Machine Learning A-Z Course Slides

Welcome to the course!

Dear student,

Welcome to the *"Machine Learning A-Z"* course brought to you by SuperDataScience. We are super-excited to have you on board! In this class you will learn many interesting and useful concepts while having lots of fun.

These slides may be updated from time-to-time. If this happens, you will be able to find them in the course materials repository with a new version indicated in the filename.

We kindly ask that you use these slides **only for the purpose of supporting your own learning** journey and we look forward to seeing you inside the class!

Enjoy machine learning,
Kirill & Hadelin

PS: if you are not yet enrolled in the course, you can find it here.

Who Are Your Instructors?

- Hello! My name is **Kirill Eremenko**
- I have a bachelor's degree in Physics & Maths and a background in Data Analytics consulting
- I used to host the SuperDataScience podcast

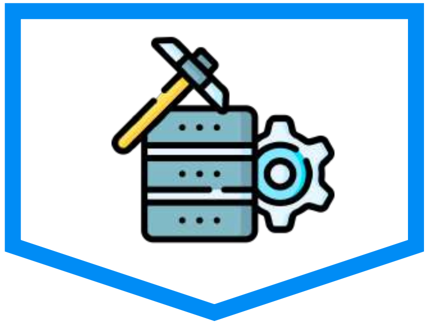
- Hi there! My name is **Hadelin de Ponteves**
- I have a master's in Machine Learning and I used to do Reinforcement Learning at Google
- I wrote a research paper on Machine Learning

We've been teaching online together since 2016 and over 1 Million students have enrolled in our Machine Learning and Data Science courses. You can be confident that you are in good hands!

Data Preprocessing

The Machine Learning Process

The Machine Learning Process



Data Pre-Processing

- Import the data
- Clean the data
- Split into training & test sets
- Feature Scaling

Modelling

- Build the model
- Train the model
- Make predictions

Evaluation

- Calculate performance metrics
- Make a verdict

Training Set & Test Set

Training Set & Test Set

$\sim$

Train
80%

$$\hat{y} = b_0 + b_1 X_1 + b_2 X_2$$

Test
20%

V.S.

Predicted values $\hat{y}$

Actual values y

Feature Scaling

Feature Scaling

X1	X2	X3	X4
$ 179.43	56.784	34.6181	3.55
$ 641.87	62.054	47.7306	1.692
$ 556.30	64.13	55.596	1.559
$ 578.47	63.377	52.7121	1.679
$ 591.16	61.553	46.1315	1.984
$ 242.03	58.29	39.2952	2.942
$ 364.66	59.93	42.4628	2.494
$ 190.68	57.271	36.2725	3.419
$ 547.23	63.763	54.1971	1.634
$ 359.69	59.375	41.5105	2.128
$ 438.08	60.484	43.493	2.47
$ 637.17	62.525	49.428	1.725

Feature Scaling

Normalization

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

[0 ; 1]

Standardization

$$X' = \frac{X - \mu}{\sigma}$$

[-3 ; +3]

Feature Scaling

	70,000 $	45 yrs
	10,000	1
	60,000 $	44 yrs
	8,000	4
	52,000 $	40 yrs

Feature Scaling

Normalization

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

[0 ; 1]

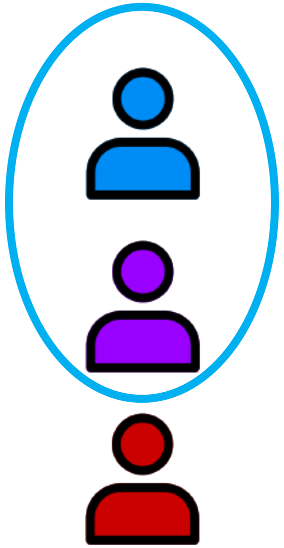
Feature Scaling

	
70,000 $	45 yrs
60,000 $	44 yrs
52,000 $	40 yrs

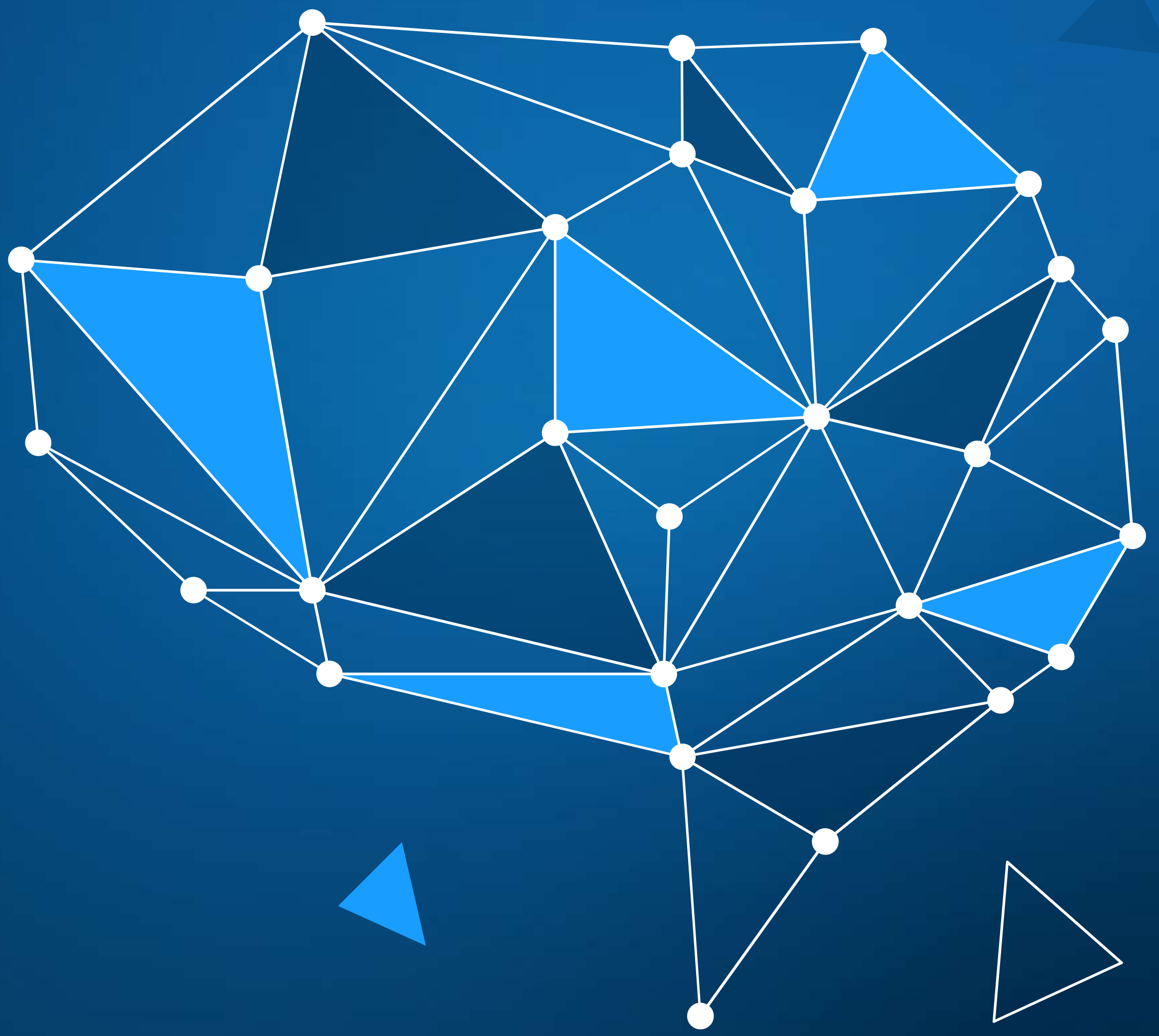
Feature Scaling

		
	1	45 yrs
	0.444	44 yrs
	0	40 yrs

Feature Scaling

		
	1	1
	0.444	0.75
	0	0

Regression



Simple Linear Regression

Simple Linear Regression

$$\underbrace{\hat{y}}_{\text{Dependent variable}} = \underbrace{b_0}_{\text{y-intercept (constant)}} + \underbrace{b_1}_{\text{Slope coefficient}} \underbrace{X_1}_{\text{Independent variable}}$$

Simple Linear Regression

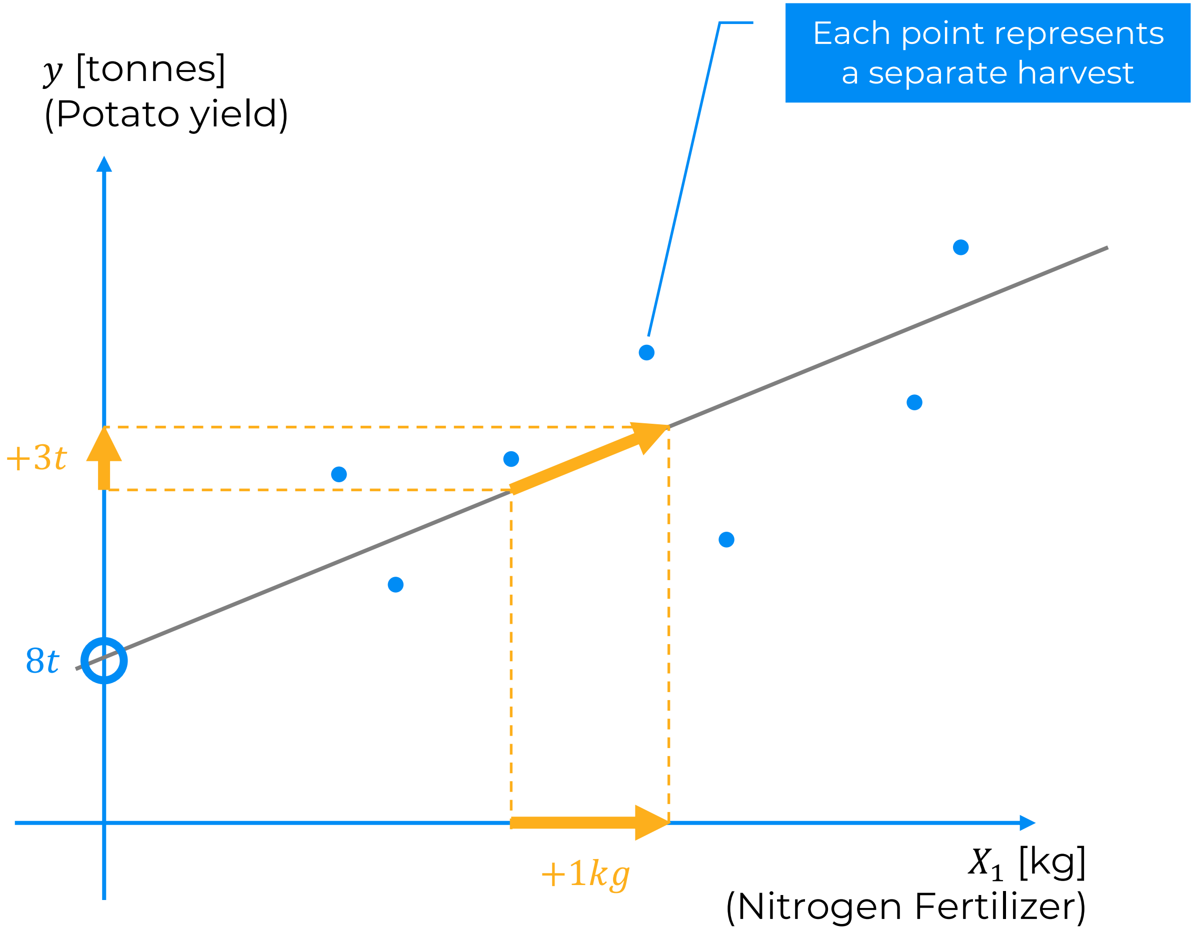
$\sim$

$$\hat{y} = b_0 + b_1 X_1$$

$$Potatoes[t] = b_0 + b_1 \times Fertilizer[kg]$$

$$b_0 = 8[t]$$

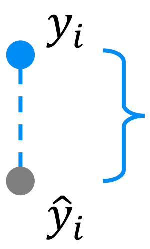
$$b_1 = 3[\frac{t}{kg}]$$



Ordinary Least Squares

Simple Linear Regression

Ordinary Least Squares:

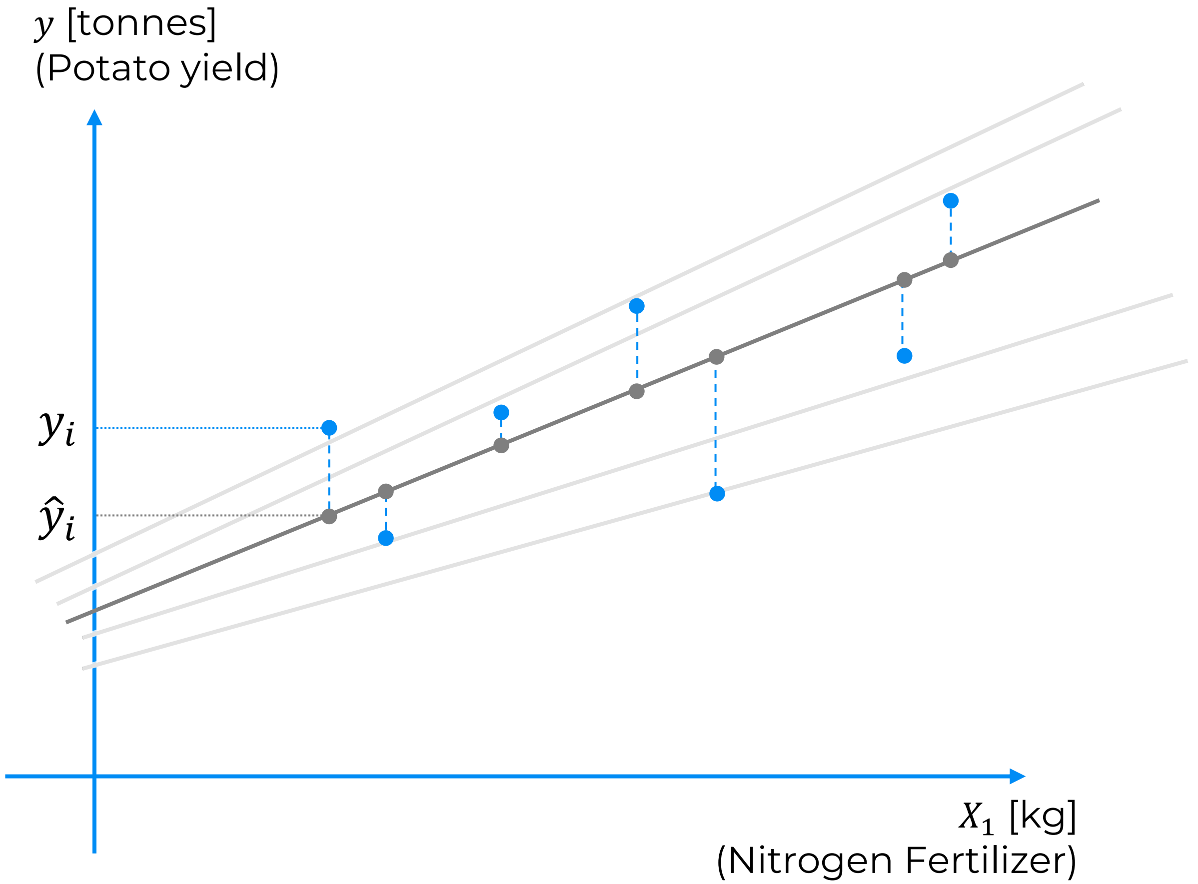


$$\textit{residual}: \varepsilon_i = y_i - \hat{y}_i$$

$$\hat{y} = b_0 + b_1 X_1$$

b_0, b_1 such that:

$SUM(y_i - \hat{y}_i)^2$ is minimized



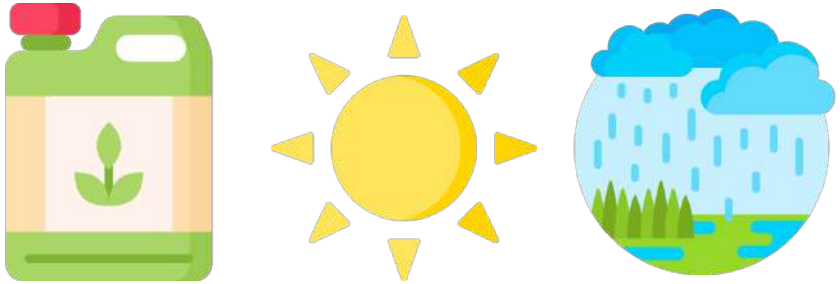
Multiple Linear Regression

Multiple Linear Regression

$$\underbrace{\hat{y}}_{\text{Dependent variable}} = \underbrace{b_0}_{\substack{\text{y-intercept}\\\text{(constant)}}} + \underbrace{b_1}_{\text{Slope coefficient 1}}\underbrace{X_1}_{\text{Independent variable 1}} + \underbrace{b_2}_{\text{Slope coefficient 2}}\underbrace{X_2}_{\text{Independent variable 2}} + \cdots + \underbrace{b_n}_{\text{Slope coefficient n}}\underbrace{X_n}_{\text{Independent variable n}}$$

Multiple Linear Regression

$\sim$



$$Potatoes[t] = 8t + 3\frac{t}{kg} \times Fertilizer[kg] - 0.54\frac{t}{°C} \times AvgTemp[°C] + 0.04\frac{t}{mm} \times Rain[mm]$$

Additional Reading

The Application of Multiple Linear Regression and Artificial Neural Network Models for Yield Prediction of Very Early Potato Cultivars before Harvest

Magdalena Piekutowska et. al. (2021)

Link:

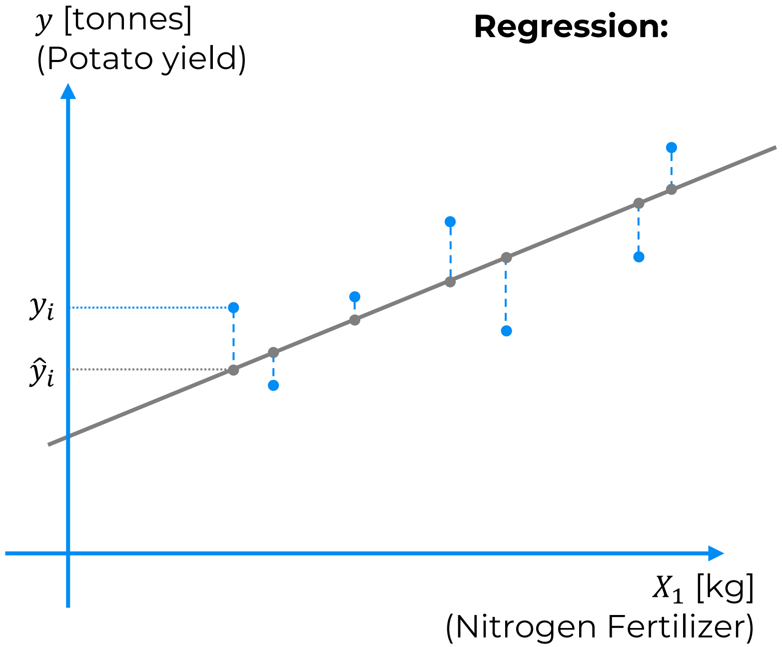
https://www.mdpi.com/2073-4395/11/5/885

Quantitative Yield Forecast		
Models RY1 and NY1	Yield Forecast before Harvest (40 Days from Full Emergence)	Data Range
INSO	insolation sum [h] in the periods: planting—June 20,	275.3–711.7
TEMP	average daily air temperature [°C] in the periods: planting—20 June	10.8–15.7
PREC	precipitation [mm] in the periods: planting—20 June	38.7–258.2
NITRO	sum of nitrogen fertilization [$kg \cdot ha^{-1}$] in the periods: planting—20 June	80–155
PHOSP	sum of phosphorus fertilization [$kg \cdot ha^{-1}$]	28.2–150
POTAS	sum of potassium fertilization [$kg \cdot ha^{-1}$]	80–306.5
PLANT	planting date [number of days since the beginning of the year]	107–127
EMERG	date of emergence [number of days since the beginning of the year], yield forecast 20th of June	130–151
DENST	densification [plants/plot], yield forecast June 20	35–60
PH	Soil pH [in 1 mol KCl]	5.8–7
SFERTP	soil fertility in phosphorus [mg $P_2O_5 \cdot 100\ g^{-1}$ soil]	14–26.2
SFERTK	soil fertility in potassium [mg $K_2O \cdot 100\ g^{-1}$ soil]	11.7–19.2
SFERTM	soil fertility in magnesium [mg $Mg \cdot 100\ g^{-1}$ soil]	3–9.1
YIELDP1	tuber yield [$t \cdot ha^{-1}$], harvest 40 days from full emergence	11.6–41.3

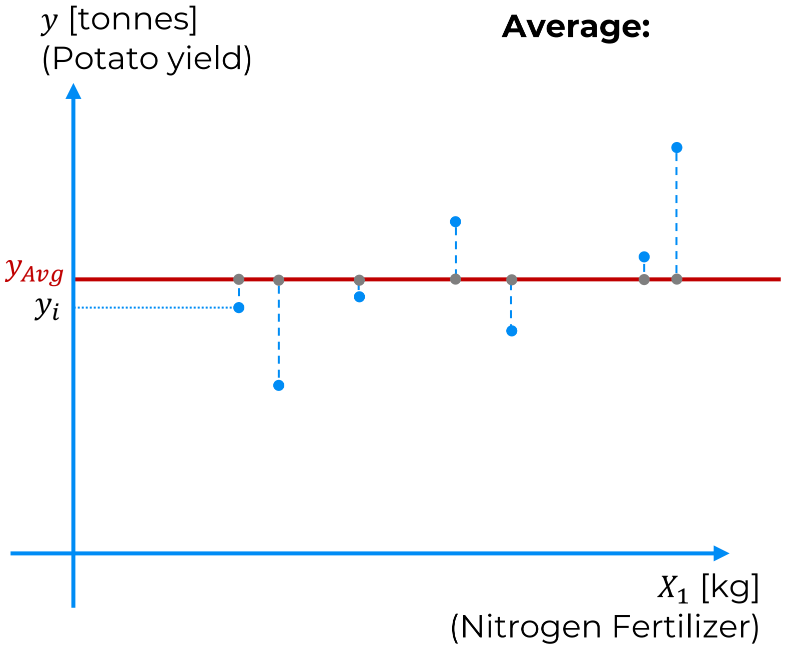
R Squared

R Squared

Regression:



Average:



$$SS_{res} = SUM(y_i - \hat{y}_i)^2$$

$$SS_{tot} = SUM(y_i - y_{avg})^2$$

$$\boxed{R^2 = 1 - \frac{SS_{res}}{SS_{tot}}}$$

Rule of thumb (for our tutorials)*:

1.0 = Perfect fit (suspicious)
~0.9 = Very good
<0.7 = Not great
<0.4 = Terrible
<0 = Model makes no sense for this data

This is highly dependent on the context

Adjusted R Squared

Adjusted R Squared

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

R^2 – Goodness of fit
(greater is better)

Problem:

$\hat{y} = b_0 + b_1X_1 + b_2X_2 \leftarrow \boxed{+ b_3X_3}$

$SS_{res} = SUM(y_i - \hat{y}_i)^2$

SS_{tot} doesn't change
SS_{res} will decrease or stay the same

(This is because of Ordinary Least Squares: SS_{res} -> Min)

Solution:

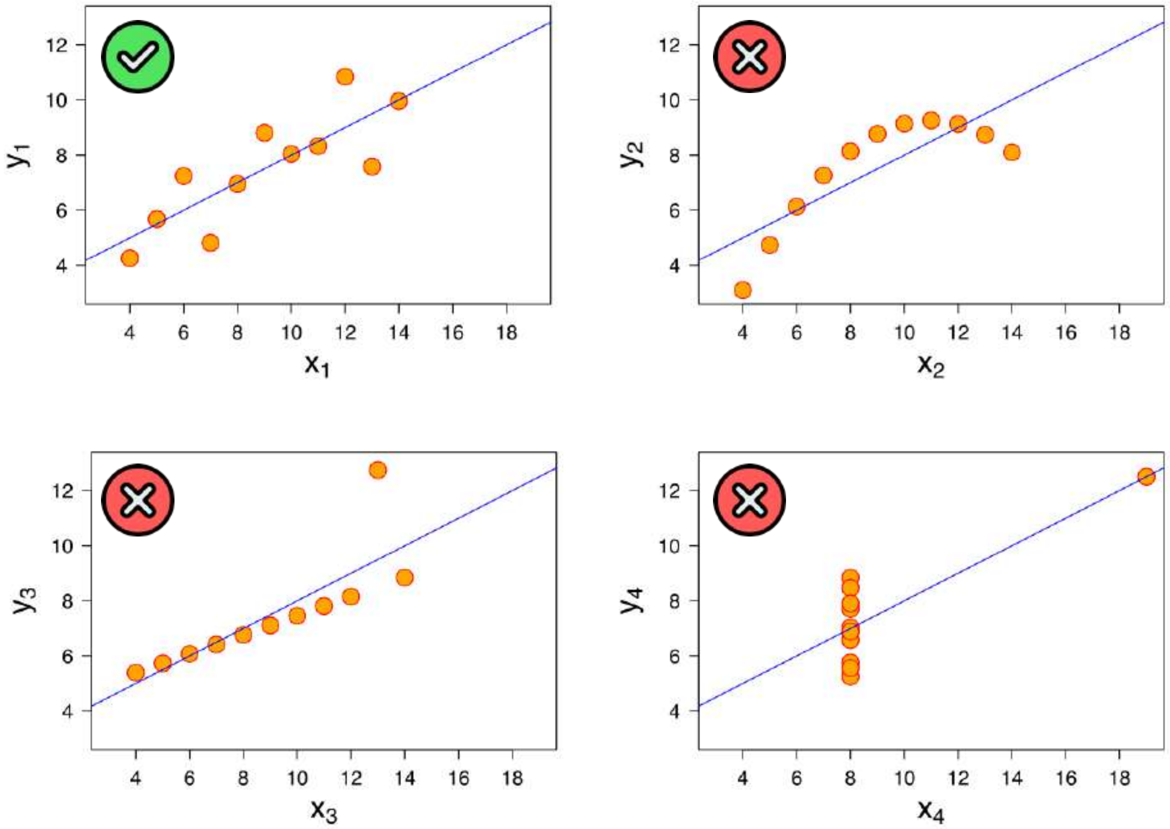
$$Adj\ R^2 = 1 - (1 - R^2) \times \frac{n-1}{n-k-1}$$

k – number of independent variables
n – sample size

Assumptions Of Linear Regression

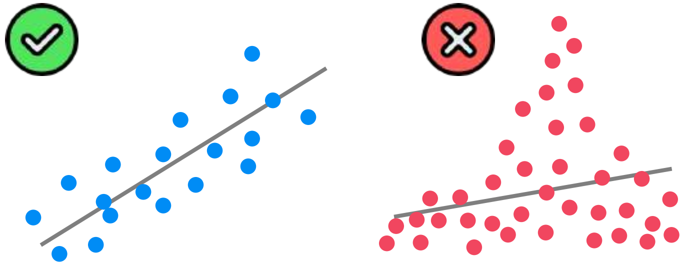
Assumptions of Linear Regression

Anscombe's quartet (1973):

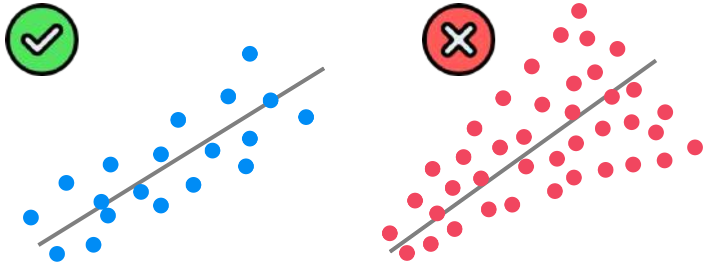


Assumptions of Linear Regression

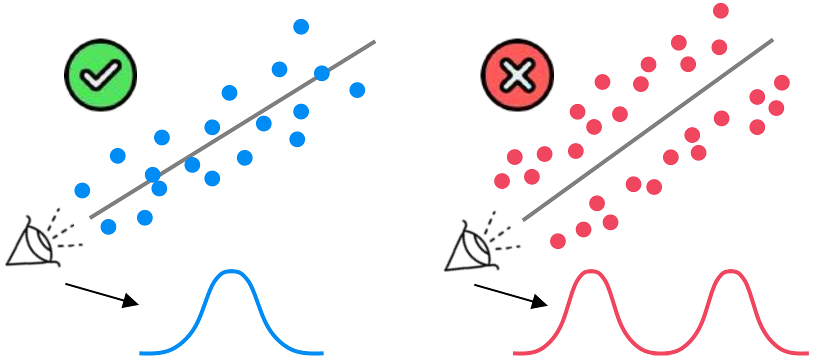
1. Linearity
(Linear relationship between Y and each X)



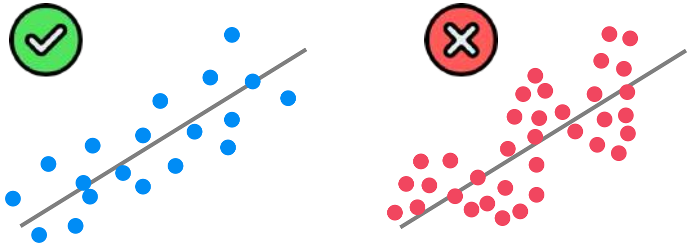
2. Homoscedasticity
(Equal variance)



3. Multivariate Normality
(Normality of error distribution)



4. Independence
(of observations. Includes “no autocorrelation”)

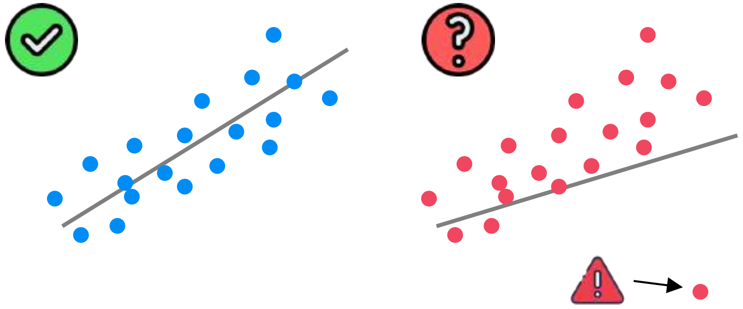


5. Lack of Multicollinearity
(Predictors are not correlated with each other)

$X_1 \nsim X_2$

$X_1 \sim X_2$

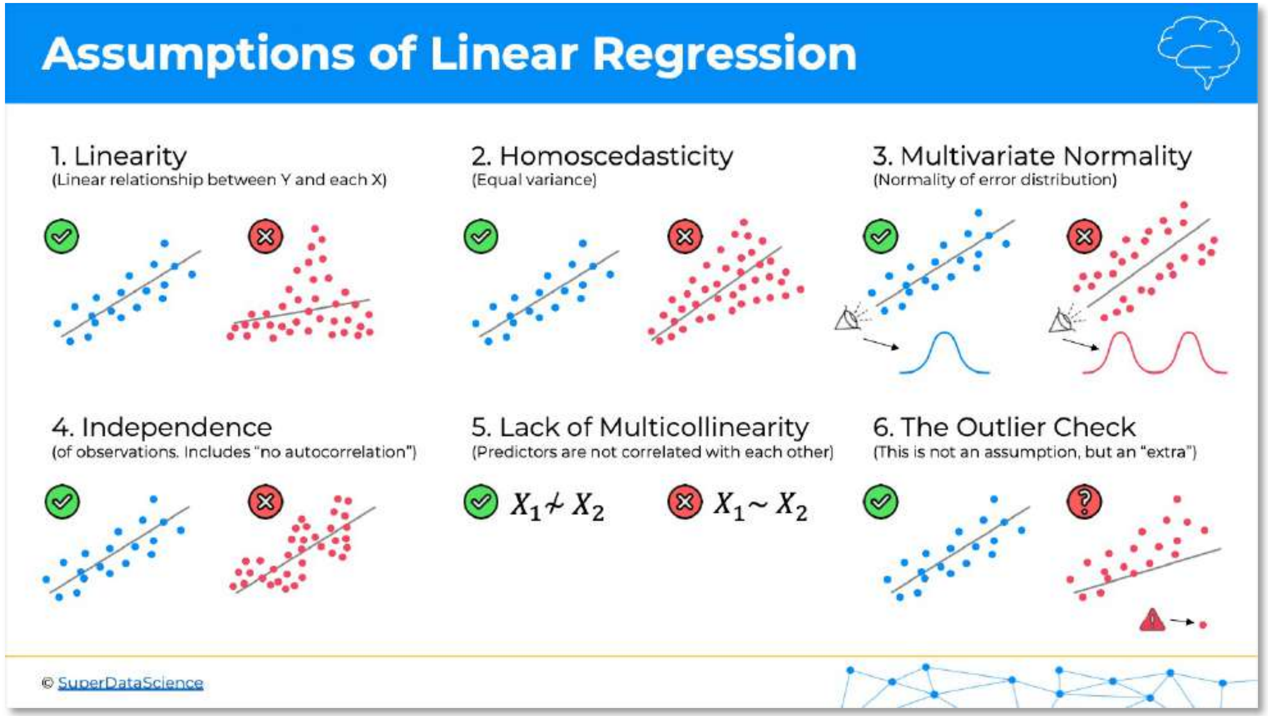
6. The Outlier Check
(This is not an assumption, but an “extra”)



Bonus

Download the Assumptions poster at:

superdatascience.com/assumptions



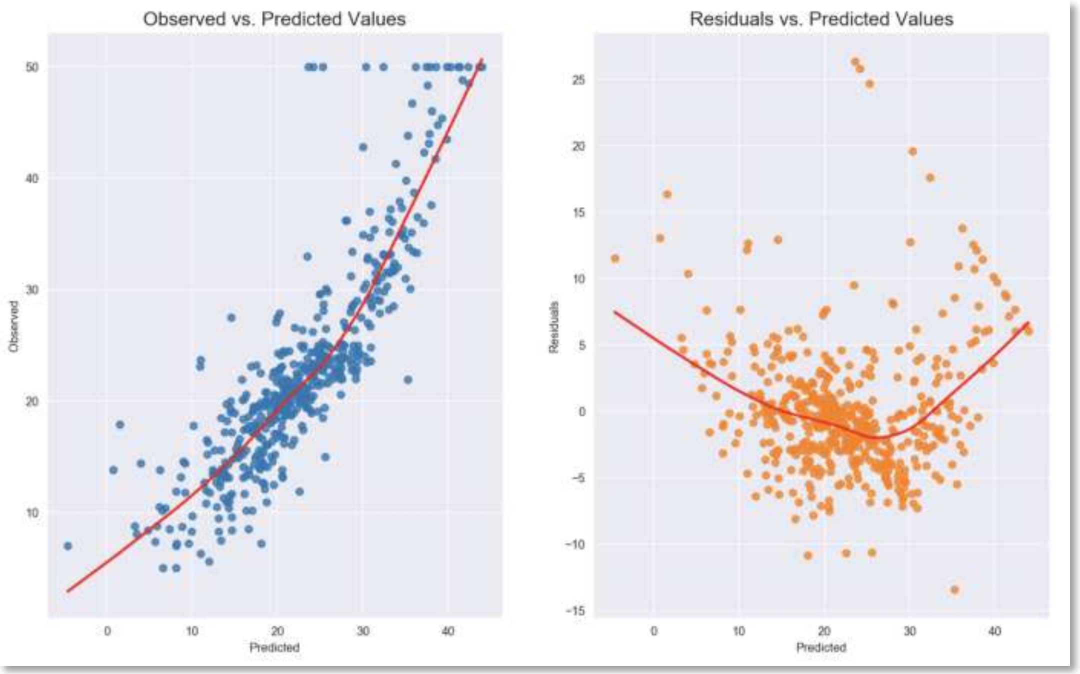
Additional Reading

Verifying the Assumptions of Linear Regression in Python and R

Eryk Lewinson (2019)

Link:

towardsdatascience.com/verifying-the-assumptions-of-linear-regression-in-python-and-r-f4cd2907d4c0



Dummy Variables

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 + \ ???$$

Dummy Variables

Profit	R&D Spend	Admin	Marketing	~~State~~
192,261.83	165,349.20	136,897.80	471,784.10	~~New York~~
191,792.06	162,597.70	151,377.59	443,898.53	~~California~~
191,050.39	153,441.51	101,145.55	407,934.54	~~California~~
182,901.99	144,372.41	118,671.85	383,199.62	~~New York~~
166,187.94	142,107.34	91,391.77	366,168.42	~~California~~

Dummy Variables

New York	~~California~~
1	~~0~~
0	~~1~~
0	~~1~~
1	~~0~~
0	~~1~~

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 \quad + b_4{*}D_1$$

Dummy Var. Trap

Dummy Variable Trap

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

New York	California
1	0
0	1
0	1
1	0
0	1

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 \quad + b_4{}^*D_1$$

Dummy Variable Trap

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	[illegible]	[illegible]	…nia
191,050.39	153,441.51	[illegible]	[illegible]	…nia
182,901.99	144,372.41	[illegible]	[illegible]	…ork
166,187.94	142,107.34	[illegible]	[illegible]	…nia

$$D_2 = 1 - D_1$$

Dummy Variables

New York	California
1	0
0	1
0	1
1	0
0	1

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 \quad + b_4{*}D_1 + \underline{b_5{*}D_2}$$

Dummy Variable Trap

Dummy Variables

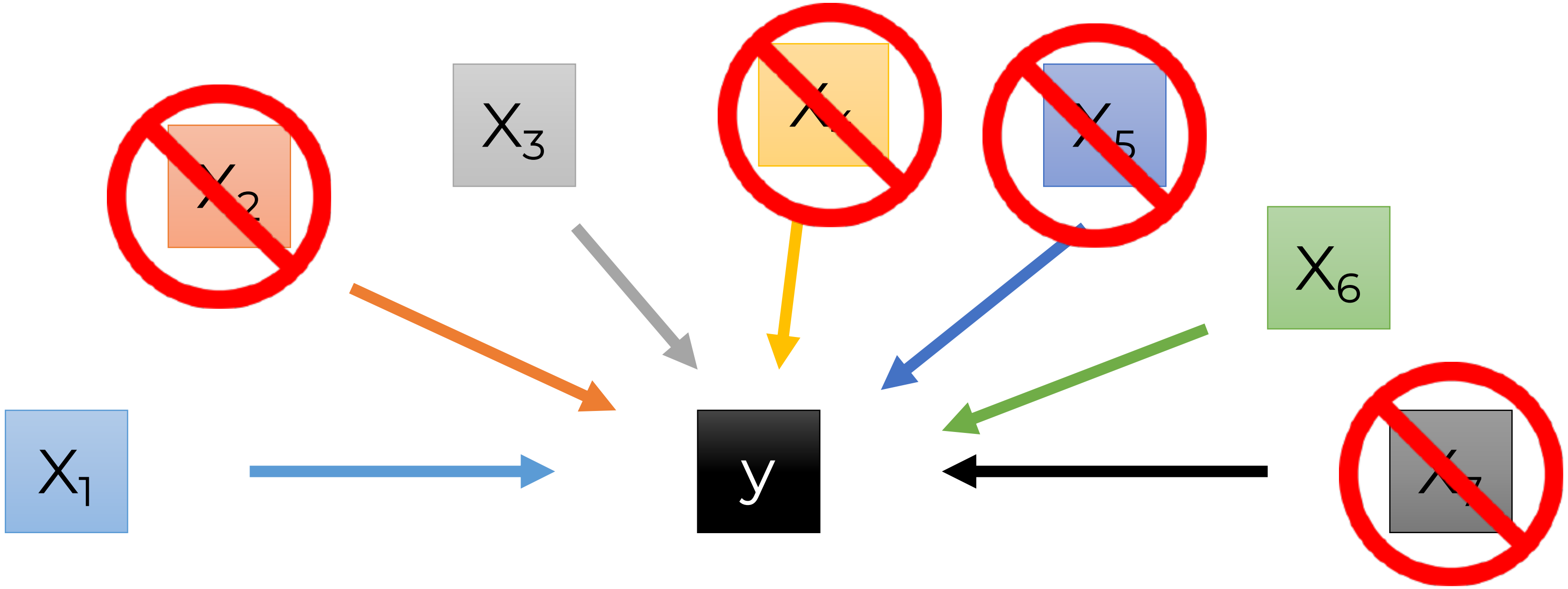
Profit	R&D Spend	Admin	Marketing	State	New York	California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70	151,377.59	443,898.53	California	0	1
191,050.39	153,441.51	101,145.55	407,934.54	California	0	1
182,901.99	144,372.41	118,671.85	383,199.62	New York	1	0
166,187.94	142,107.34	91,391.77	366,168.42	California	0	1

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 \quad + b_4{}^*D_1 + \cancel{b_5{}^*D_2}$$

Always omit one dummy variable

Building A Model (Step-By-Step)

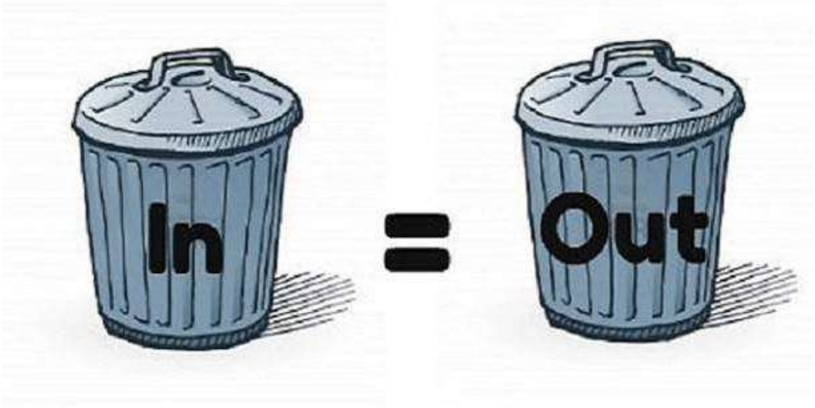
Building A Model



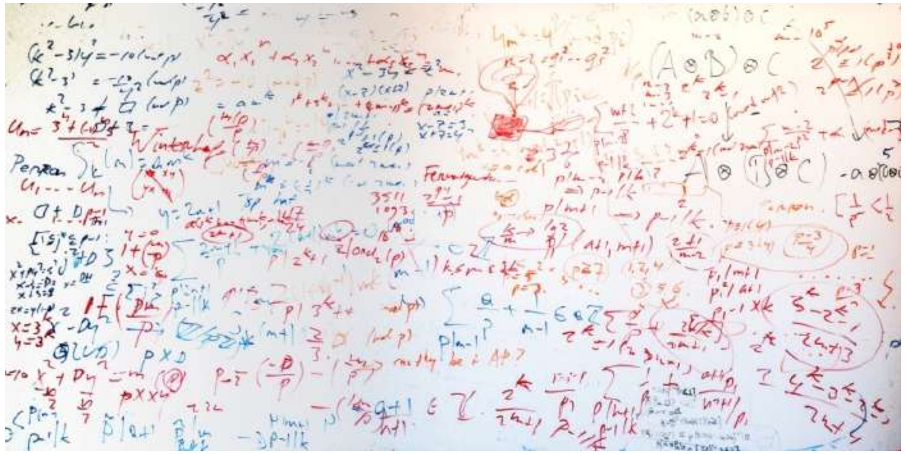
Why?

