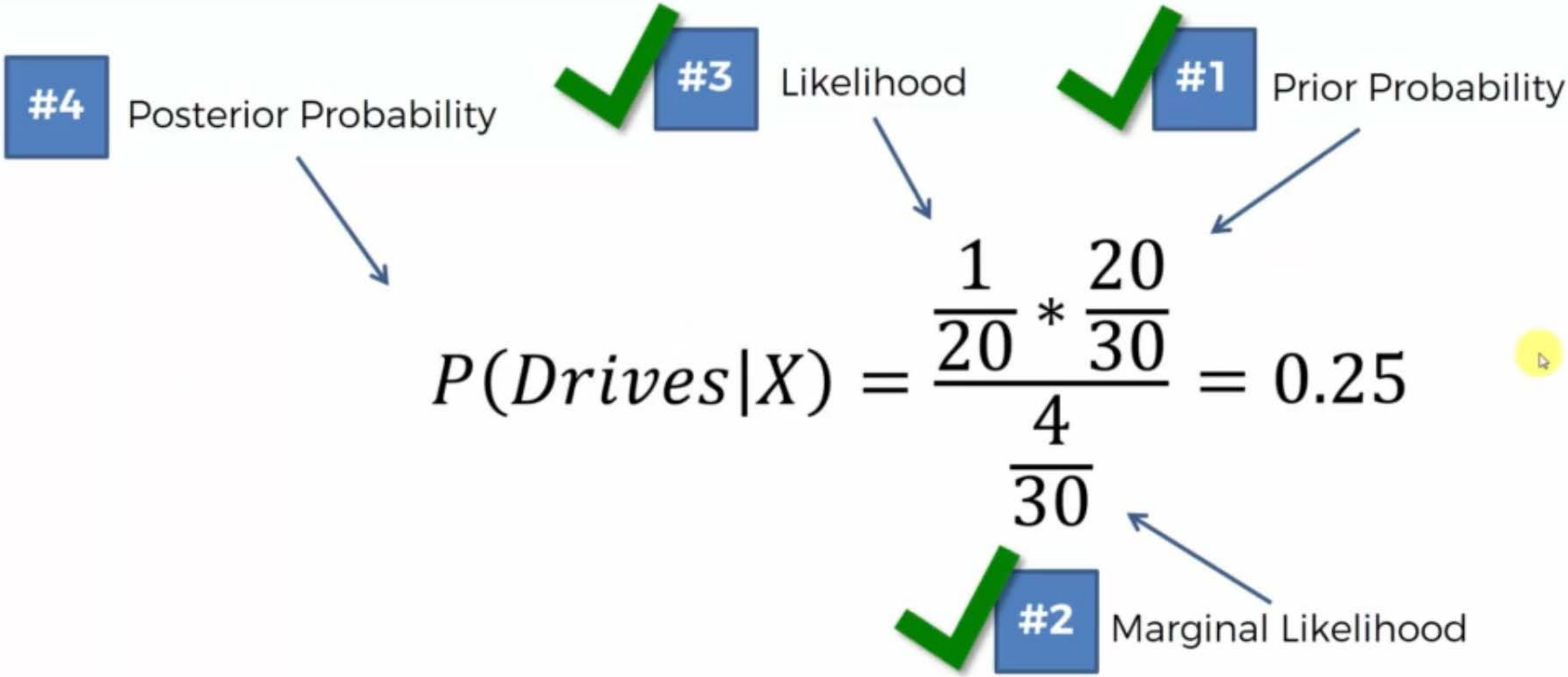
#3 Likelihood

#1 Prior Probability

$$P(Walks|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$$

#2 Marginal Likelihood

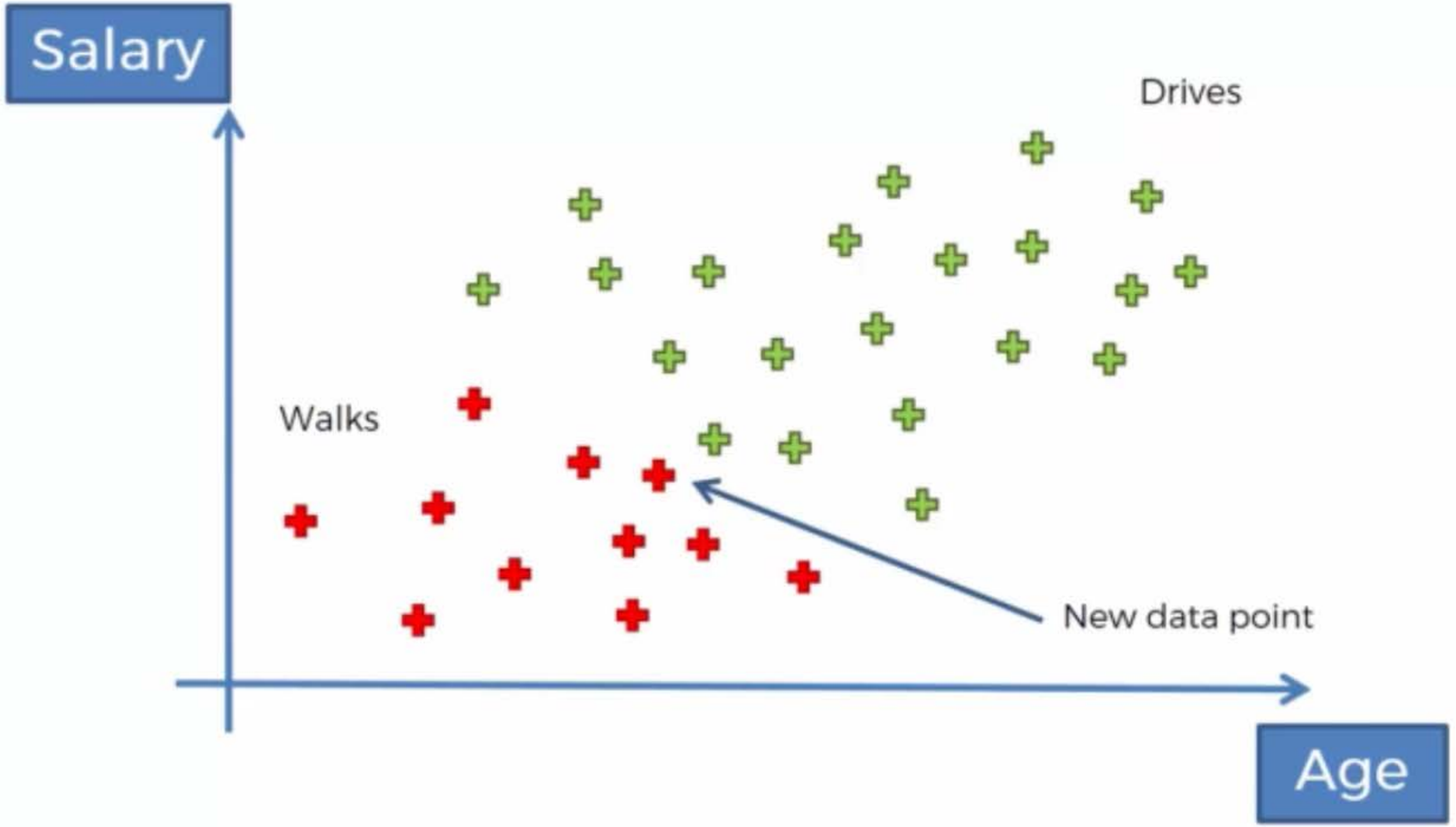
Naïve Bayes: Step 2



Step 3

$$P(Walks|X) > P(Drives|X)$$

Naïve Bayes



Naïve Bayes Classifier Additional Comments

Naïve Bayes

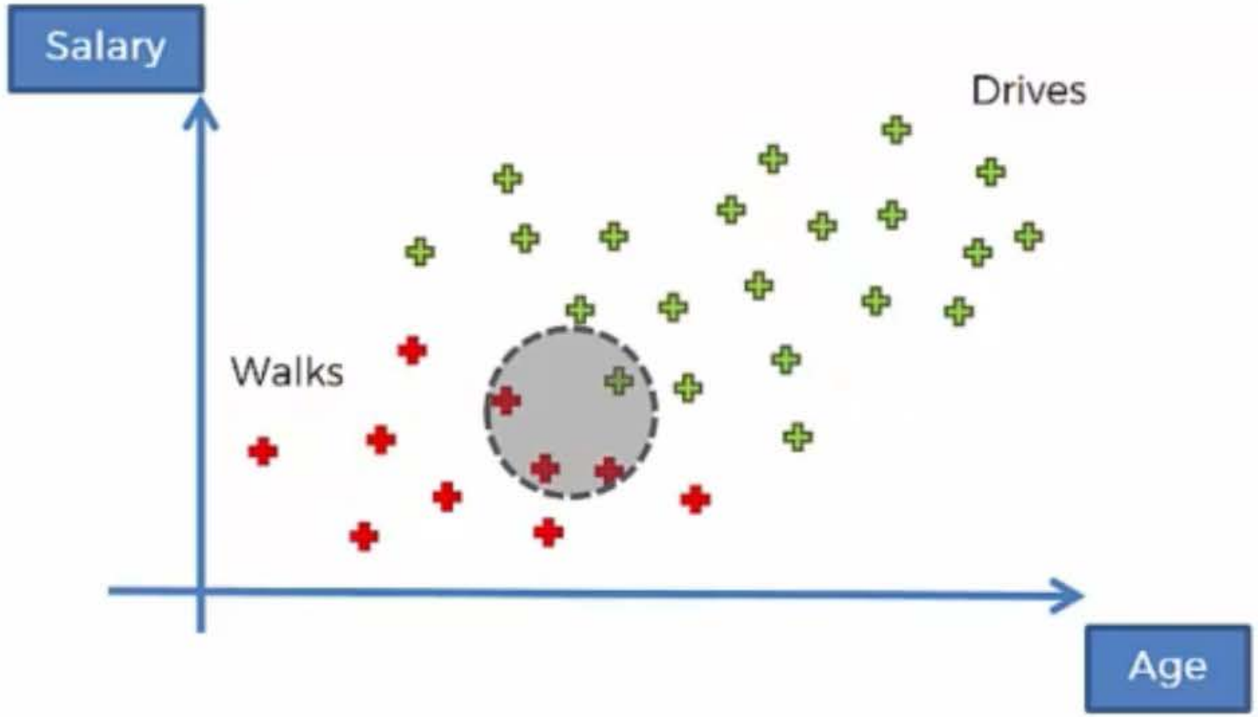
1. Q: Why "Naïve"?
2. P(X)
3. More than 2 features

Naïve Bayes

Q: Why "Naïve"?