Building A Model

1)



2)



Building A Model

5 methods of building models:

1. All-in
2. Backward Elimination
3. Forward Selection
4. Bidirectional Elimination
5. Score Comparison

Stepwise Regression

Building A Model

"All-in" – cases:

- Prior knowledge; OR
- You have to; OR
- Preparing for Backward Elimination

Building A Model

Backward Elimination

STEP 1: Select a significance level to stay in the model (e.g. SL = 0.05)

STEP 2: Fit the full model with all possible predictors

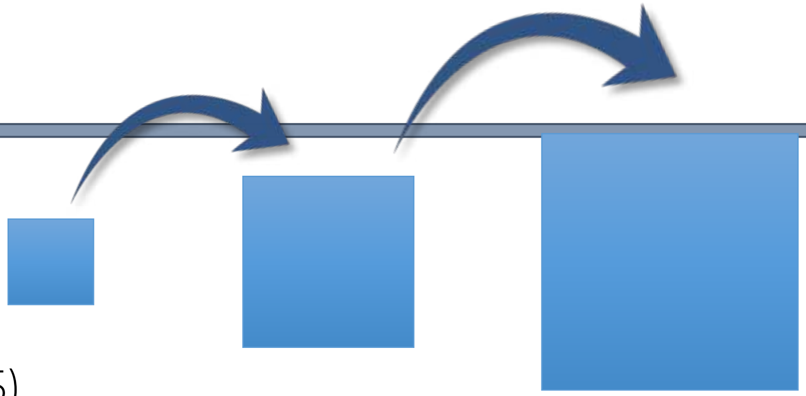
STEP 3: Consider the predictor with the <u>highest</u> P-value. If P > SL, go to STEP 4, otherwise go to FIN

STEP 4: Remove the predictor

STEP 5: Fit model without this variable*

FIN: Your Model Is Ready

Building A Model



Forward Selection

STEP 1: Select a significance level to enter the model (e.g. SL = 0.05)

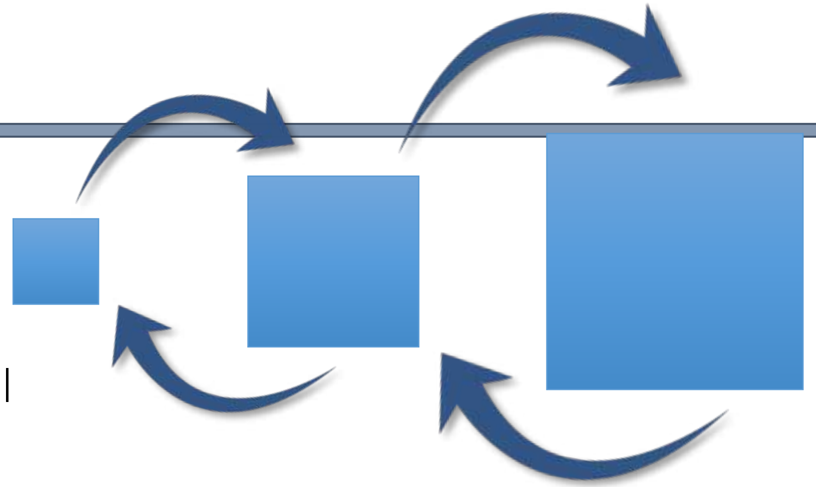
STEP 2: Fit all simple regression models **y ~ x_n** Select the one with the lowest P-value

STEP 3: Keep this variable and fit all possible models with one extra predictor added to the one(s) you already have

STEP 4: Consider the predictor with the <u>lowest</u> P-value. If P < SL, go to STEP 3, otherwise go to FIN

FIN: Keep the previous model

Building A Model



Bidirectional Elimination

STEP 1: Select a significance level to enter and to stay in the model
e.g.: SLENTER = 0.05, SLSTAY = 0.05

STEP 2: Perform the next step of Forward Selection (new variables must have: P < SLENTER to enter)

STEP 3: Perform ALL steps of Backward Elimination (old variables must have P < SLSTAY to stay)

STEP 4: No new variables can enter and no old variables can exit

FIN: Your Model Is Ready

Building A Model

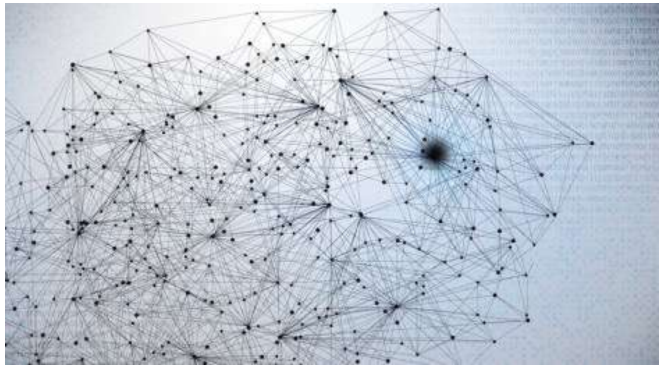
All Possible Models

STEP 1: Select a criterion of goodness of fit (e.g. Akaike criterion)

STEP 2: Construct All Possible Regression Models: 2^N-1 total combinations

STEP 3: Select the one with the best criterion

FIN: Your Model Is Ready



Example:
10 columns means
1,023 models

Building A Model

5 methods of building models:

1. All-in
2. Backward Elimination
3. Forward Selection
4. Bidirectional Elimination
5. Score Comparison

Section Recap

Section Recap

In this section we learned:

1. How to create dummies for categorical IVs
2. How to avoid the dummy variable trap
3. Backward, Forward, Bidirectional, All Possible
4. We actually built a model. Step-By-Step!!
5. How to use adjusted R-squared in modelling
6. How to interpret coefficients of a MLR

Polynomial Regression

Regressions

Simple Linear Regression

$$y = b_0 + b_1 x_1$$

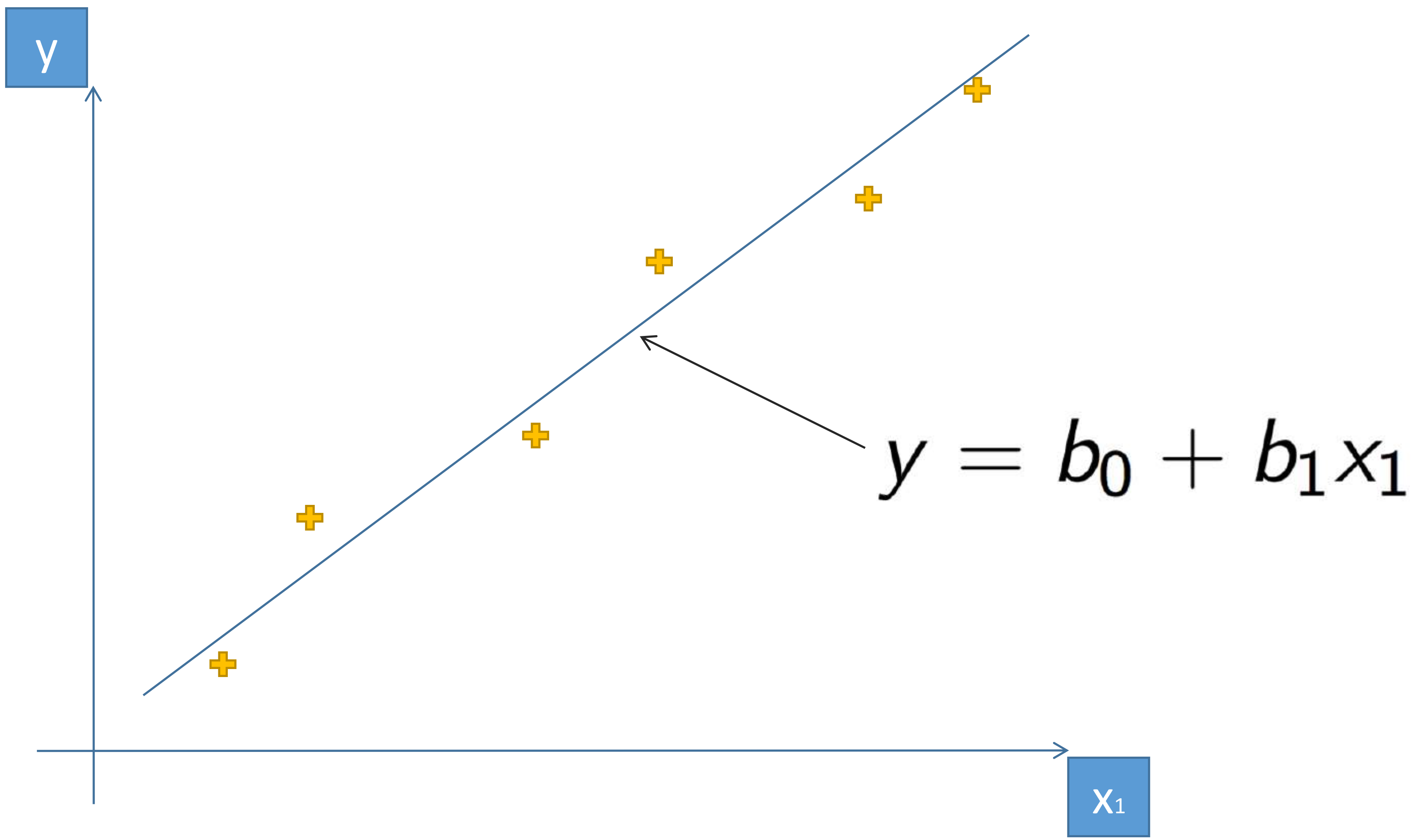
Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

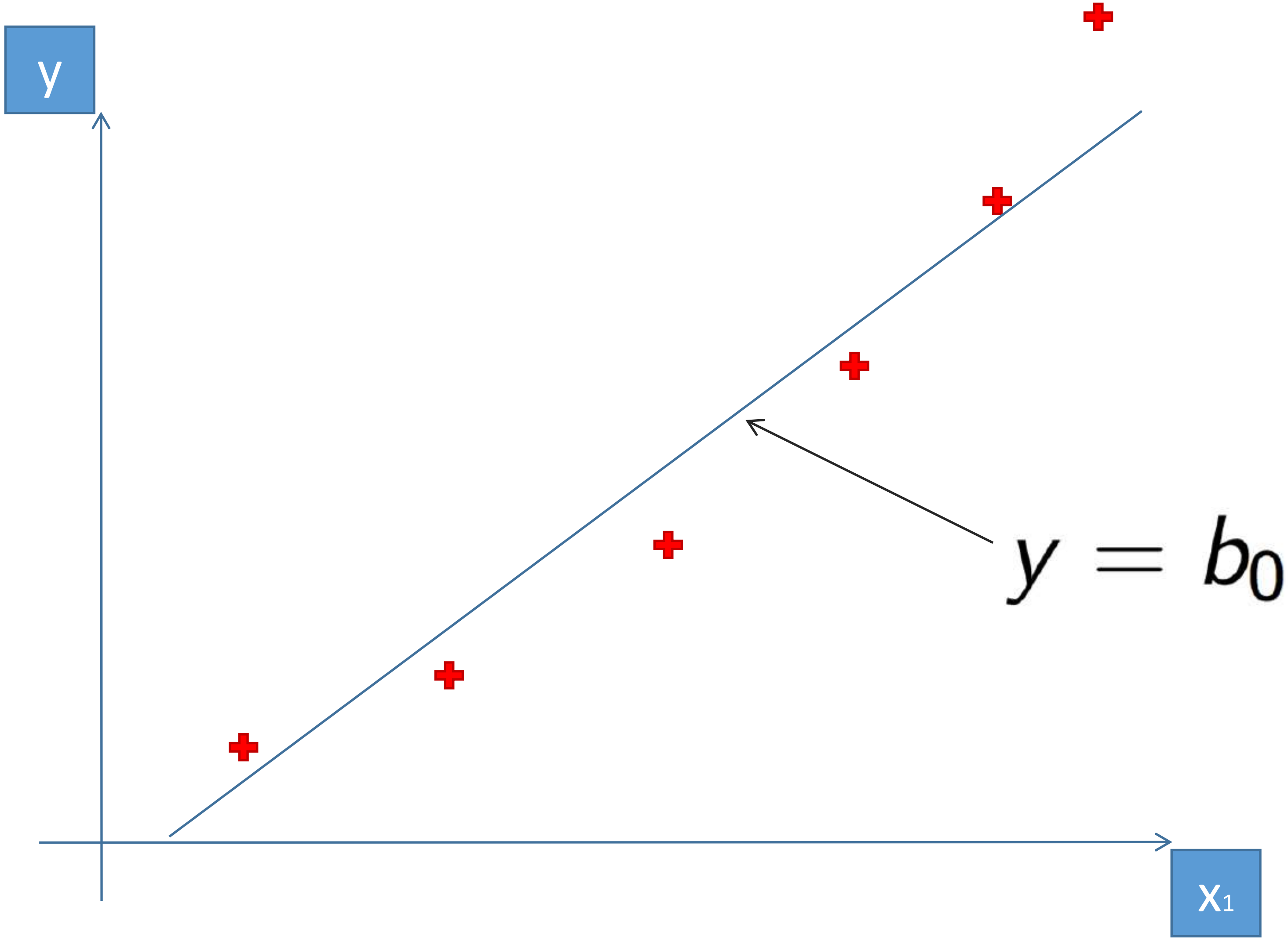
Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

Simple Linear Regression

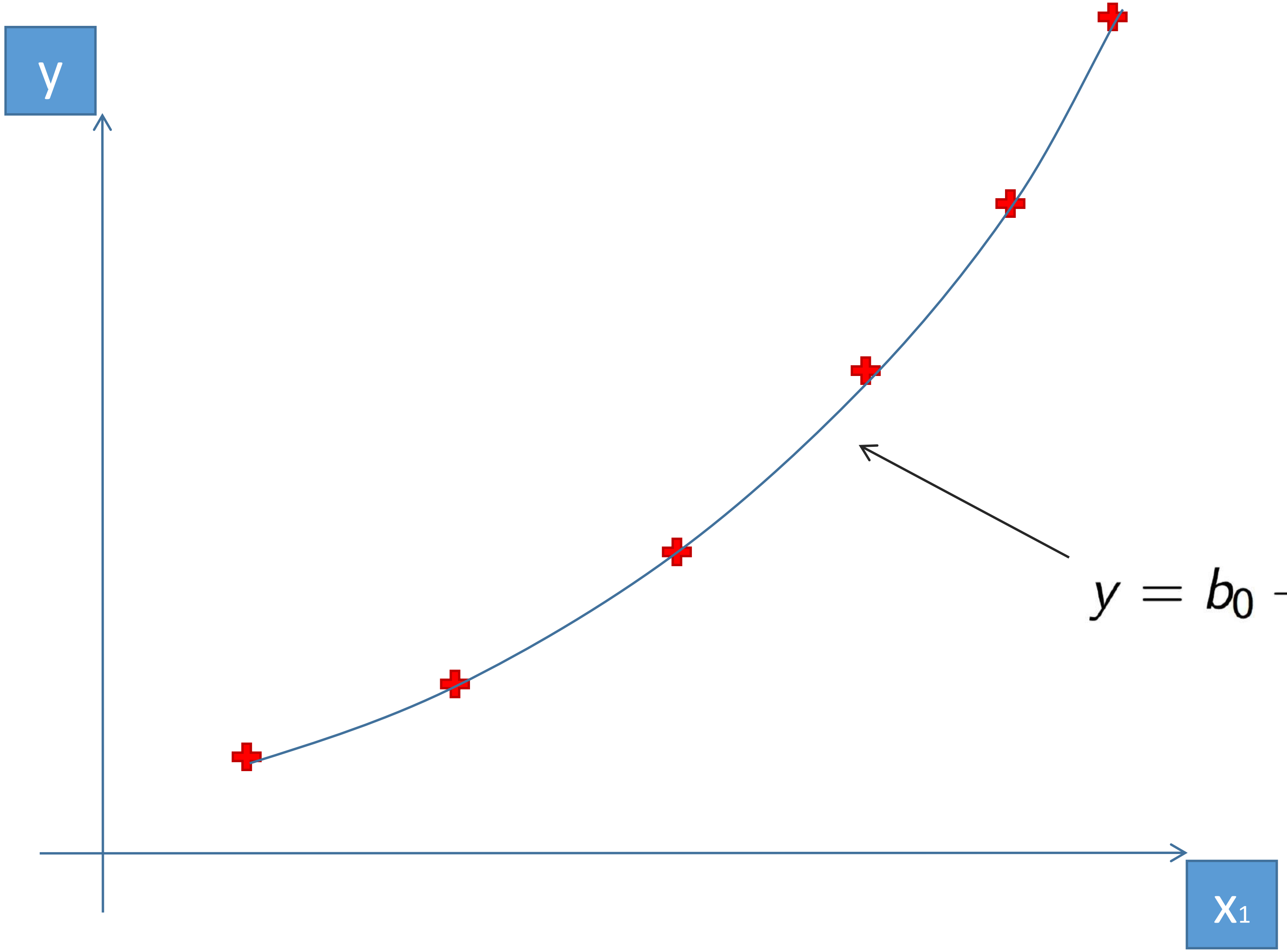


Simple Linear Regression



$$y = b_0 + b_1 x_1$$

Polinomial Regression



$$y = b_0 + b_1 x_1 \boxed{+ b_2 x_1^2}$$

Polinomial Regression

One Question: Why "Linear"?

Polinomial Regression

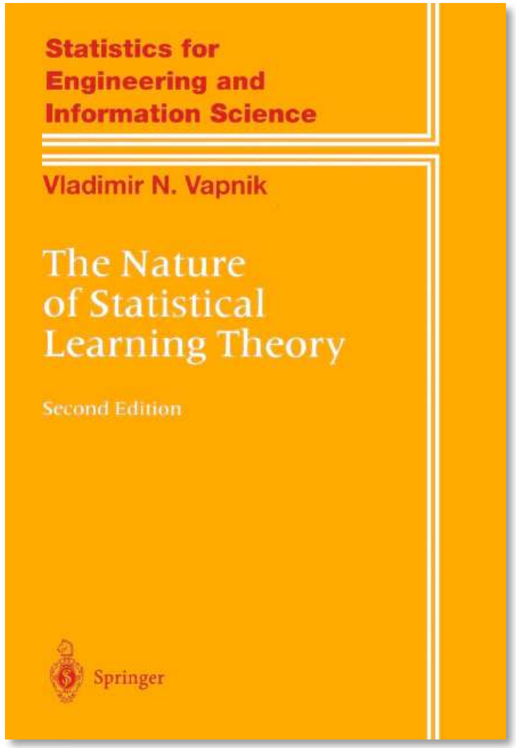
Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

SVR Intuition

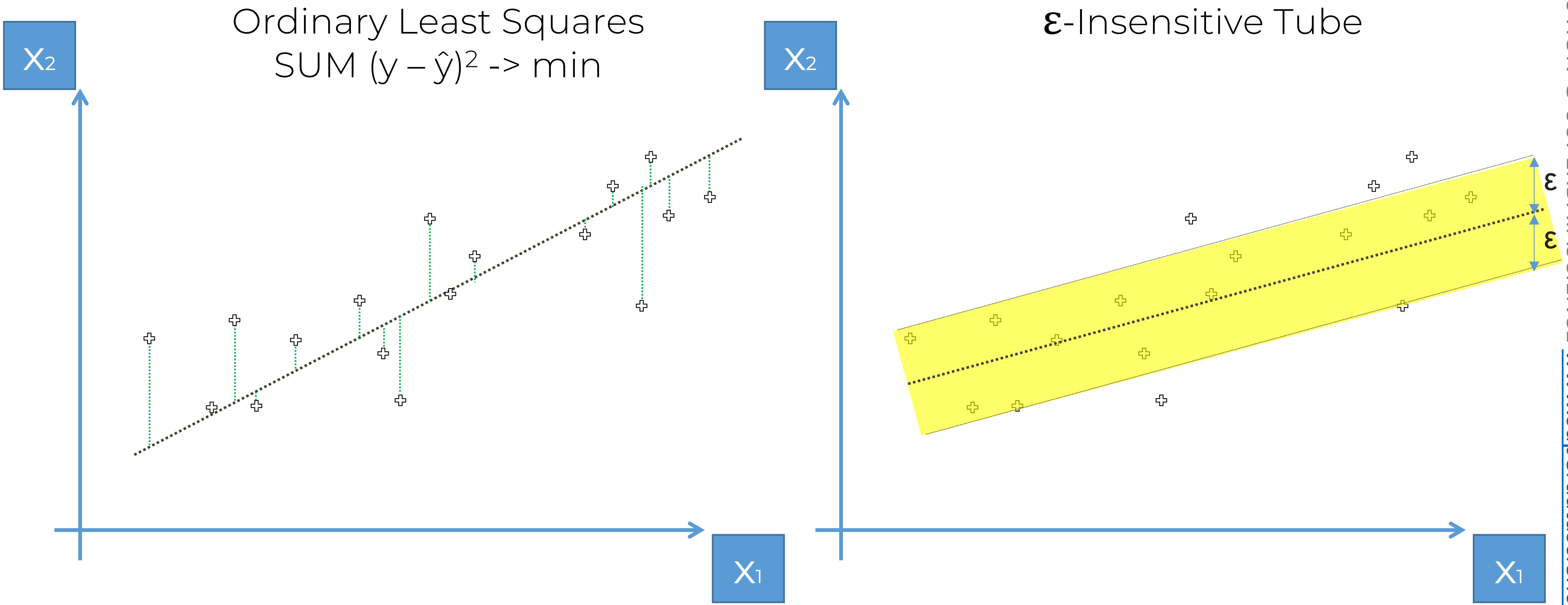
SVR Intuition

Vladimir Vapnik



1992

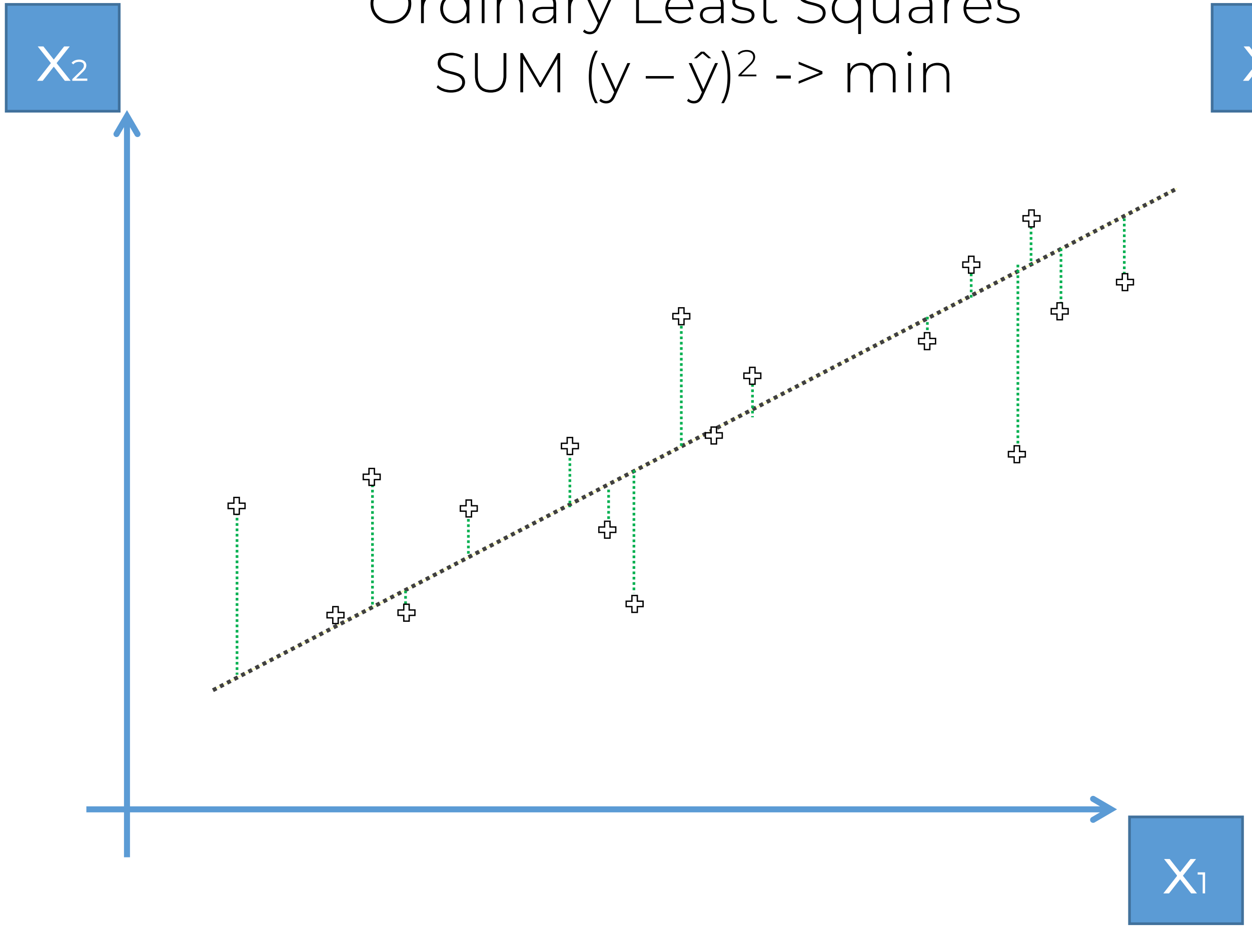
SVR Intuition



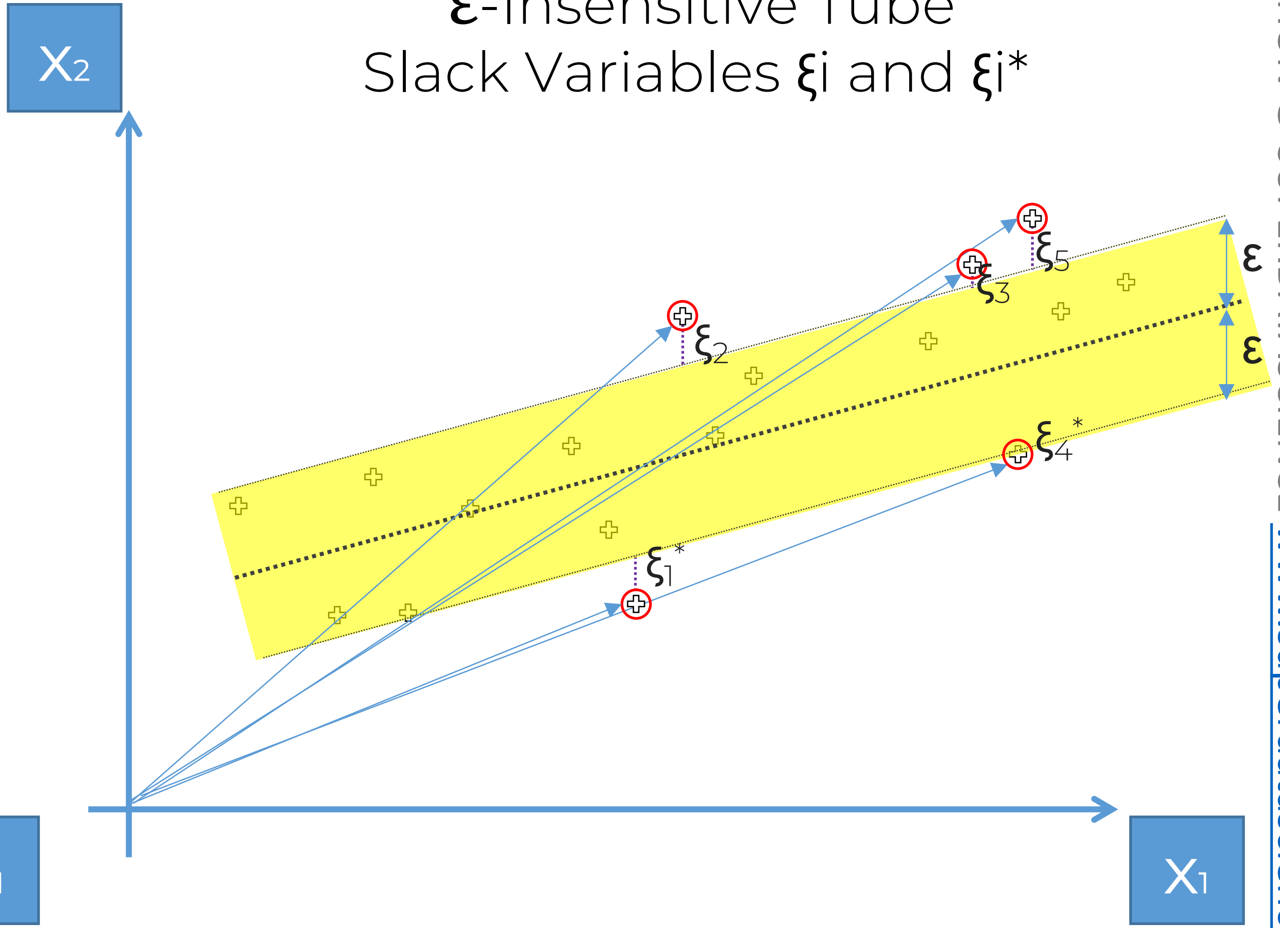
SVR Intuition

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \rightarrow min$$

Ordinary Least Squares
SUM (y – ŷ)² -> min



ε-Insensitive Tube
Slack Variables ξi and ξi*



SVR Intuition

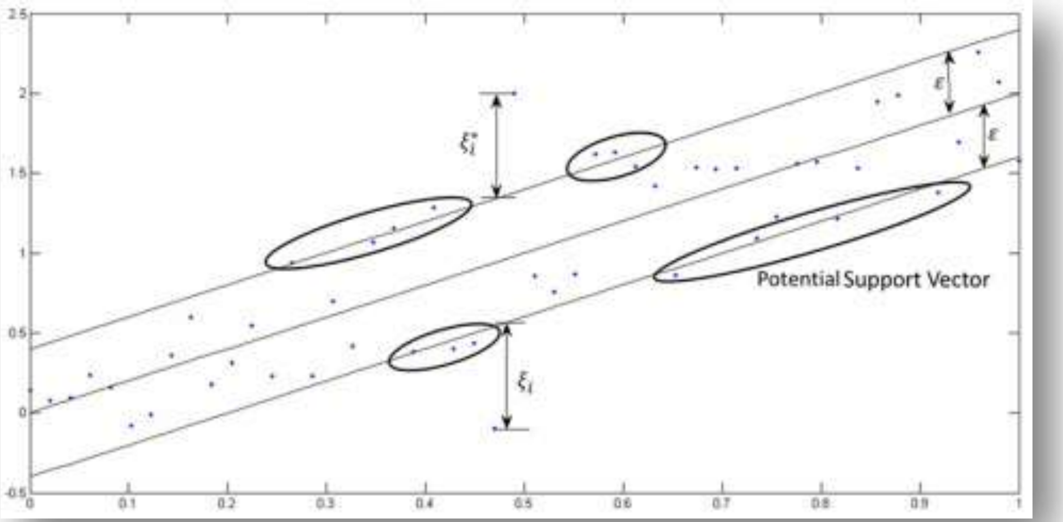
Additional Reading:

Chapter 4 – Support Vector Regression (from: Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers)

By Mariette Awad & Rahul Khanna (2015)

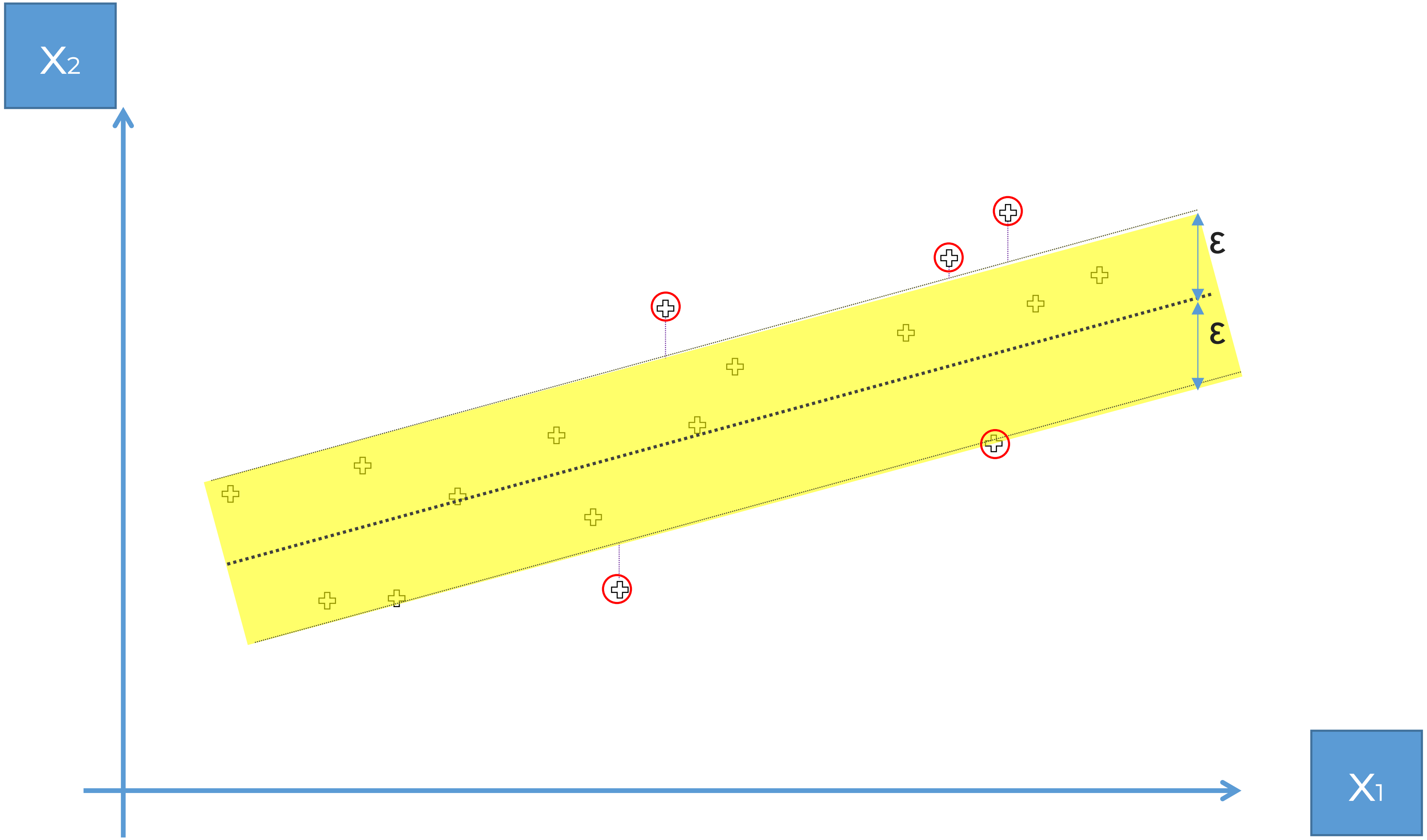
Link:

https://core.ac.uk/download/pdf/81523322.pdf



Heads-up about Non-Linear SVR

SVR Intuition



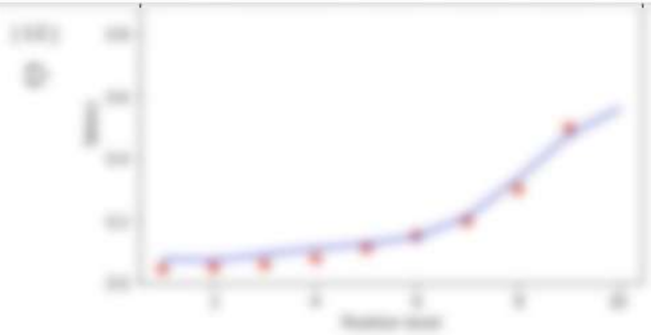
Copy of support_vector_regression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