A: Independence assumption

Naïve Bayes: Step 2



#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

NOTE: Same both times

Step 3

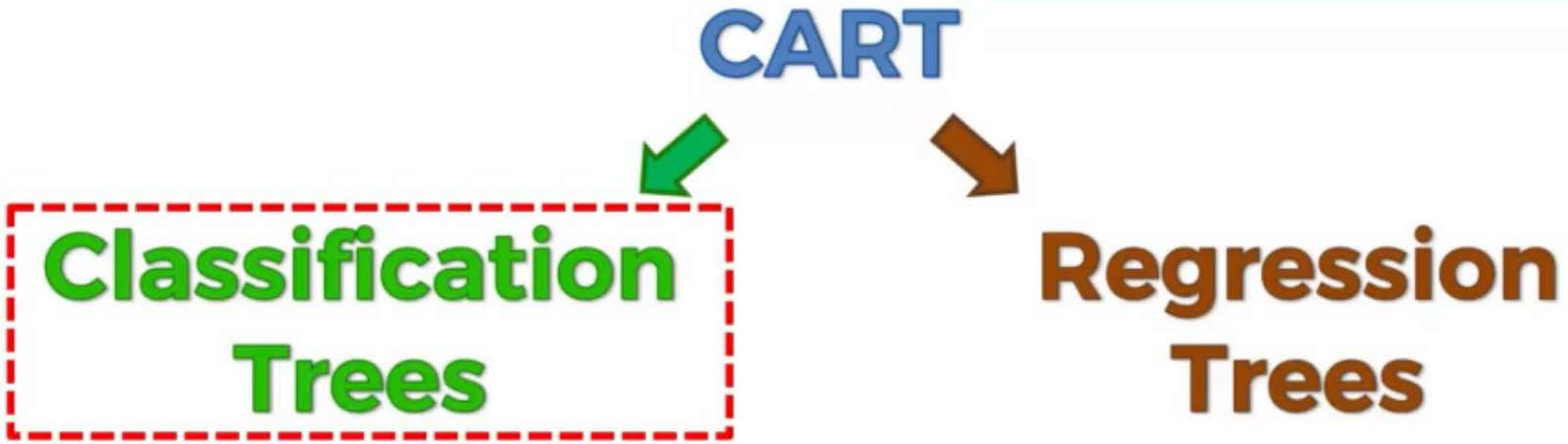
$$\frac{P(X|Walks) * P(Walks)}{\cancel{P(X)}} \quad v.s. \quad \frac{P(X|Drives) * P(Drives)}{\cancel{P(X)}}$$

Naïve Bayes

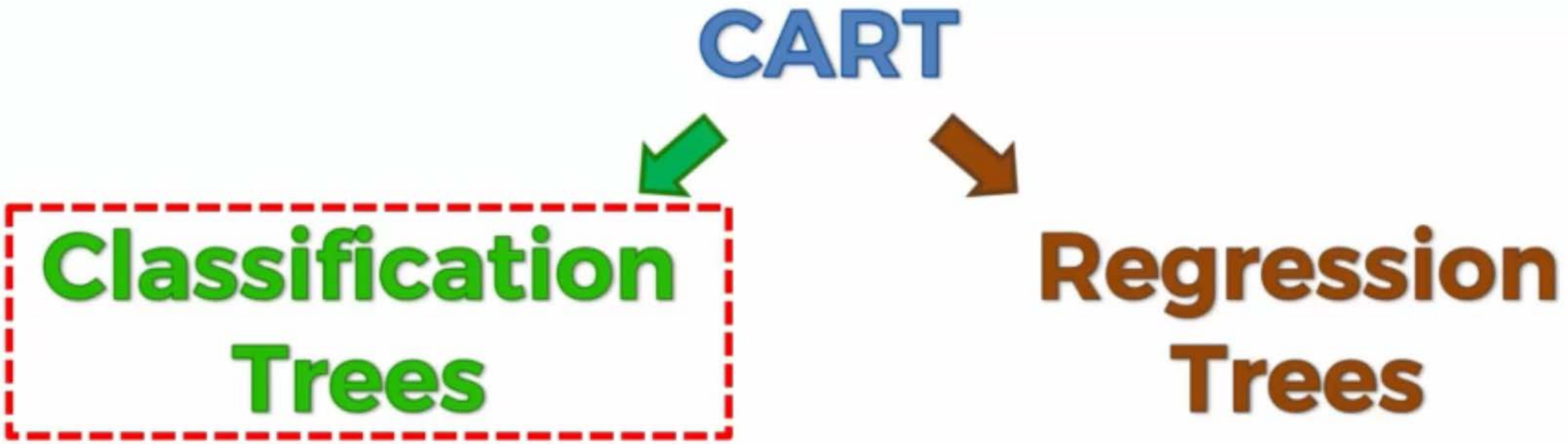
More than 2 classes

Decision Tree Intuition

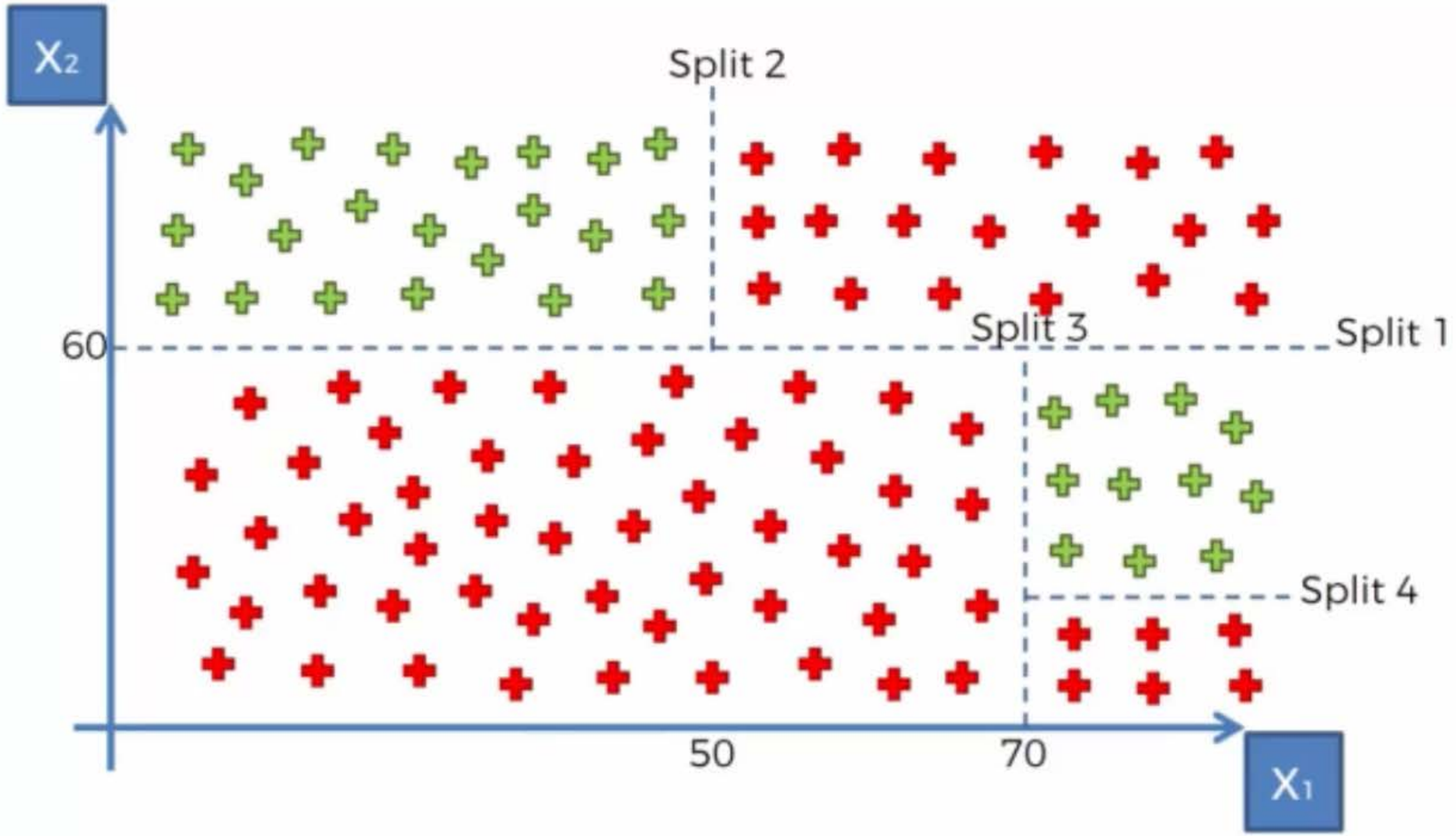
Decision Tree Intuition



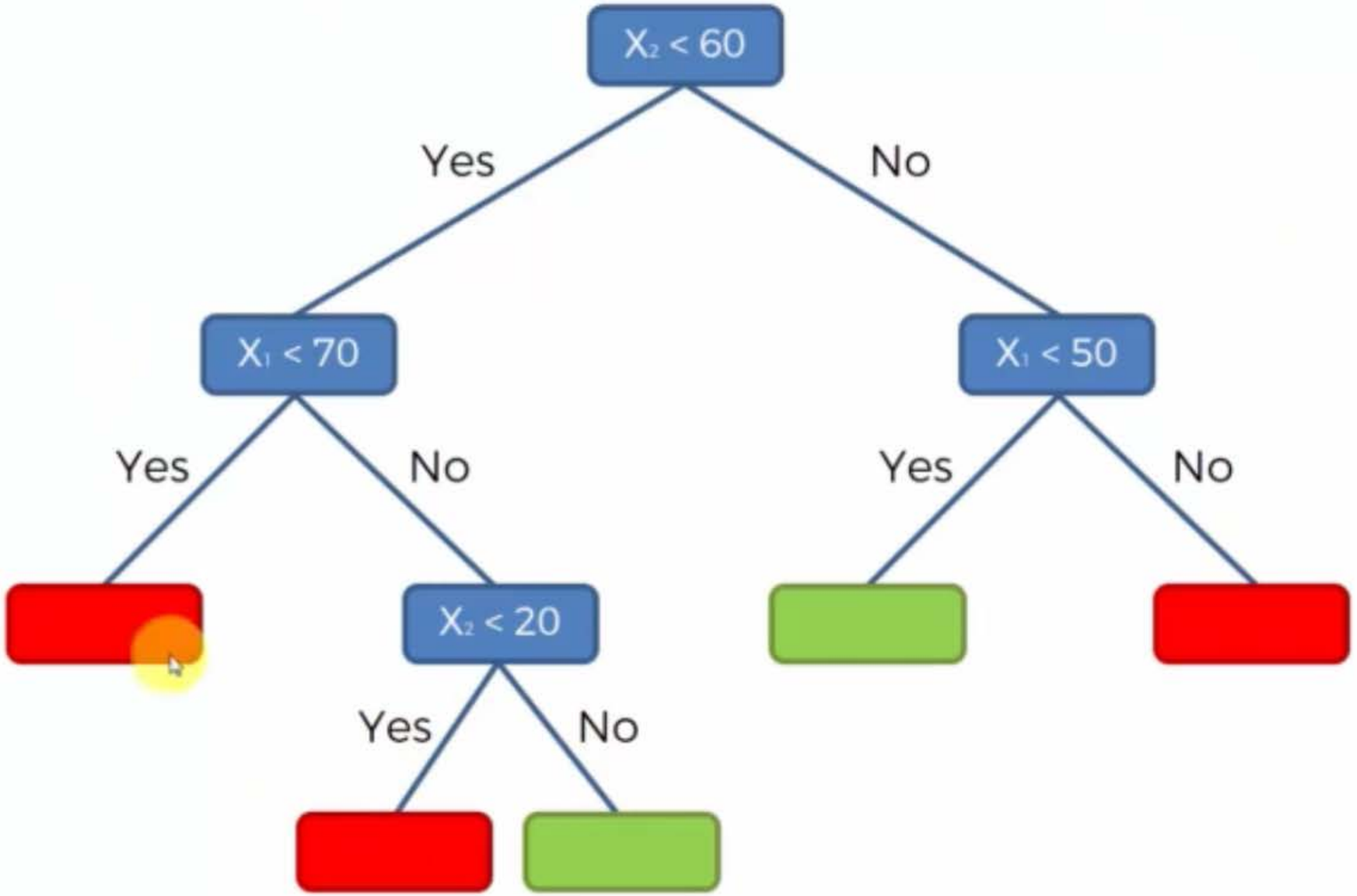
Decision Tree Intuition



Decision Tree Intuition



Split 4



Decision Trees

- Old Method
- Reborn with upgrades
- Random Forest
- Gradient Boosting
- etc.

Random Forest Intuition

Random Forest Intuition

Ensemble Learning

Random Forest Intuition

STEP 1: Pick at random K data points from the Training set.

STEP 2: Build the Decision Tree associated to these K data points.

STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

Random Forest Intuition

Random Forest Intuition

Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

Abstract

We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. Our large and highly varied training dataset allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes.

The system runs at 200 frames per second on consumer hardware. Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton nearest neighbor matching.

1. Introduction

Robust interactive human body tracking has applications including gaming, human-computer interaction, security, telepresence, and even health-care. The task has recently been greatly simplified by the introduction of real-time depth cameras [16, 19, 44, 37, 28, 13]. However, even

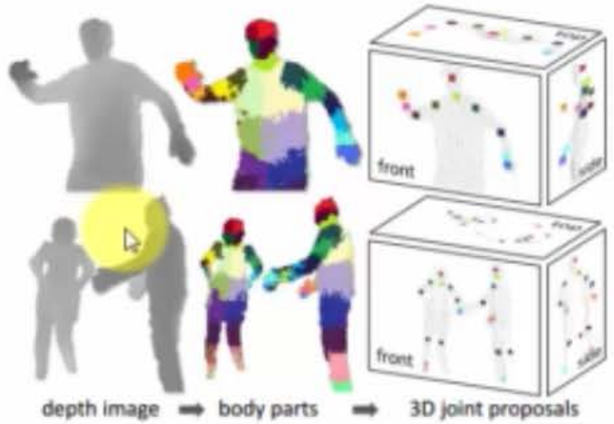


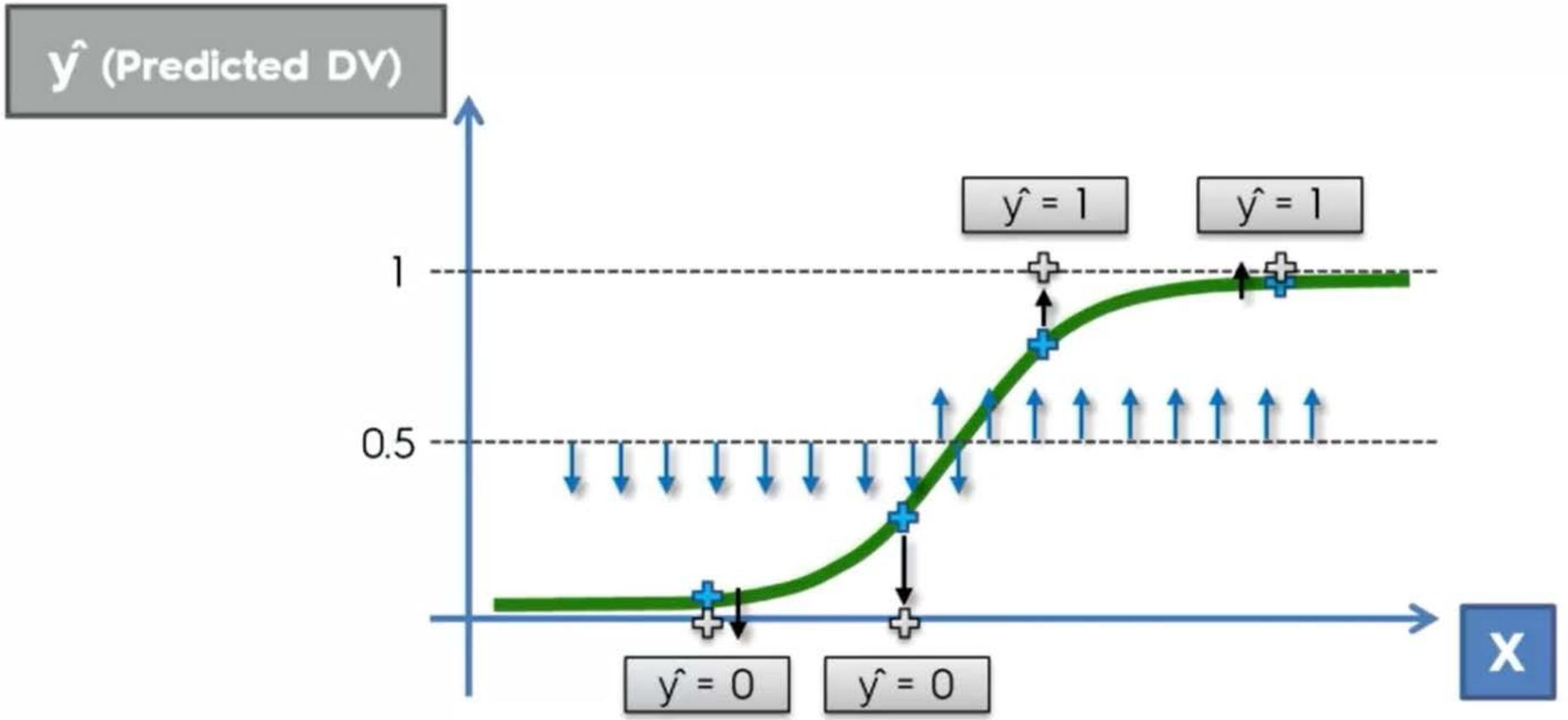
Figure 1. **Overview.** From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.

joints of interest. Reprojecting the inferred parts into world space, we localize spatial modes of each part distribution and thus generate (possibly several) confidence-weighted proposals for the 3D locations of each skeletal joint.

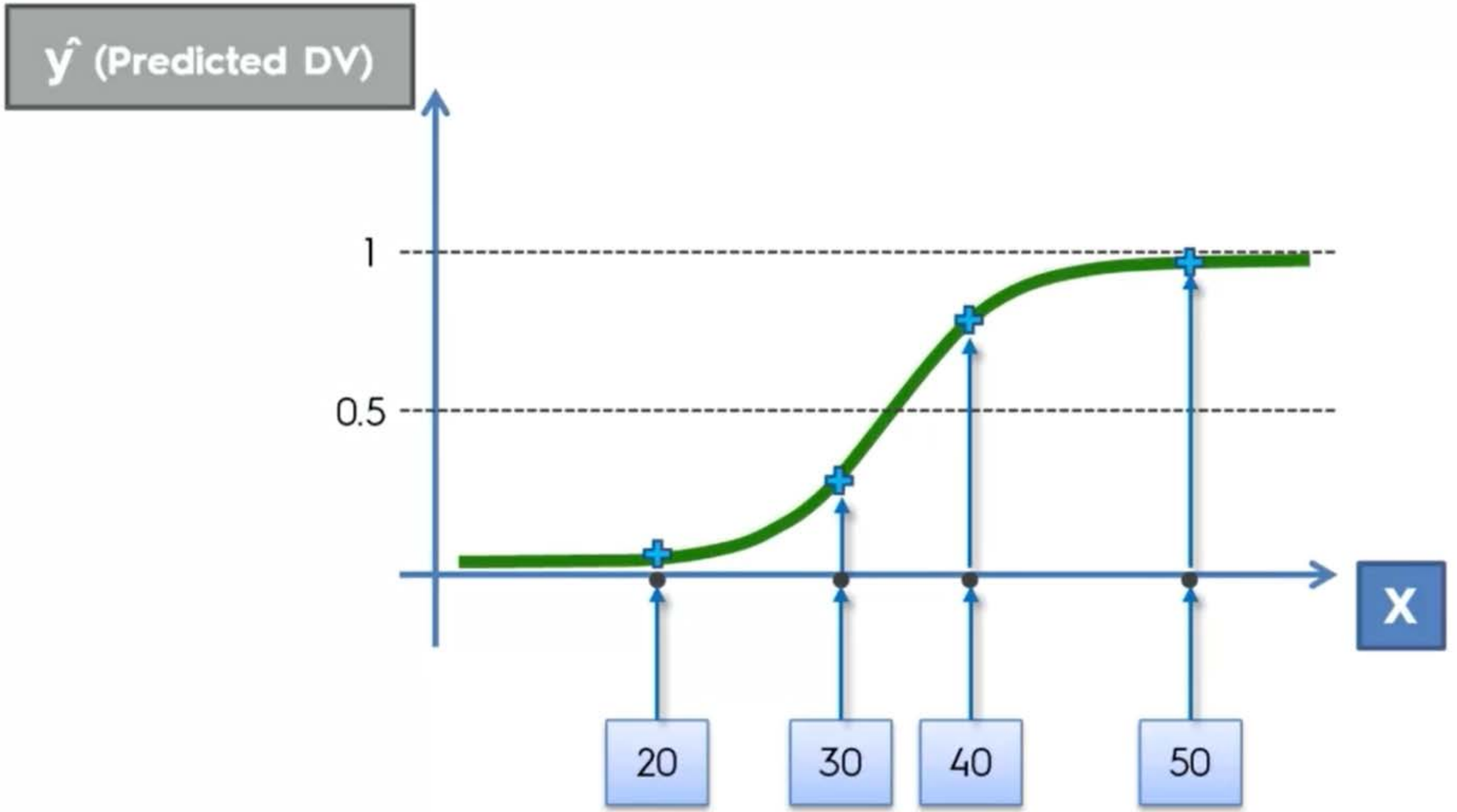
We treat the segmentation into body parts as a per-pixel classification task (no pairwise terms or CRF have proved necessary). Evaluating each pixel separately avoids a com-