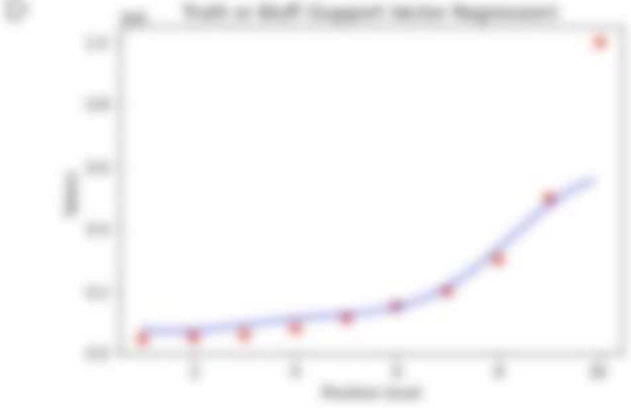
Upload Refresh Mount Drive

..
sample_data
Position_Salaries.csv



Visualising the SVR results (for higher resolution and smoother curve)

```
X_grid = np.arange(min(sc_X.inverse_transform(X)), max(sc_X.inverse_transform(X)), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(sc_X.inverse_transform(X), sc_y.inverse_transform(y), color = 'red')
plt.plot(X_grid, sc_y.inverse_transform(regressor.predict(sc_X.transform(X_grid))), color = 'blue')
plt.title('Truth or Bluff (Support Vector Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Position_Salaries.csv

1 to 10 of 10 entries

Position	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Show 10 per page

Disk 76.87 GB available

Heads-up about Non-Linear SVR

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

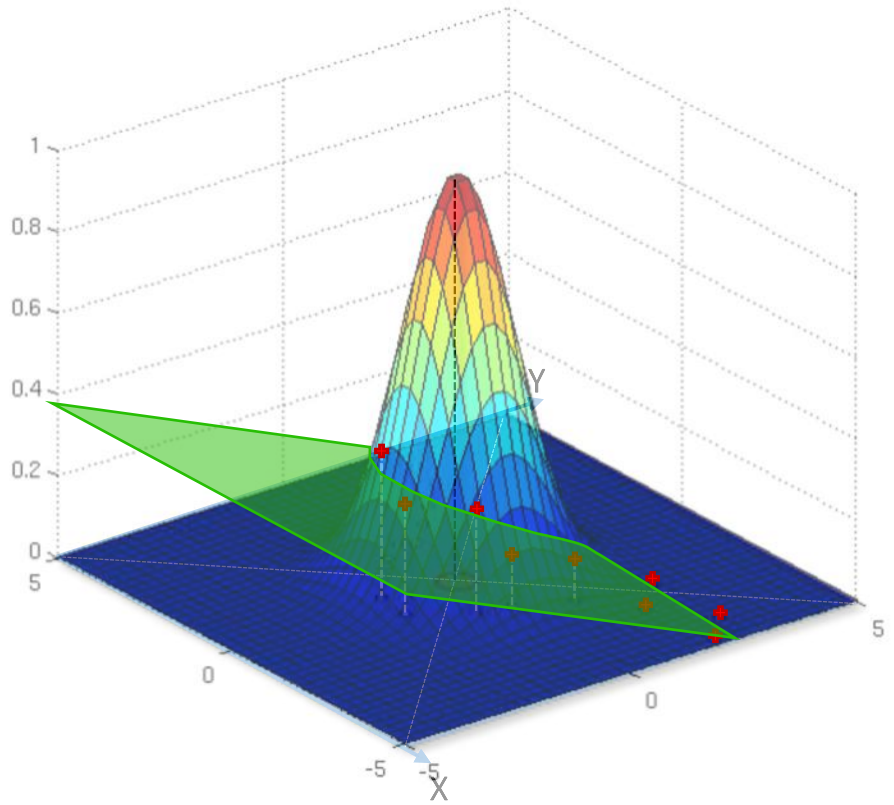


Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

Decision Tree Intuition

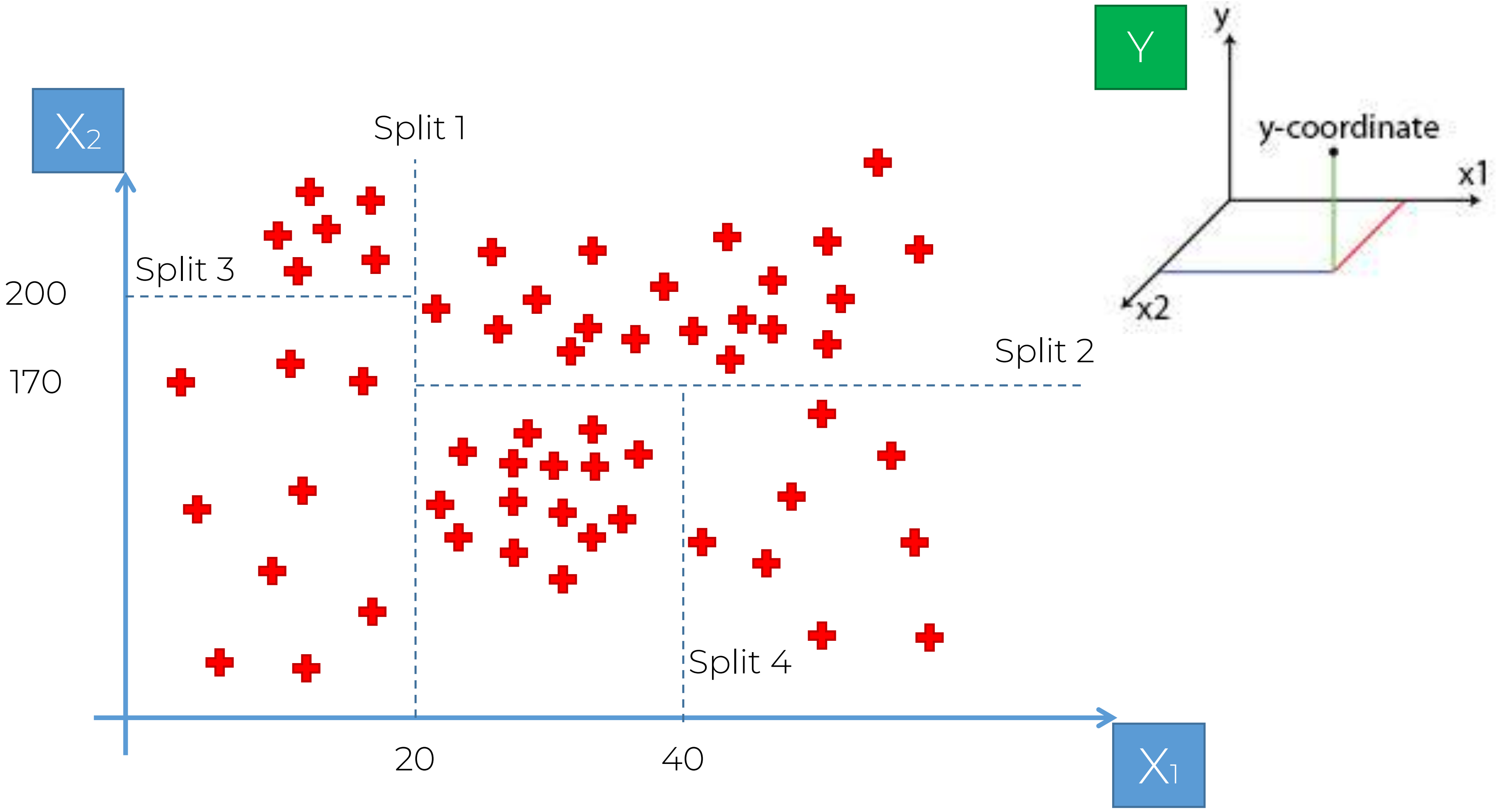
Decision Tree Intuition

CART

Classification Trees

Regression Trees

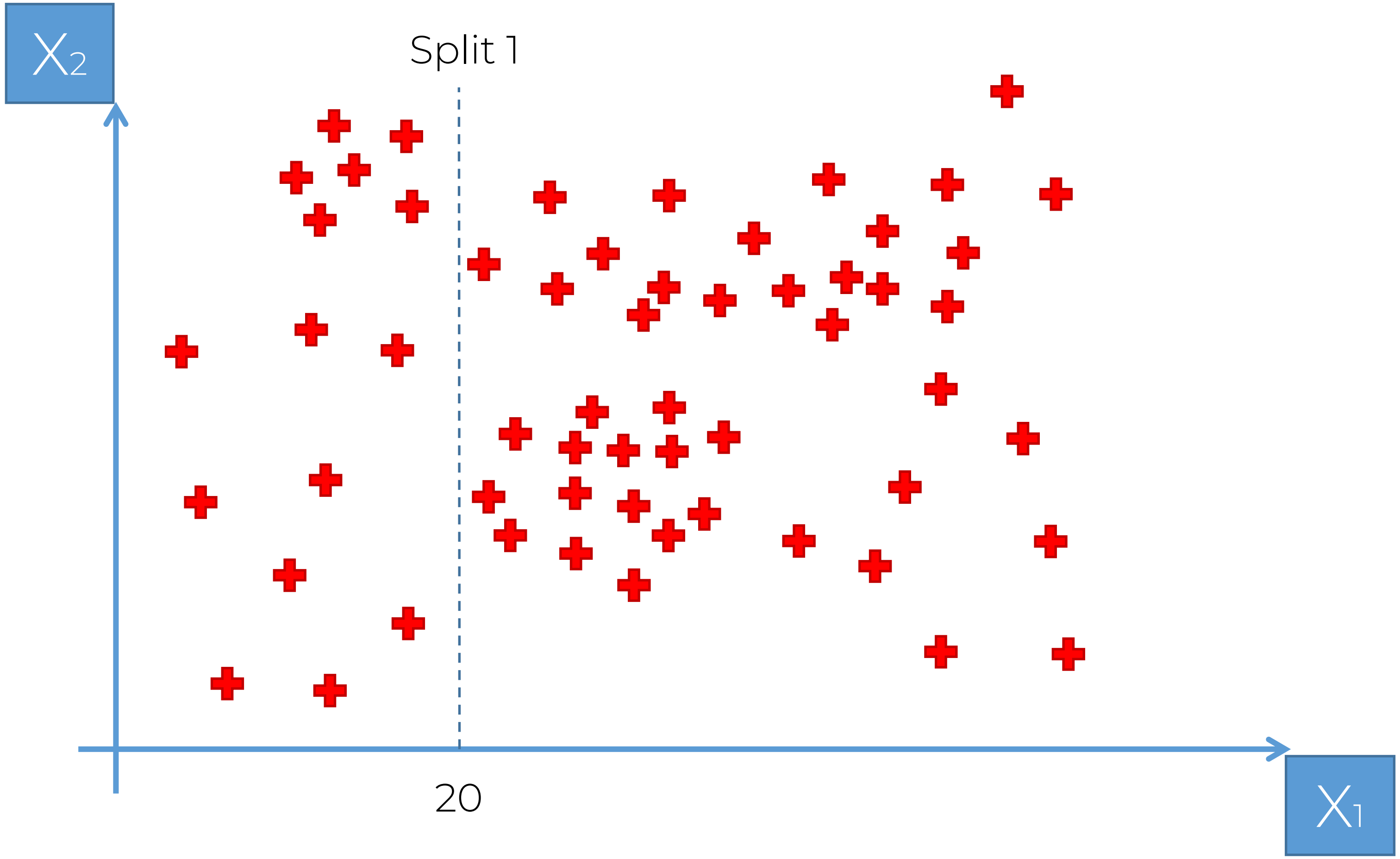
Decision Tree Intuition



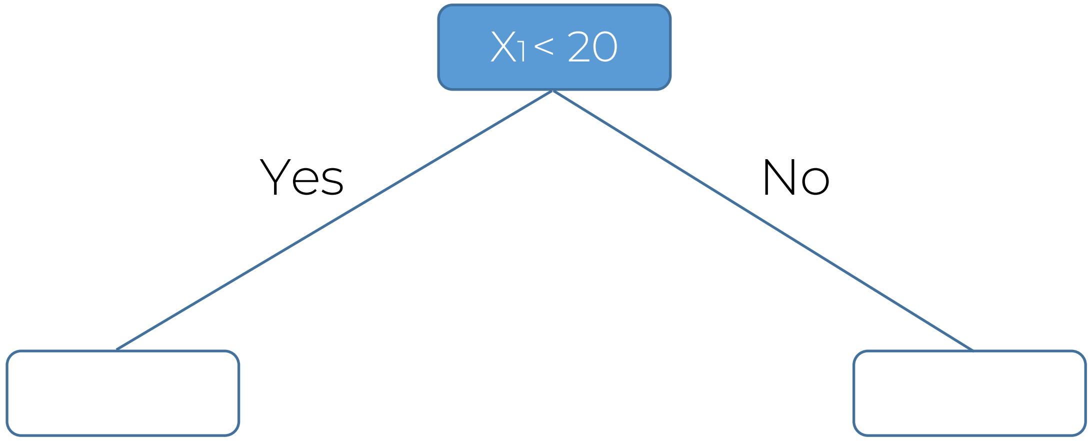
Decision Tree Intuition

Rewind…

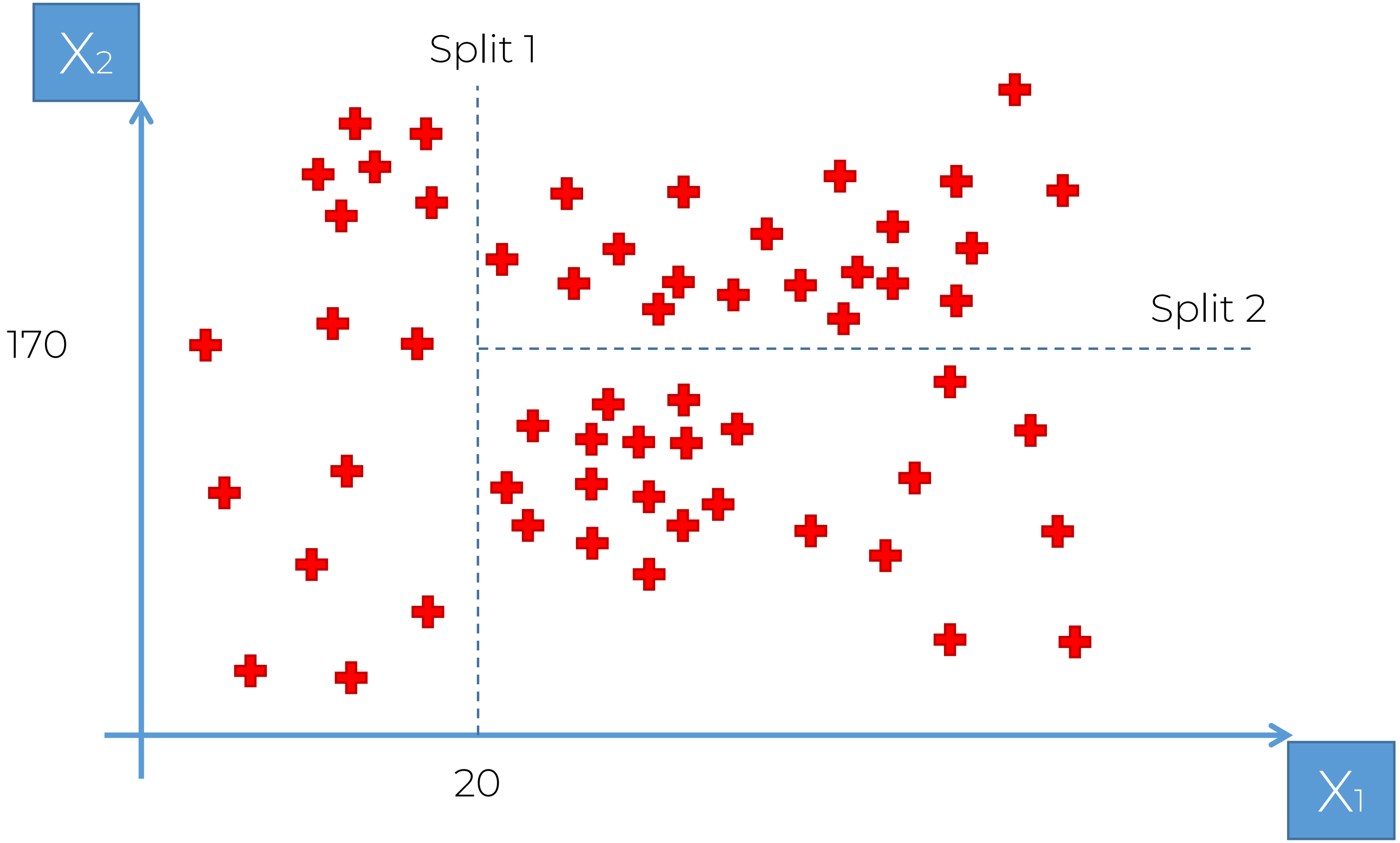
Decision Tree Intuition



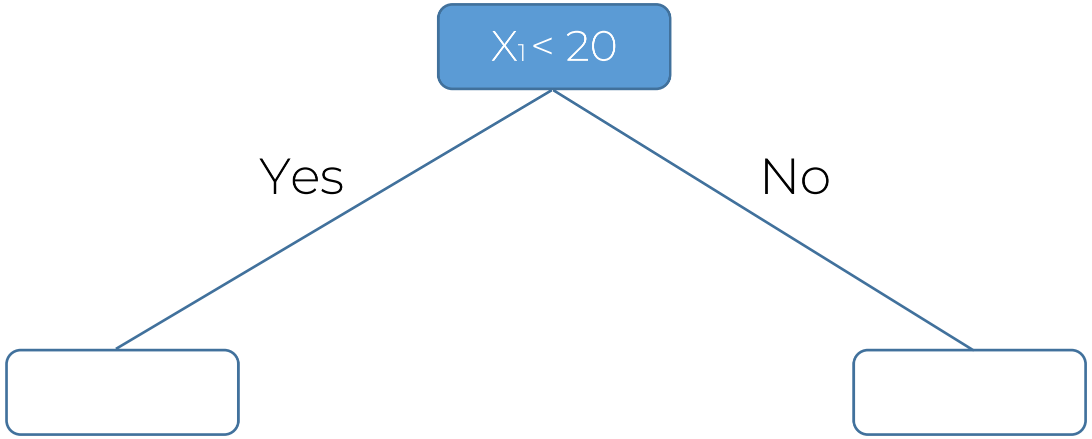
Decision Tree Intuition



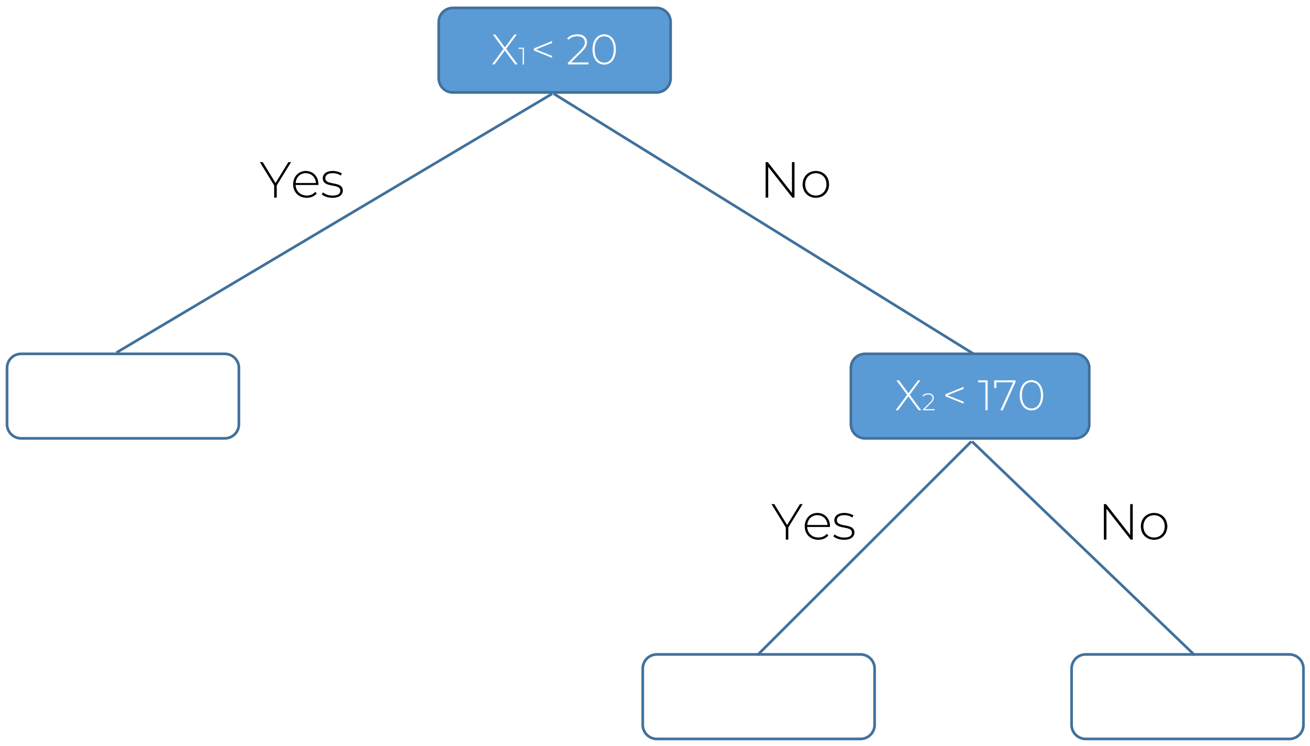
Decision Tree Intuition



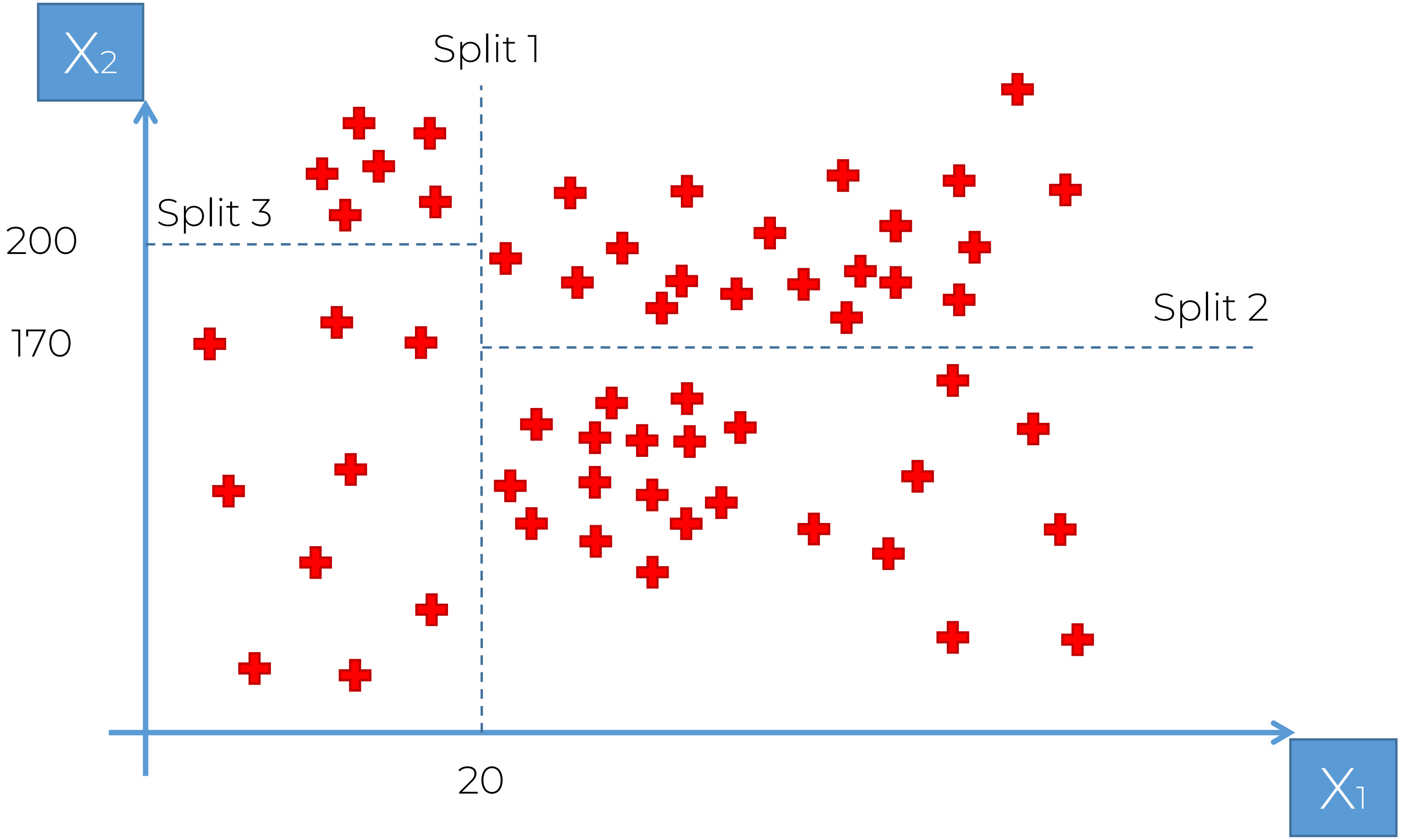
Decision Tree Intuition



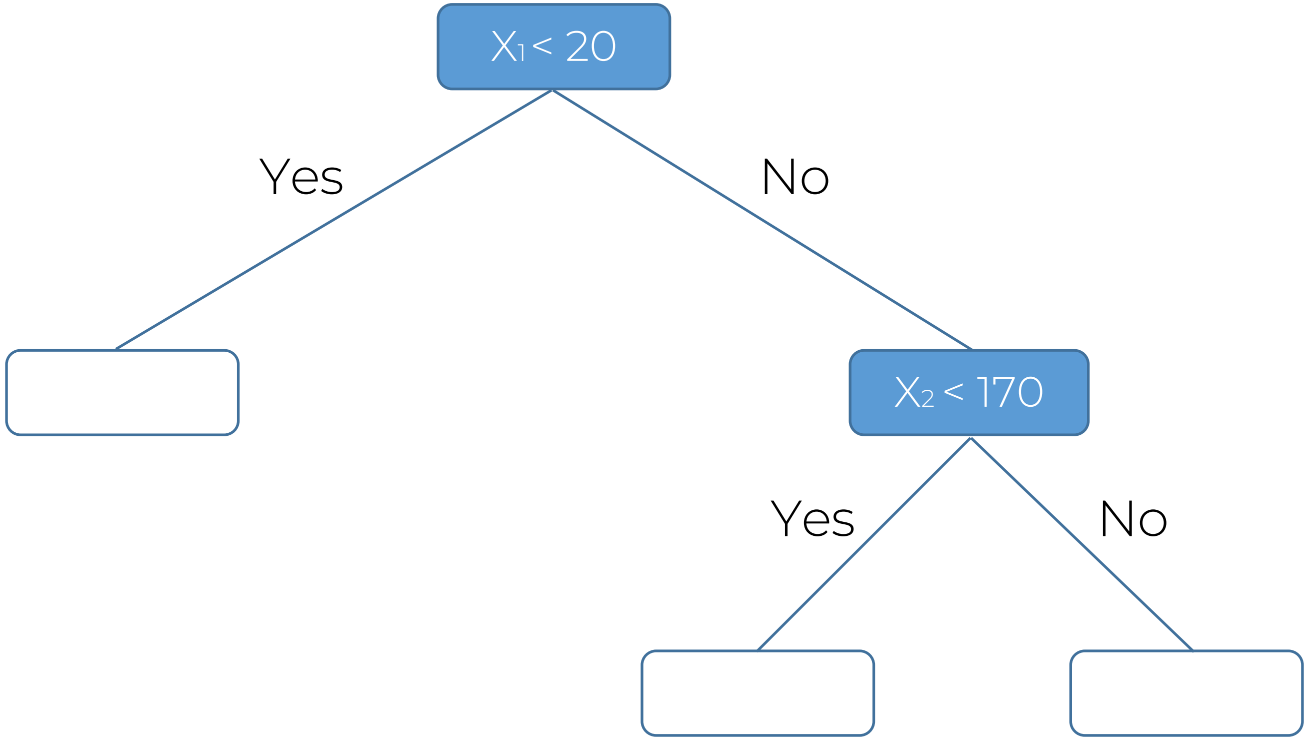
Decision Tree Intuition



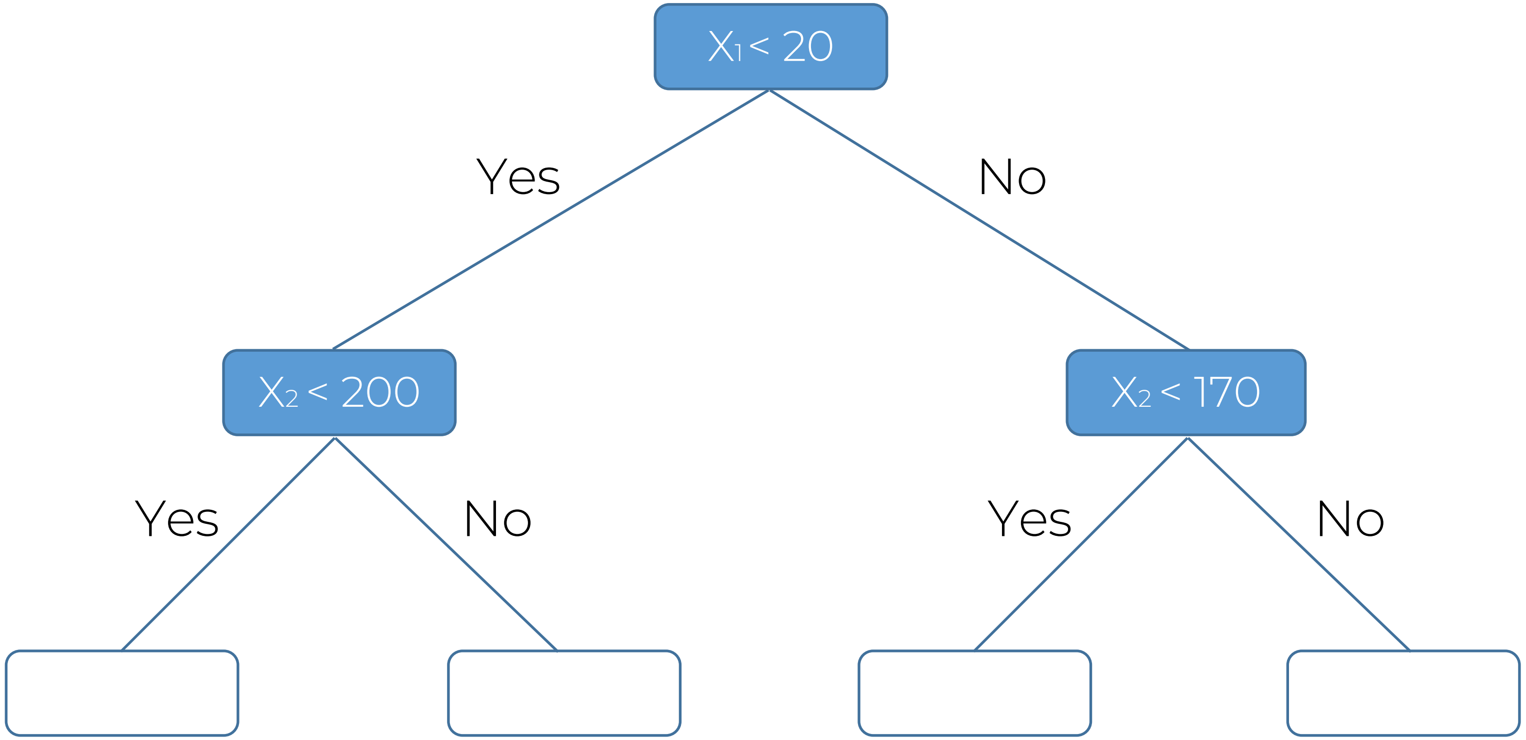
Decision Tree Intuition



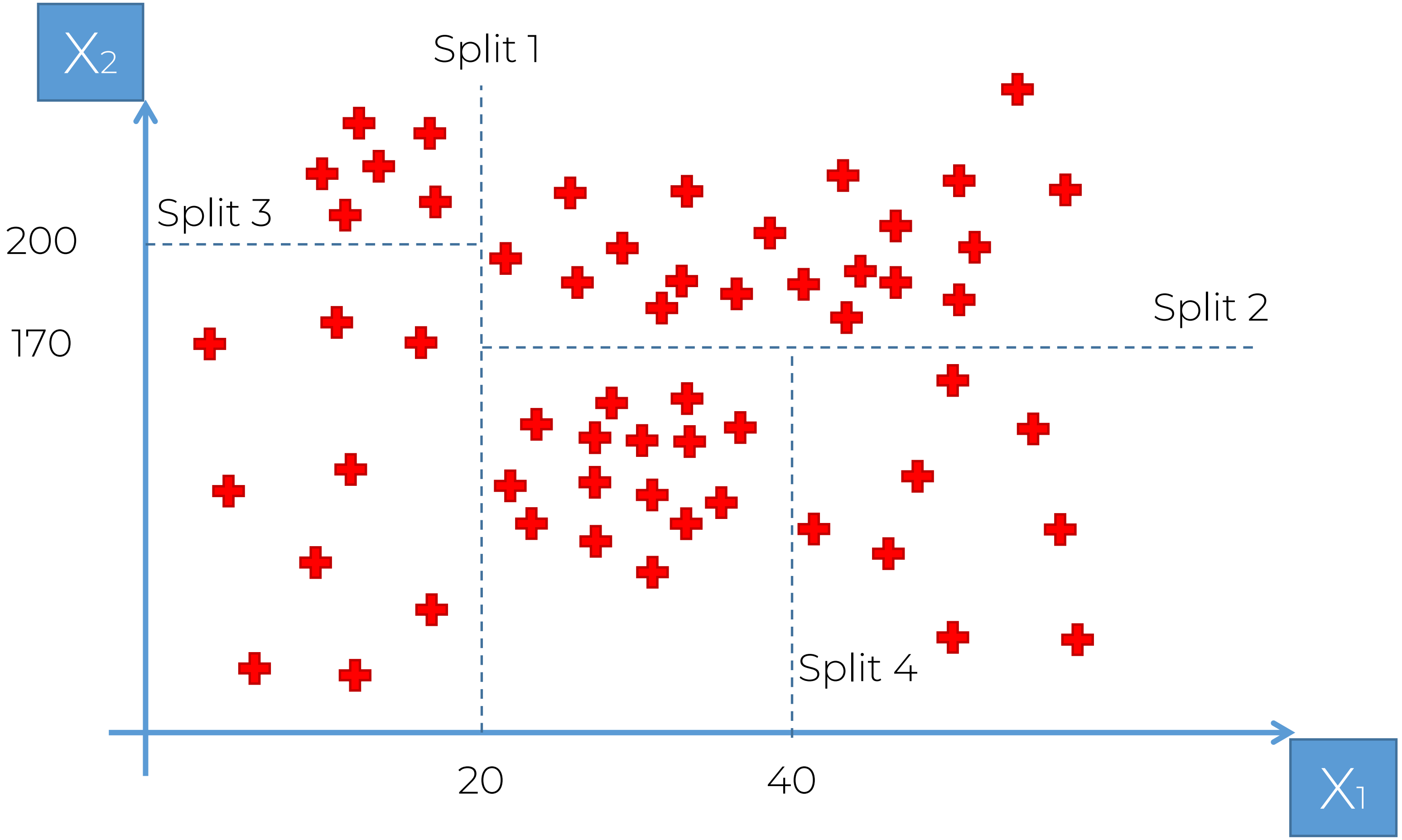
Decision Tree Intuition



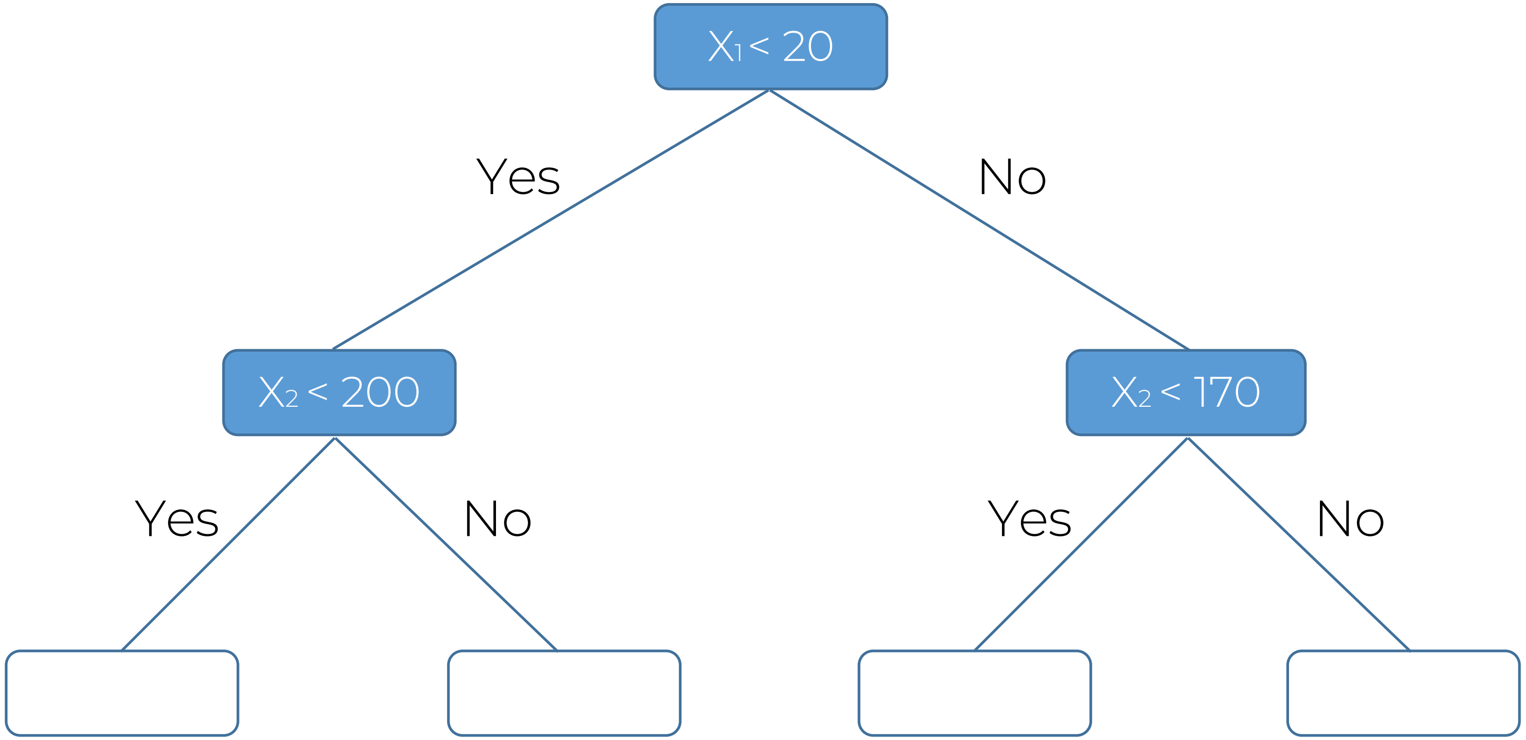
Decision Tree Intuition



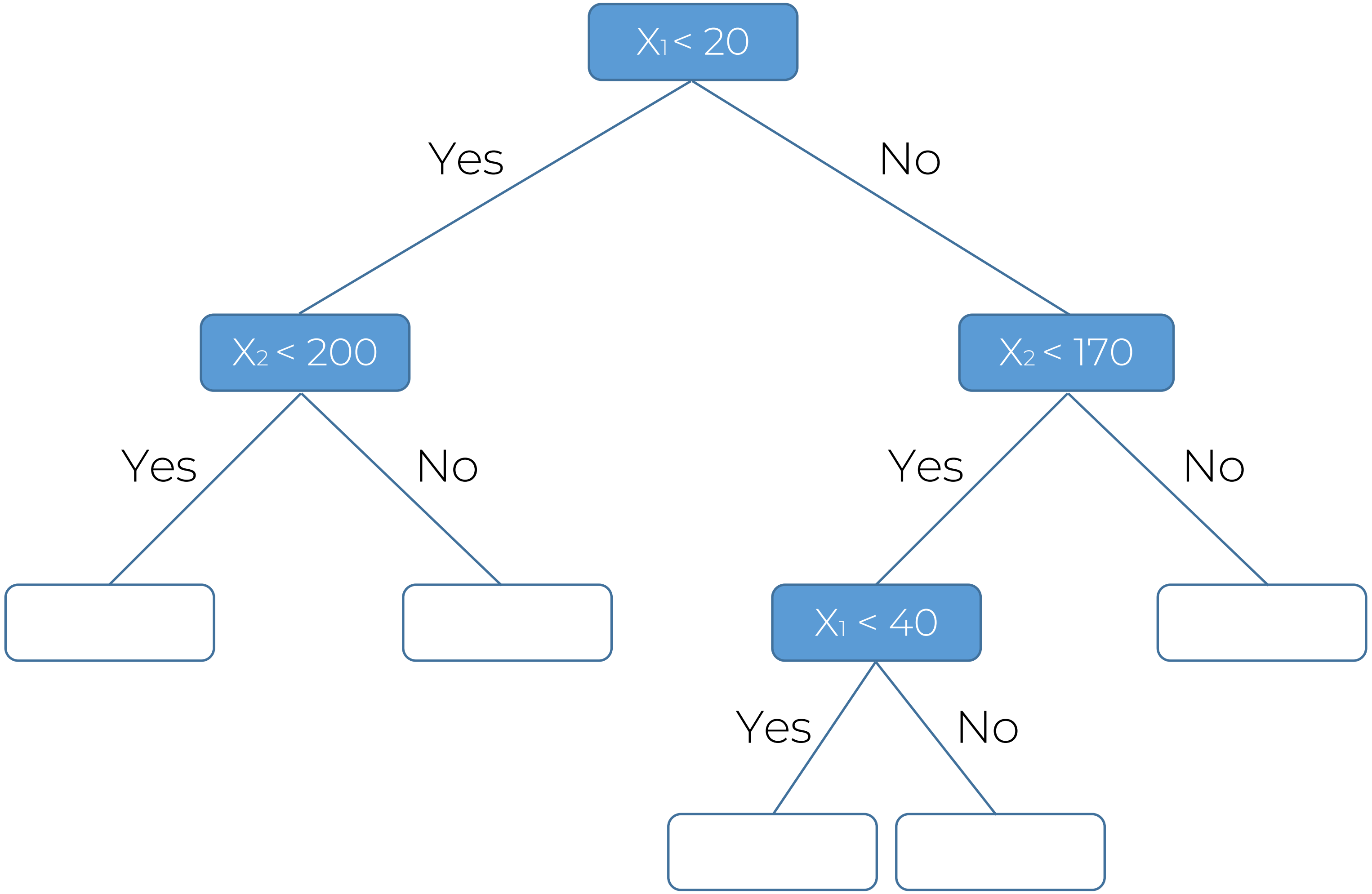
Decision Tree Intuition



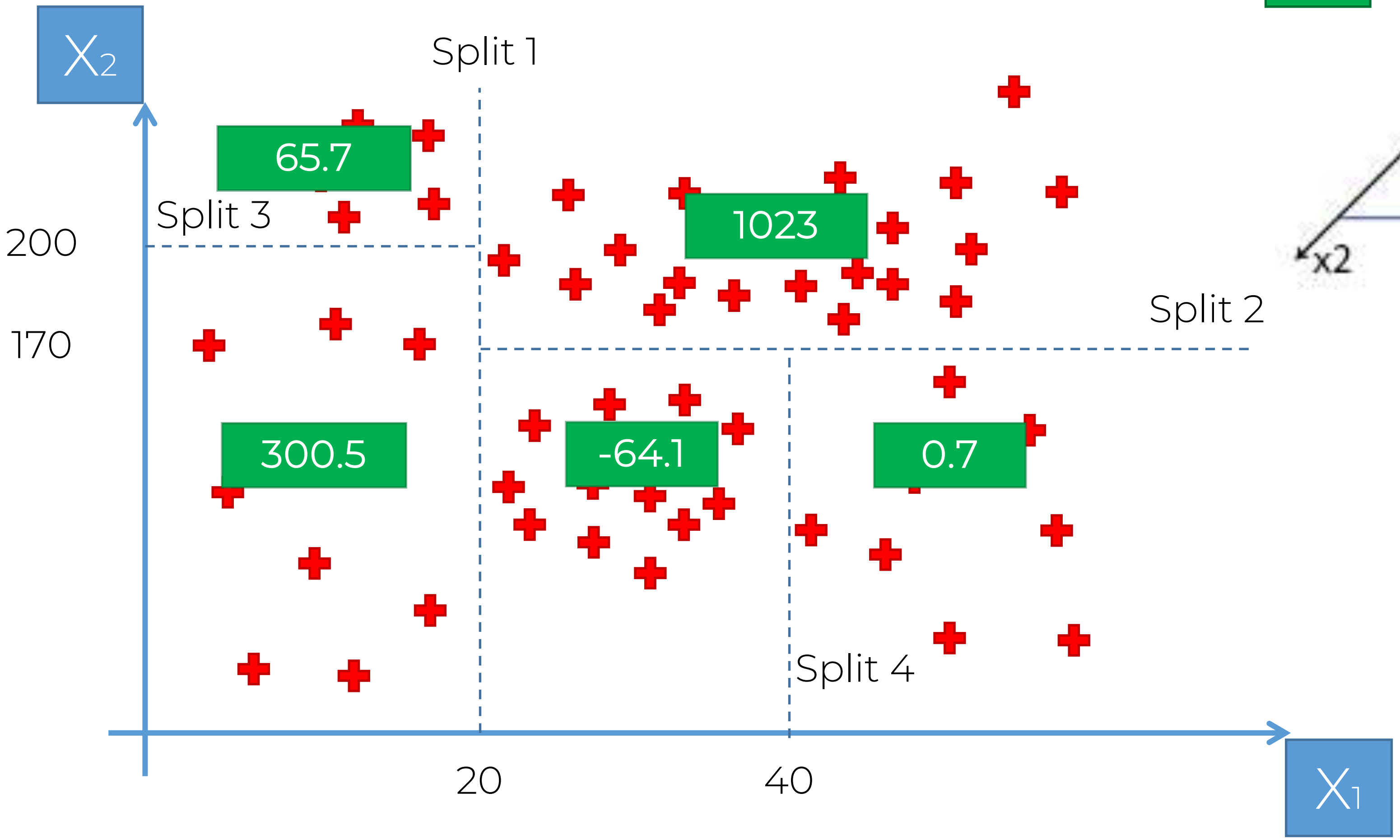
Decision Tree Intuition

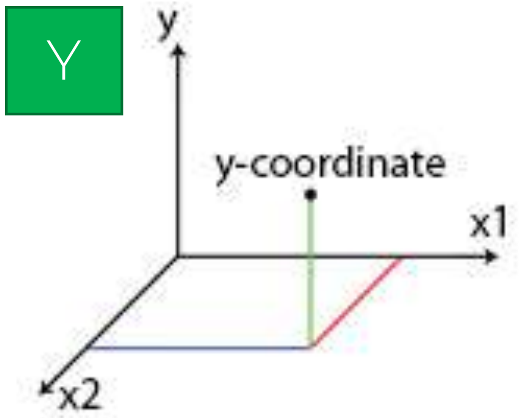


Decision Tree Intuition

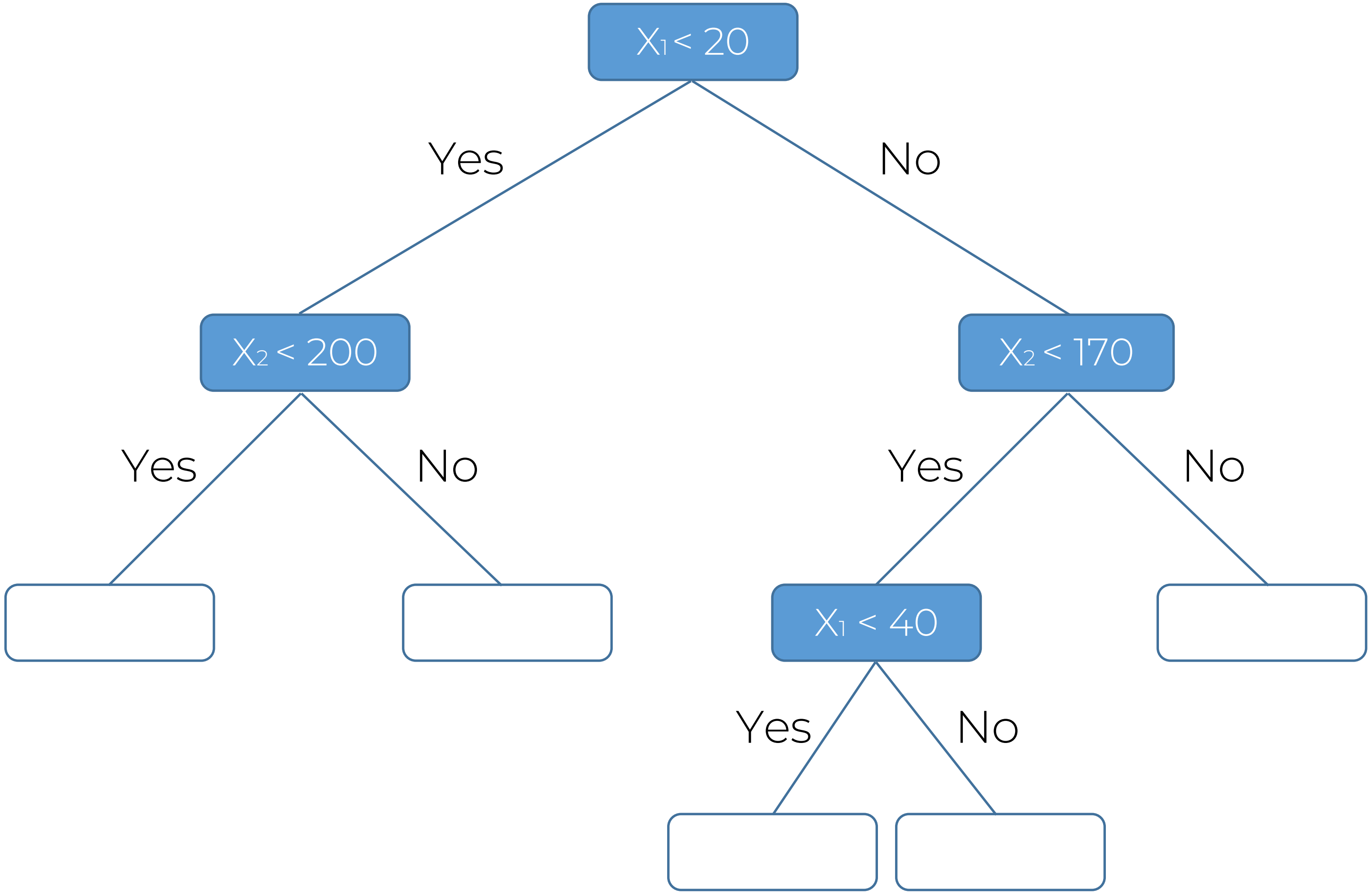


Decision Tree Intuition

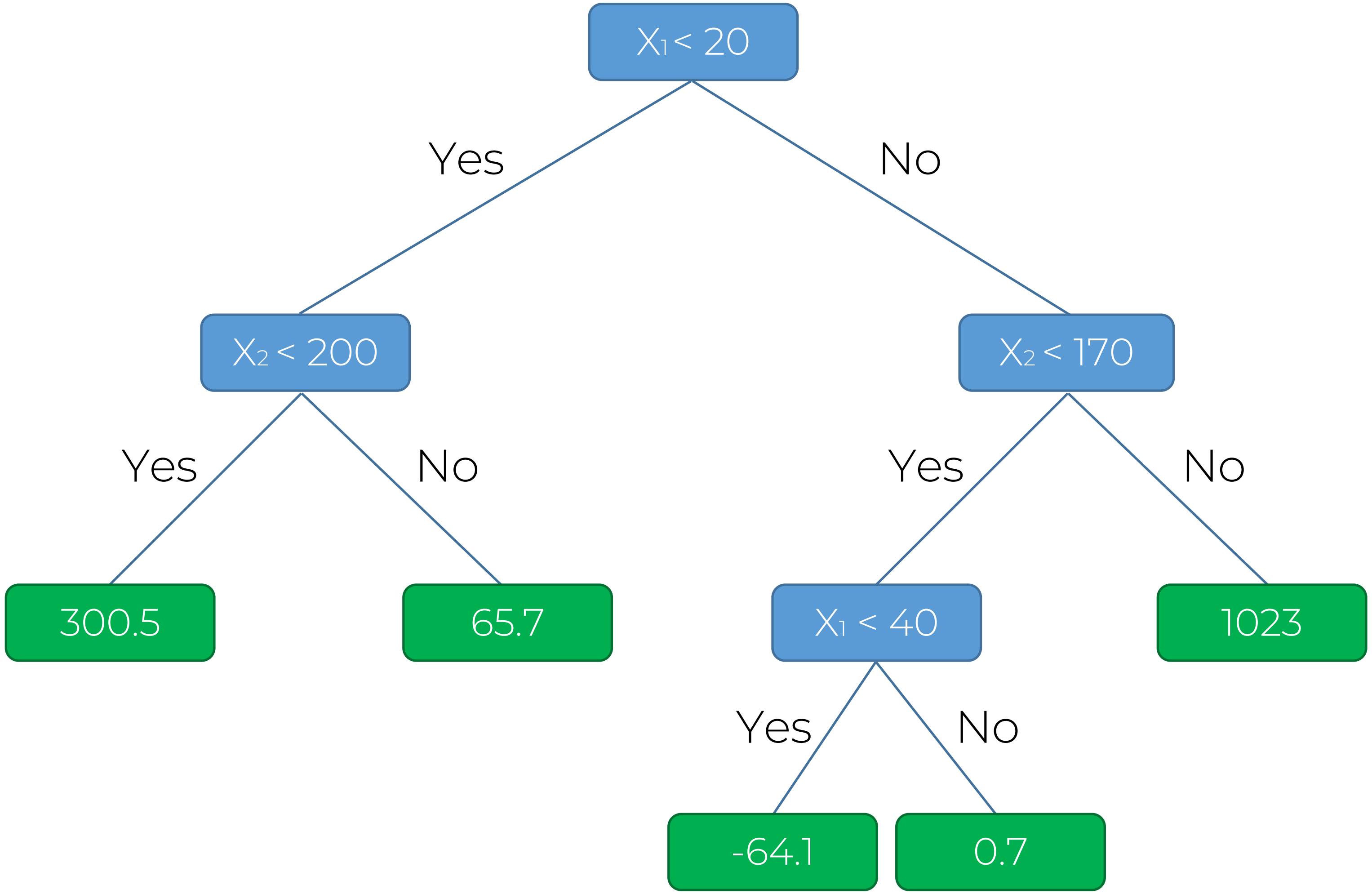




Decision Tree Intuition



Decision Tree 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.

↓

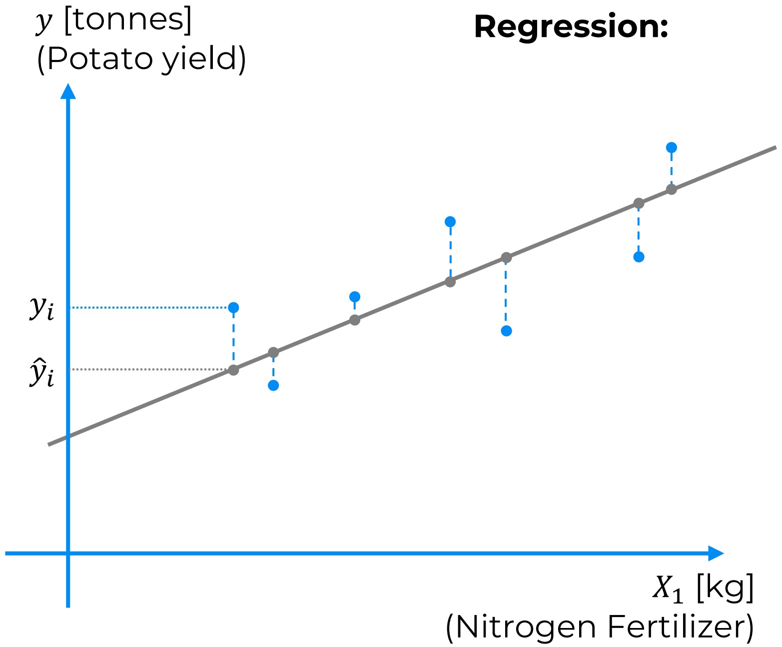
STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

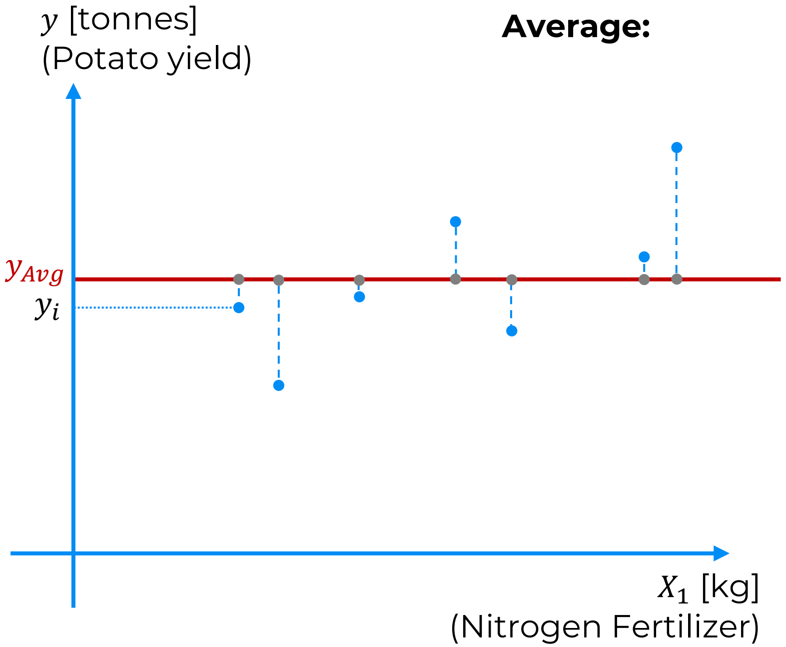
↓

STEP 4: For a new data point, make each one of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values.

R Squared

R Squared





$$SS_{res} = SUM(y_i - \hat{y}_i)^2$$

$$SS_{tot} = SUM(y_i - y_{avg})^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Rule of thumb (for our tutorials)*:

1.0 = Perfect fit (suspicious)
~0.9 = Very good
<0.7 = Not great
<0.4 = Terrible
<0 = Model makes no sense for this data

**This is highly dependent on the context*

Adjusted R Squared

Adjusted R Squared

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

R^2 – Goodness of fit
(greater is better)

Problem:

$$\hat{y} = b_0 + b_1X_1 + b_2X_2 \leftarrow \boxed{+\, b_3X_3}$$

$$SS_{res} = SUM(y_i - \hat{y}_i)^2$$

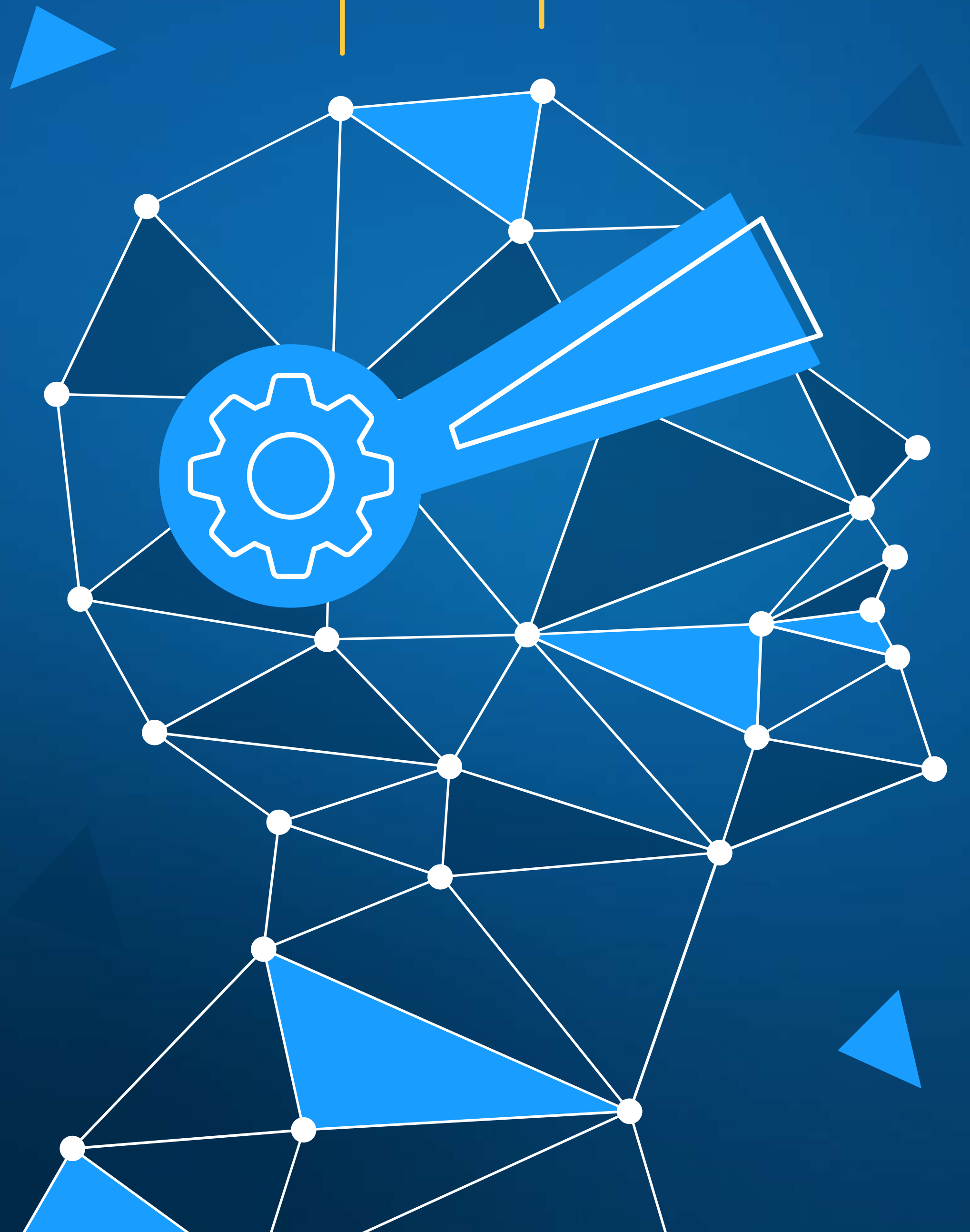
SS_{tot} doesn't change
SS_{res} will decrease or stay the same

(This is because of Ordinary Least Squares: SS_{res} -> Min)

Solution:

$$Adj\ R^2 = 1 - (1 - R^2) \times \frac{n-1}{n-k-1}$$

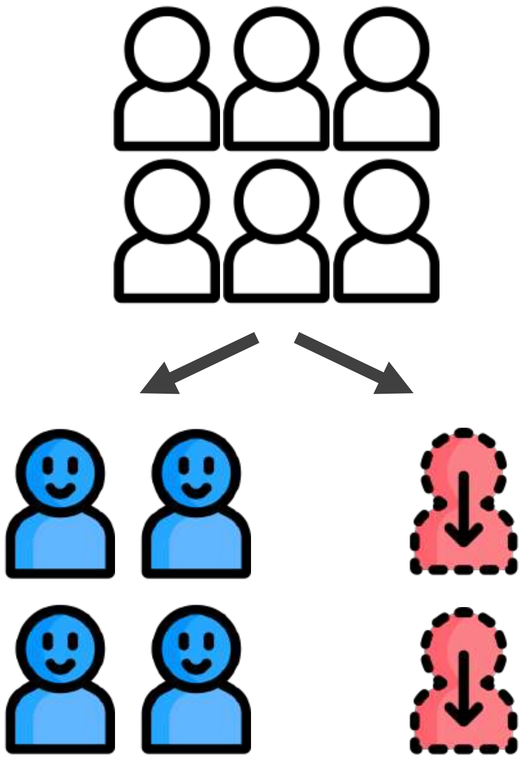
k – number of independent variables
n – sample size



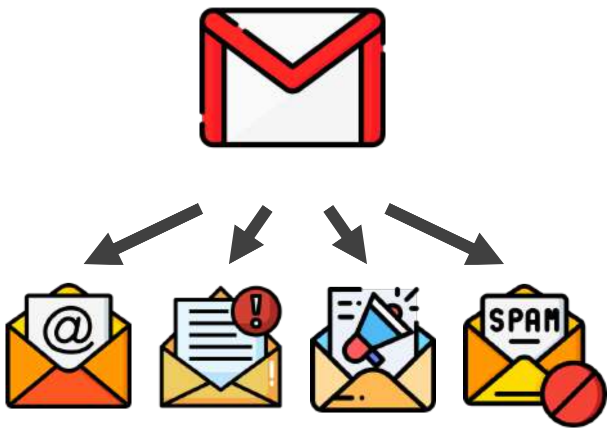
Classification

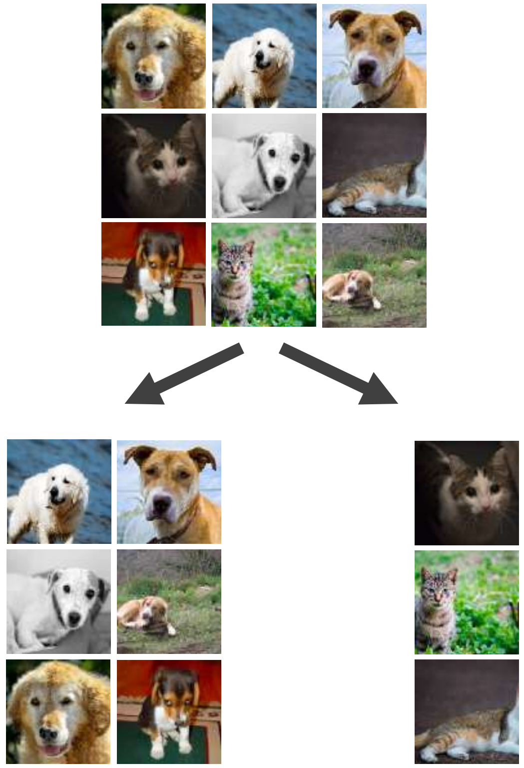
What is Classification?

Classification: a Machine Learning technique to identify the <u>category</u> of new observations based on training data.

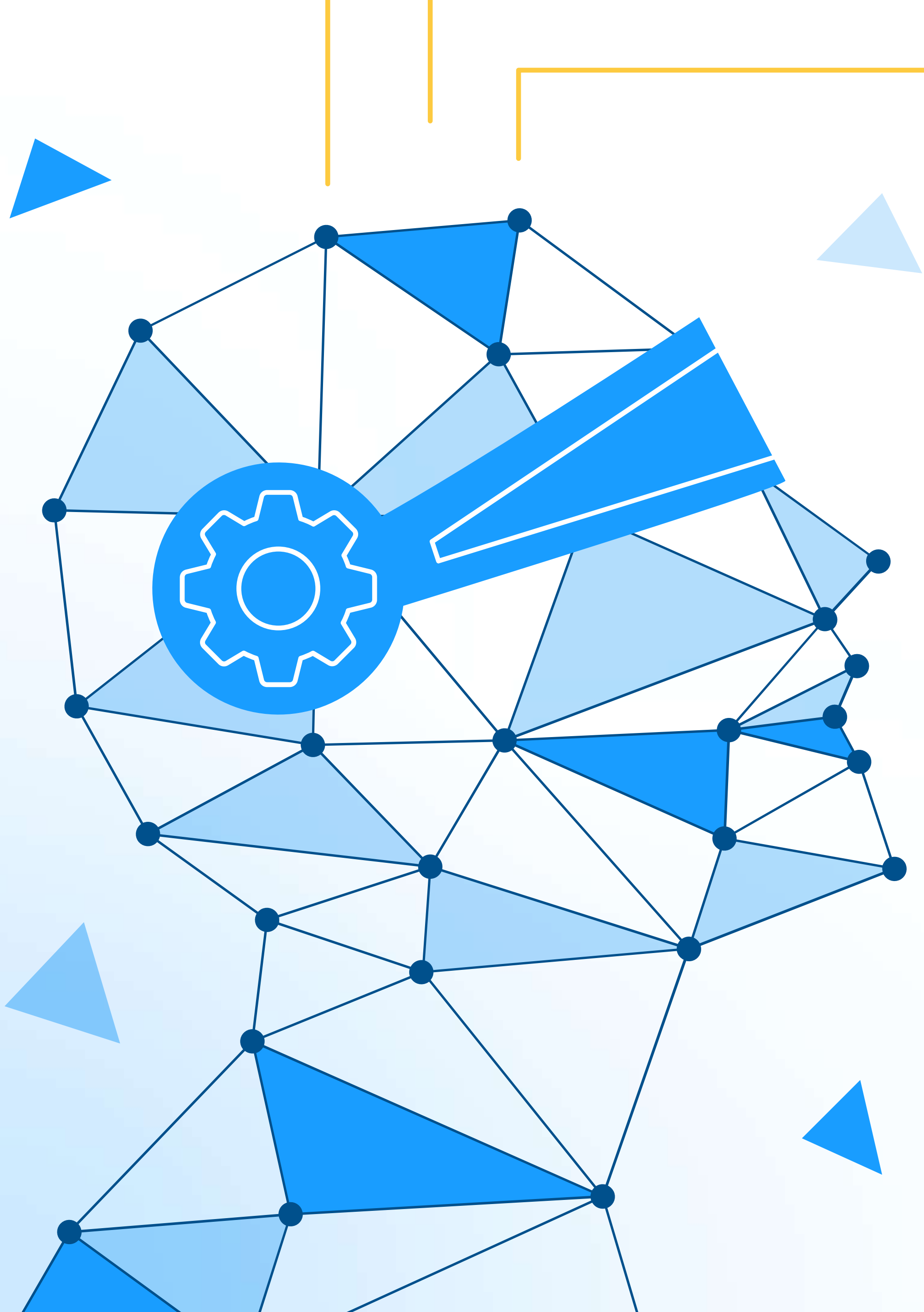


Likely to stay *Likely to leave*





Dogs *Cats*



Logistic Regression

Logistic Regression

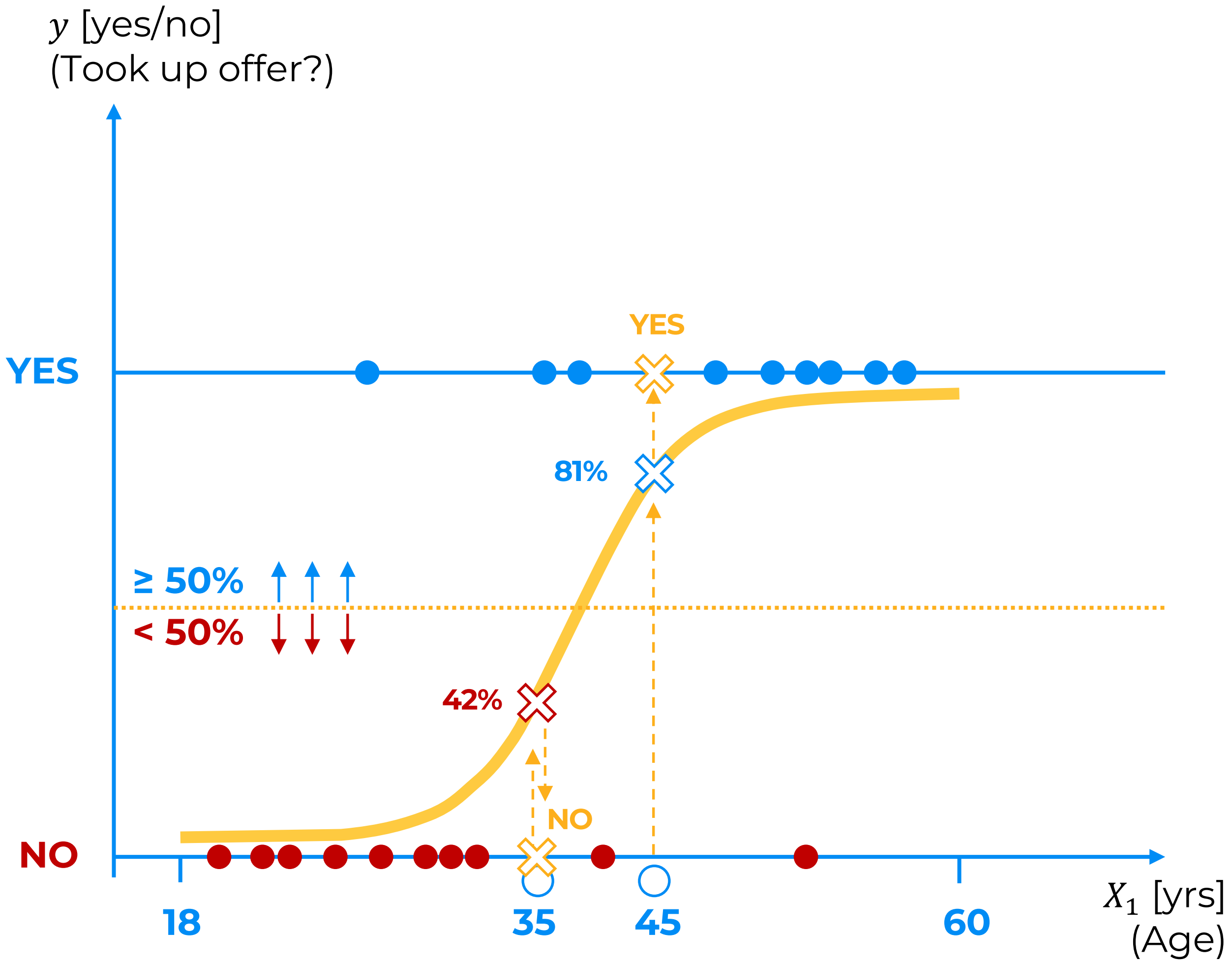
Logistic regression: predict a <u>categorical</u> dependent variable from a number of independent variables.

Will purchase health insurance: Yes / No

$\sim$

Age

$$\ln \frac{p}{1-p} = b_0 + b_1 X_1$$



Logistic Regression

$\sim$

Will purchase health insurance: Yes / No

Age

Income

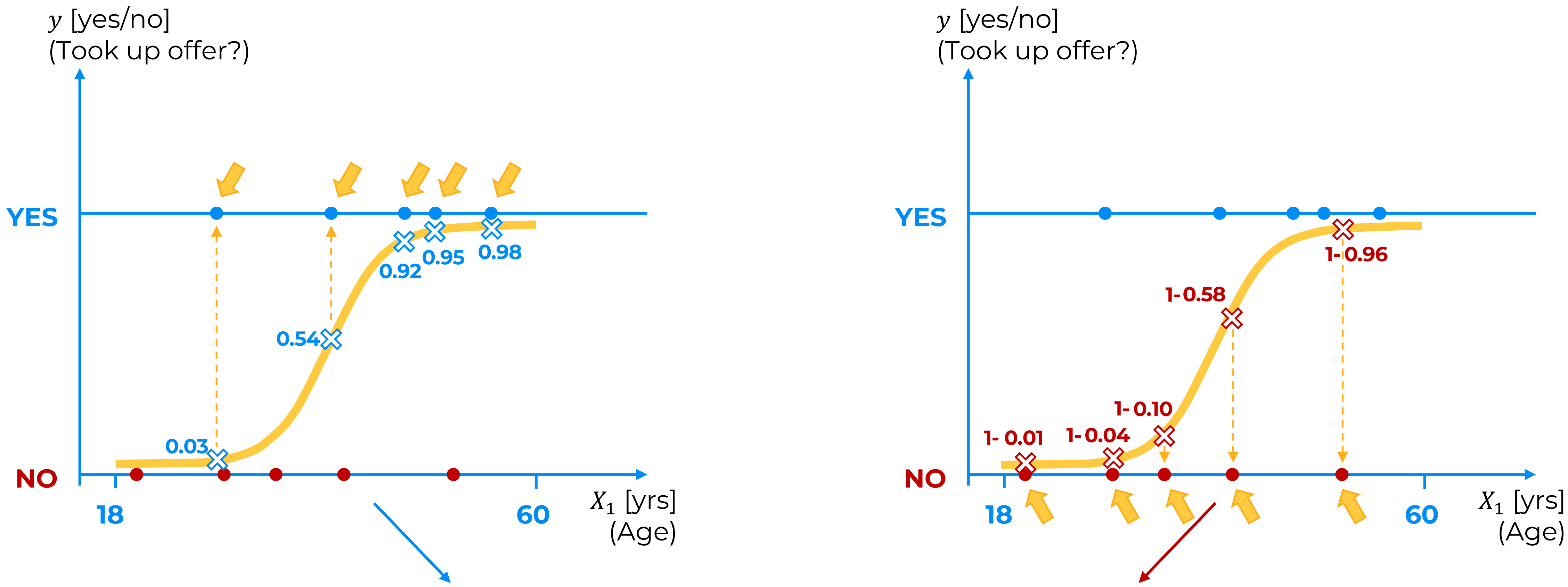
Level of Education

Family or Single

$$\ln \frac{p}{1-p} = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4$$

Maximum Likelihood

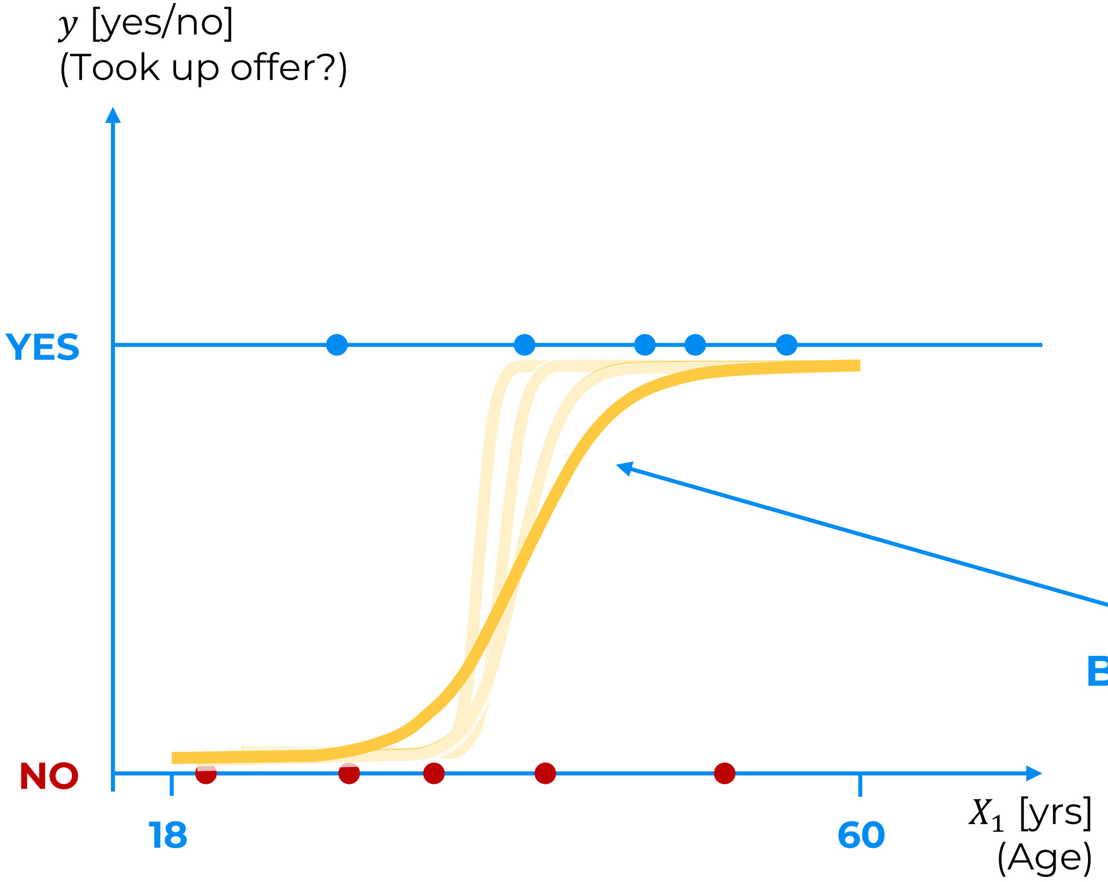
Maximum Likelihood



Likelihood = 0.03 x 0.54 x 0.92 x 0.95 x 0.98 x (1 – 0.01) x (1 – 0.04) x (1 – 0.10) x (1 – 0.58) x (1 – 0.96)

Likelihood = **0.00019939**

Maximum Likelihood



Likelihood = 0.00007418

Likelihood = 0.00012845

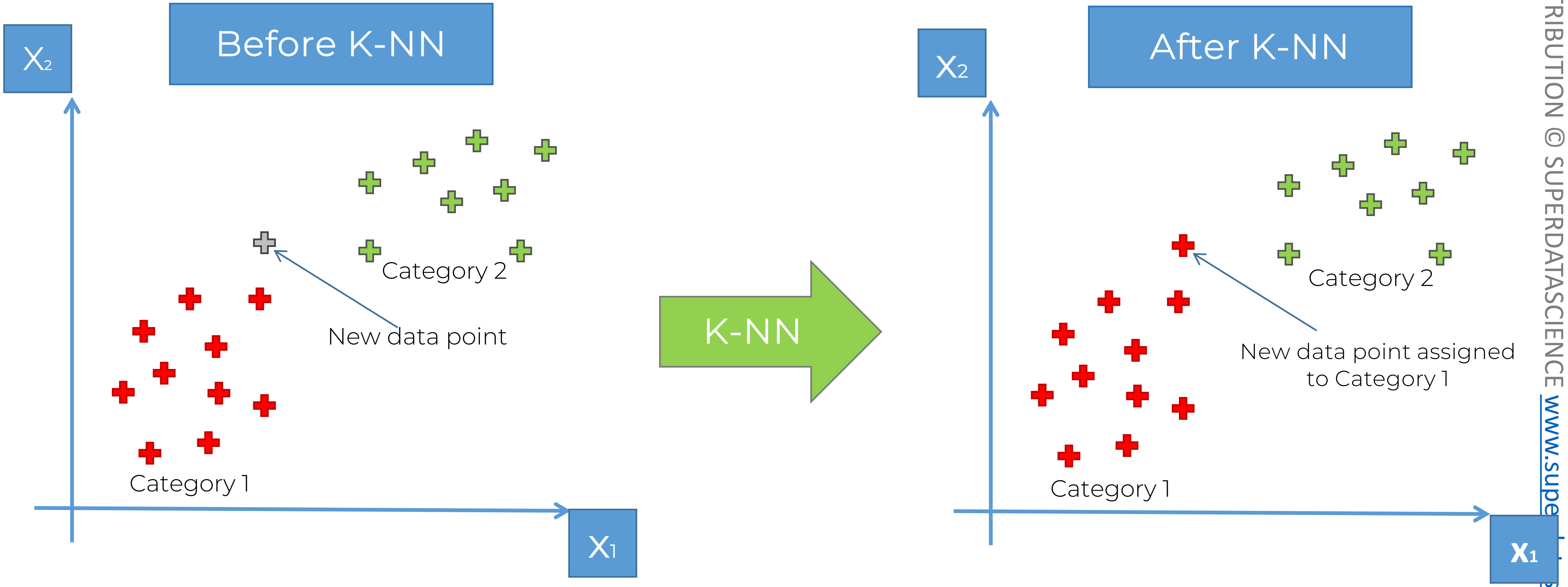
Likelihood = 0.00016553

Likelihood = 0.00019939

Best Curve <= Maximum Likelihood

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

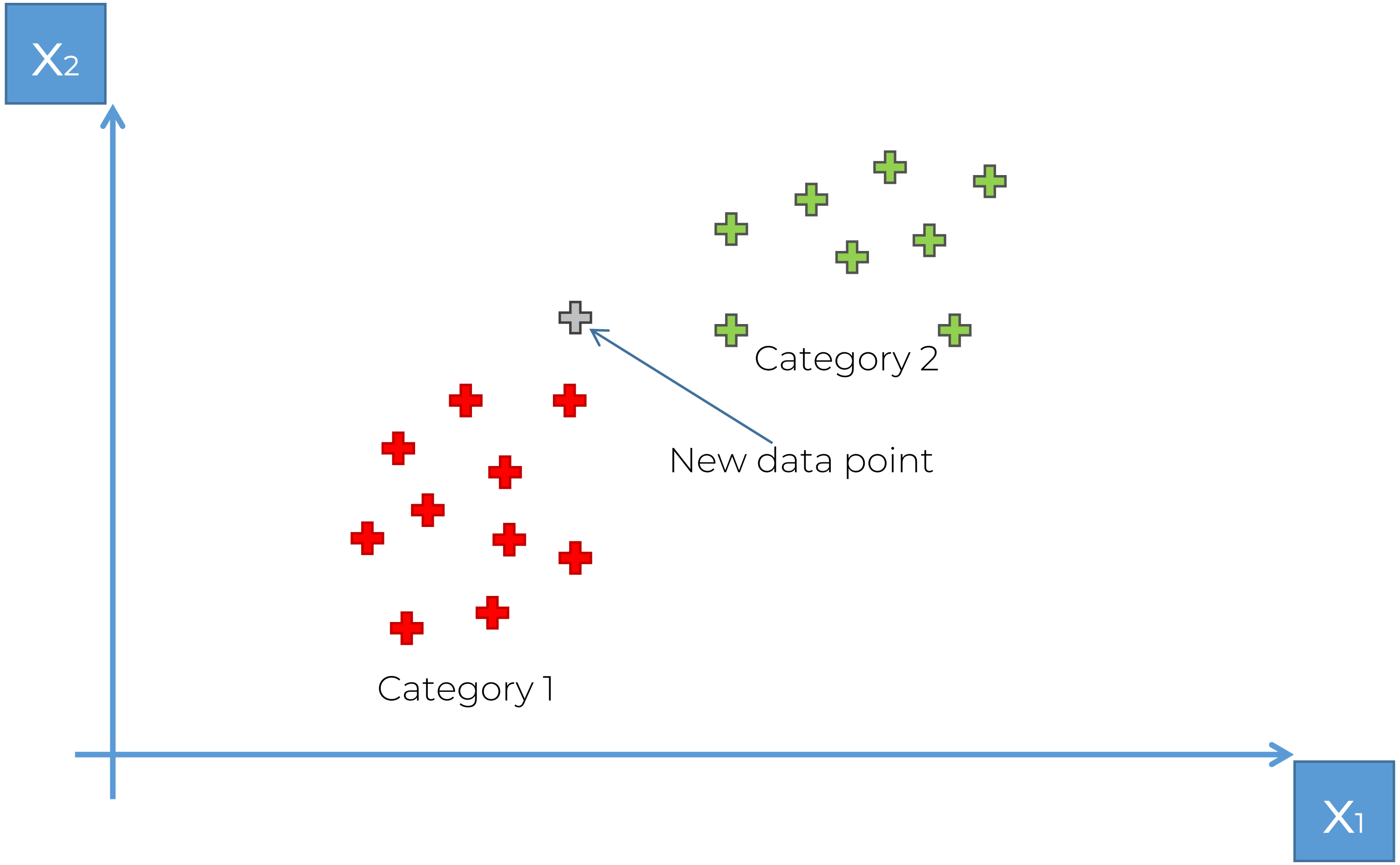
STEP 3: Among these K neighbors, count the number of data points in each category

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

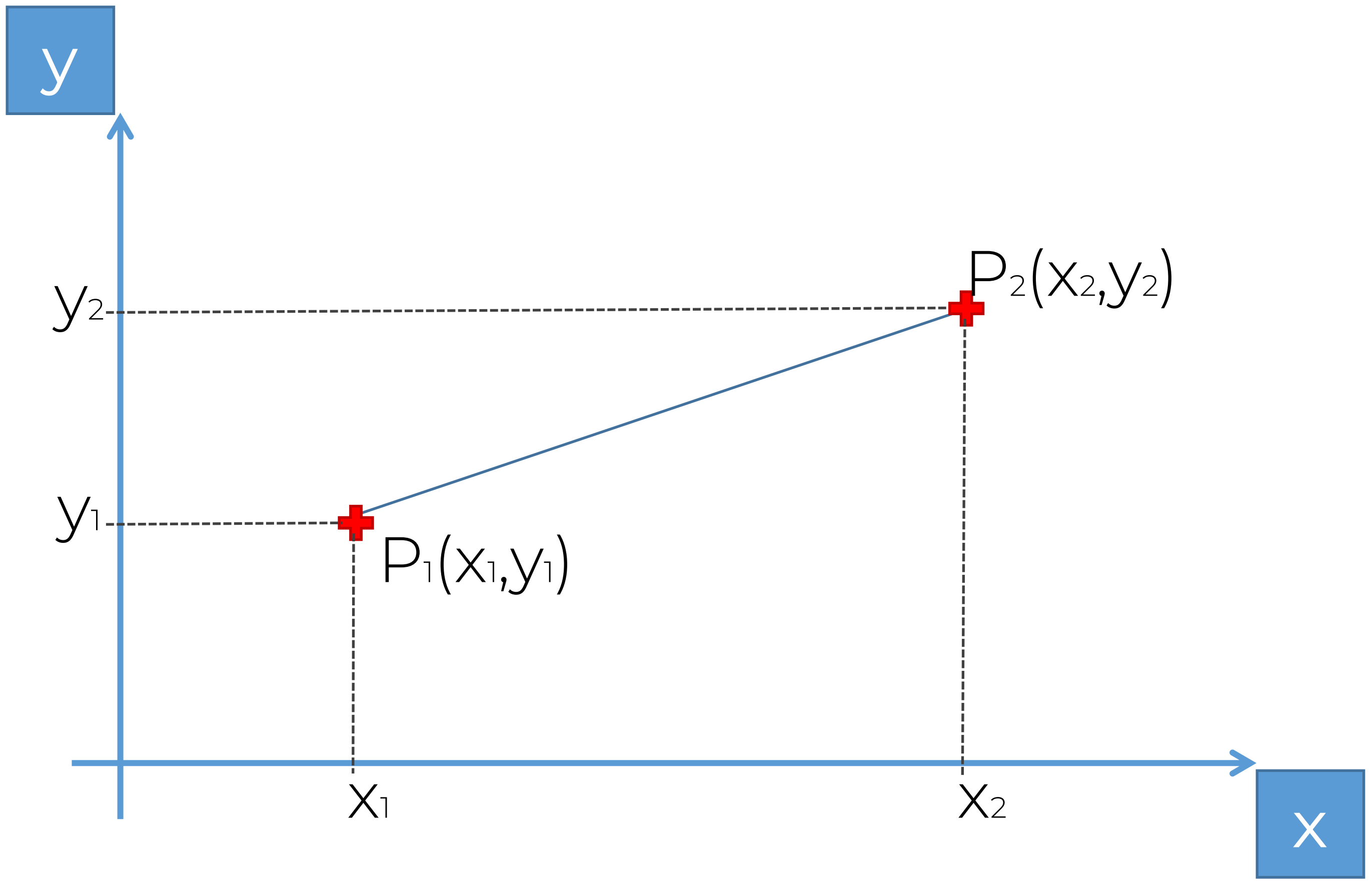
Your Model is Ready

K-NN algorithm

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