False Positives & False Negatives

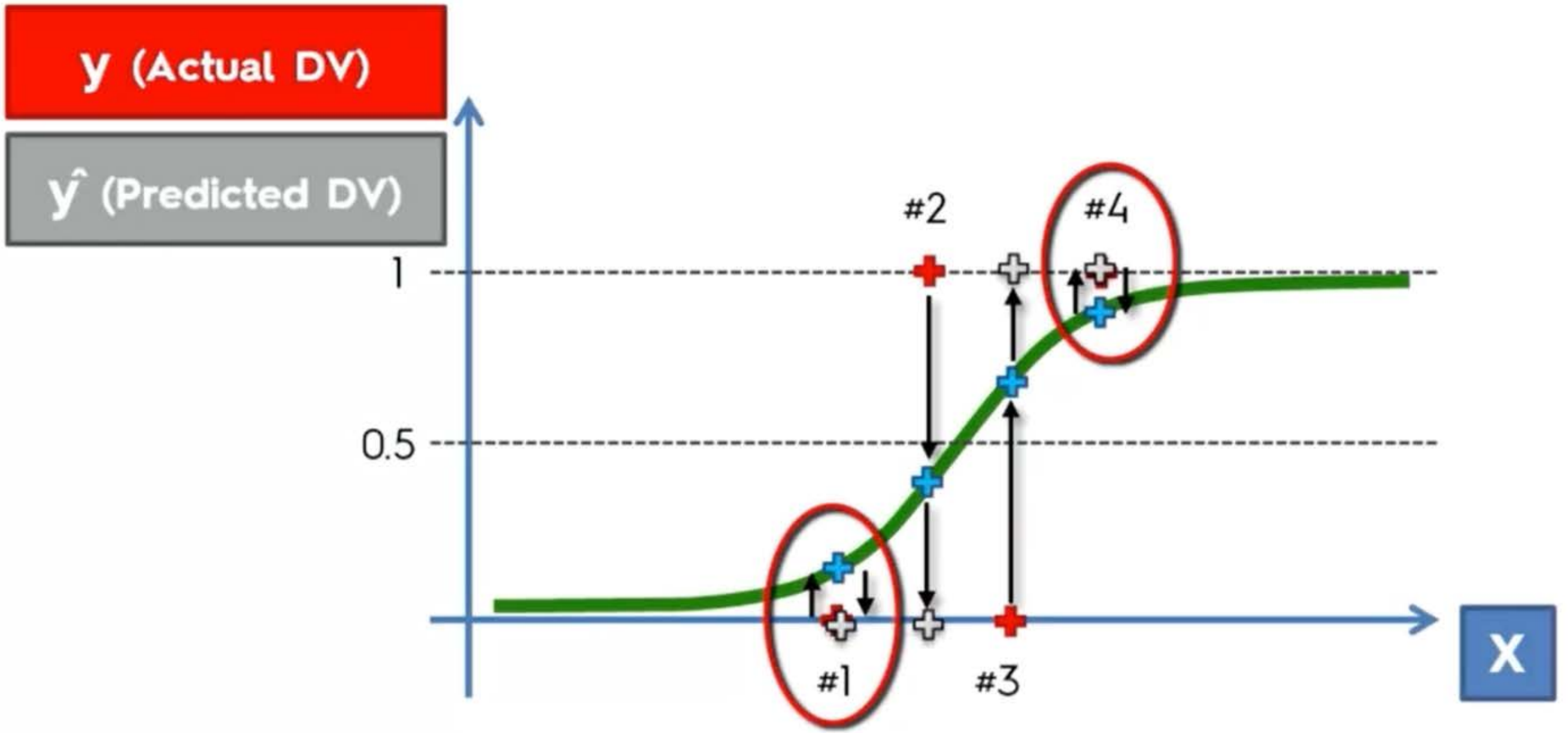
False Positives & Negatives



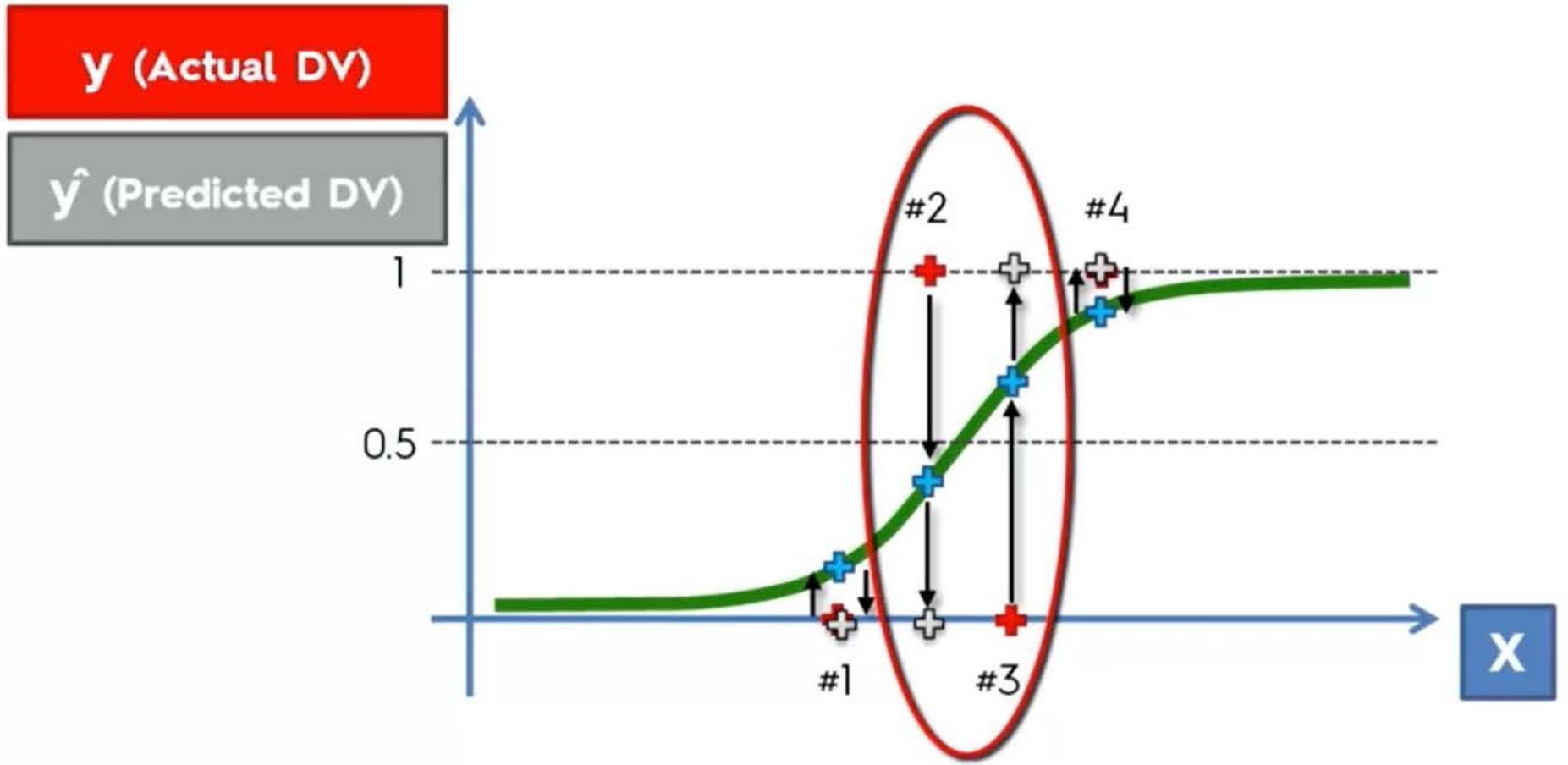
False Positives & Negatives



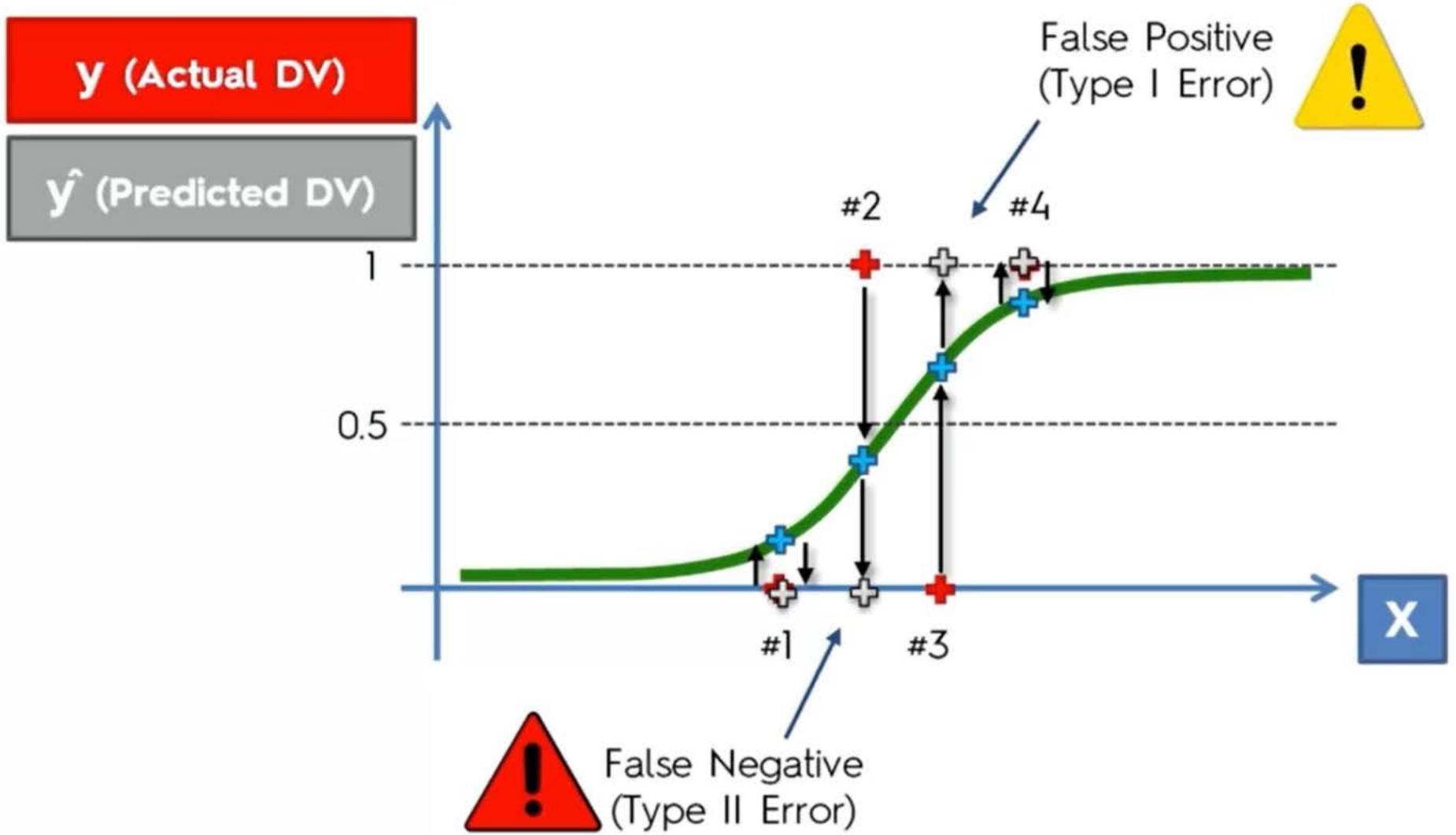
False Positives & Negatives