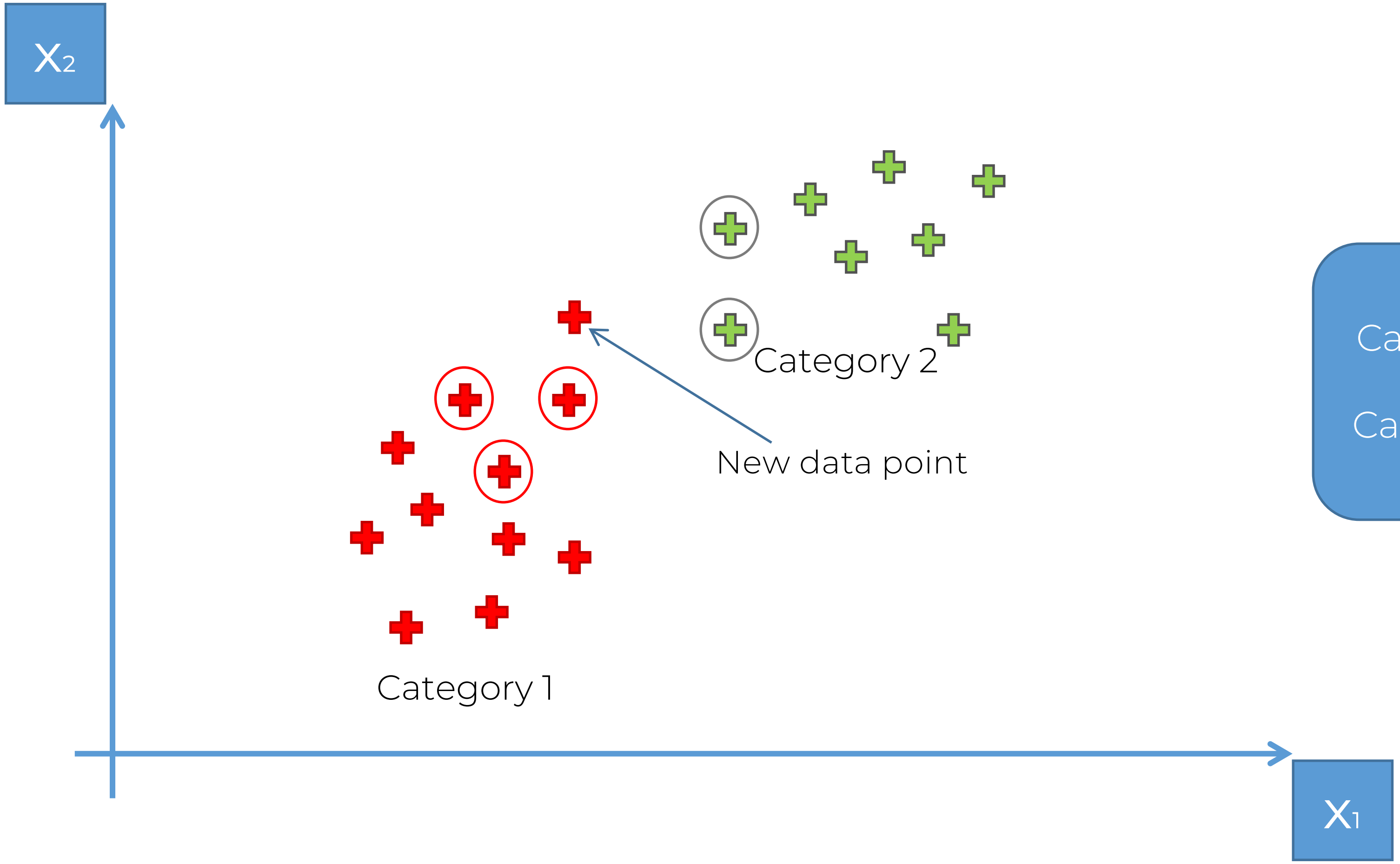


Euclidean Distance



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

K-NN algorithm

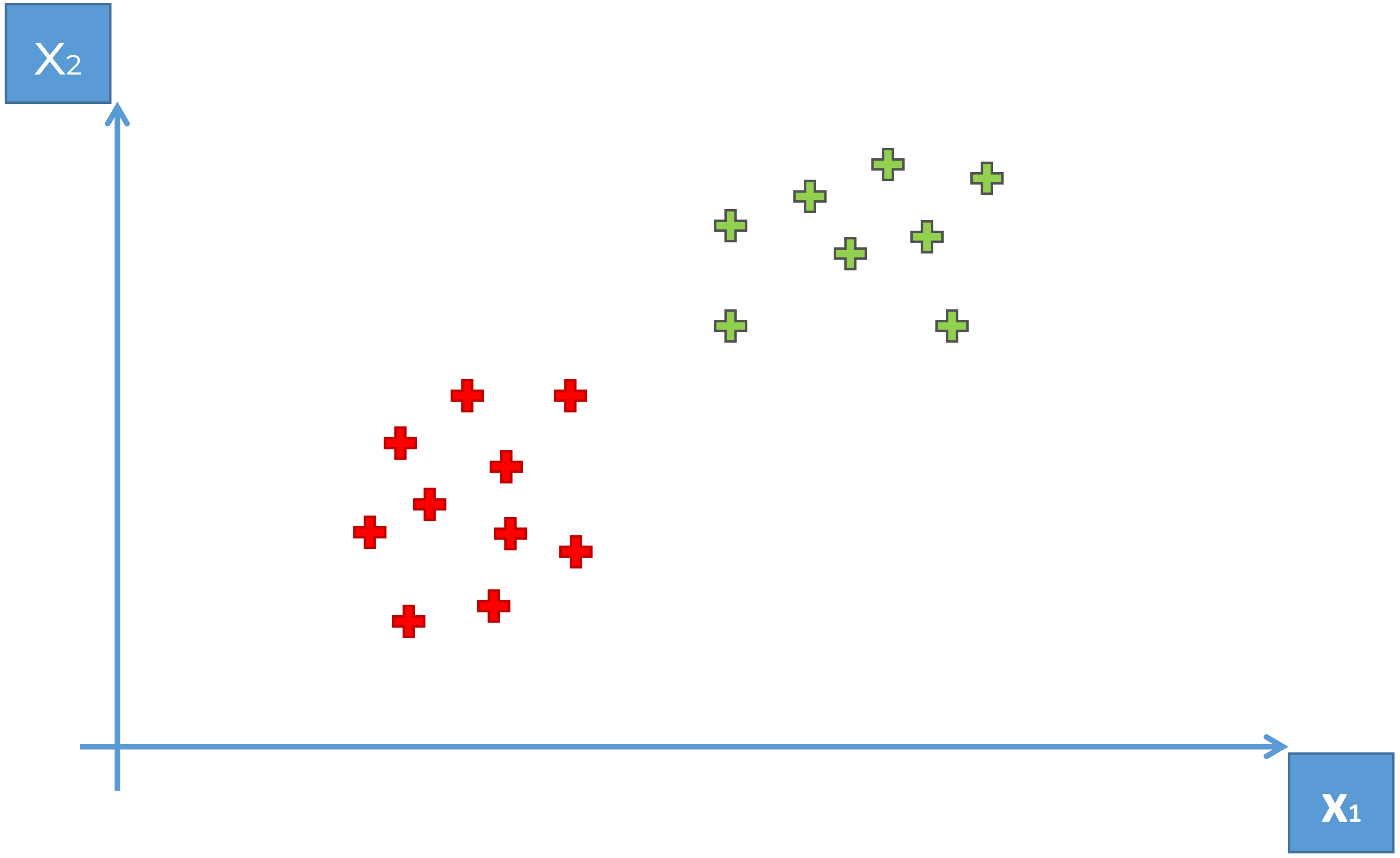


Category 1: 3 neighbors

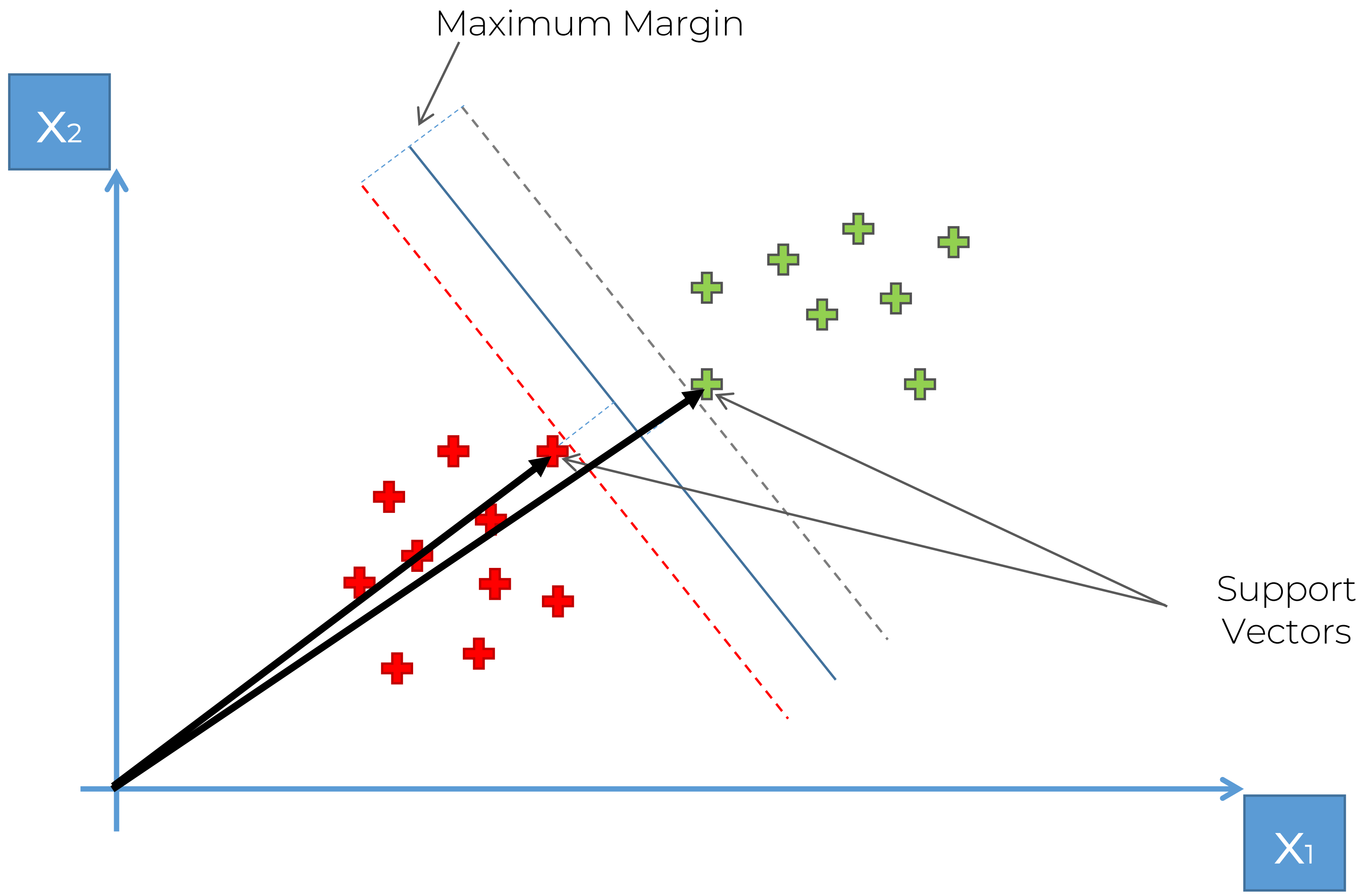
Category 2: 2 neighbors

SVM Intuition

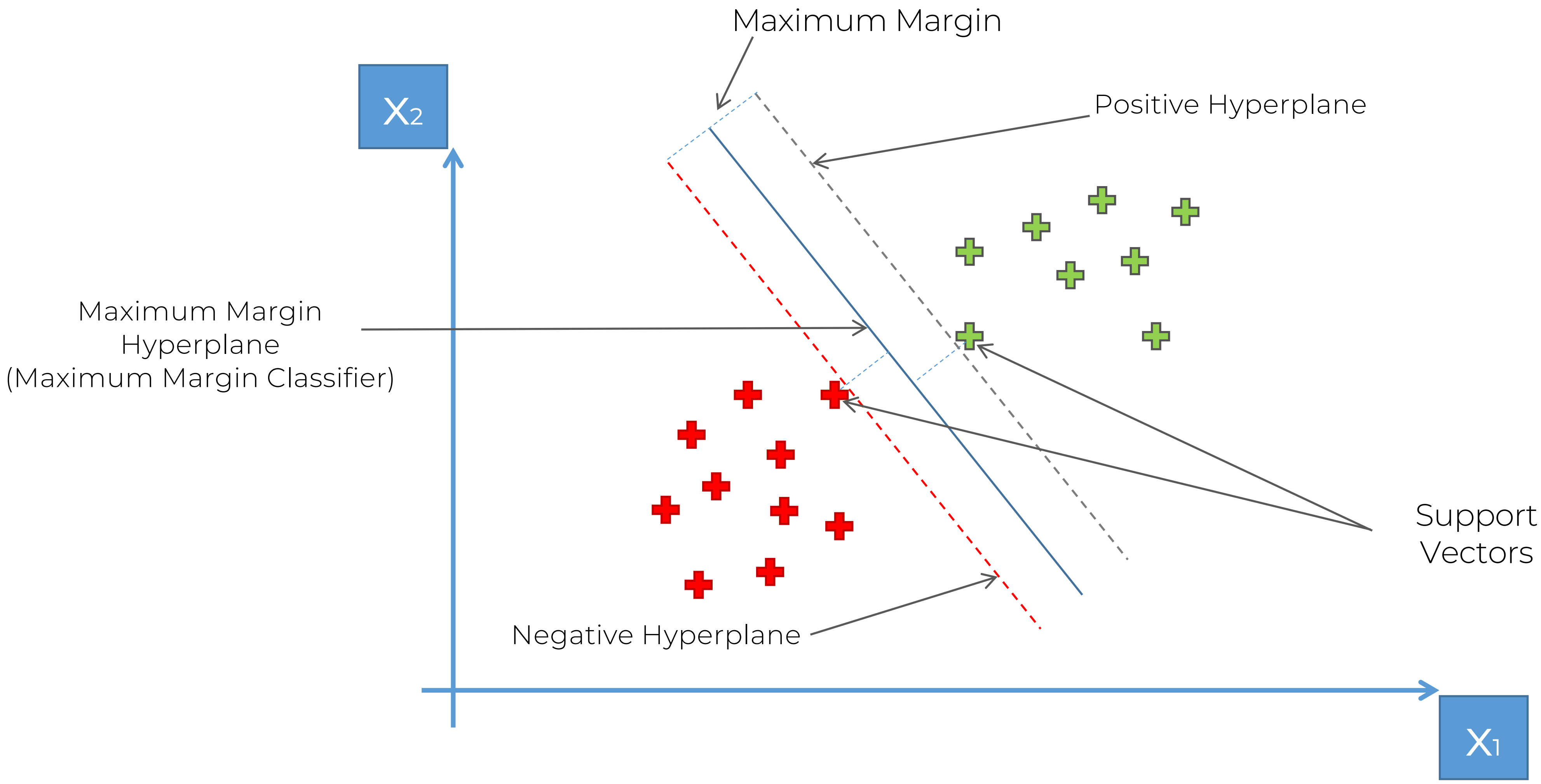
How to separate these points ?



Support Vectors

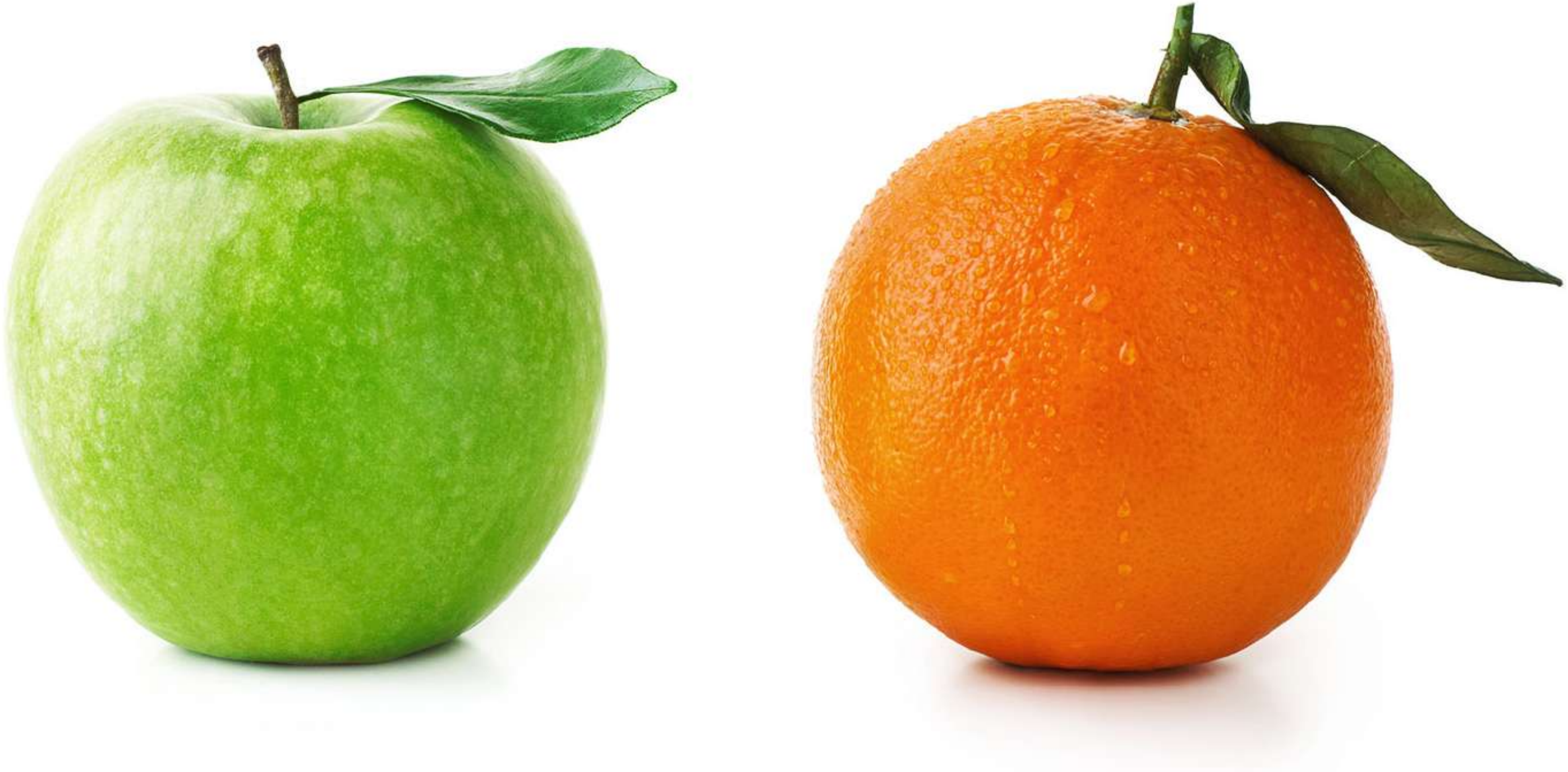


Hyperplanes

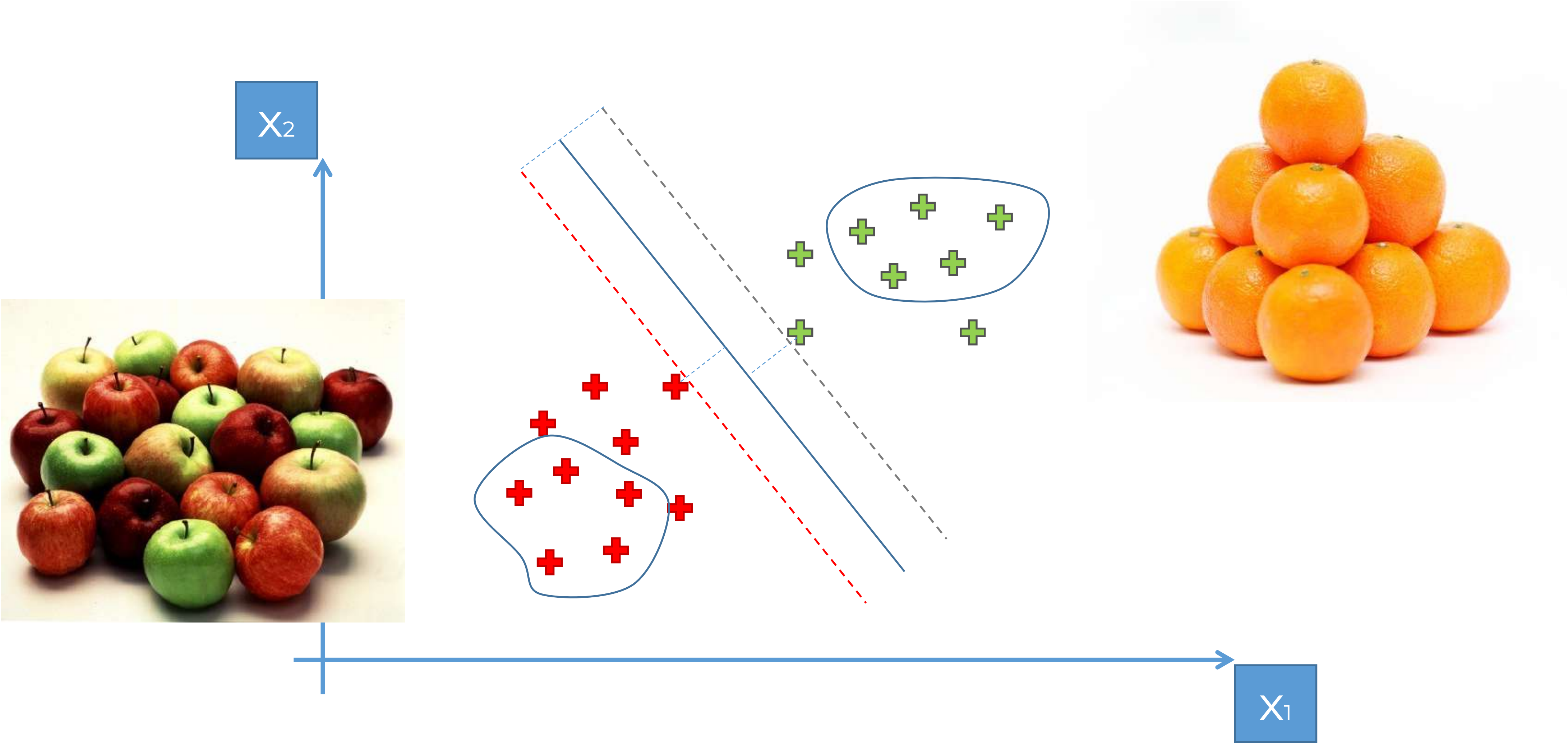


What's So Special About SVMs?

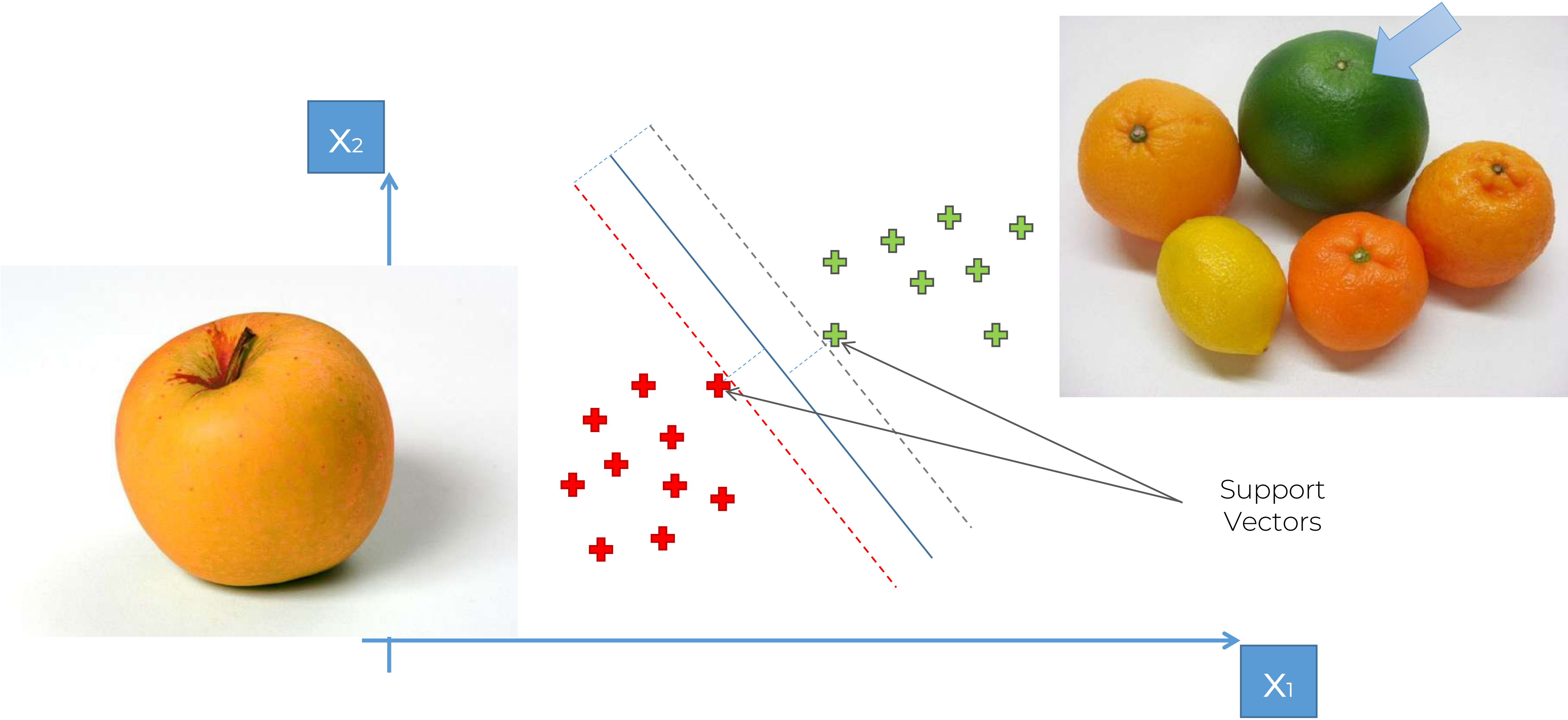
What's So Special About SVMs?



What's So Special About SVMs?



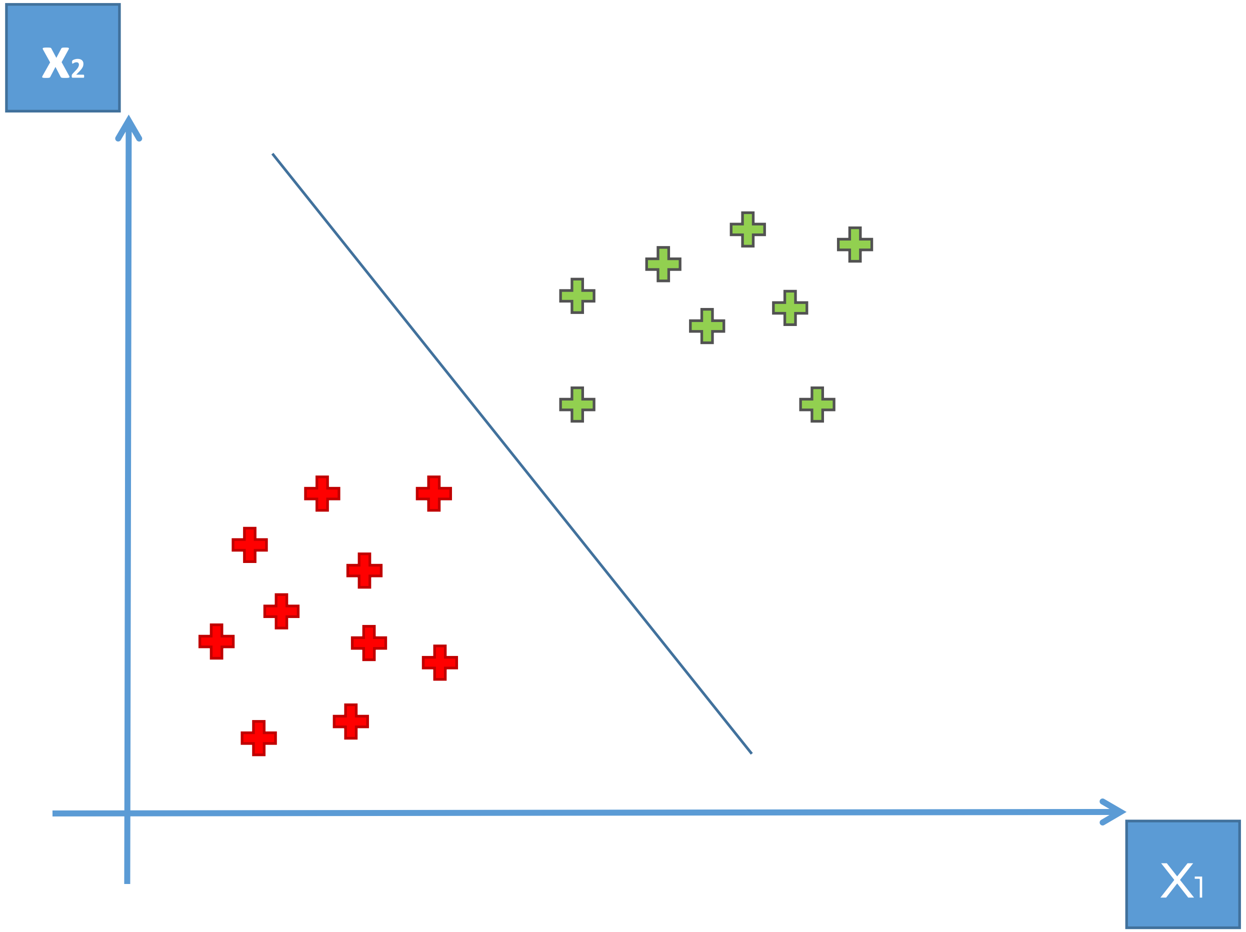
What's So Special About SVMs?



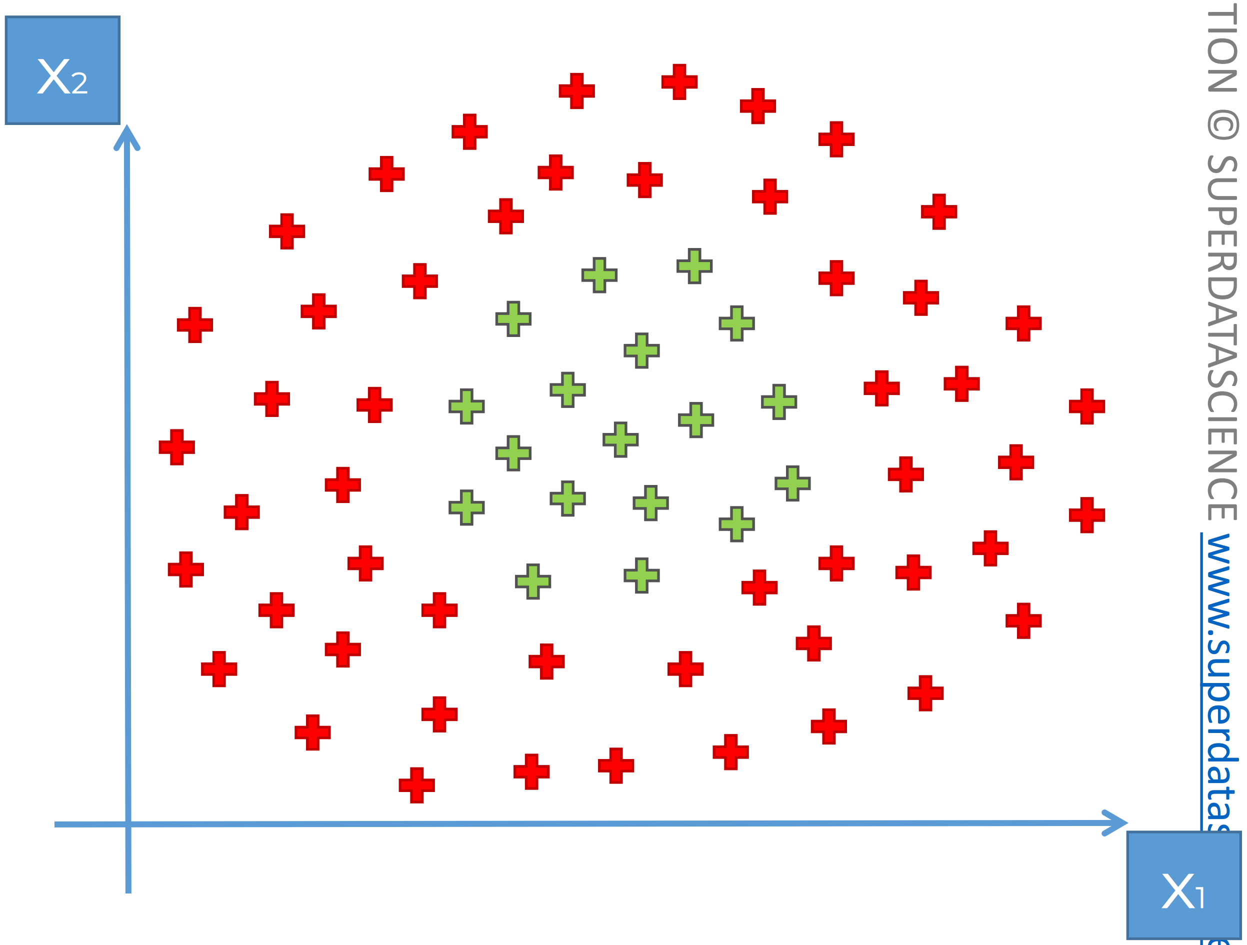
Kernel SVM Intuition

Linear Separability

Linearly Separable

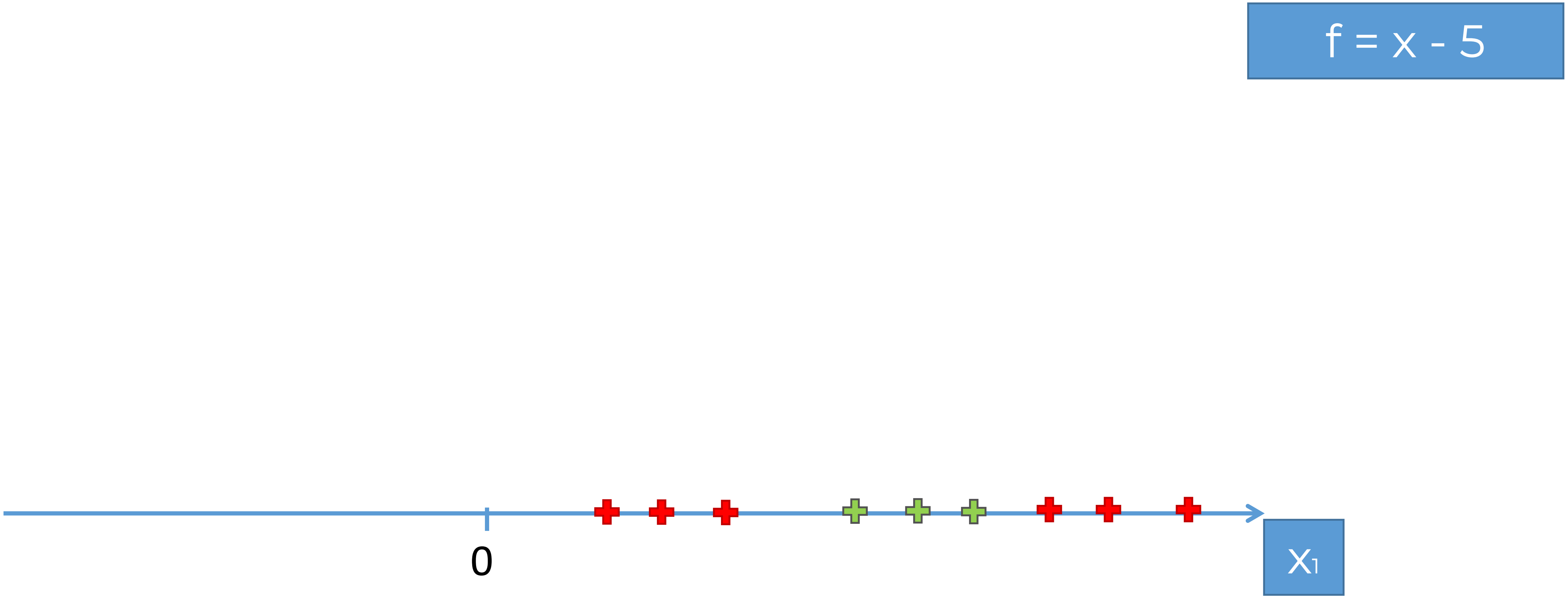


Not Linearly Separable

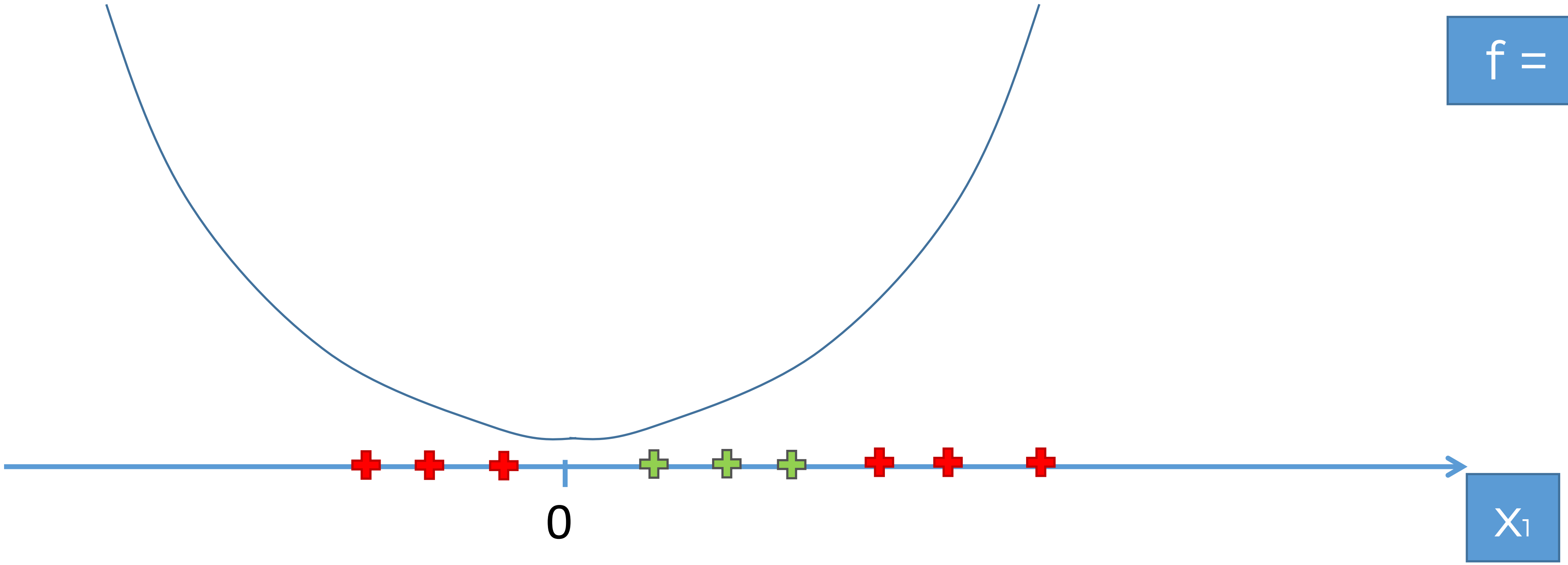


A Higher-Dimensional Space

Mapping to a Higher Dimension



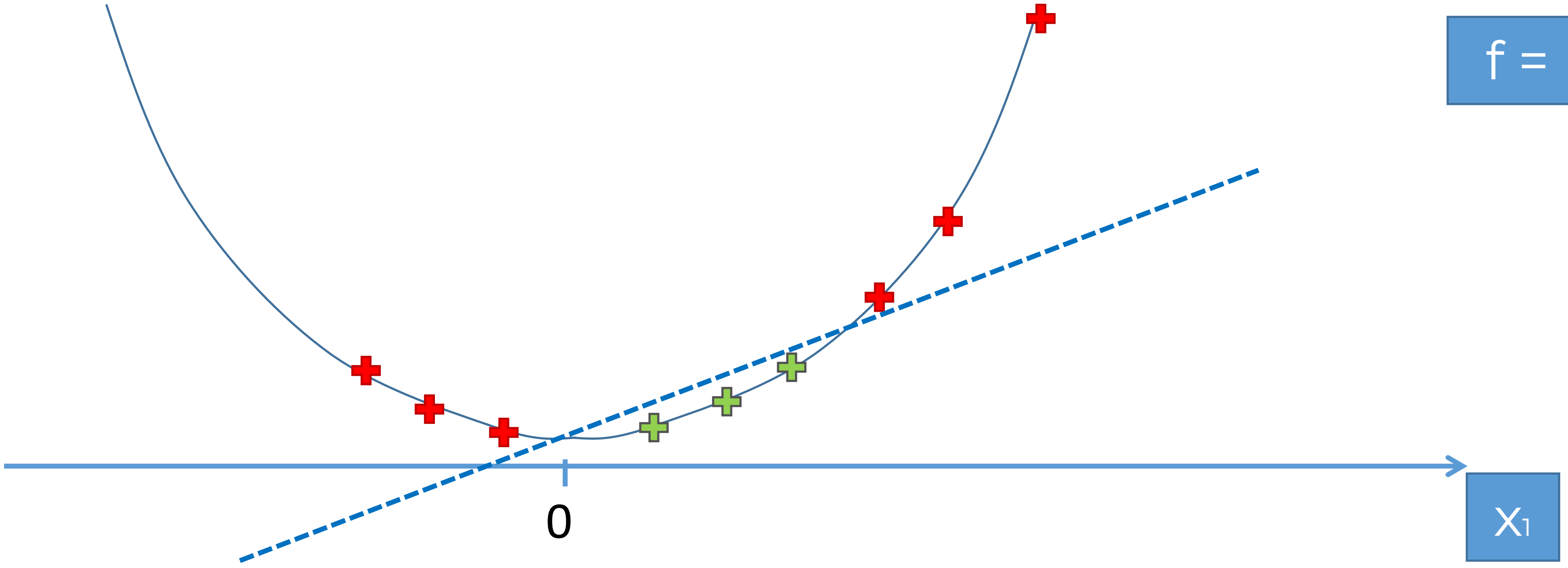
Mapping to a Higher Dimension



f = x - 5

f = (x – 5)^2

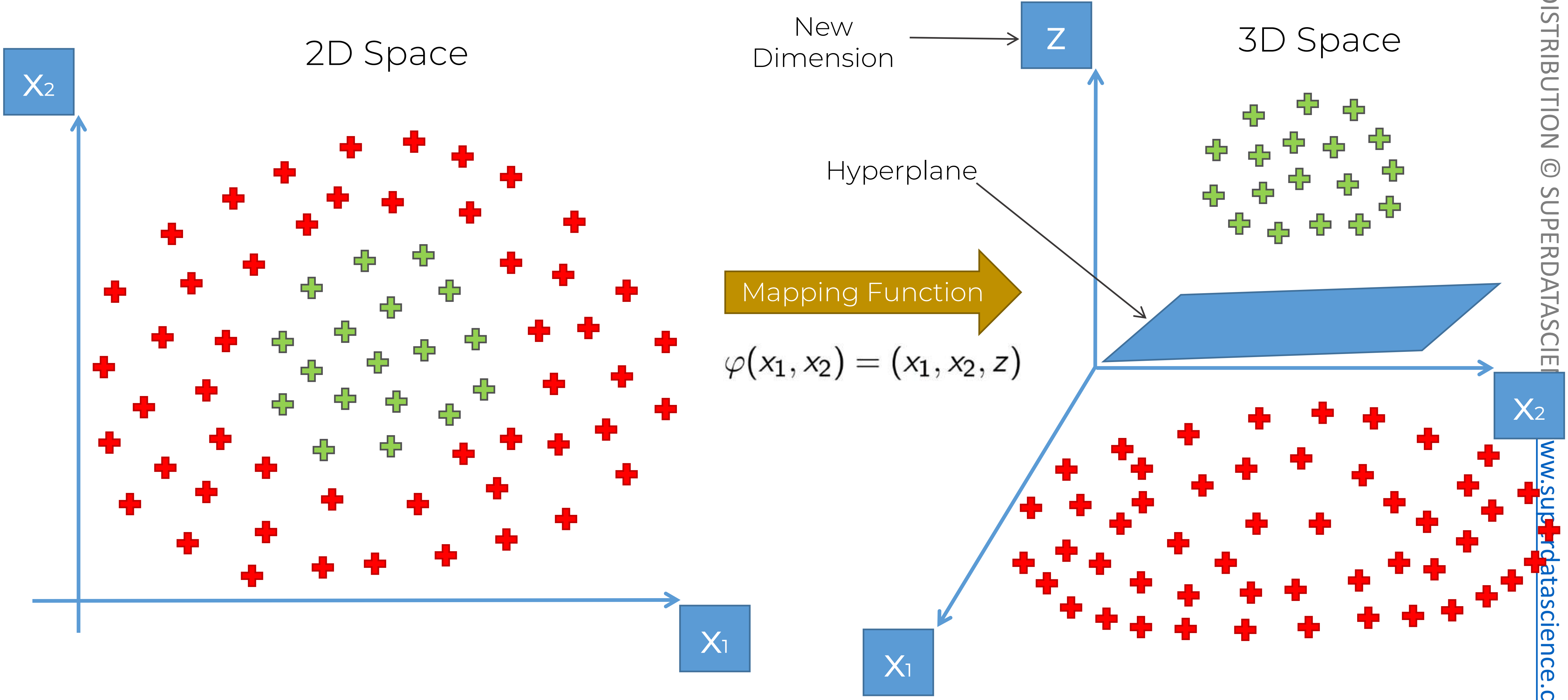
Mapping to a Higher Dimension



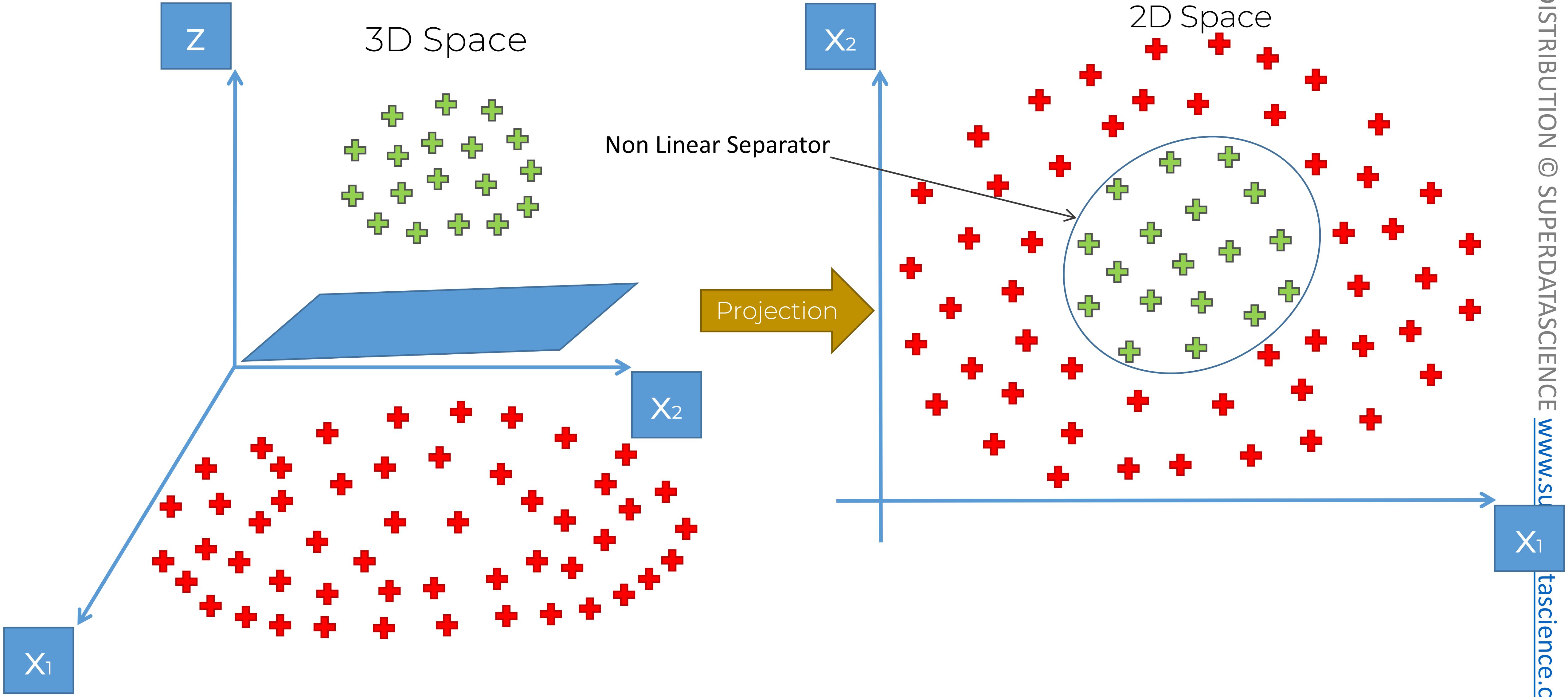
$f = x - 5$

$f = (x - 5)^2$

Mapping to a Higher Dimension



Projecting back to 2D Space



But there is a catch...

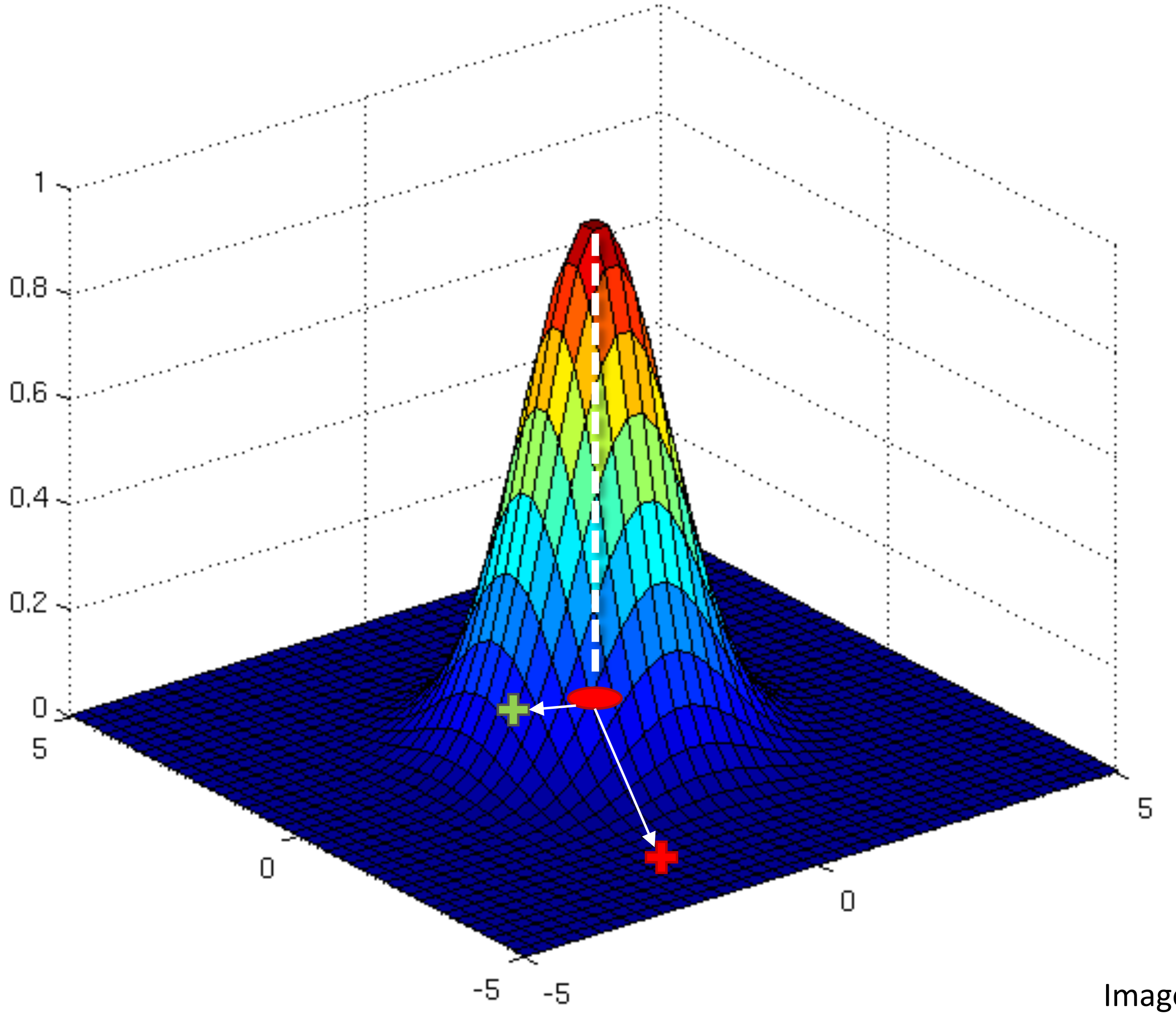
Mapping to a Higher Dimensional Space
can be highly compute-intensive

The Kernel Trick

The Gaussian RBF Kernel

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

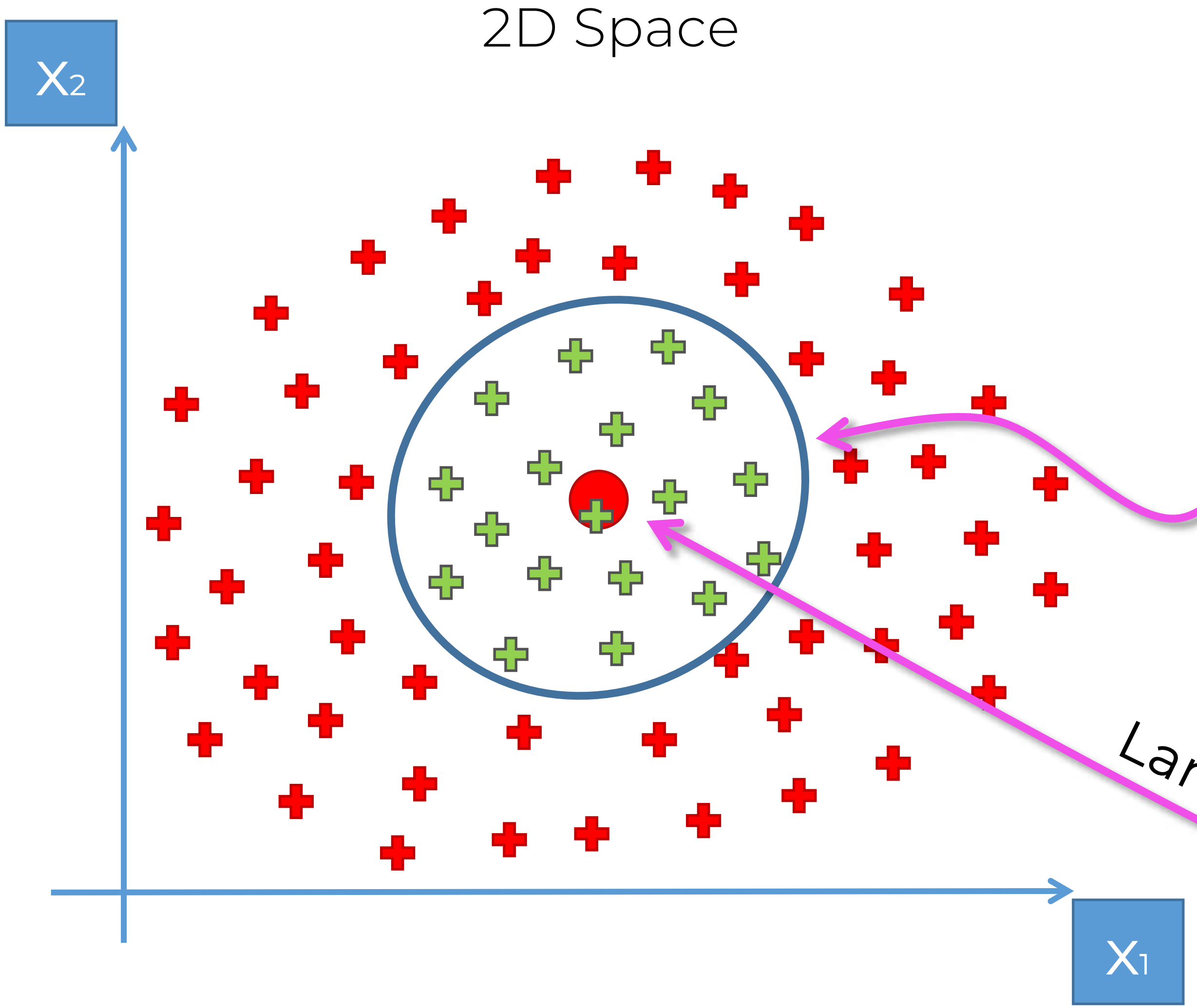
The Gaussian RBF Kernel



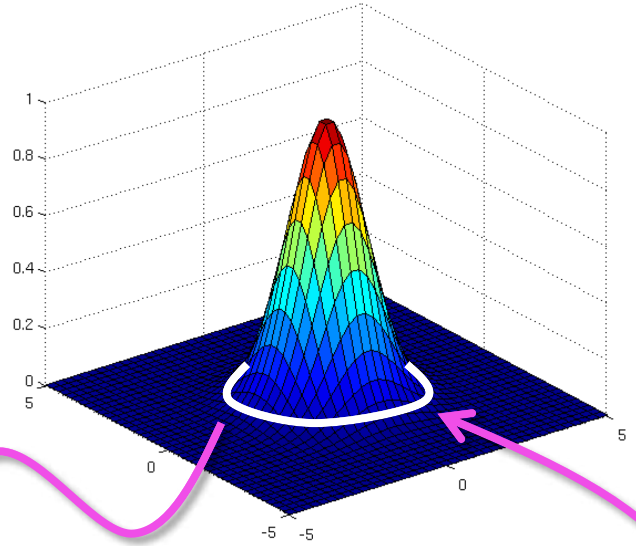
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

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

The Gaussian RBF Kernel

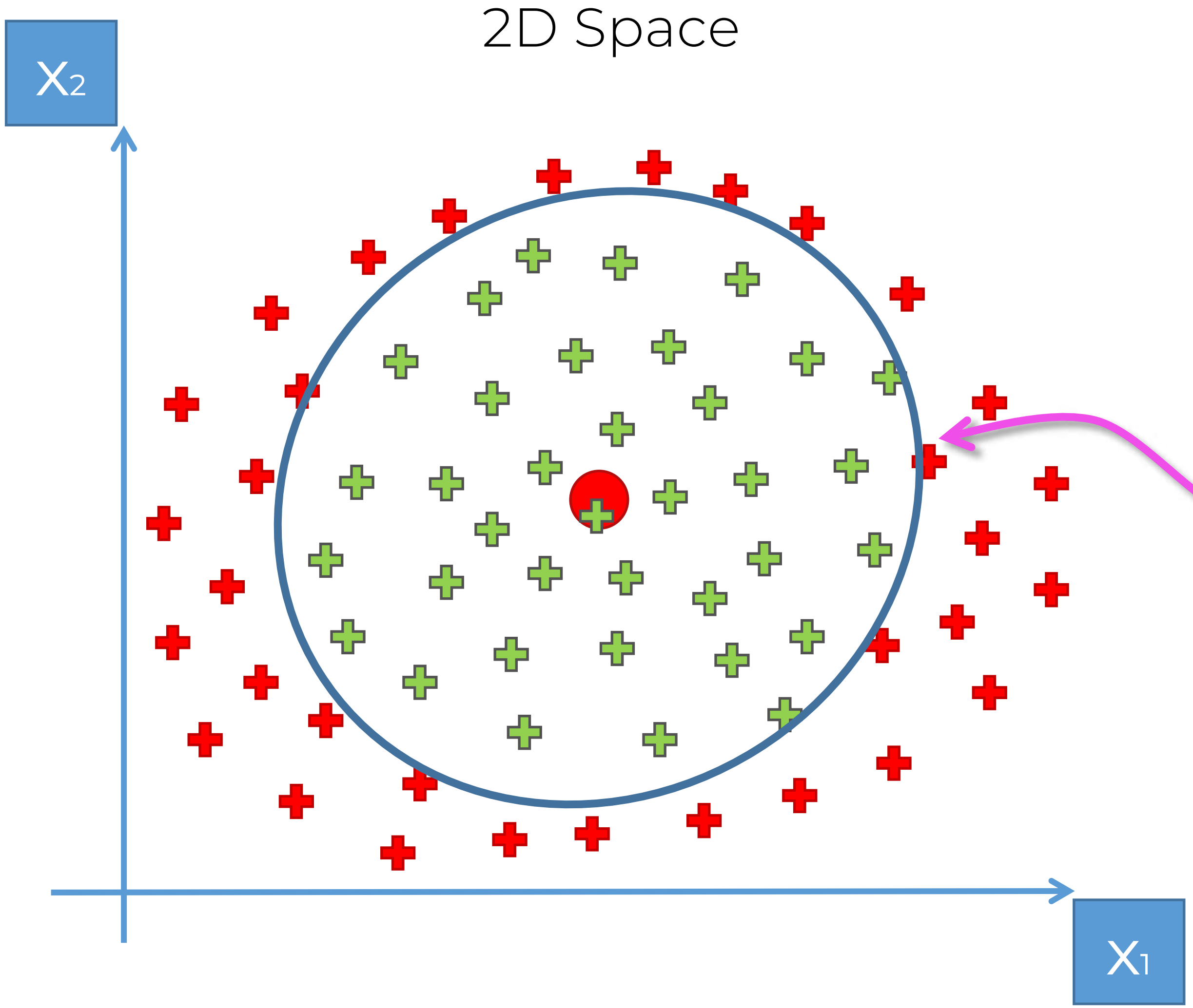


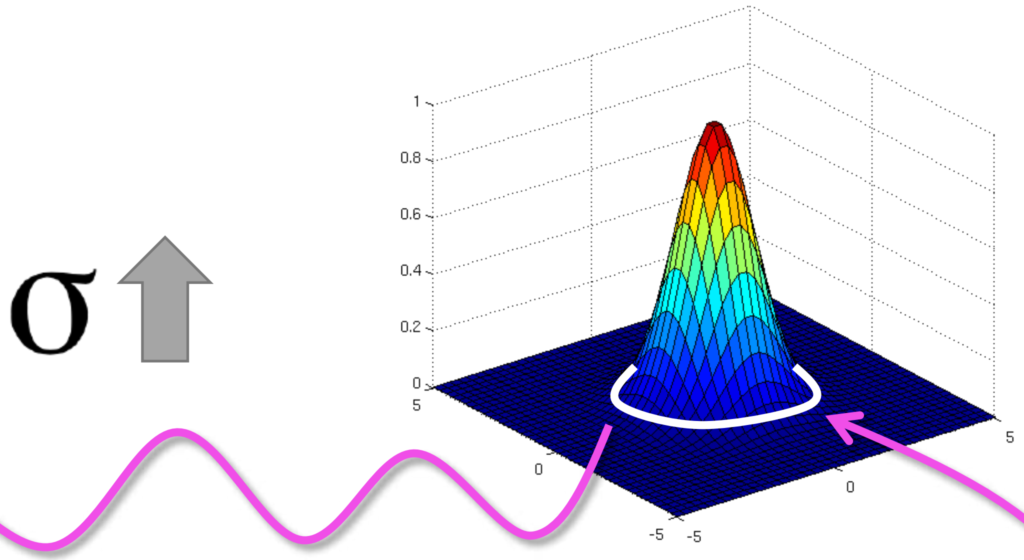
Landmark



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

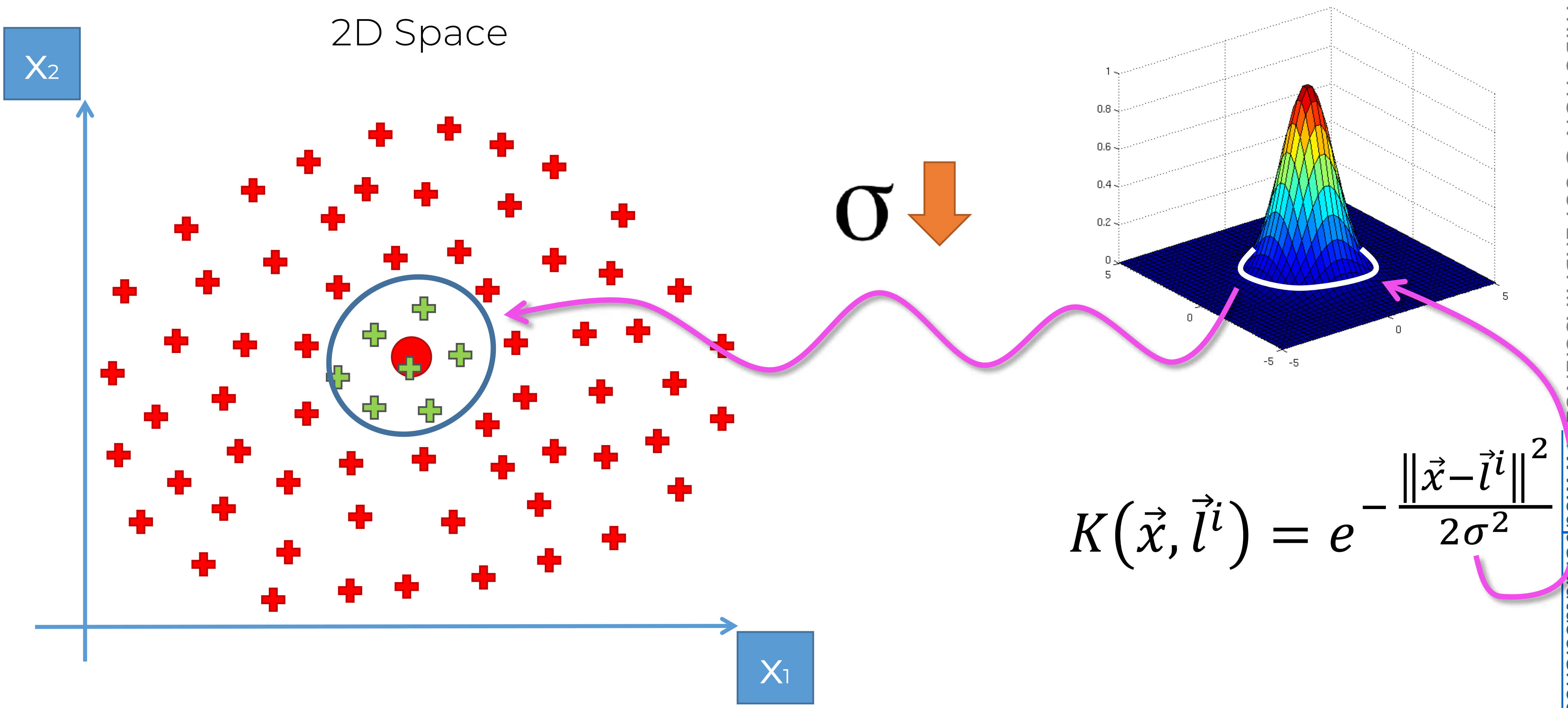
The Gaussian RBF Kernel





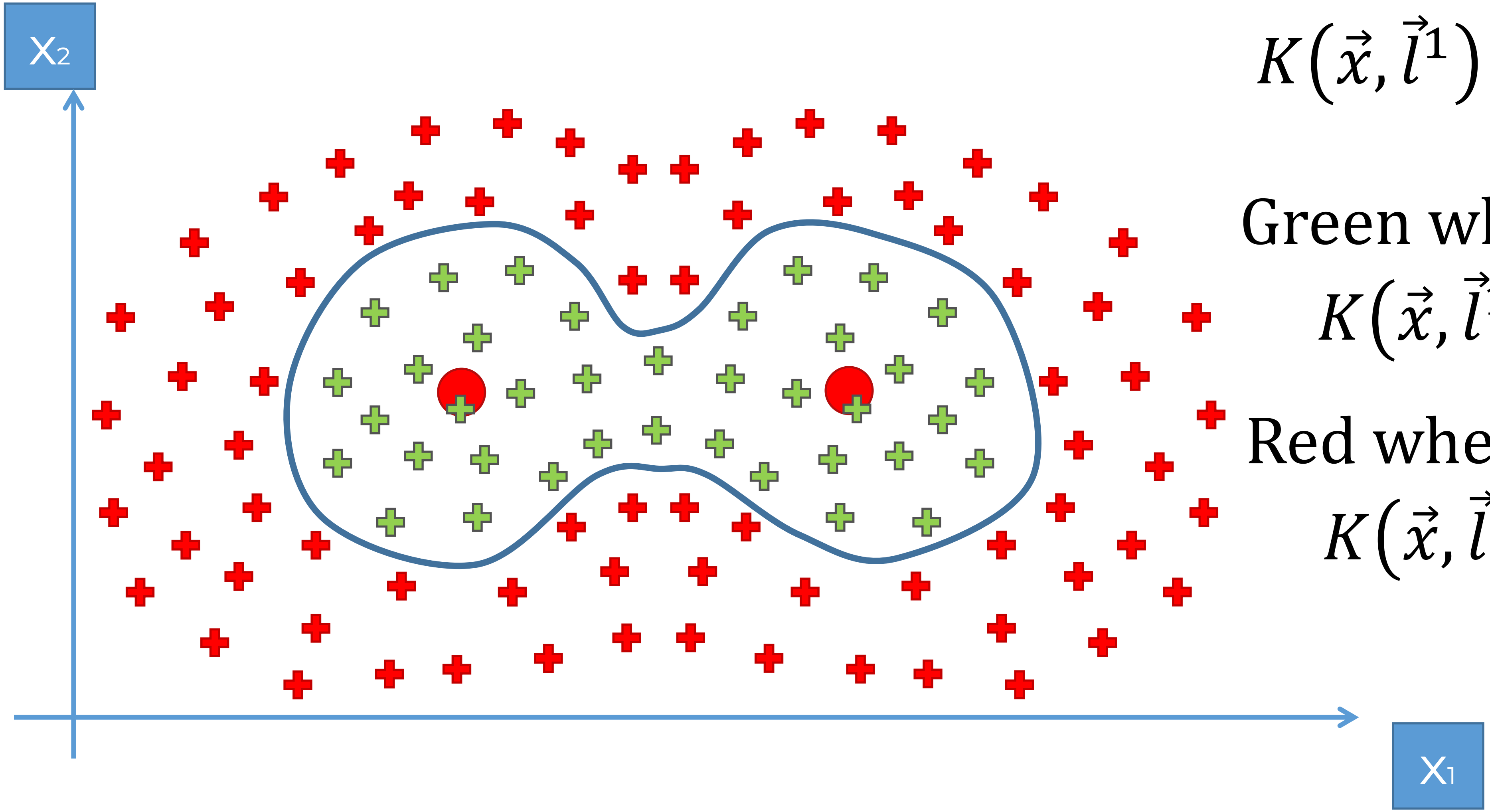
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

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



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

(Simplified Formula)

Green when:

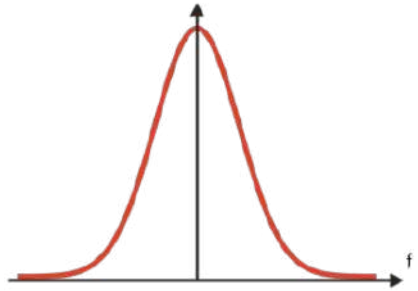
$$K\left(\vec{x},\vec{l}^{1}\right)+K\left(\vec{x},\vec{l}^{2}\right) >$$

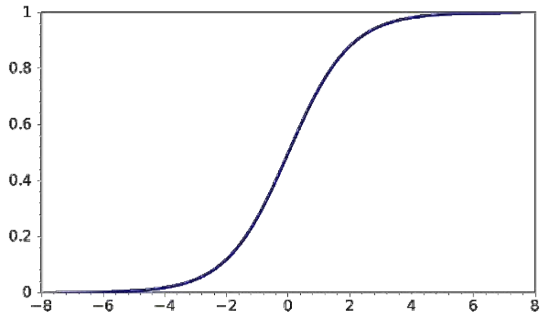
Red when:

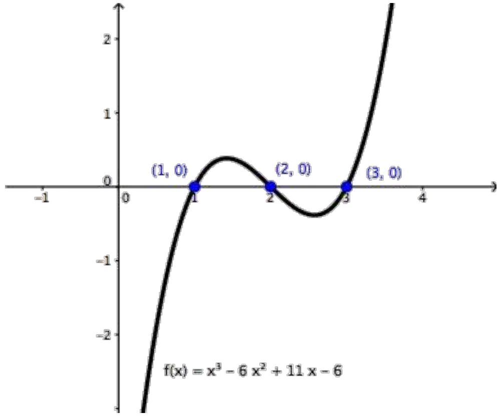
$$K\left(\vec{x},\vec{l}^{1}\right)+K\left(\vec{x},\vec{l}^{2}\right) =$$

Types of Kernel Functions

Types of Kernel Functions







Gaussian RBF Kernel

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

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

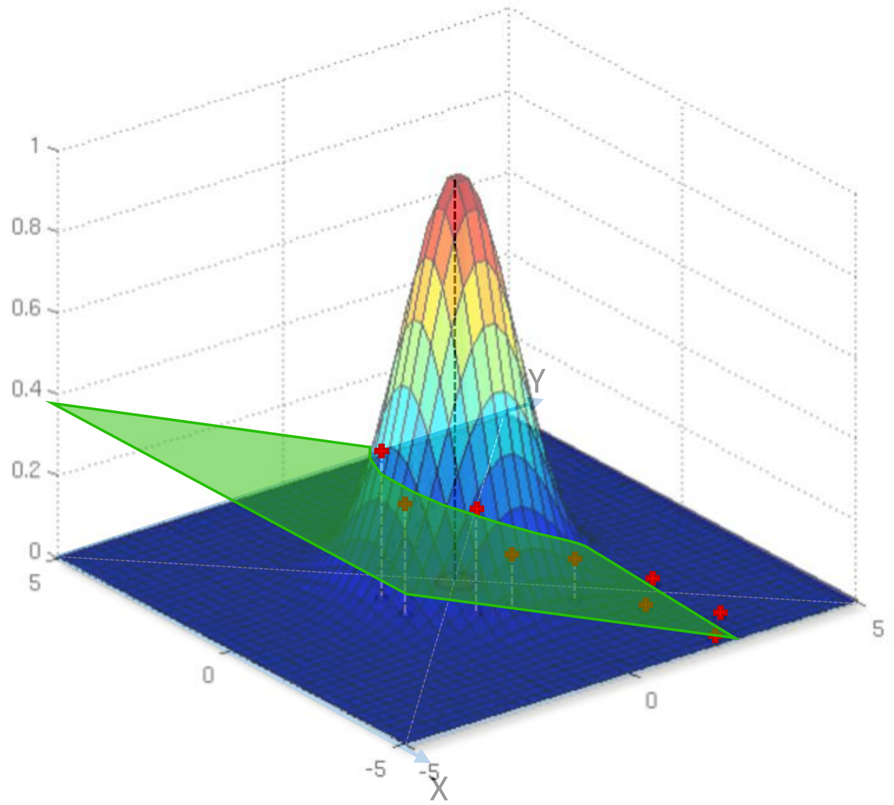
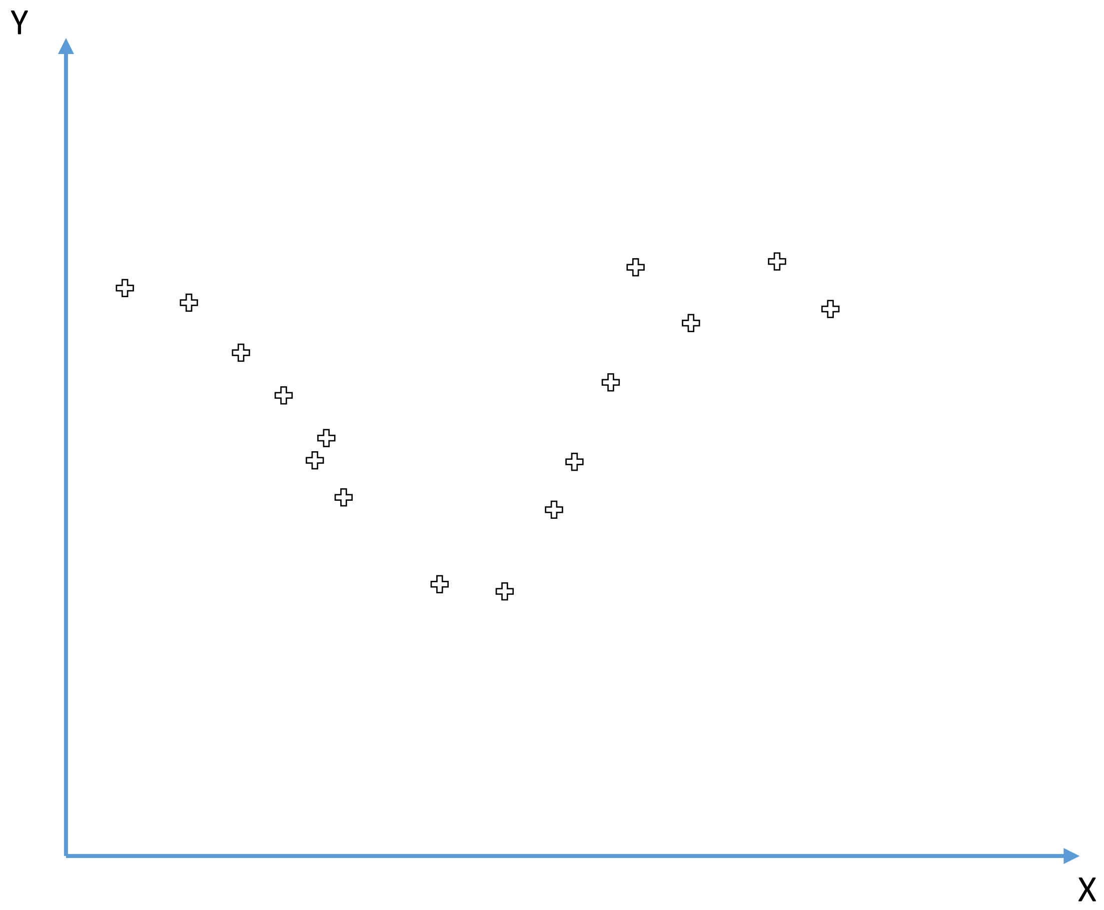
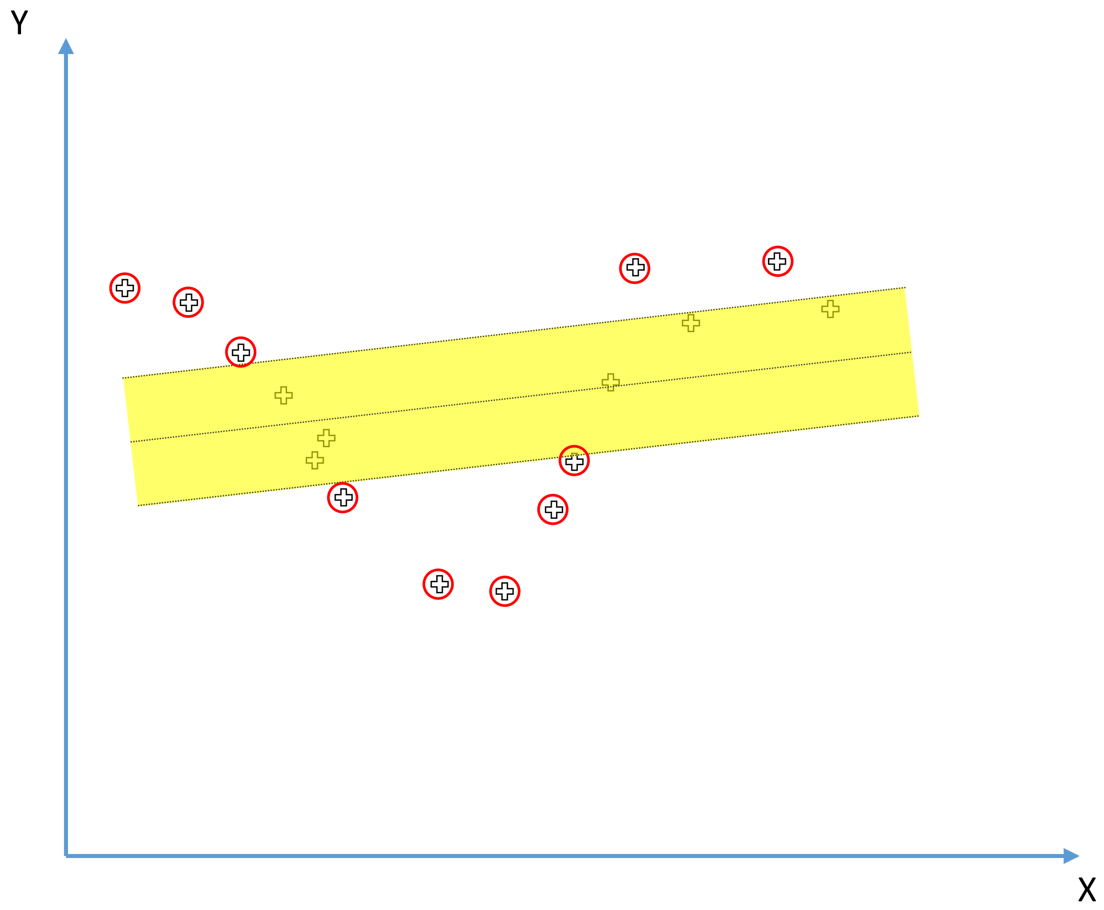


Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

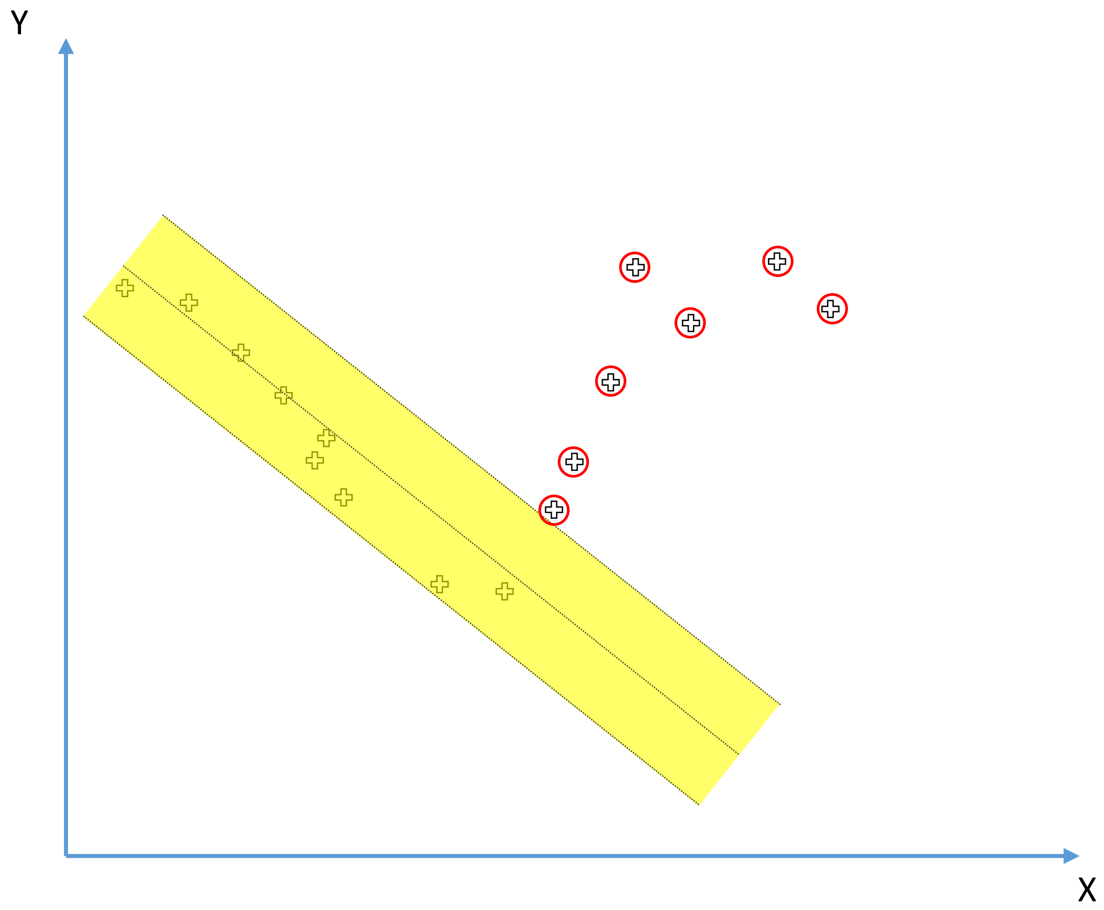
Non-Linear SVR



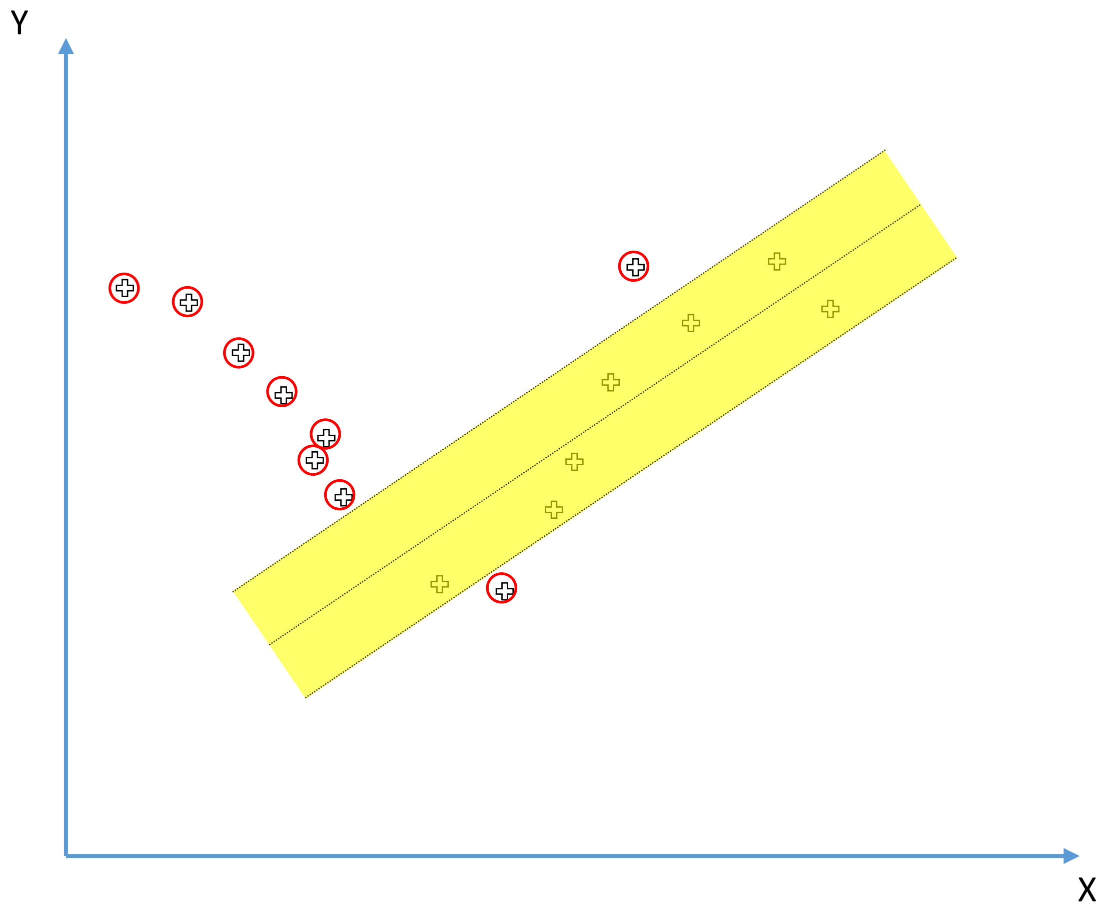
Non-Linear SVR



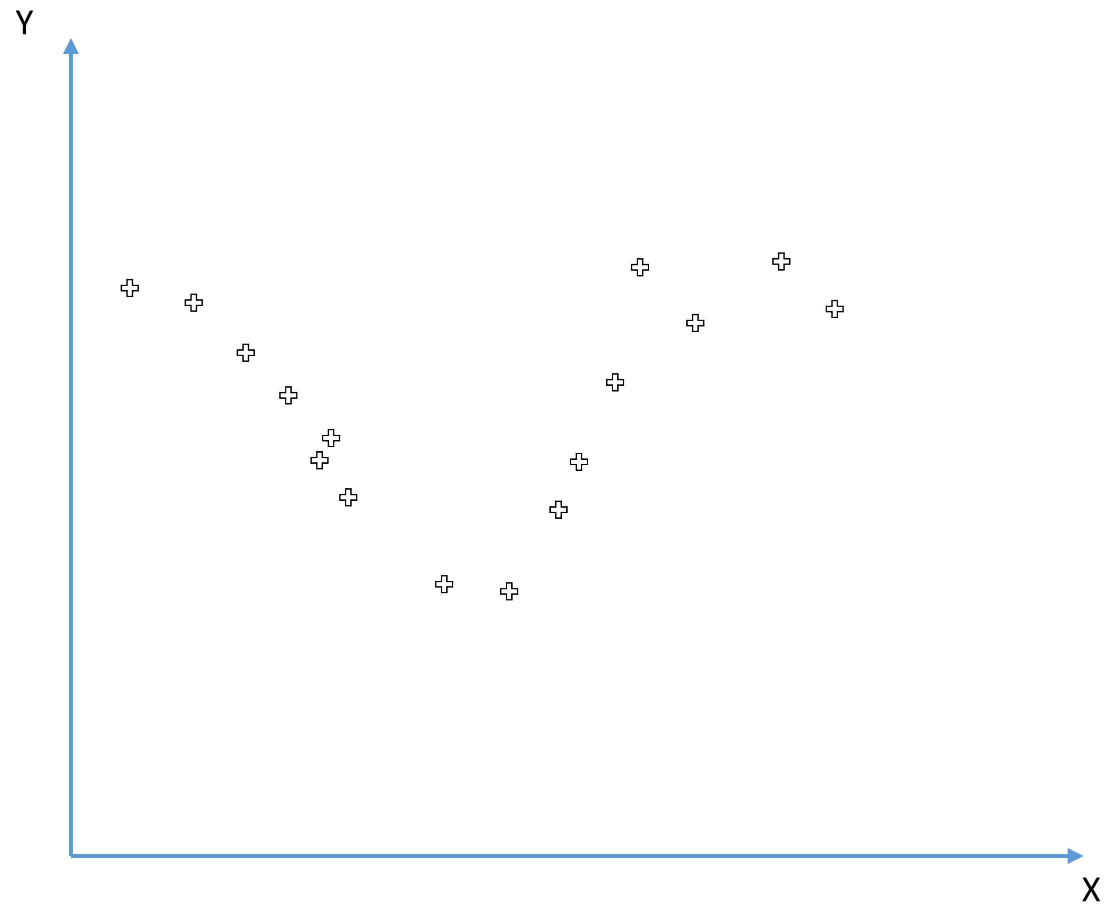
Non-Linear SVR



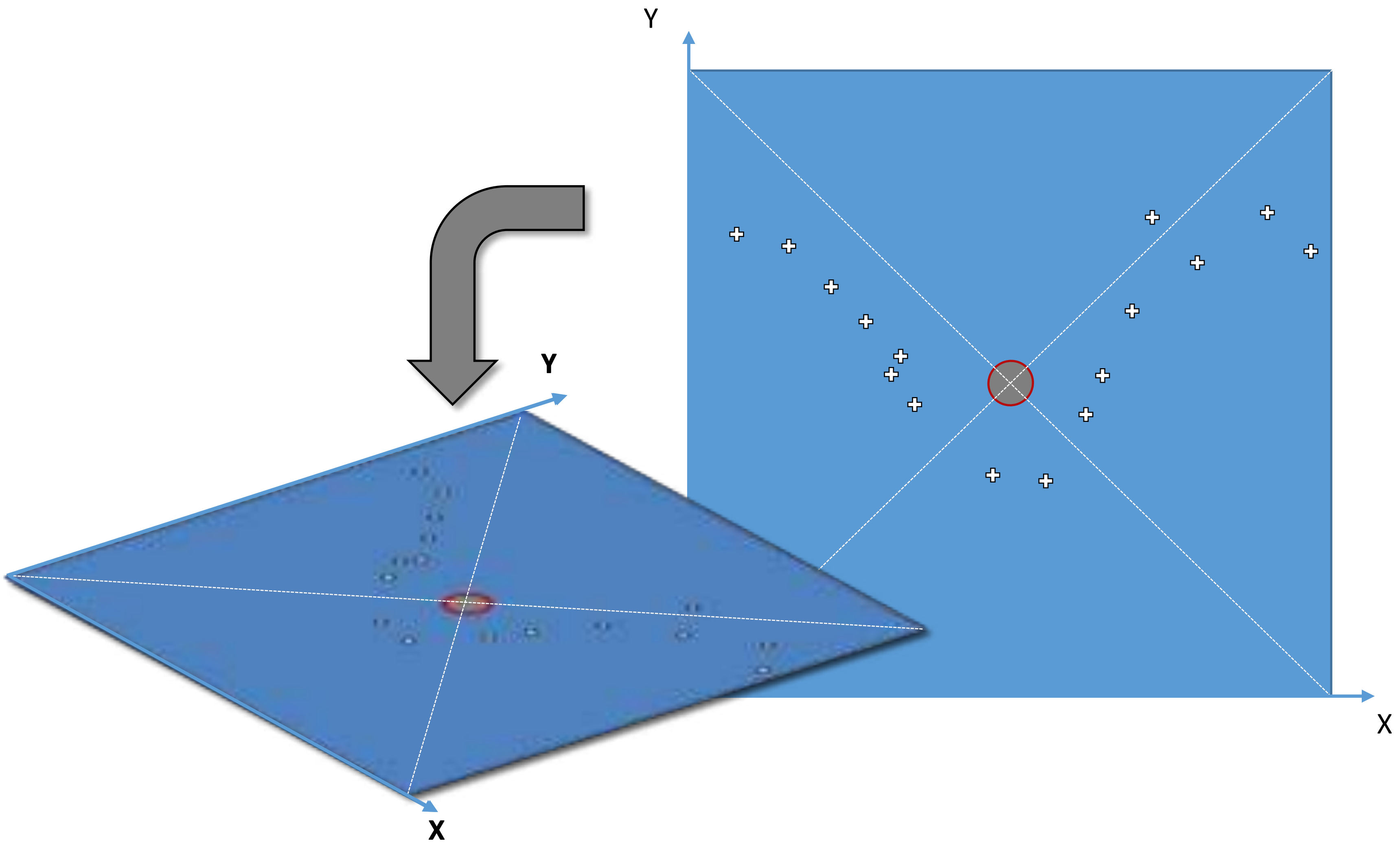
Non-Linear SVR



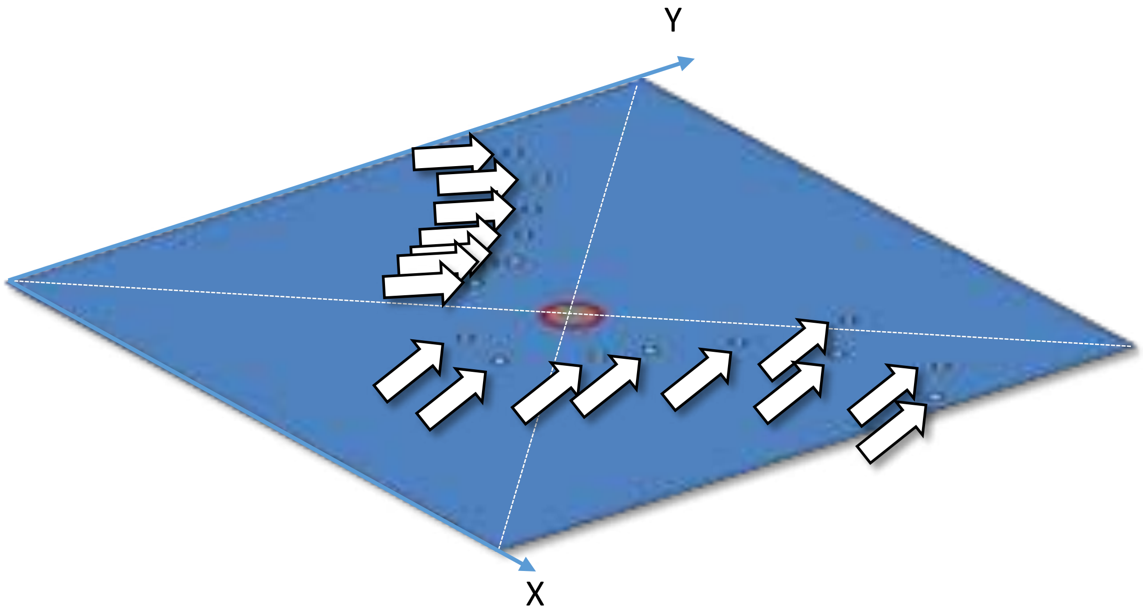
Non-Linear SVR



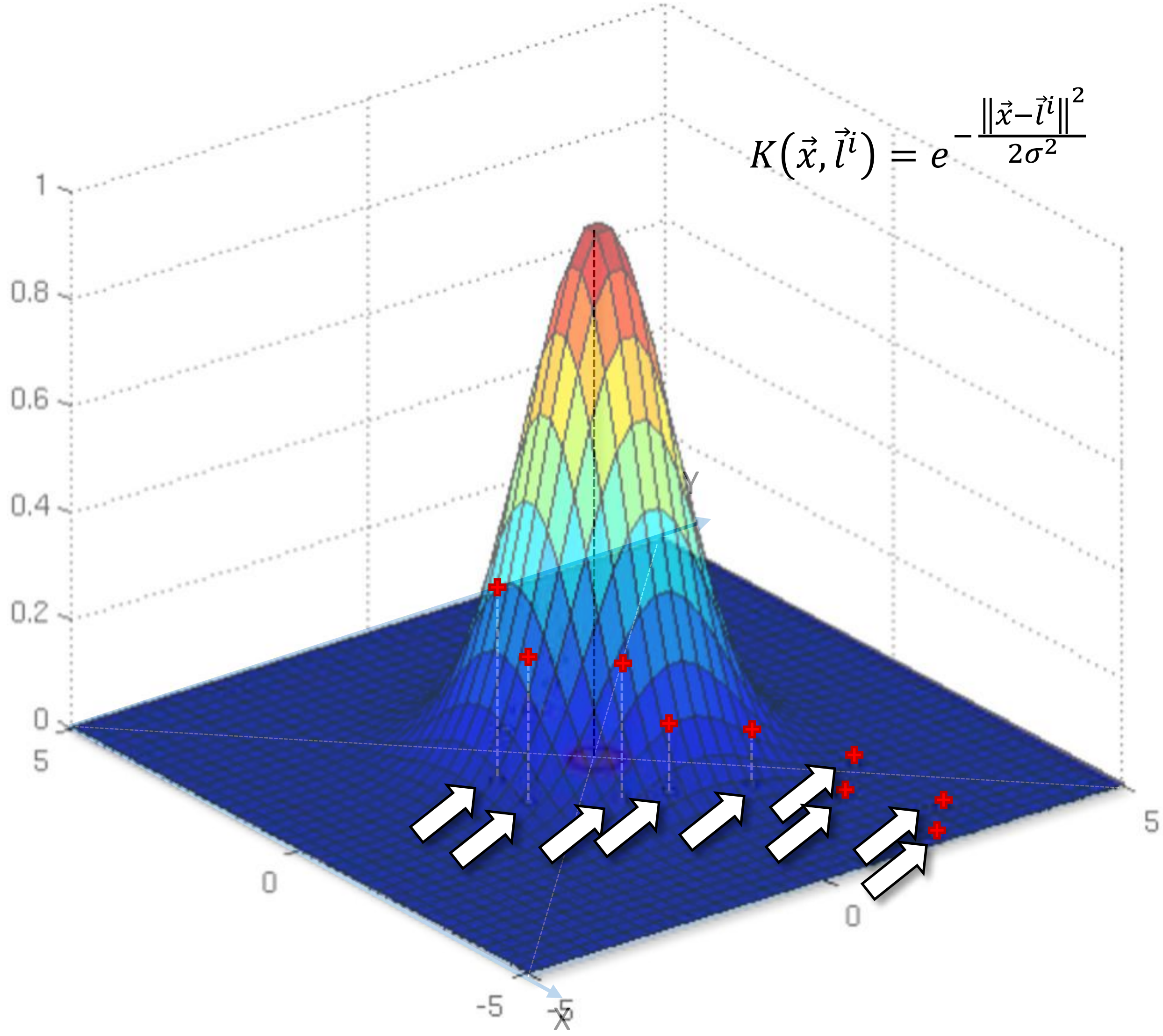
Non-Linear SVR



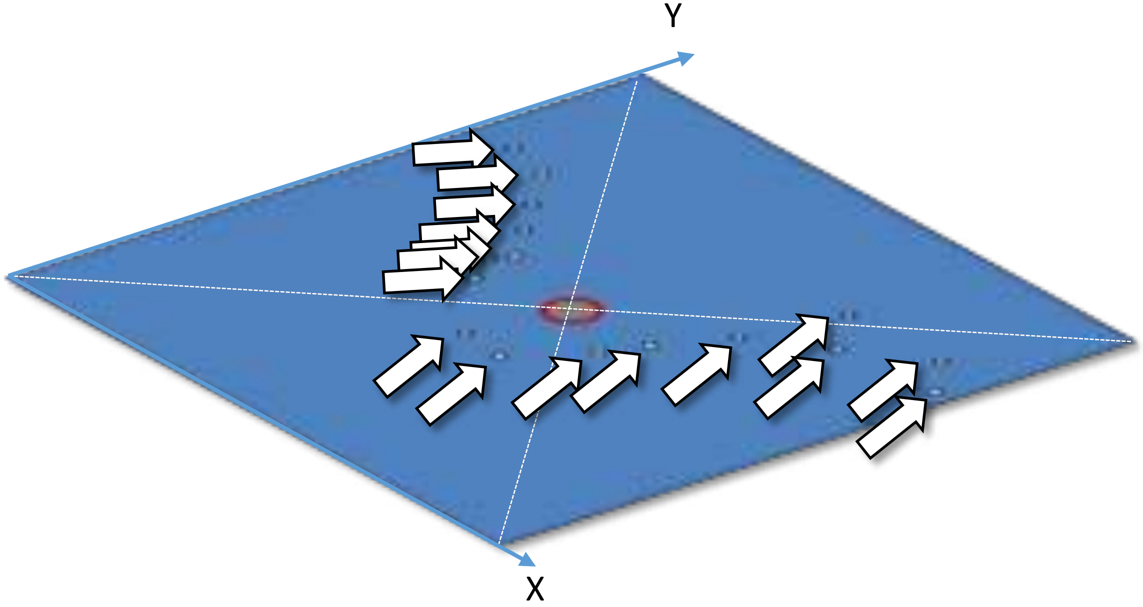
Non-Linear SVR

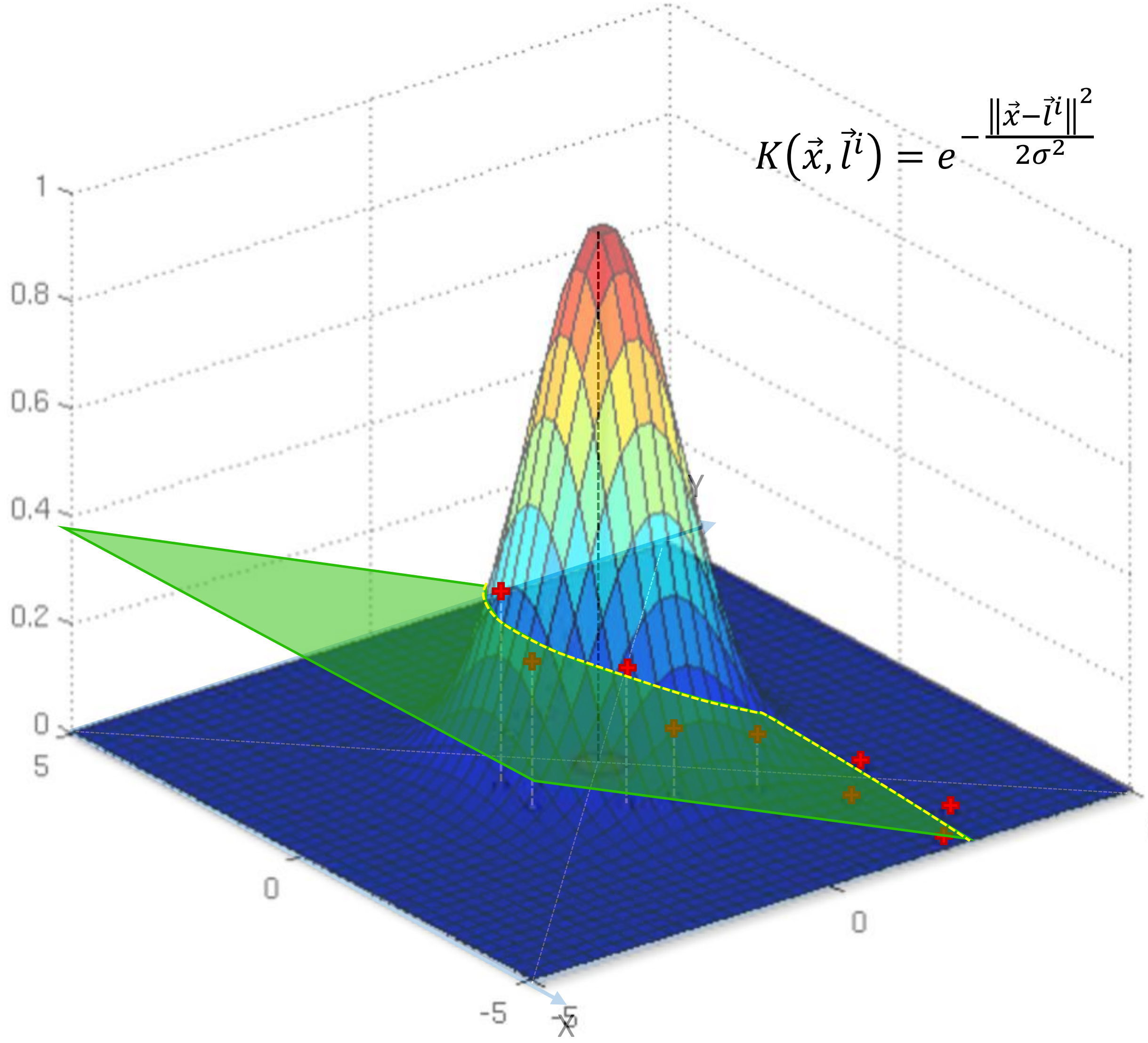


$$K(\vec{x},\vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

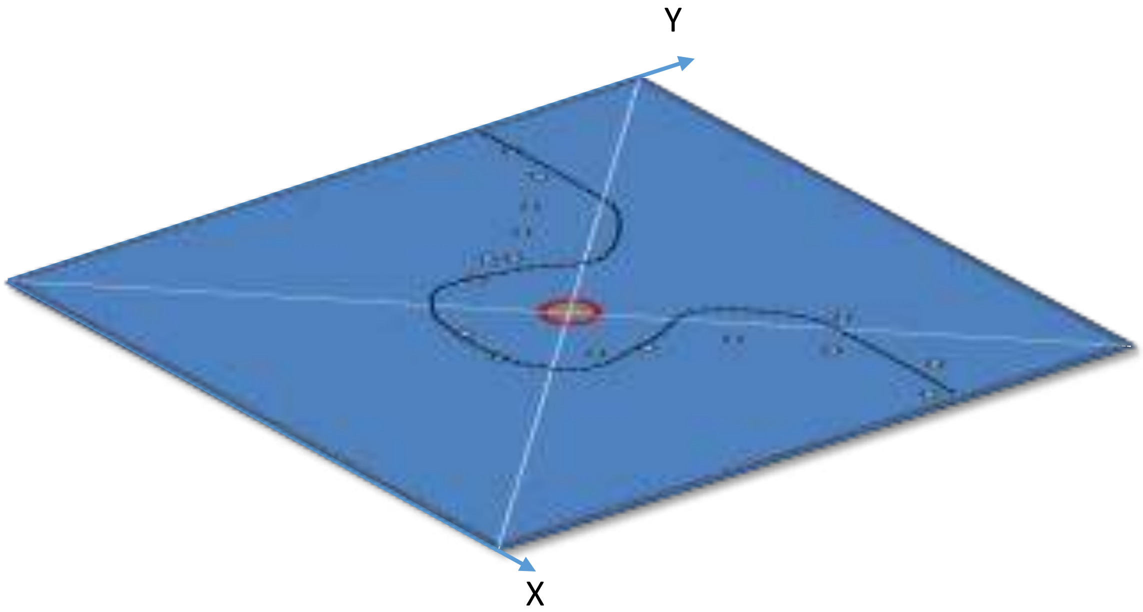


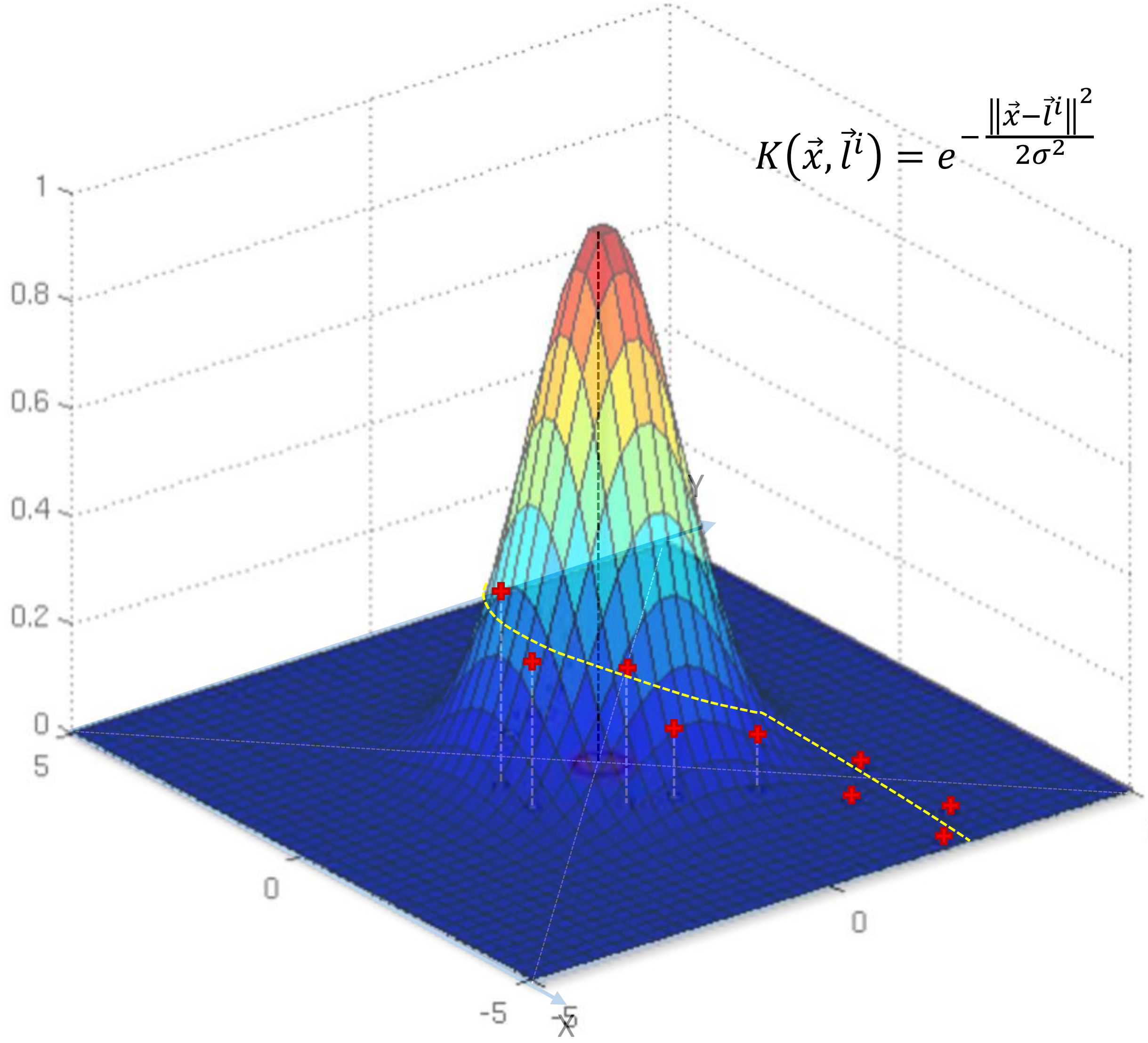
Non-Linear SVR



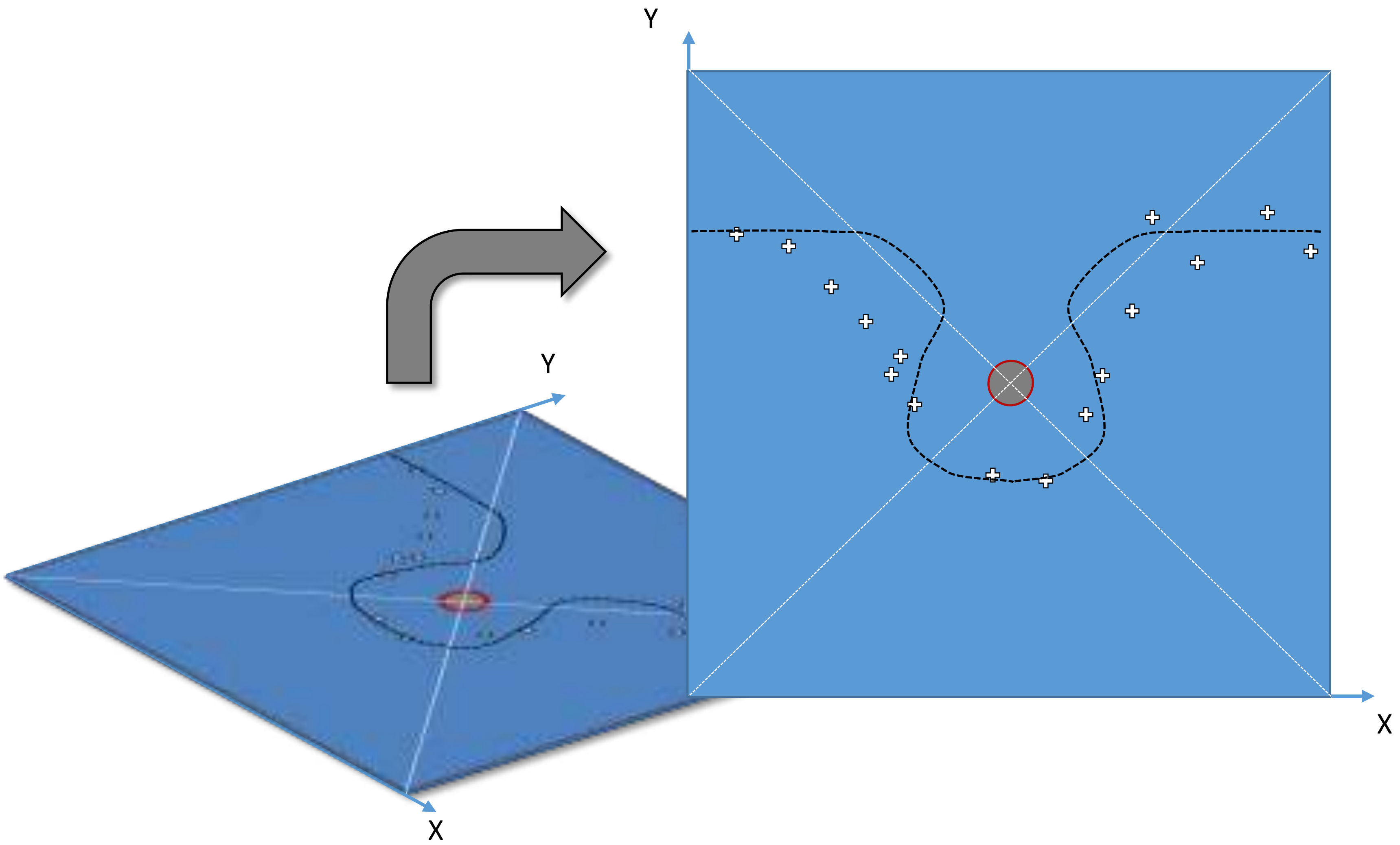


Non-Linear SVR

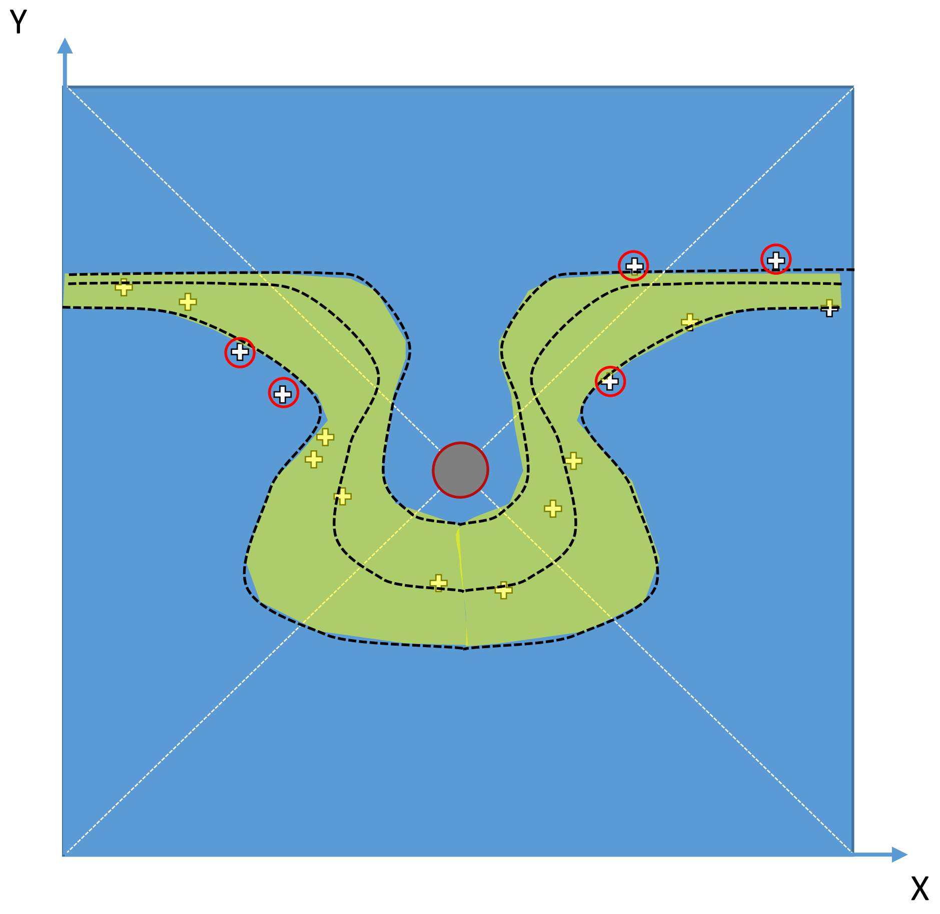


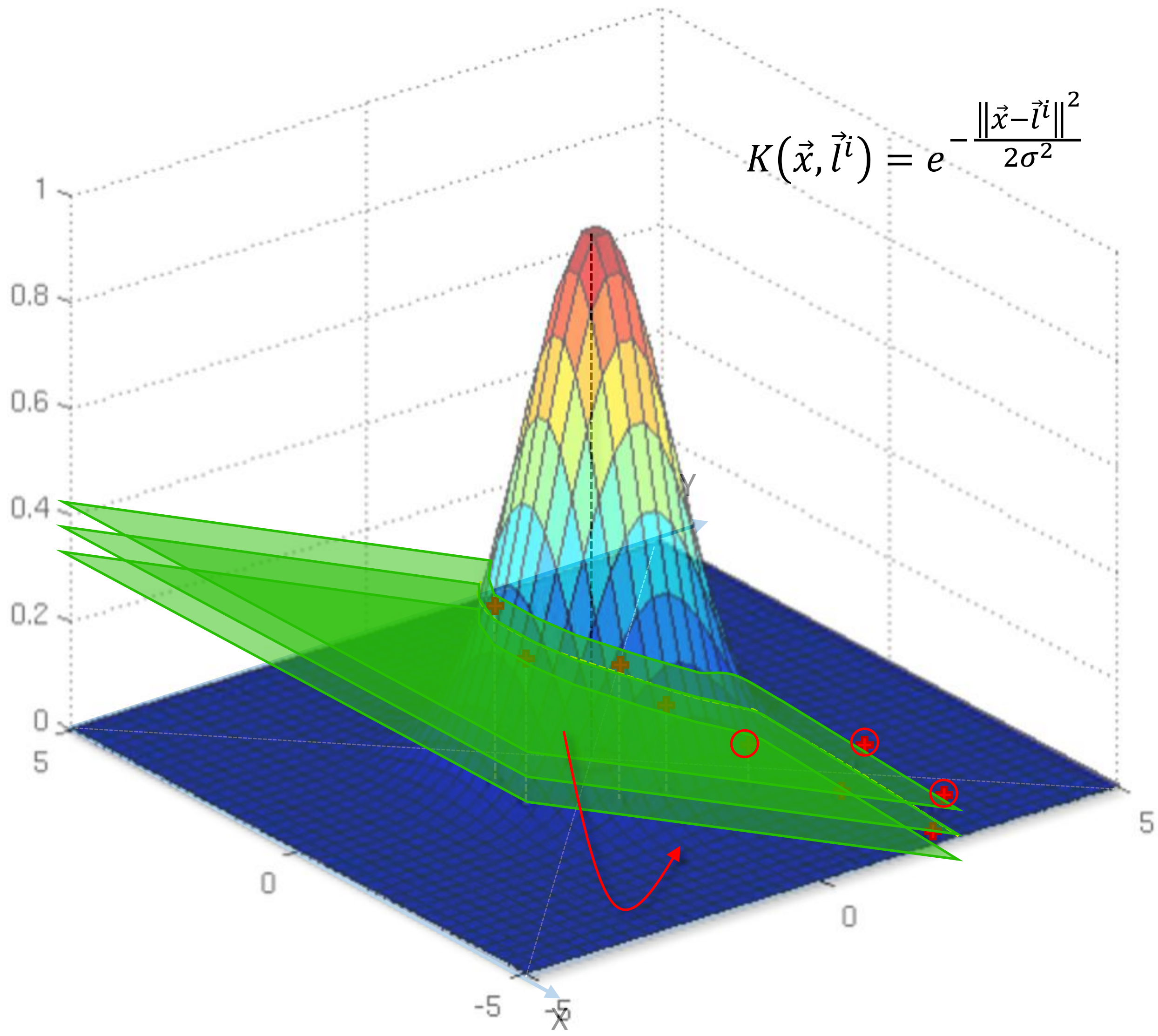


Non-Linear SVR



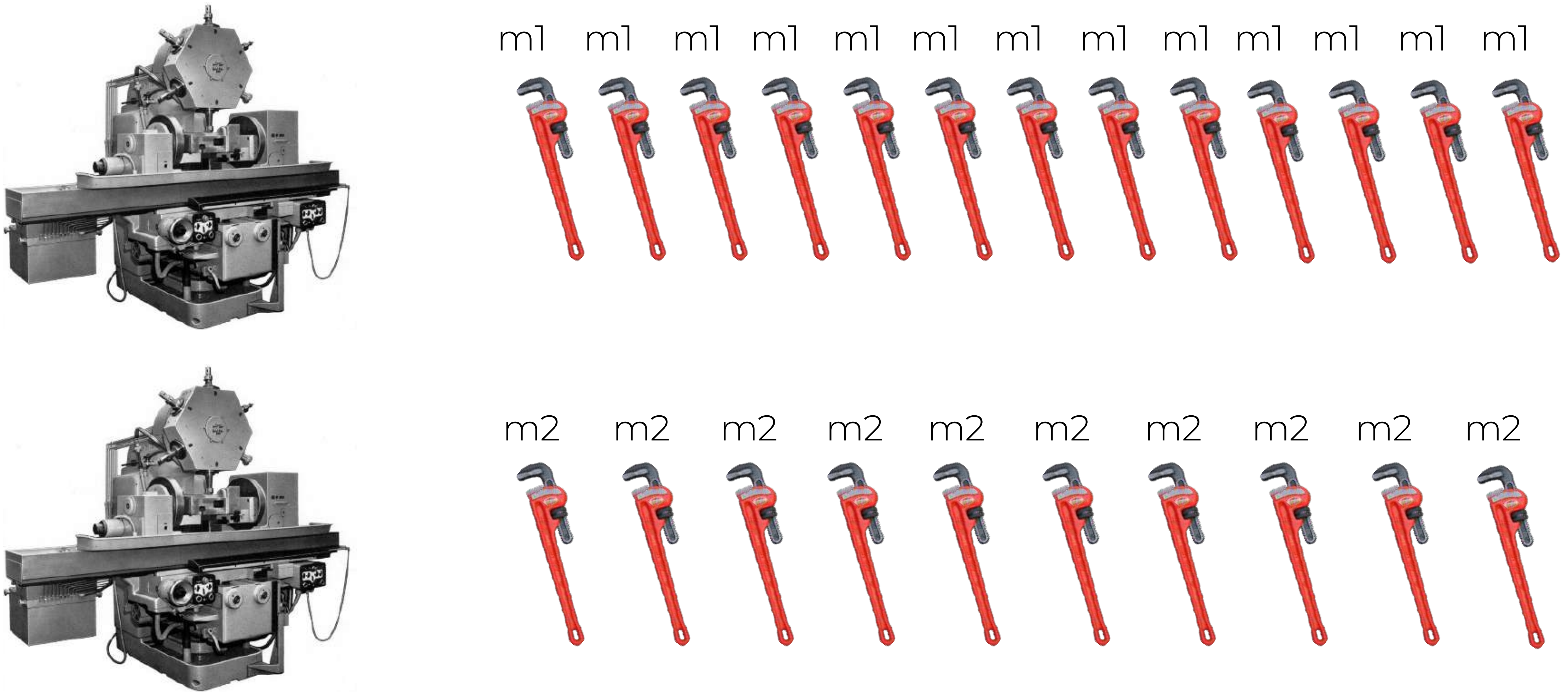
Non-Linear SVR



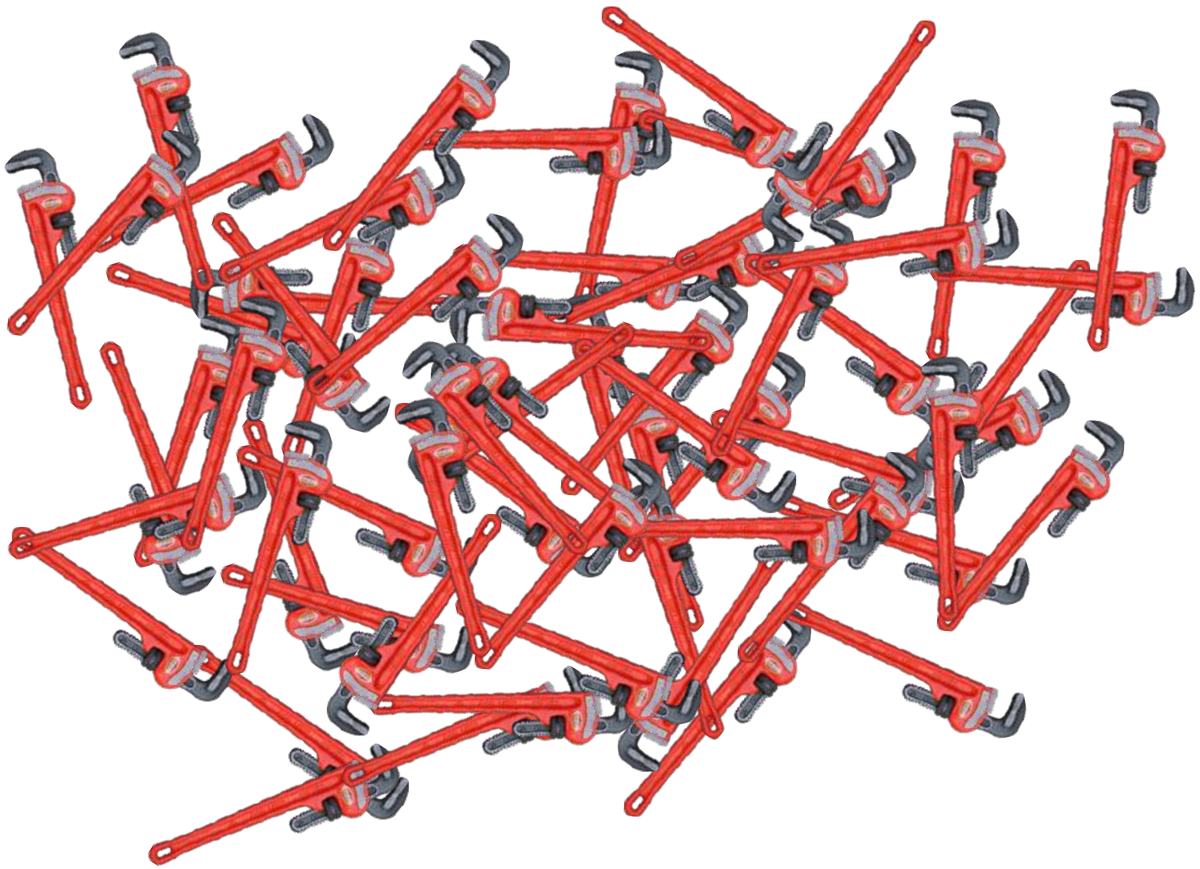


Bayes' Theorem

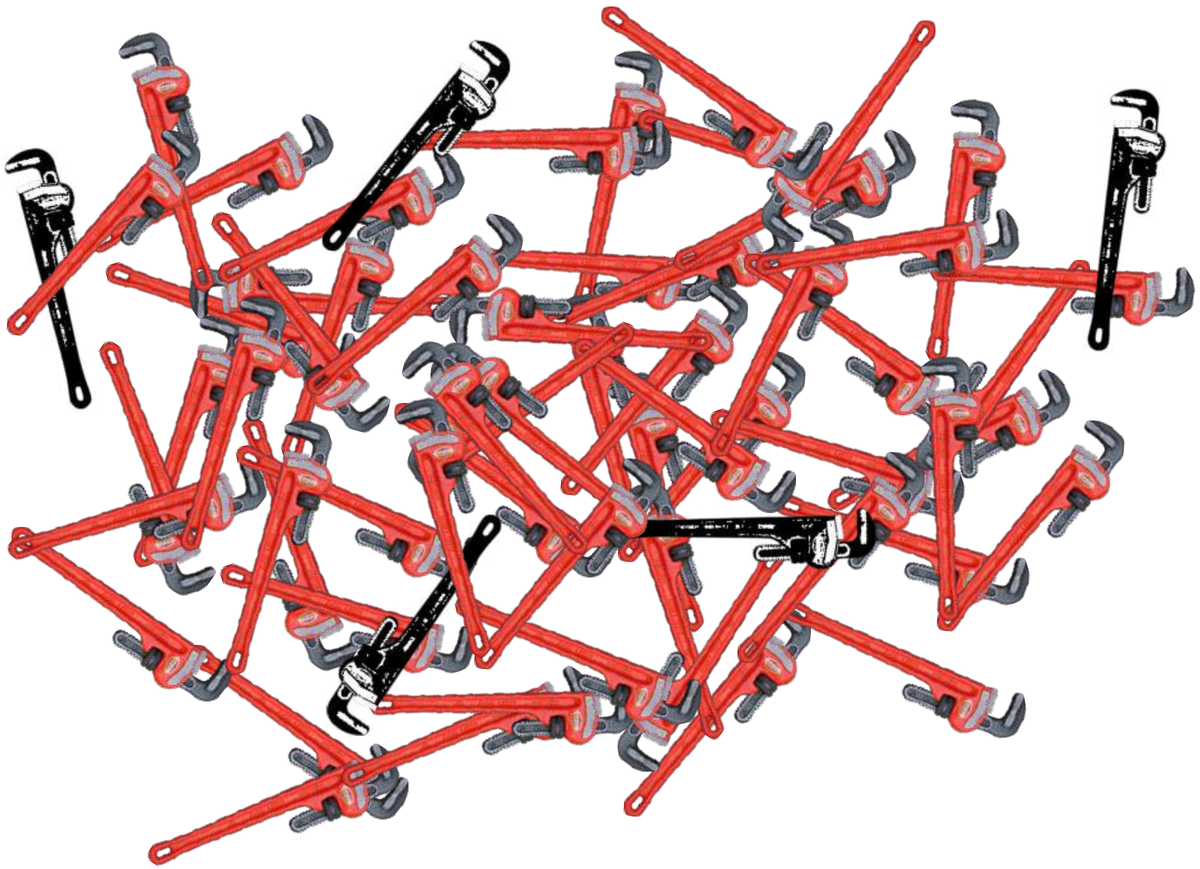
Bayes Theorem



Bayes Theorem

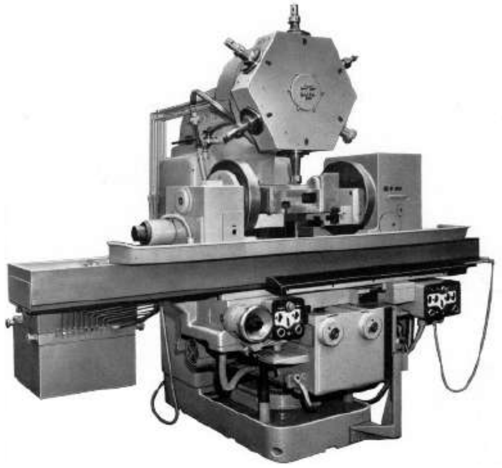


Bayes Theorem



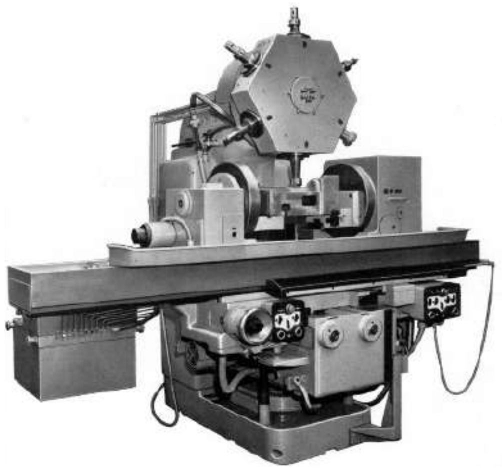
Bayes Theorem

What's the probability?



m2

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

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 = ?

Bayes Theorem

Mach1: 30 wrenches / hr -> P(Mach1) = 30/50 = 0.6
Mach2: 20 wrenches / hr -> 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 -> P(Mach1 | Defect) = 50%
And 50% came from mach2 -> 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

~~-> 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) = ?

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) = ?

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

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) = ?

$$P(Defect \mid Mach2) = \frac{0.5 \quad * \quad 0.01}{0.4}$$

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) = ?

$$P(\text{Defect} \mid \text{Mach2}) = \frac{0.5 \quad * \quad 0.01}{0.4} = 0.0125 = 1.25\%$$

It’s intuitive!

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

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?

Bayes Theorem

Quick exercise:

P(Defect | Mach1) = ?