False Positives & Negatives

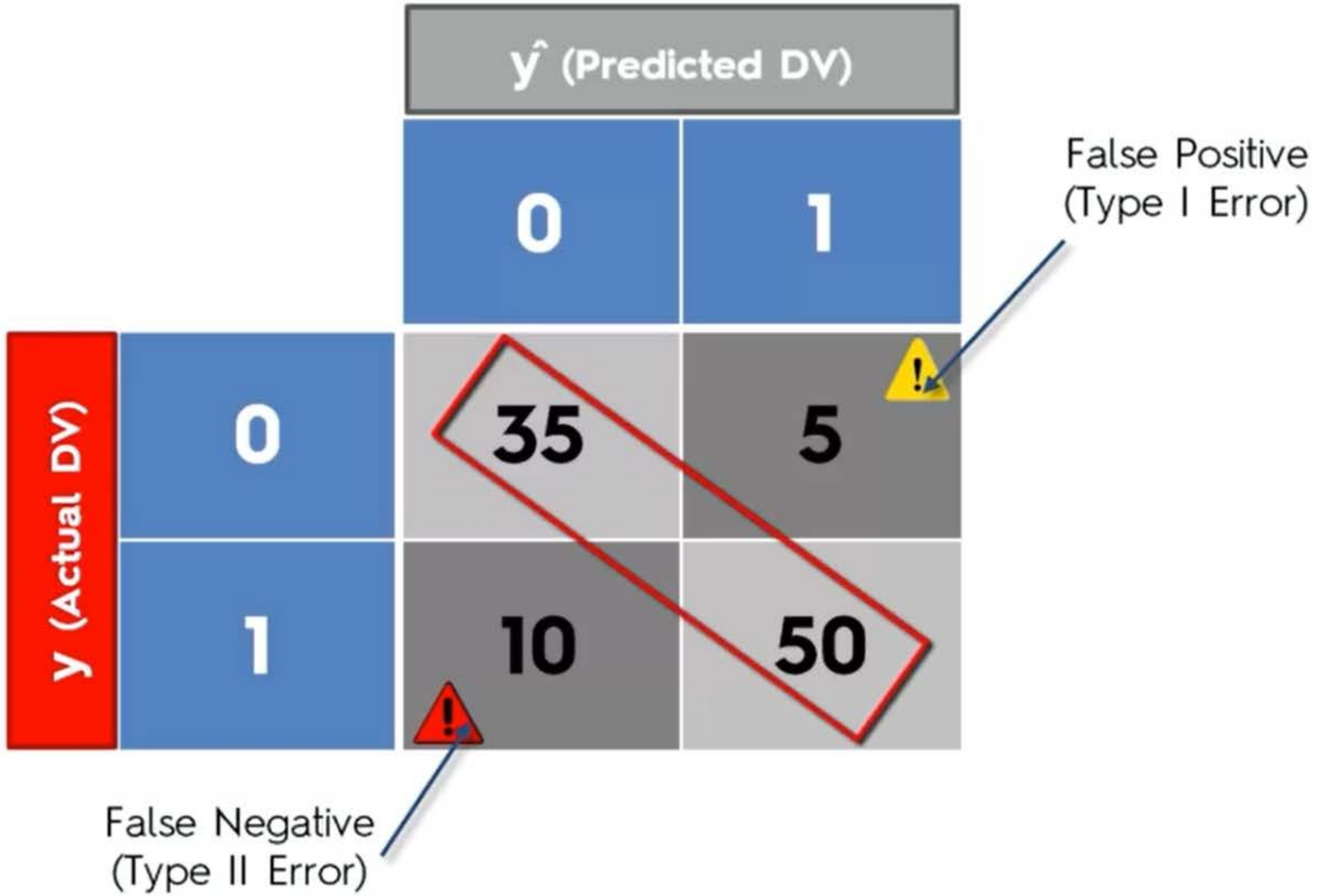


False Positives & Negatives

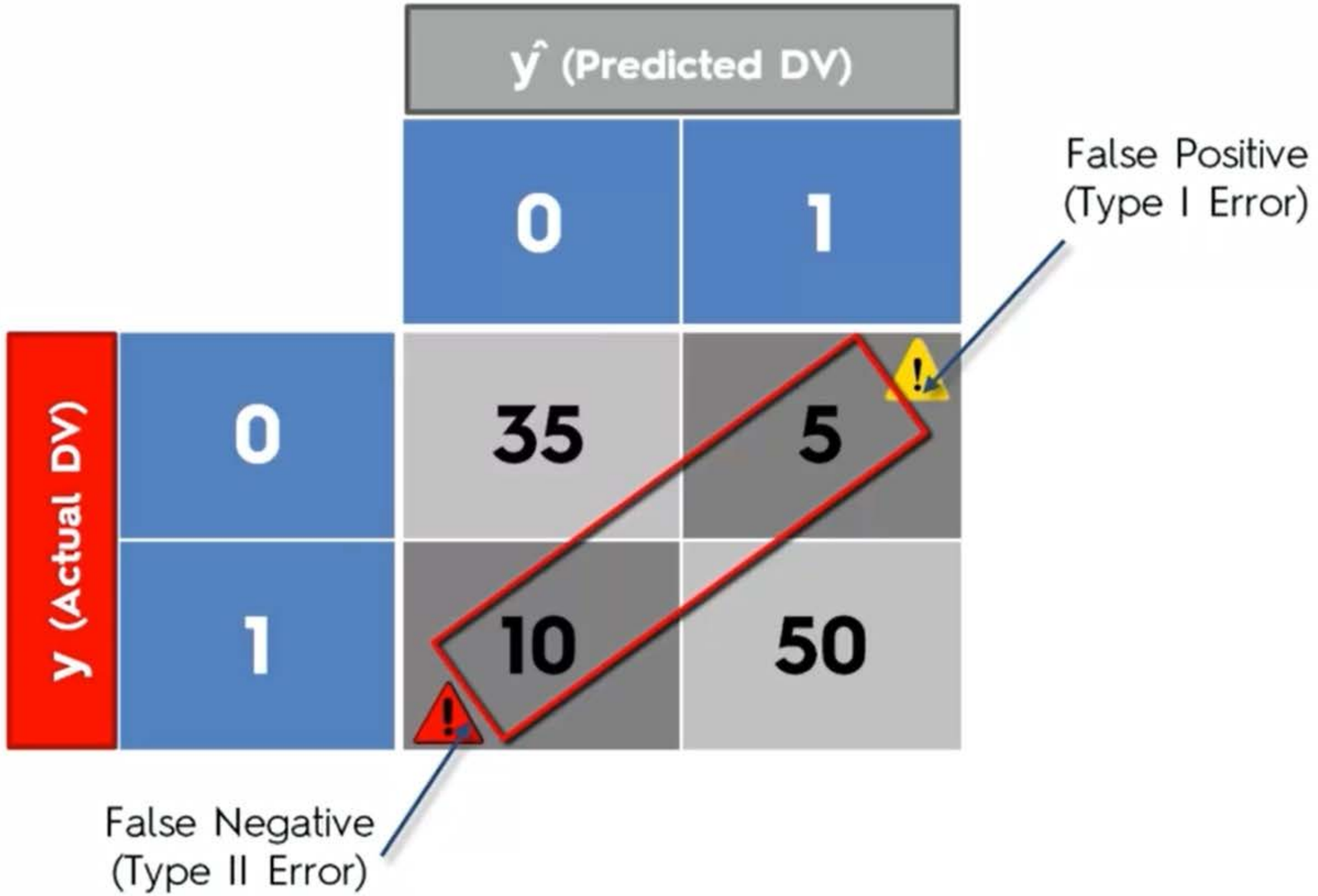


Confusion Matrix

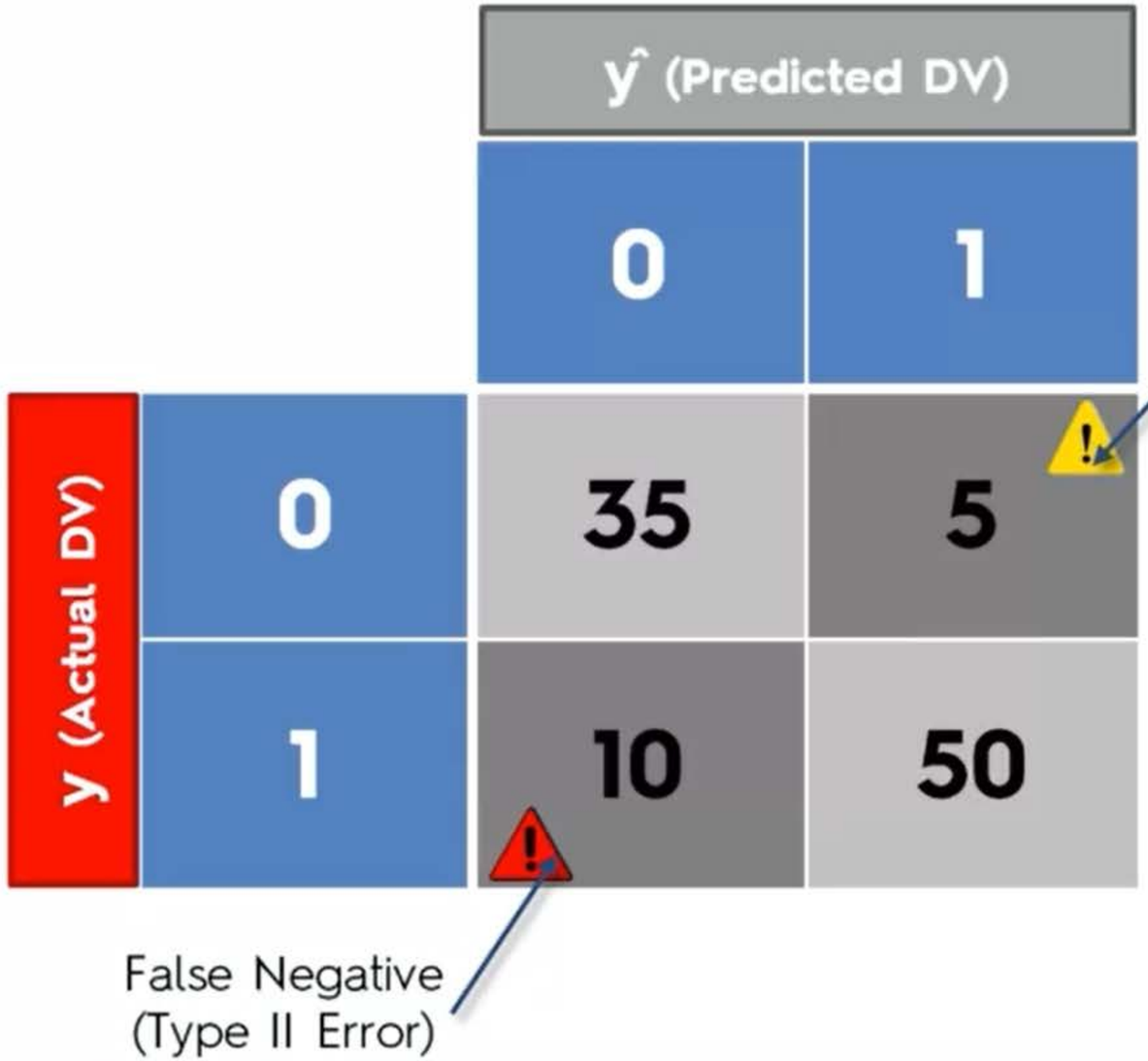
Confusion Matrix



Confusion Matrix



Confusion Matrix