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

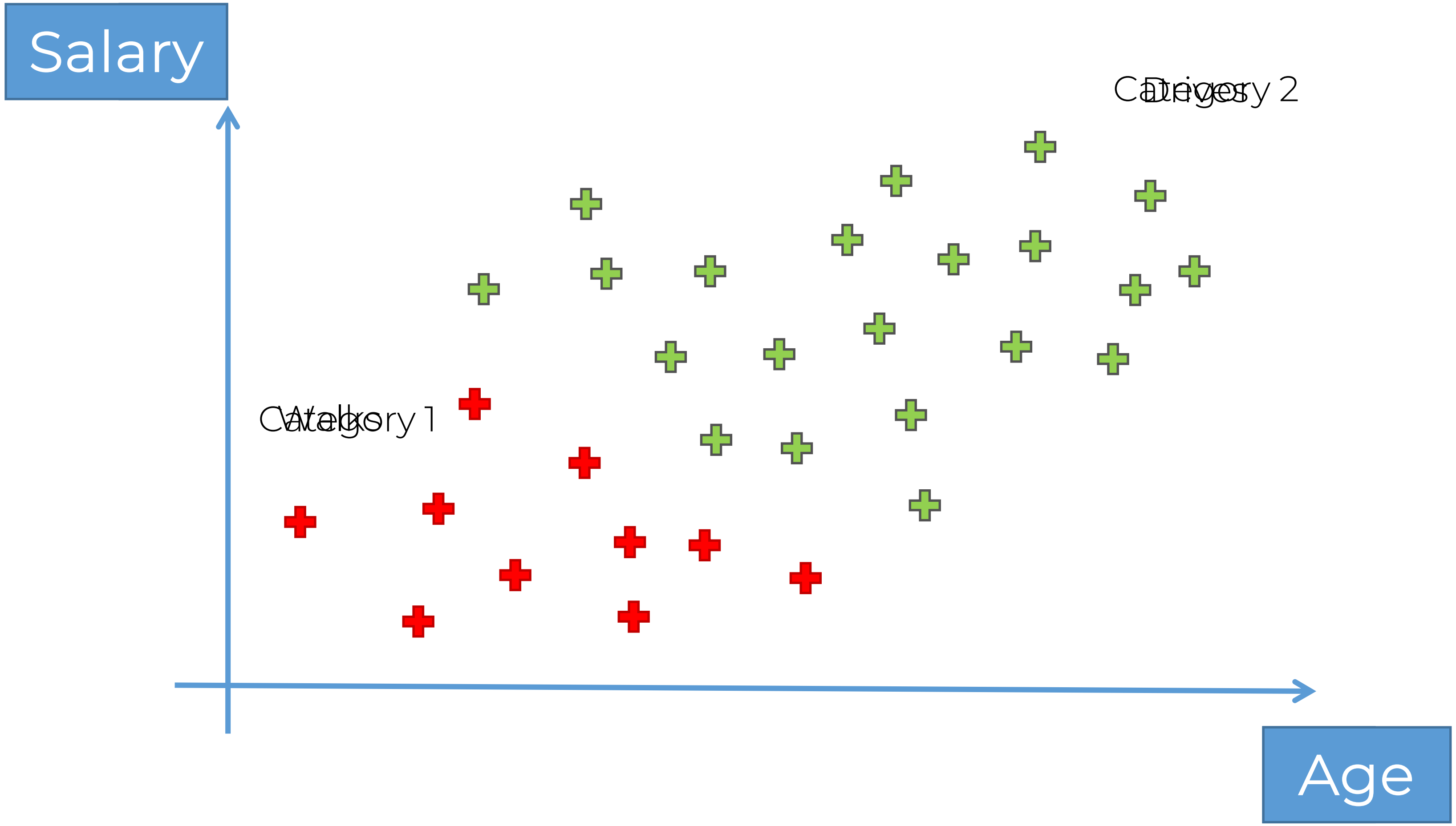
-> P(Mach1 | Defect) = 50%
-> P(Mach2 | Defect) = 50%

Naïve Bayes Classifier Intuition

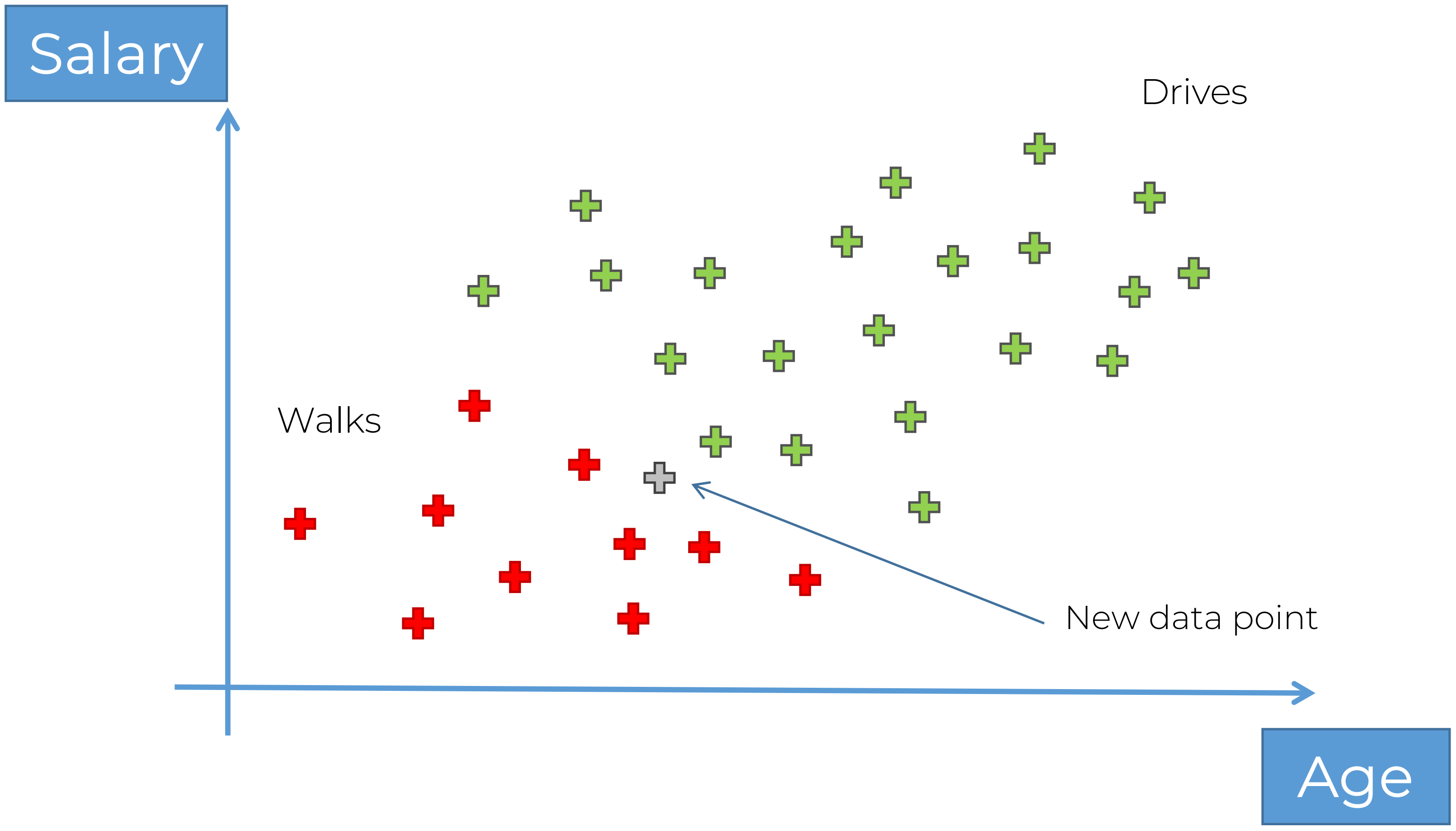
Naïve Bayes

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

Naïve Bayes



Naïve Bayes



Plan of Attack

Naïve Bayes

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

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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

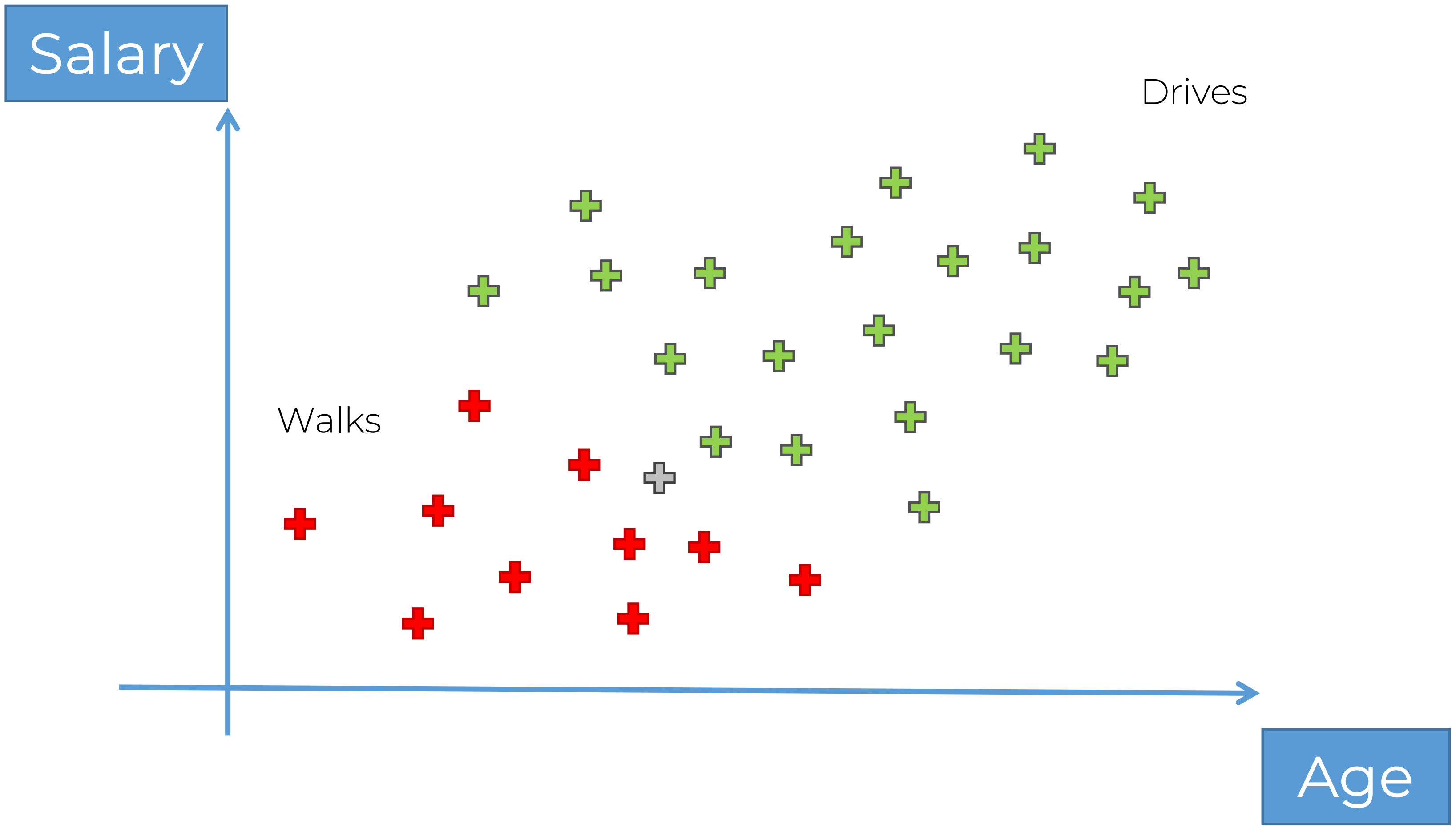
Step 3

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

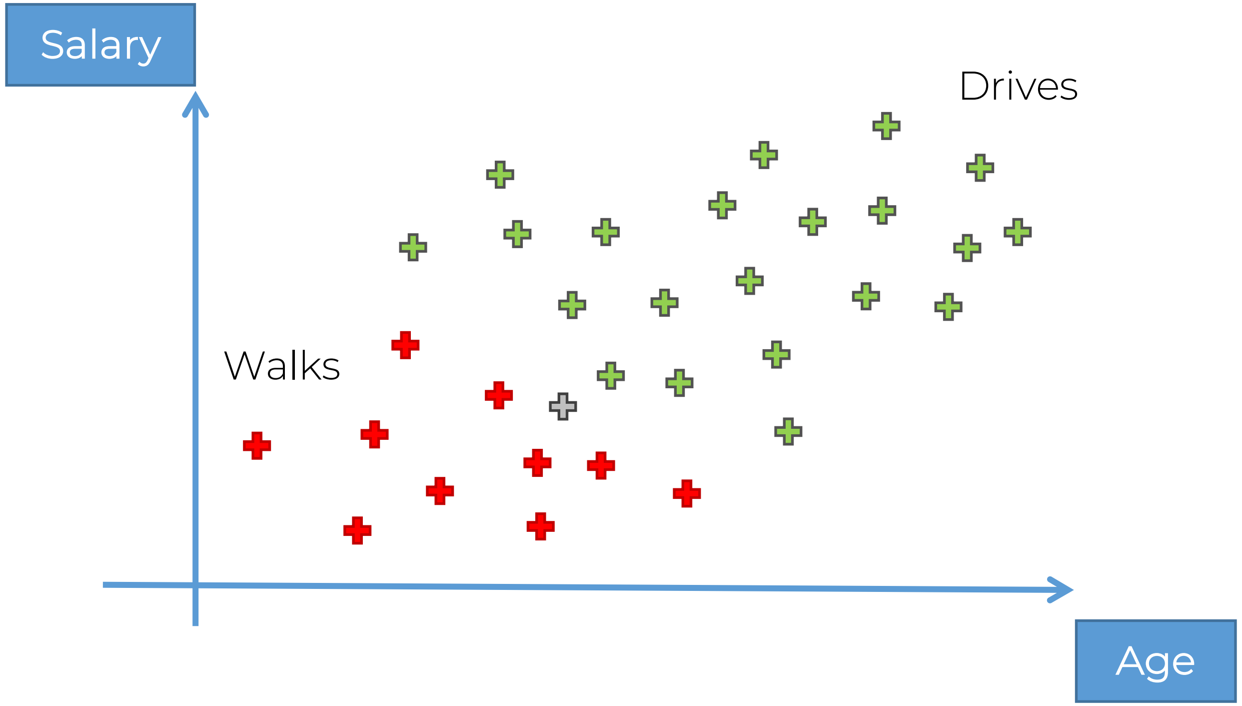
Naïve Bayes

Ready?

Naïve Bayes: Step 1



Naïve Bayes: Step 1



#1. P(Walks)

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

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

Naïve Bayes: Step 1

#4 Posterior Probability

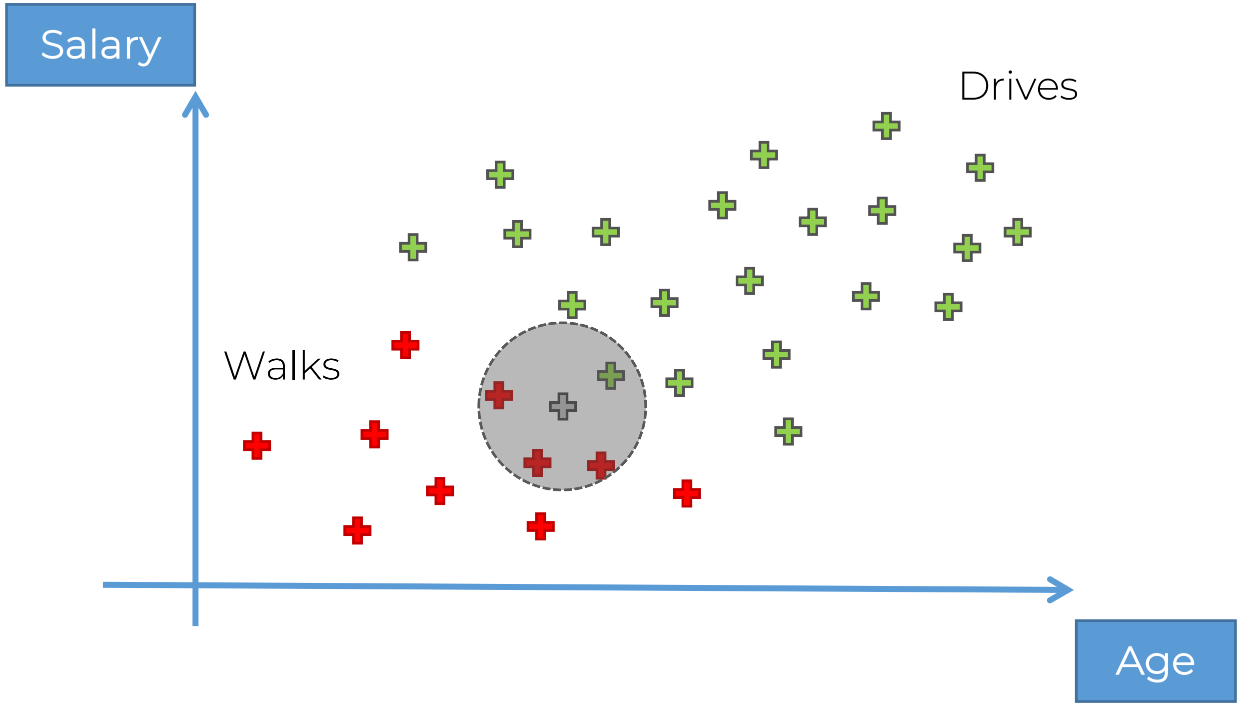
#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

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

#4 Posterior Probability

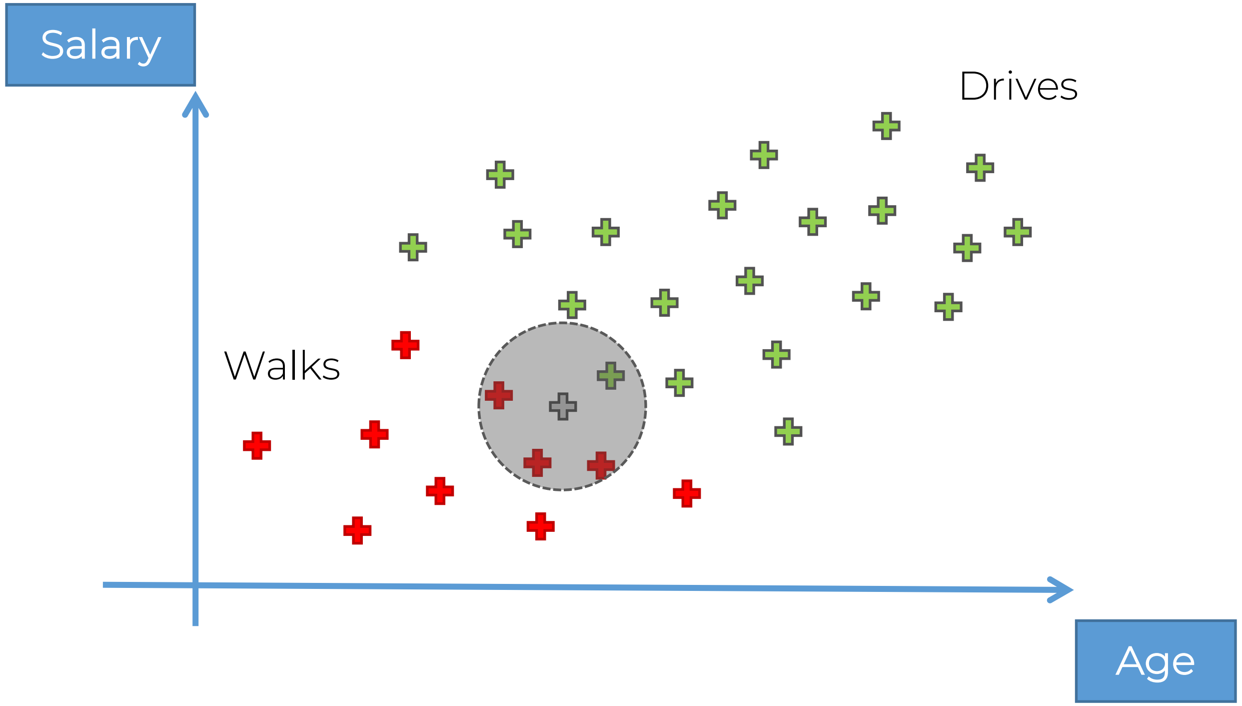
#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes: Step 1



#3. P(X|Walks)

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

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

Naïve Bayes: Step 1

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

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 1 – Done.

Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

$$P(Drives|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

#2 Marginal Likelihood

Step 2 – Done.

Step 3

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

Step 3

$$0.75 \; v.s. \; 0.25$$

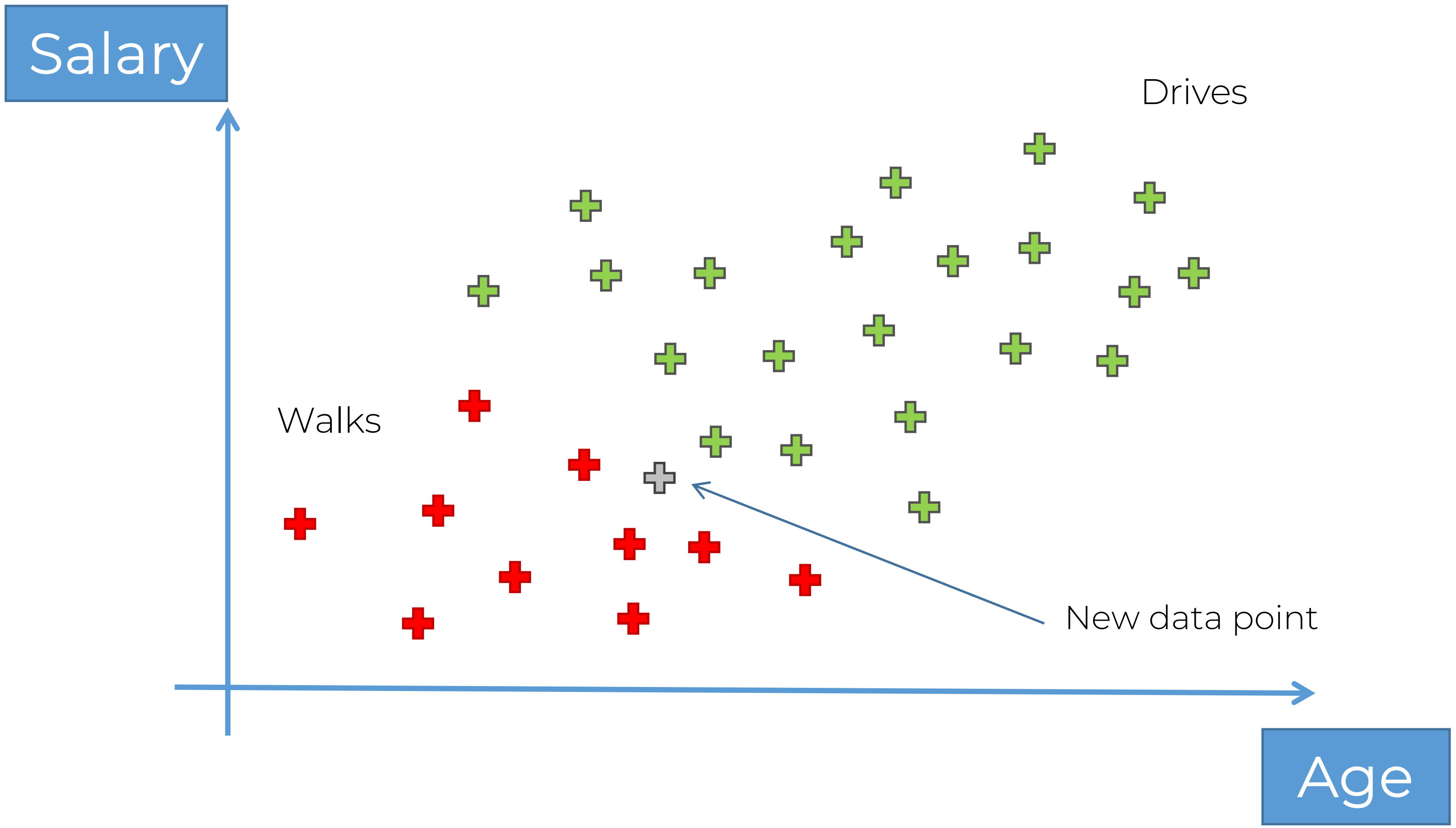
Step 3

$$0.75 > 0.25$$

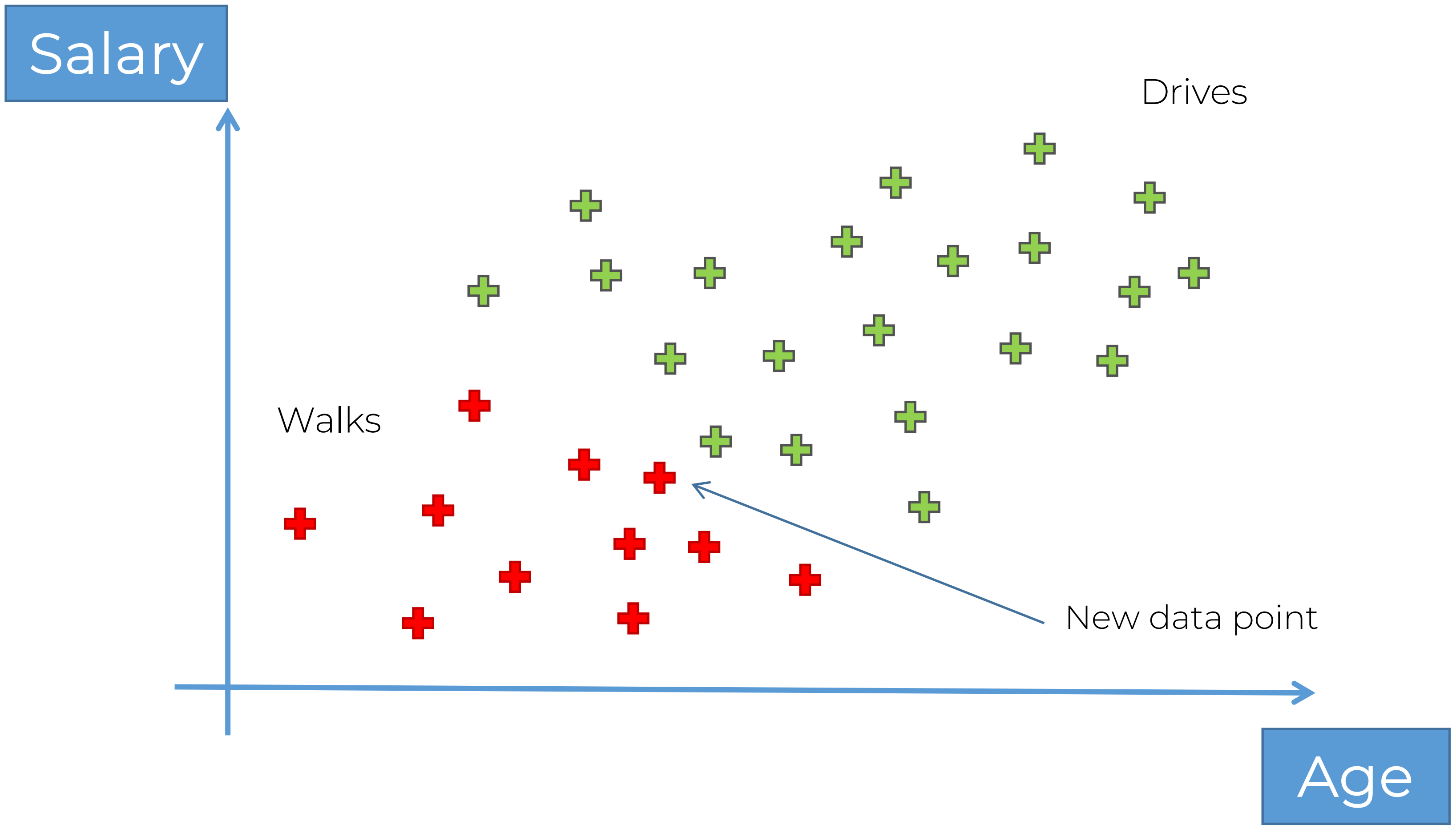
Step 3

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

Naïve Bayes



Naïve Bayes



Naïve Bayes Classifier Intuition (Challenge Reveal)

Step 2

#4 Posterior Probability

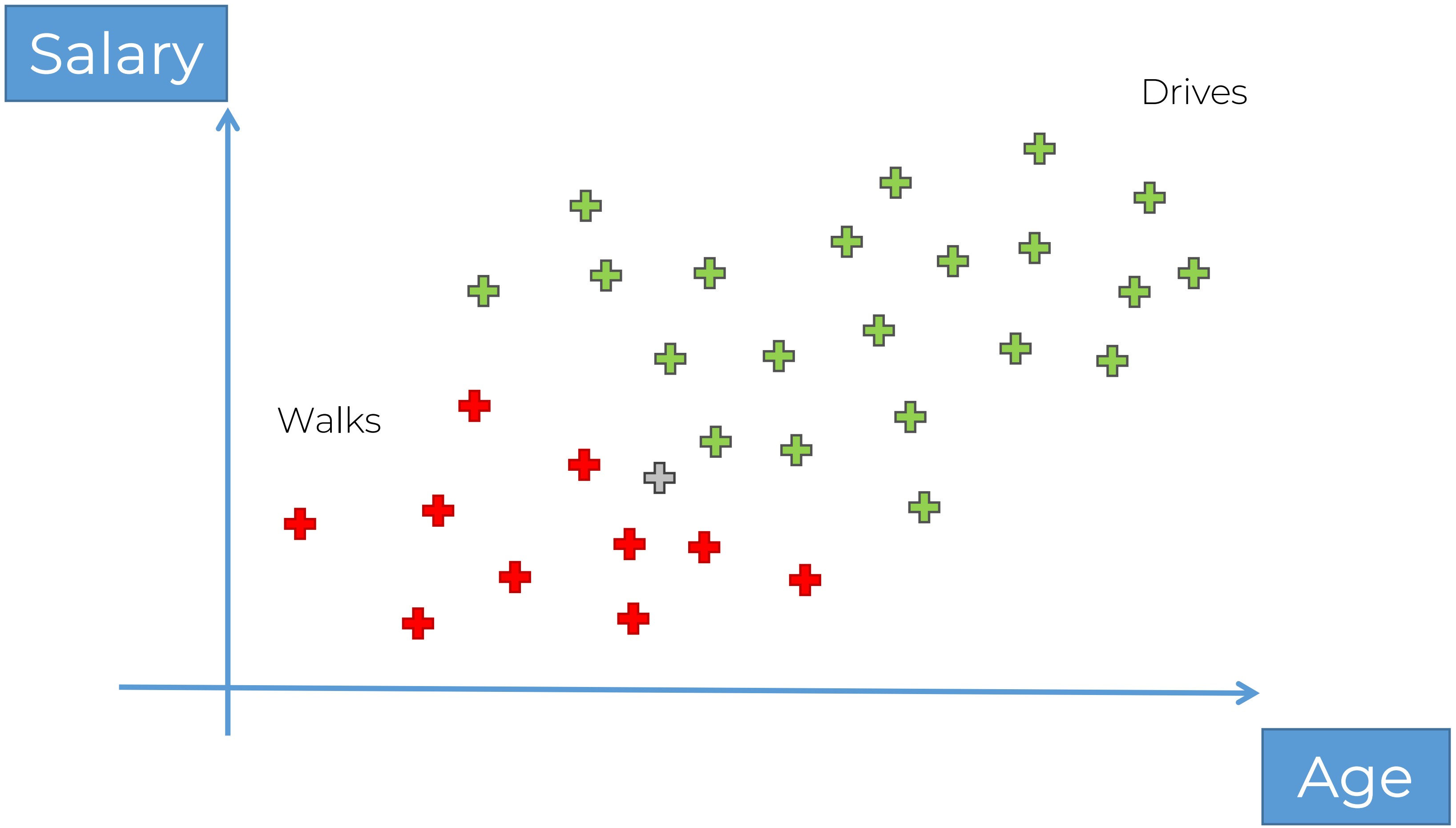
#3 Likelihood

#1 Prior Probability

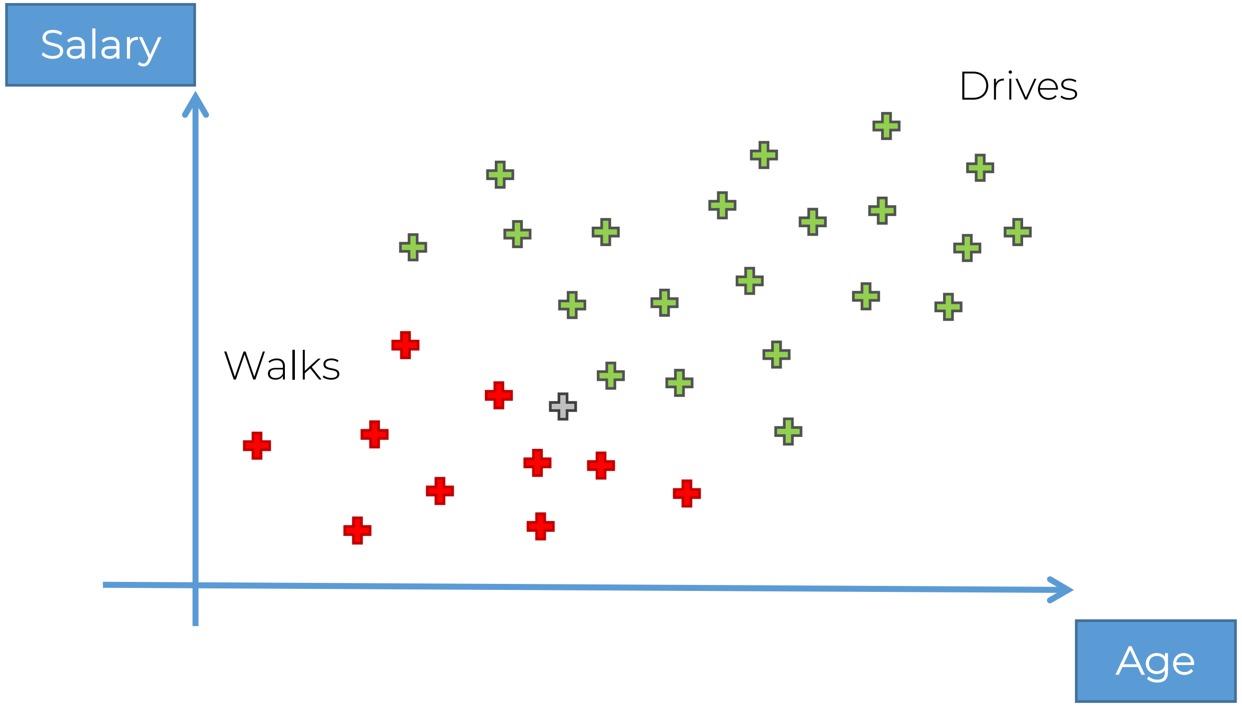
$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

#2 Marginal Likelihood

Naïve Bayes: Step 2



Naïve Bayes: Step 2



#1. P(Drives)

$$P(Drives) = \frac{Number\ of\ Drivers}{Total\ Observations}$$

$$P(Drives) = \frac{20}{30}$$

Naïve Bayes: Step 2

#4 Posterior Probability

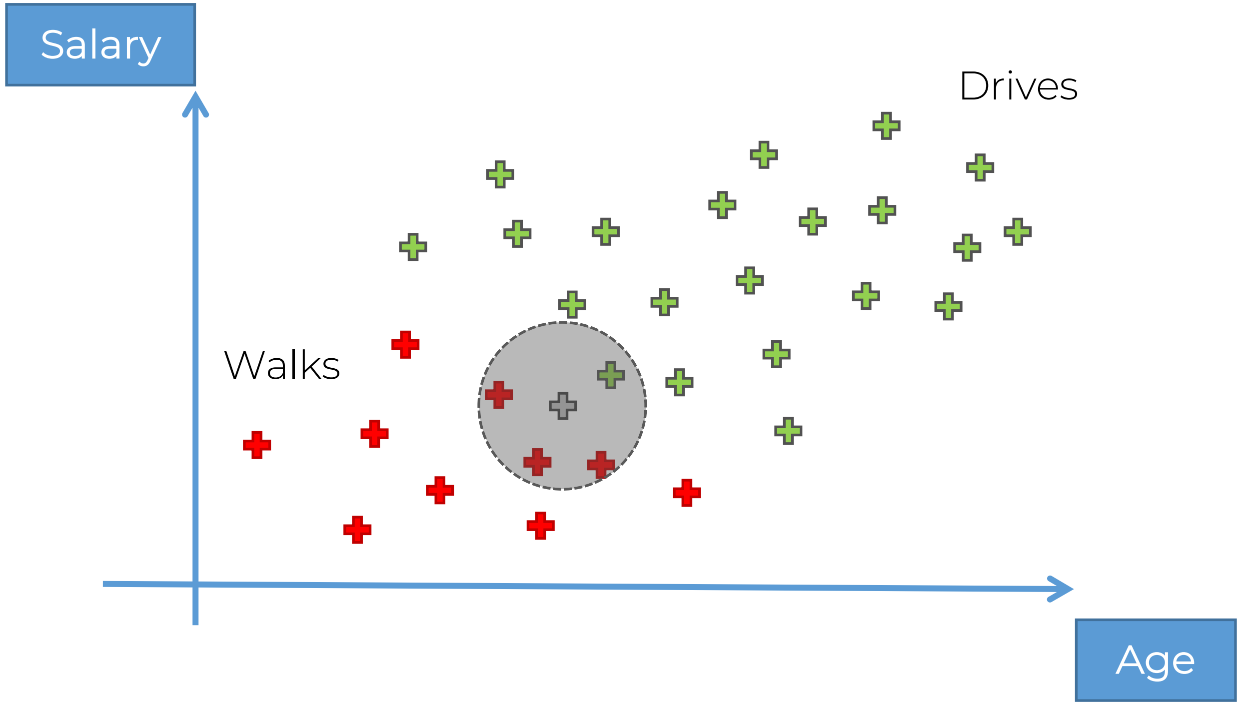
#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes: Step 2