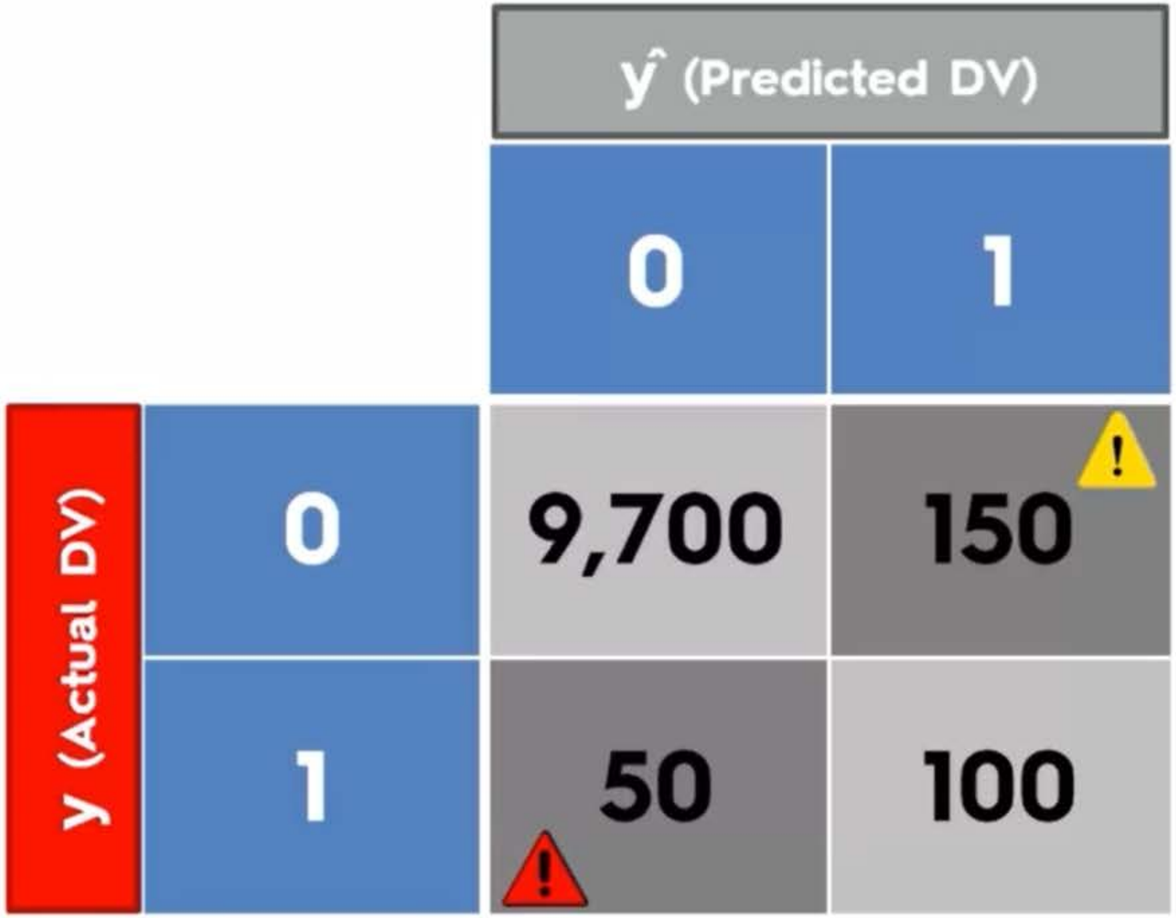
Calculate two rates

1. Accuracy Rate = Correct / Total
AR = 85/100 = 85%

2. Error Rate = Wrong / Total
ER = 15/100 = 15%

Accuracy Paradox

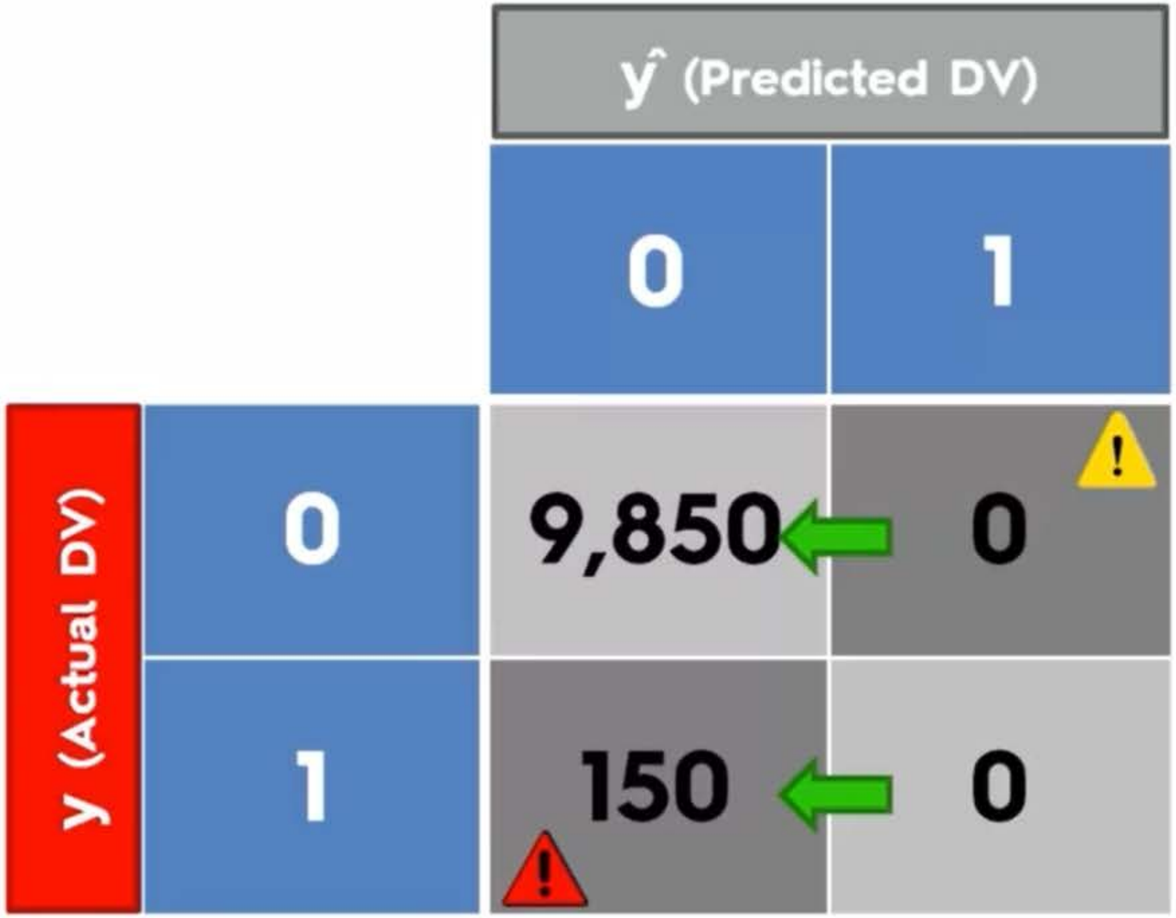
Accuracy Paradox



Scenario 1:

Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

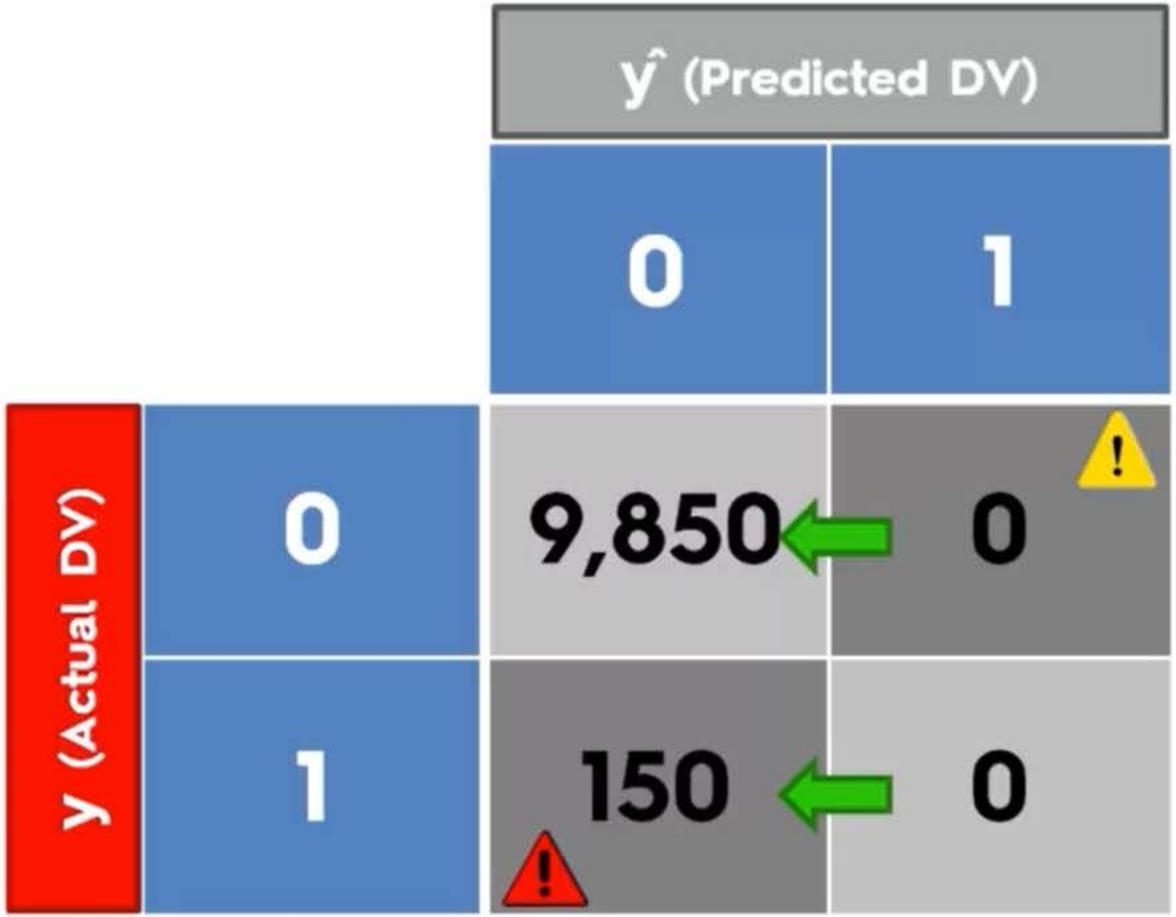
Accuracy Paradox



Scenario 1:

Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

Accuracy Paradox



Scenario 1:

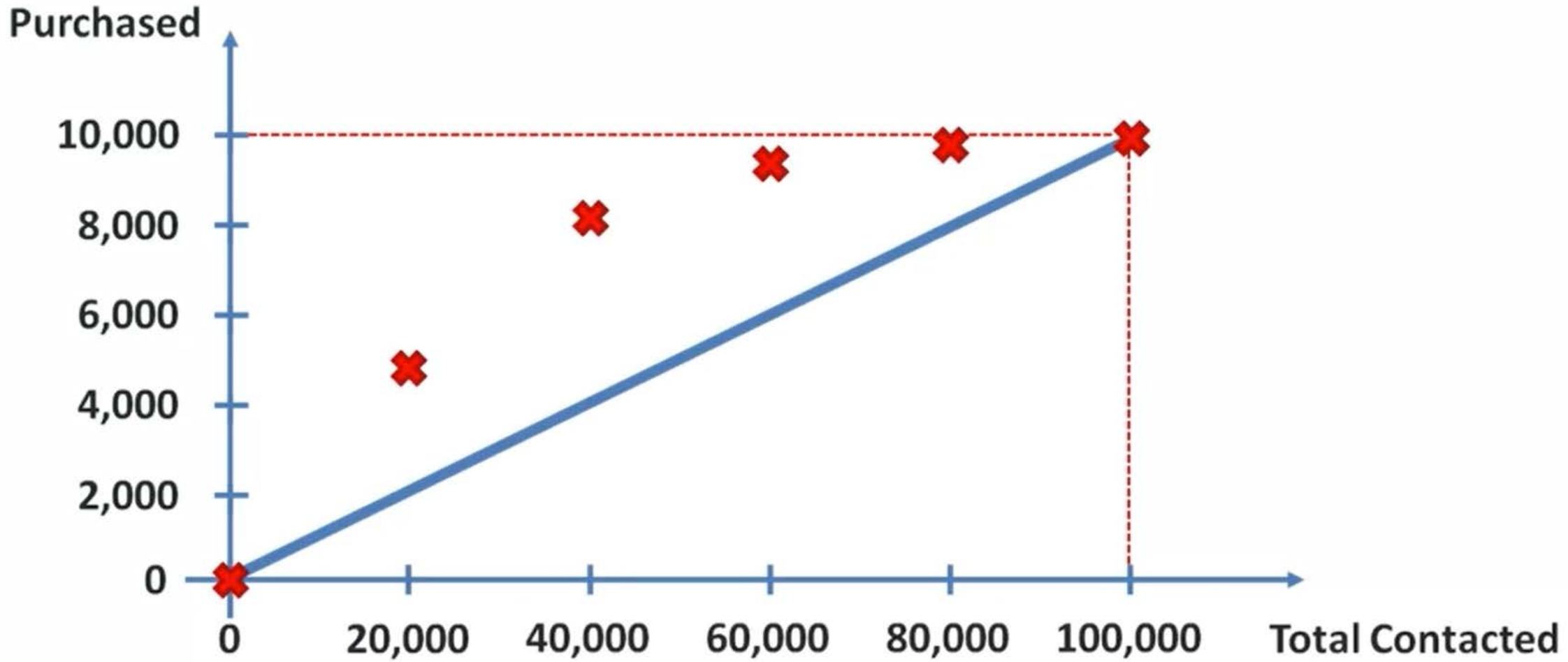
Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

Scenario 2:

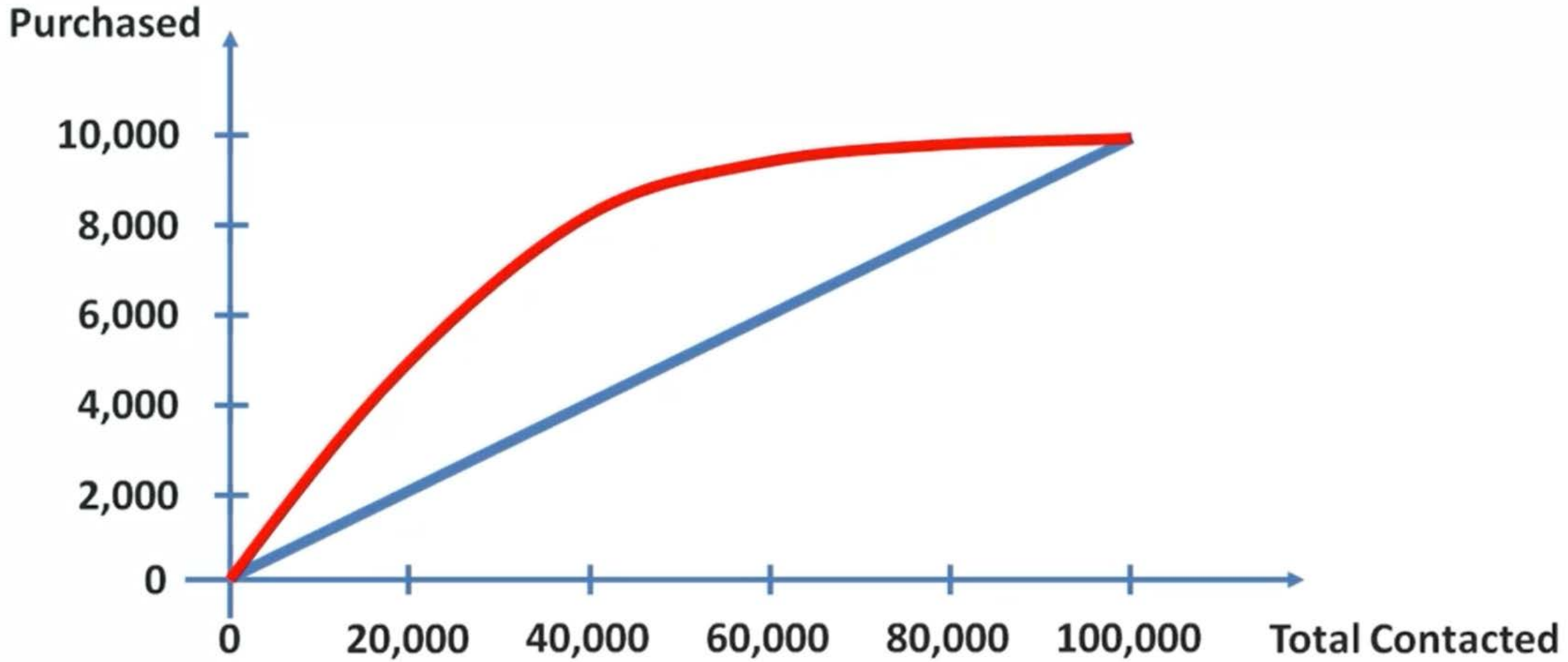
Accuracy Rate = Correct / Total
AR = 9,850/10,000 = 98.5%

CAP Curve

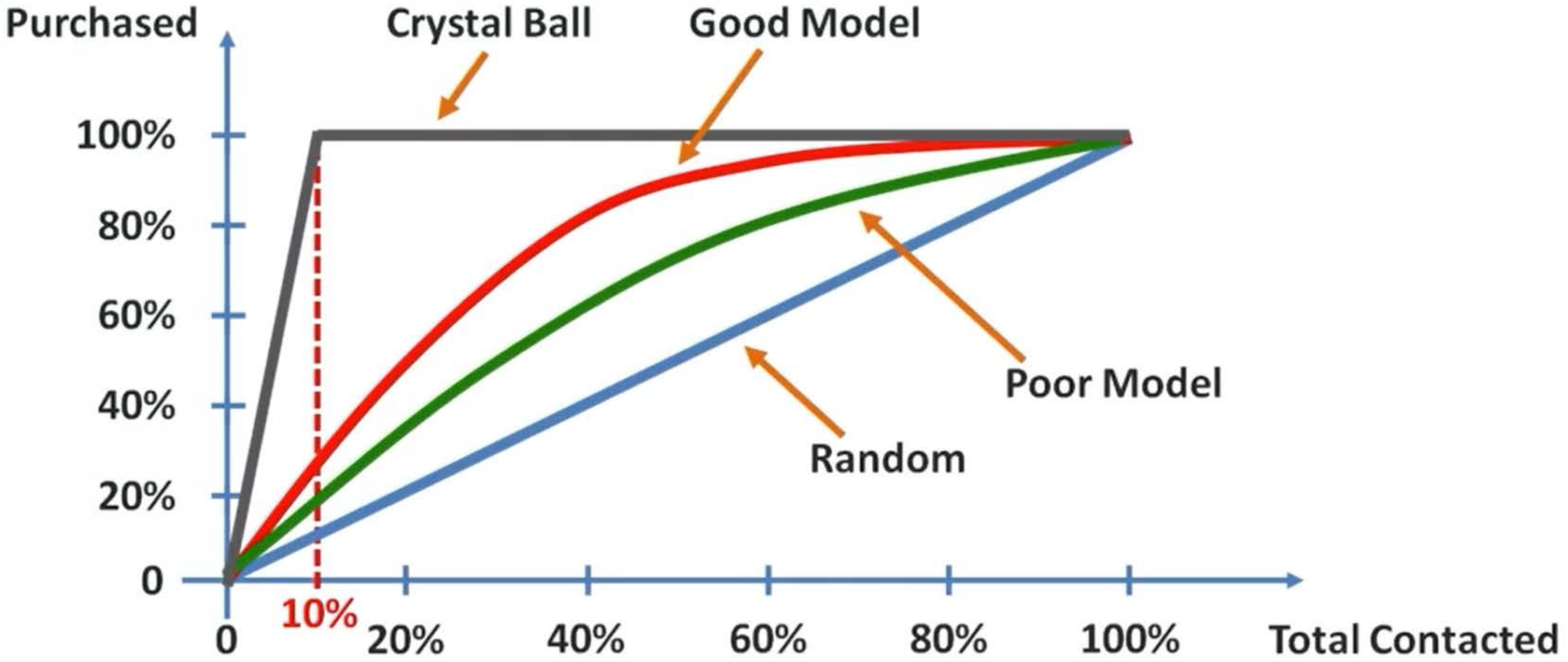
CAP



CAP



CAP



CAP

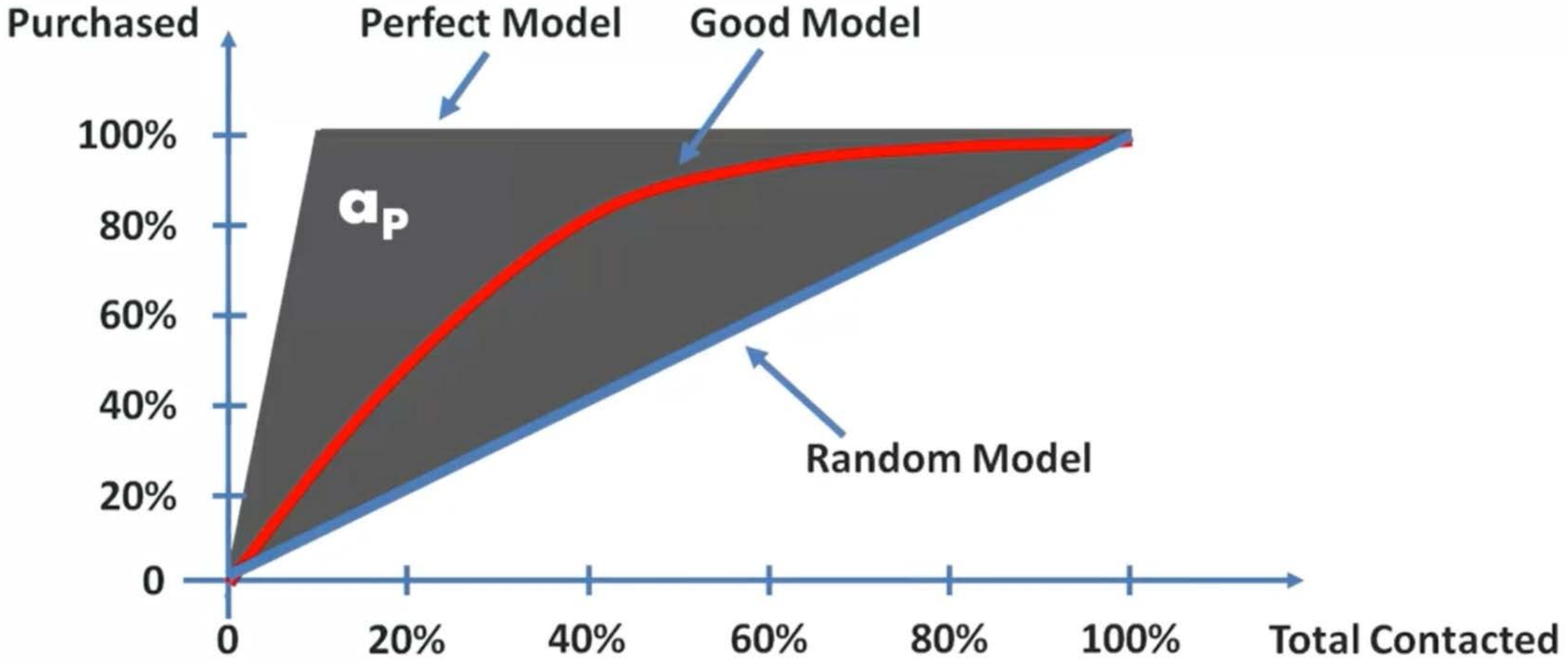
Note:

CAP = Cumulative Accuracy Profile

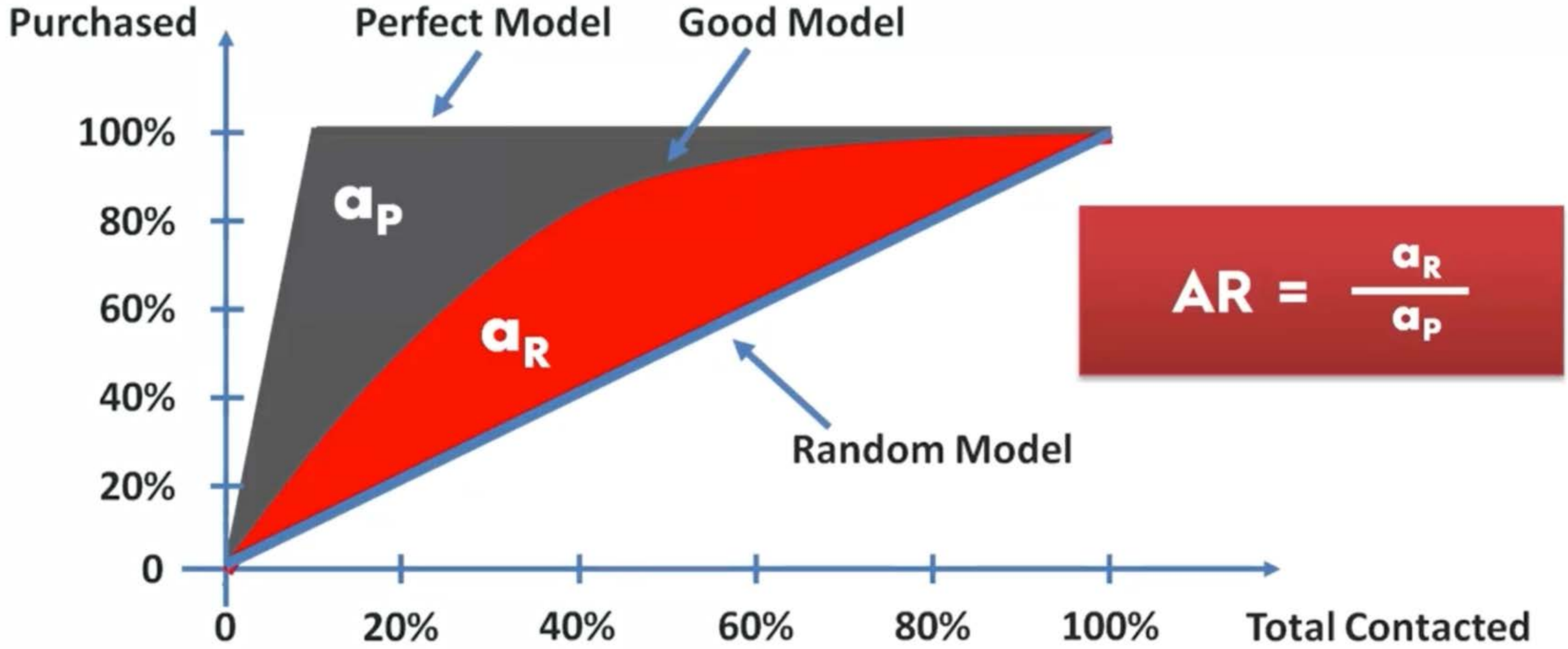
ROC = Receiver Operating Characteristic

CAP Curve Analysis

CAP Analysis



CAP Analysis



$$AR = \frac{a_R}{a_P}$$

CAP Analysis