#2. P(X)

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

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

Naïve Bayes: Step 2

#4 Posterior Probability

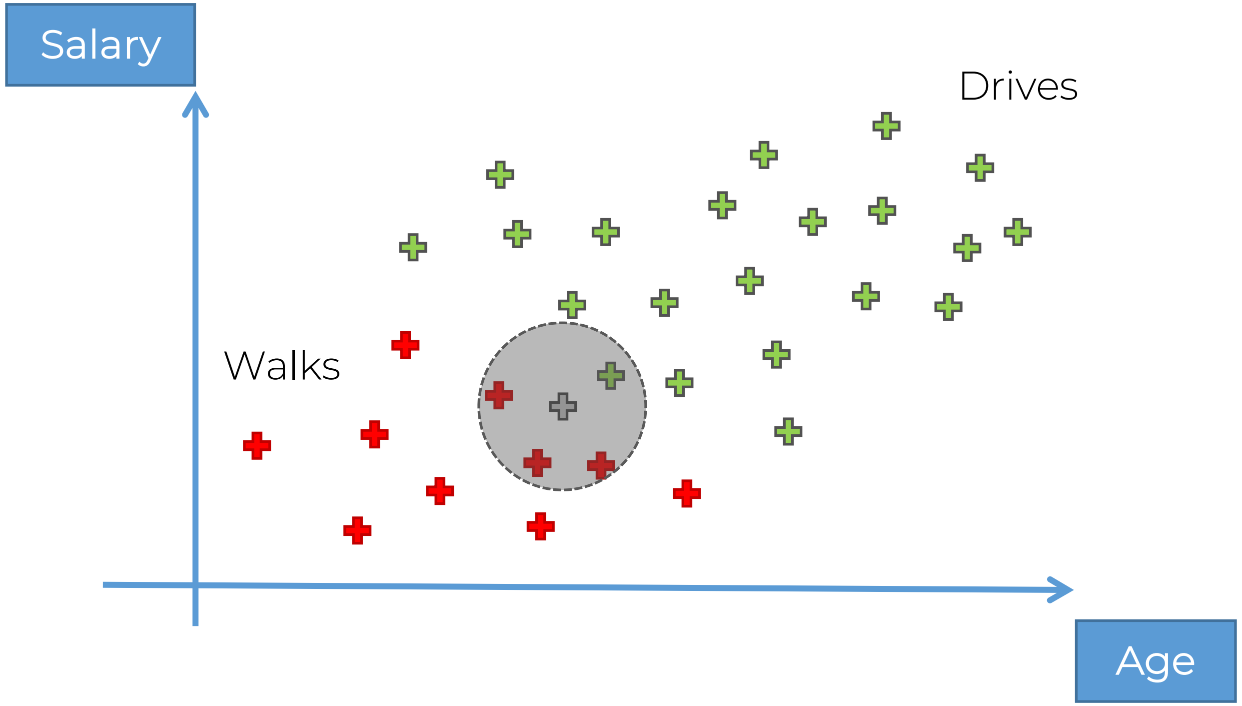
#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes: Step 2



#3. P(X|Drives)

$$P(X|Drives) = \frac{\begin{array}{c}Number\ of\ Similar\\ Observations\\ Among\ those\ who\ Walk\end{array}}{Total\ number\ of\ Walkers}$$

$$P(X|Drives) = \frac{1}{20}$$

Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

$$P(Drives|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

#2 Marginal Likelihood

Step 2 – Done.

Naïve Bayes Classifier Additional Comments

Naïve Bayes

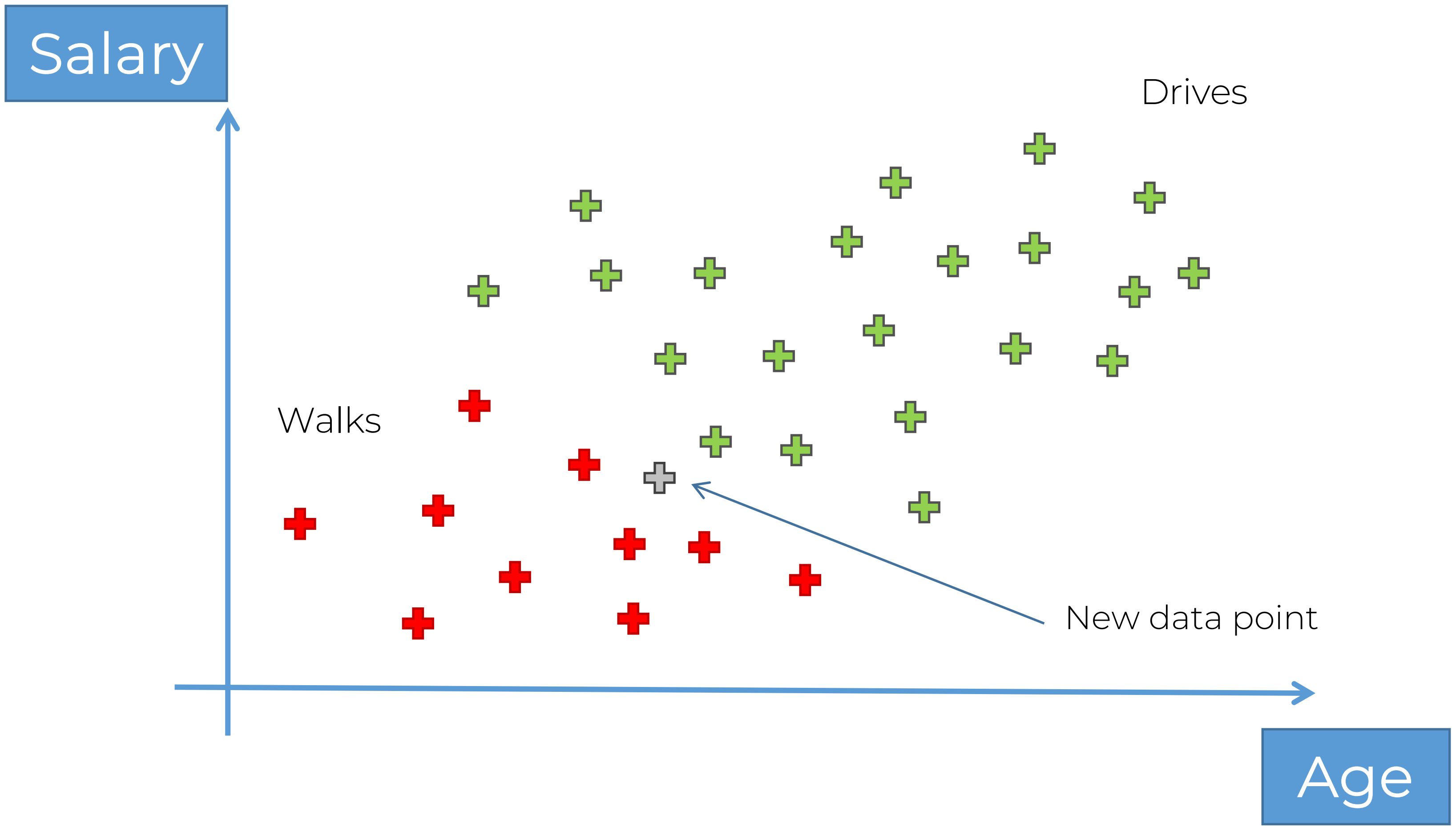
1. Q: Why “Naïve”?
2. P(X)
3. More than 2 features

Naïve Bayes

Q: Why "Naïve"?

A: Independence assumption

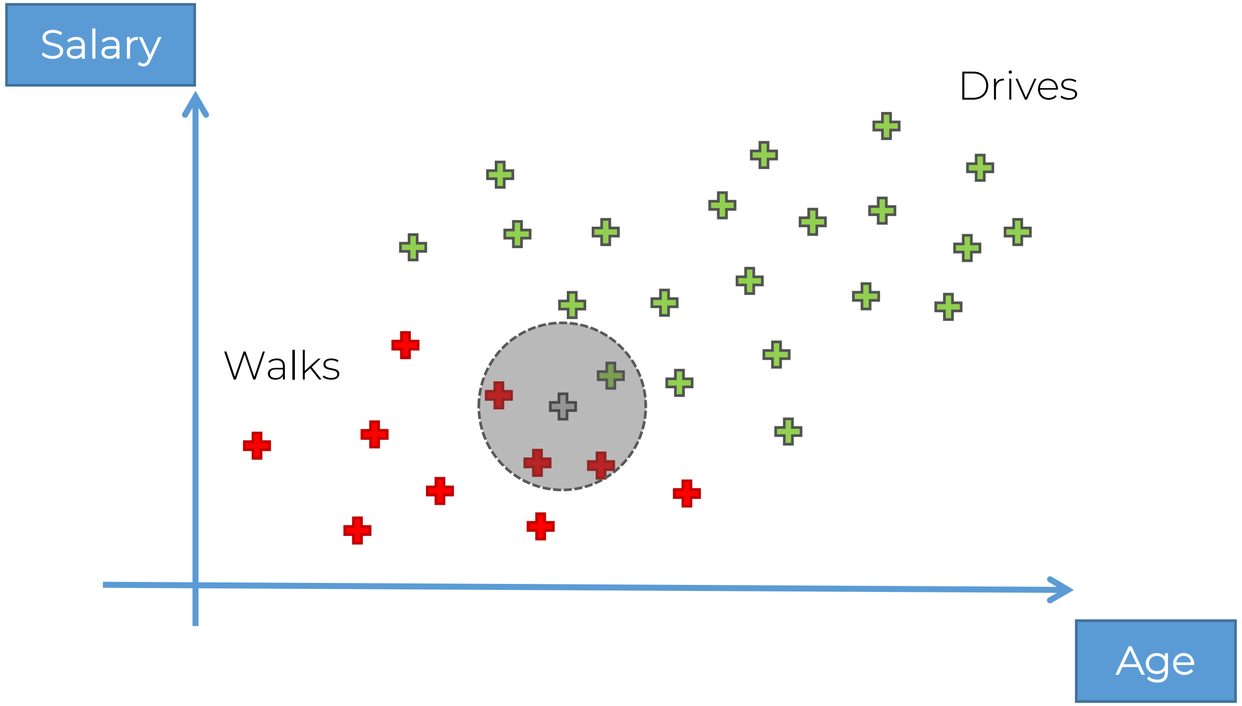
Naïve Bayes



Naïve Bayes

$$P(X)$$

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 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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Step 3

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

Step 3

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

More than 2 classes

Step 3

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

Step 3

$$0.75 \; v.s. \; 0.25$$

Step 3

$$0.75 > 0.25$$

Step 3

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

Decision Tree Intuition

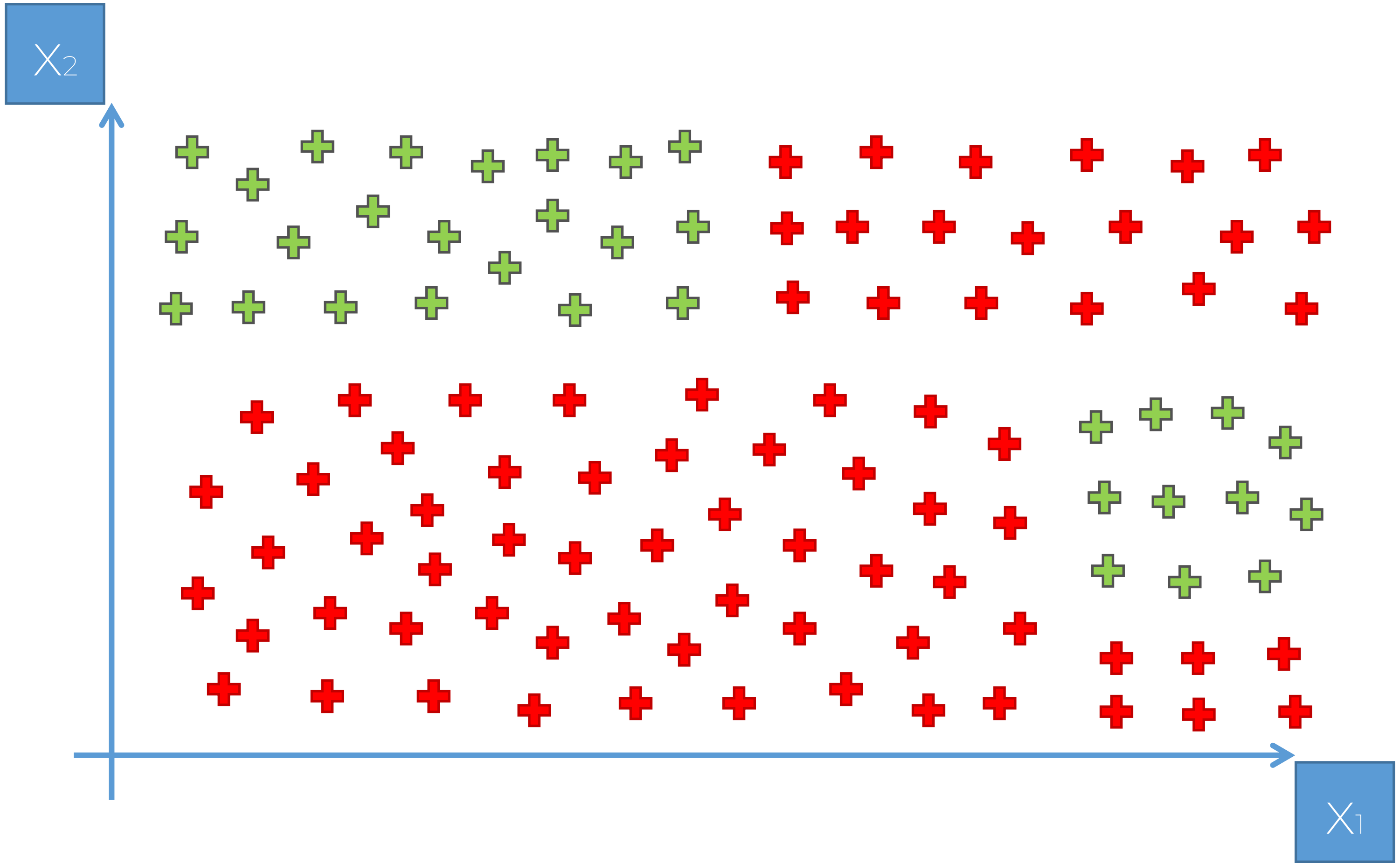
Decision Tree Intuition

CART

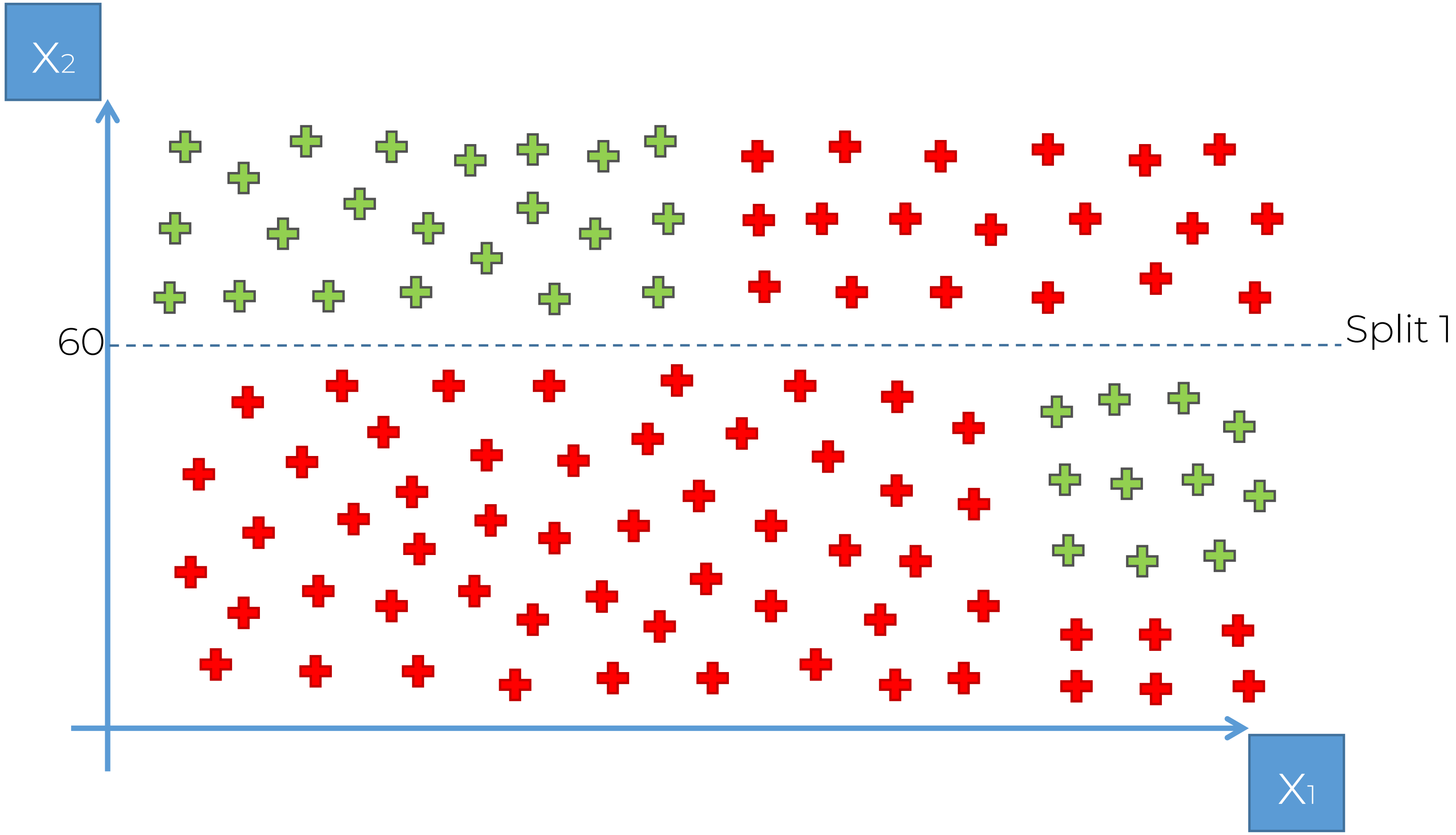
Classification Trees

Regression Trees

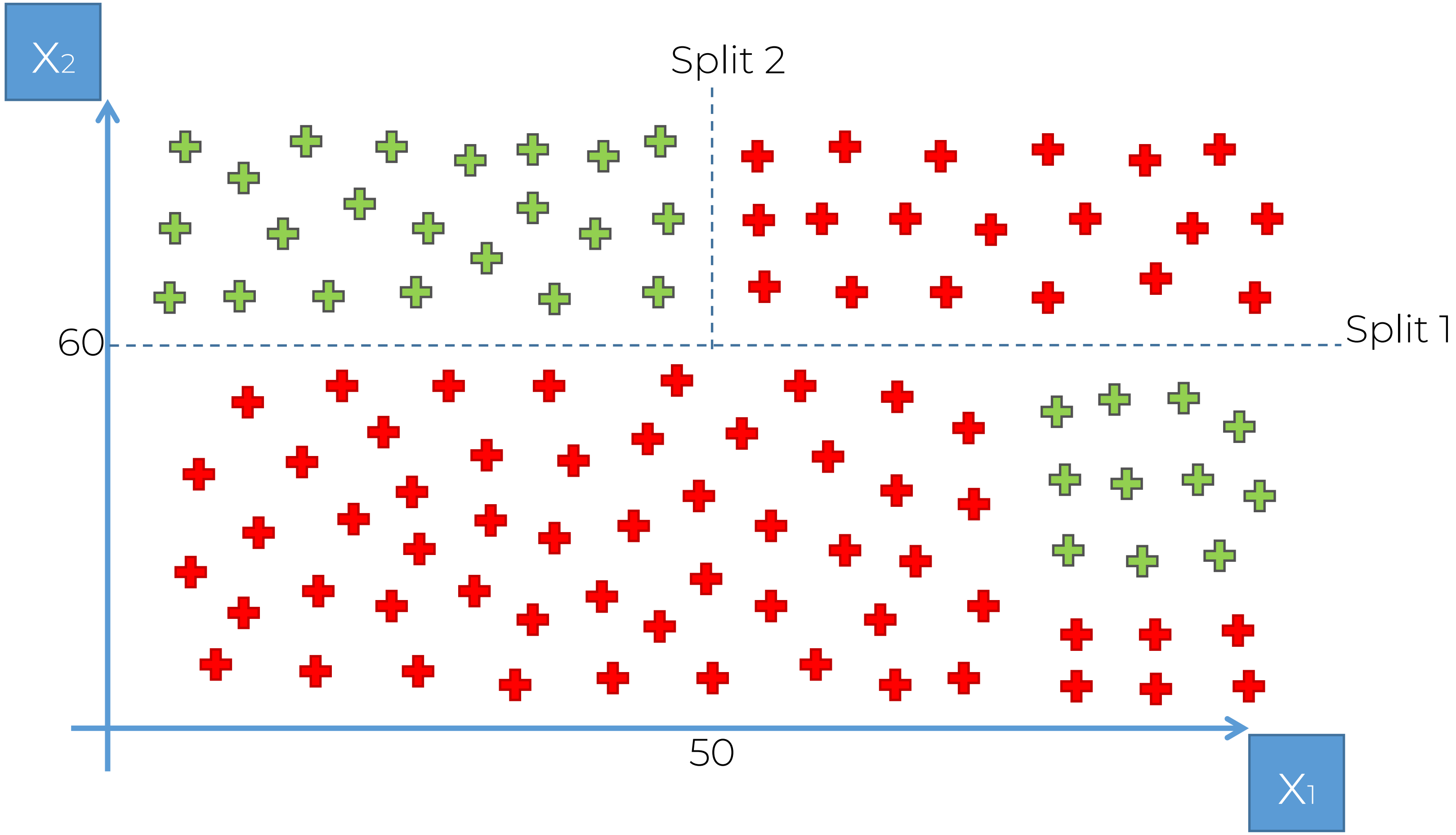
Decision Tree Intuition



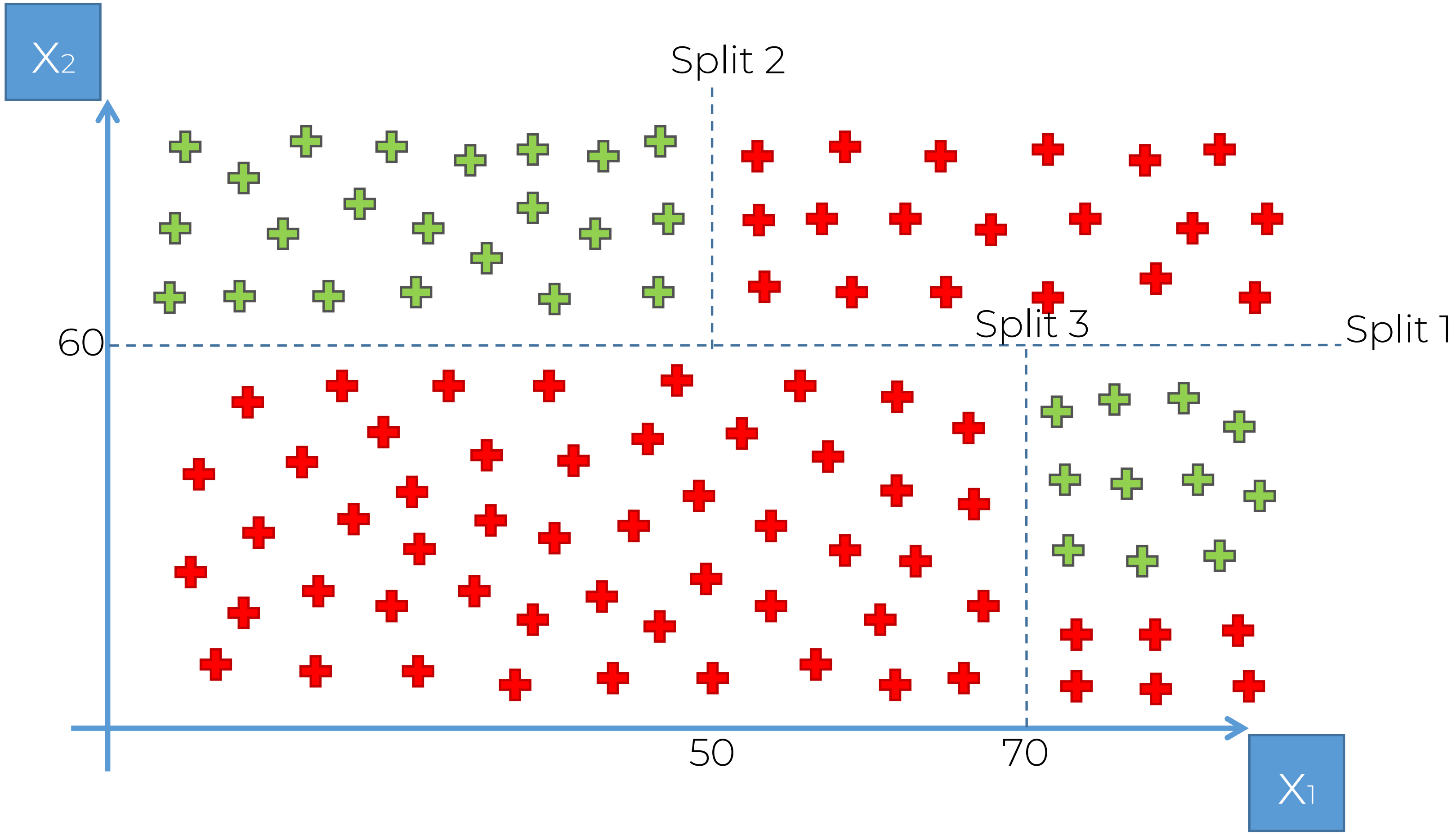
Decision Tree Intuition



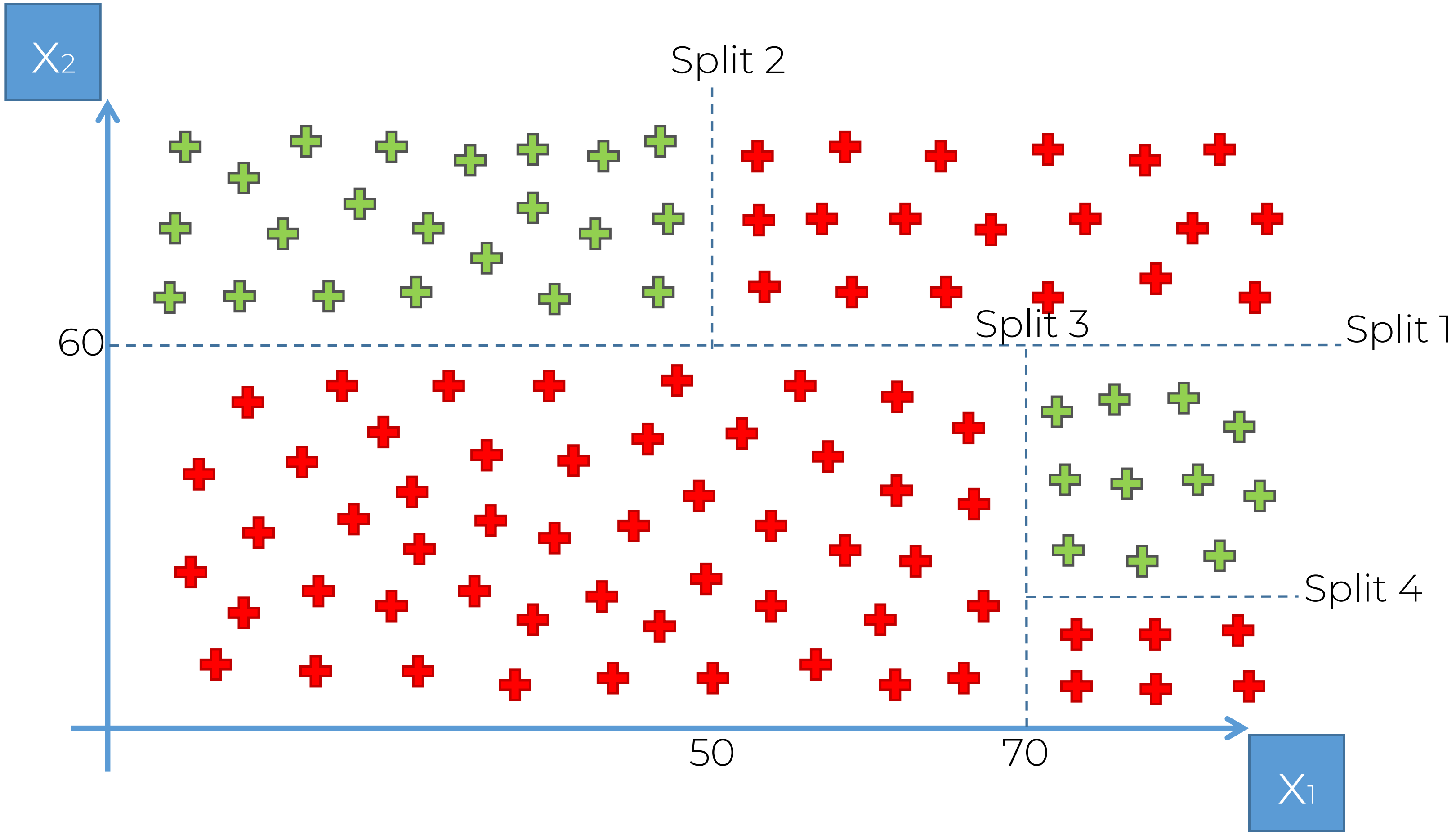
Decision Tree Intuition



Decision Tree Intuition



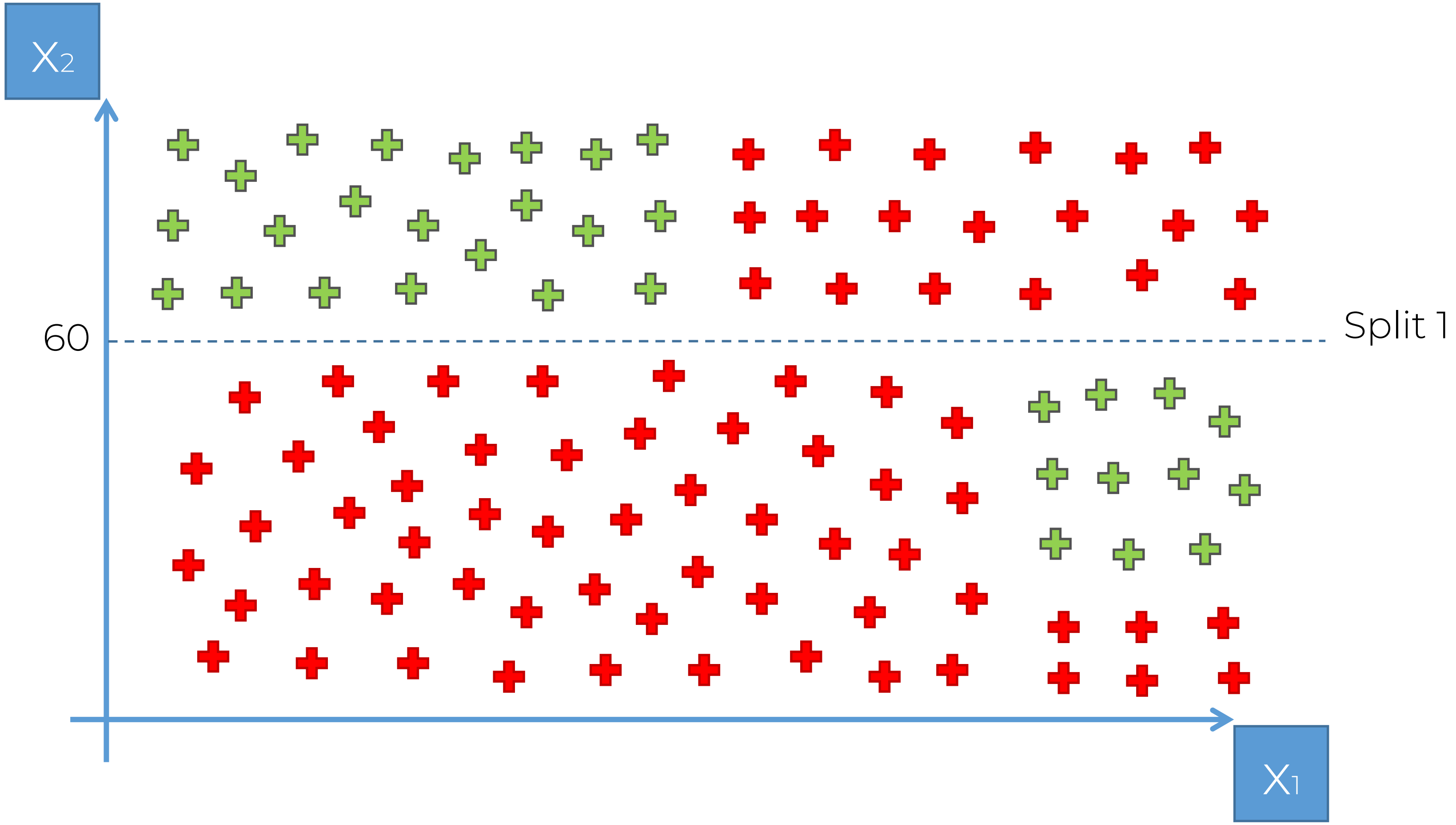
Decision Tree Intuition



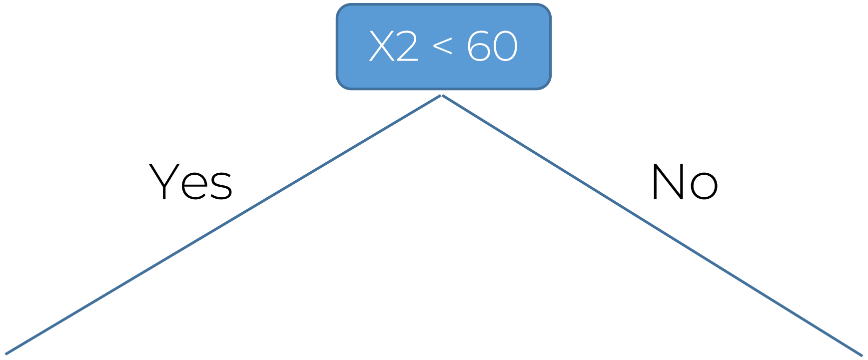
Decision Tree Intuition

Rewind…

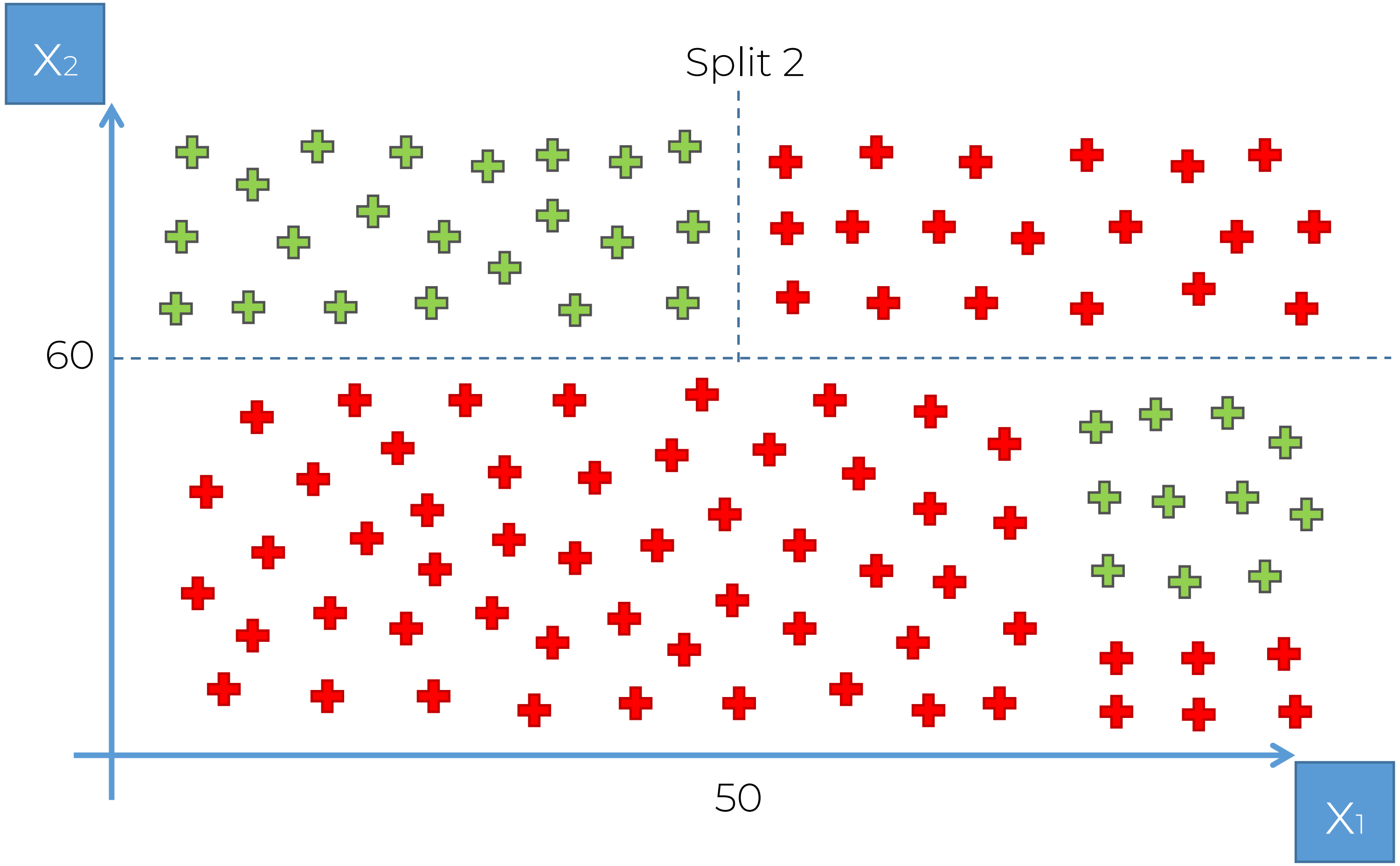
Split 1



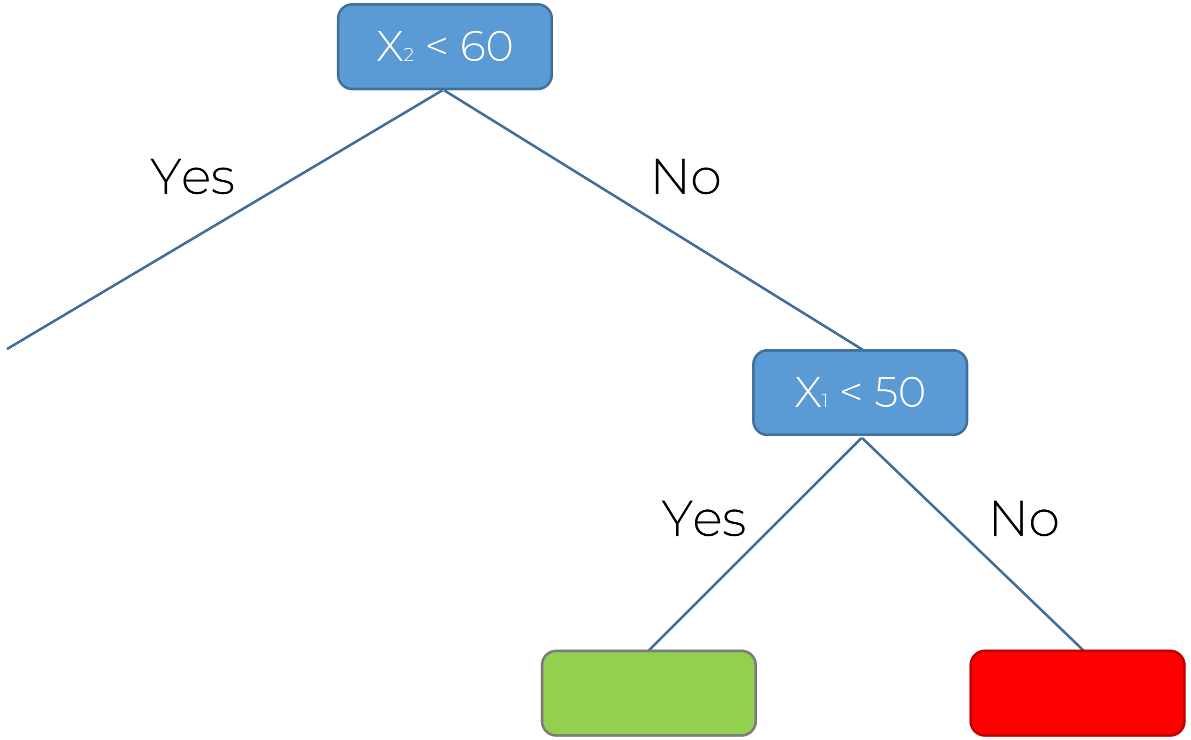
Split 1



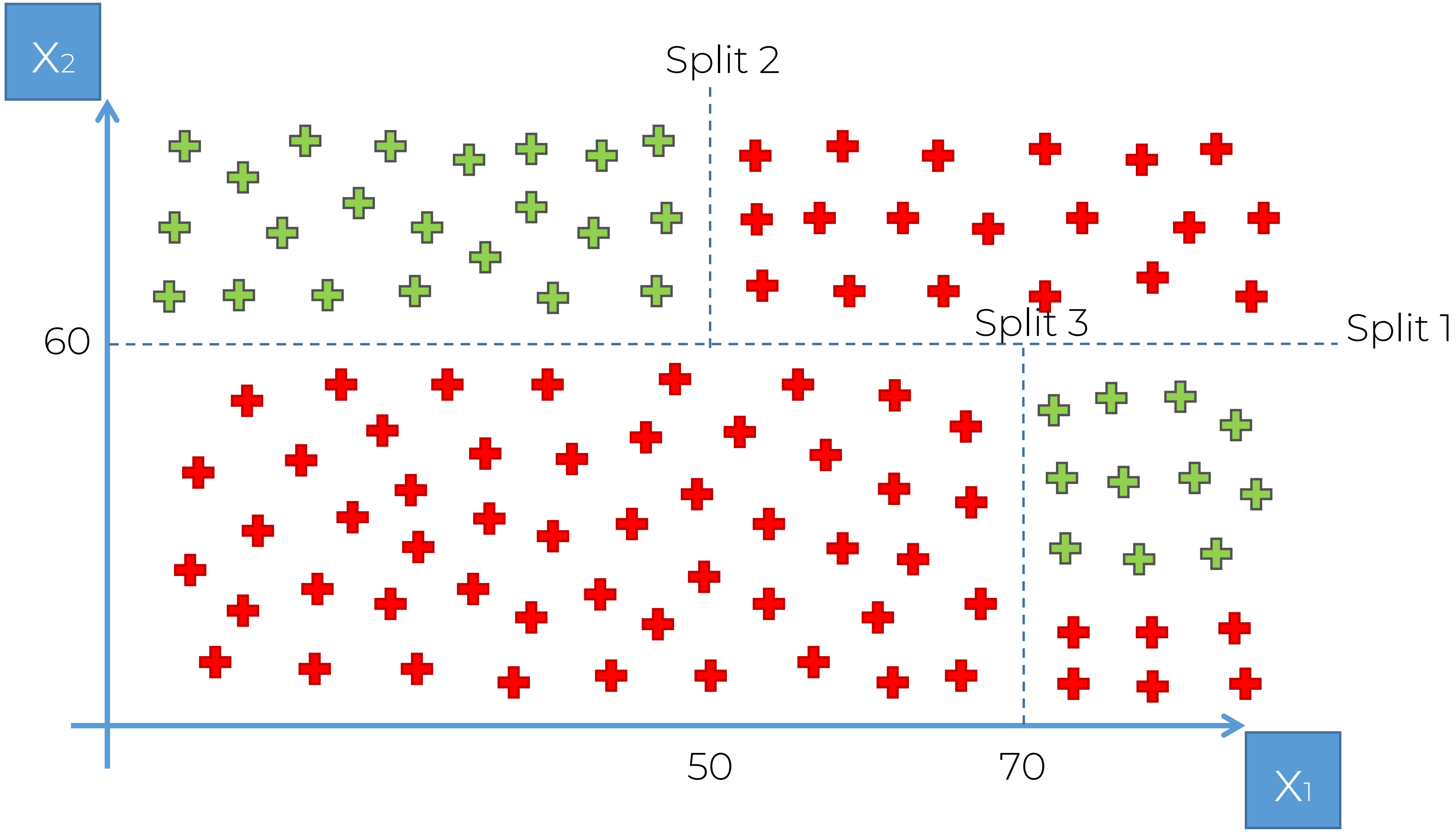
Split 2



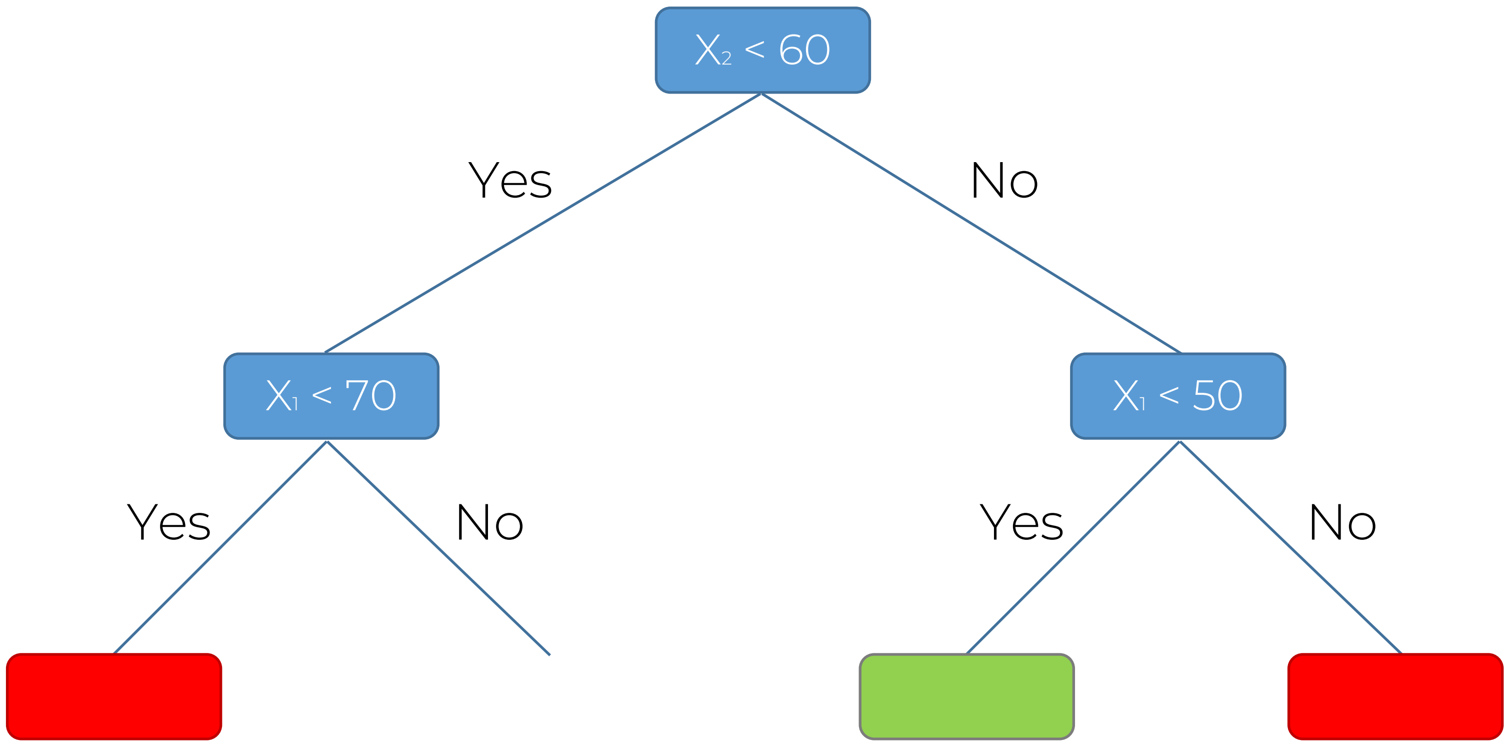
Split 2



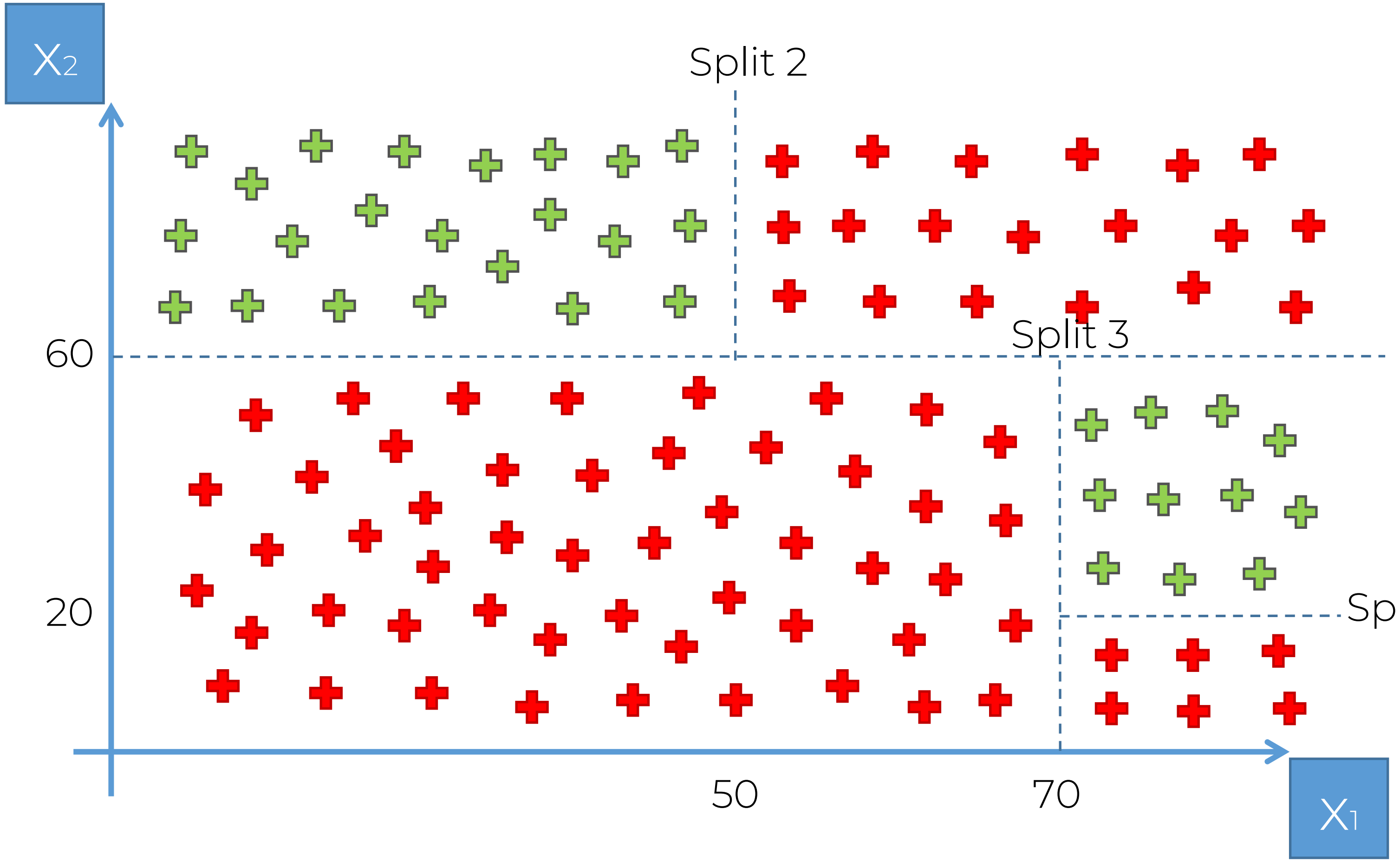
Split 3



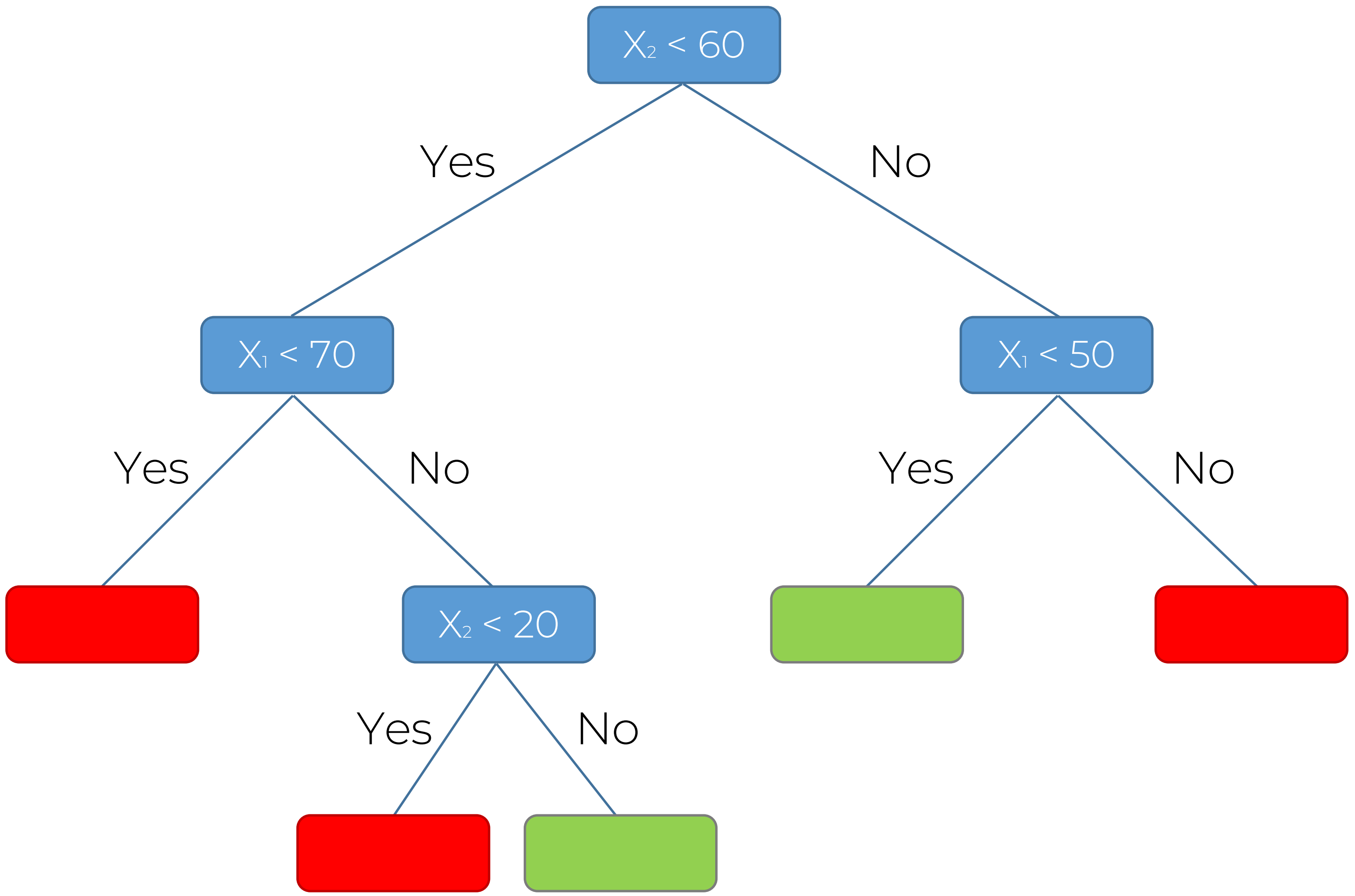
Split 3



Split 4



Split 4



Decision Trees

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

Random Forest Intuition

Ensemble Learning

Random Forest Intuition

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

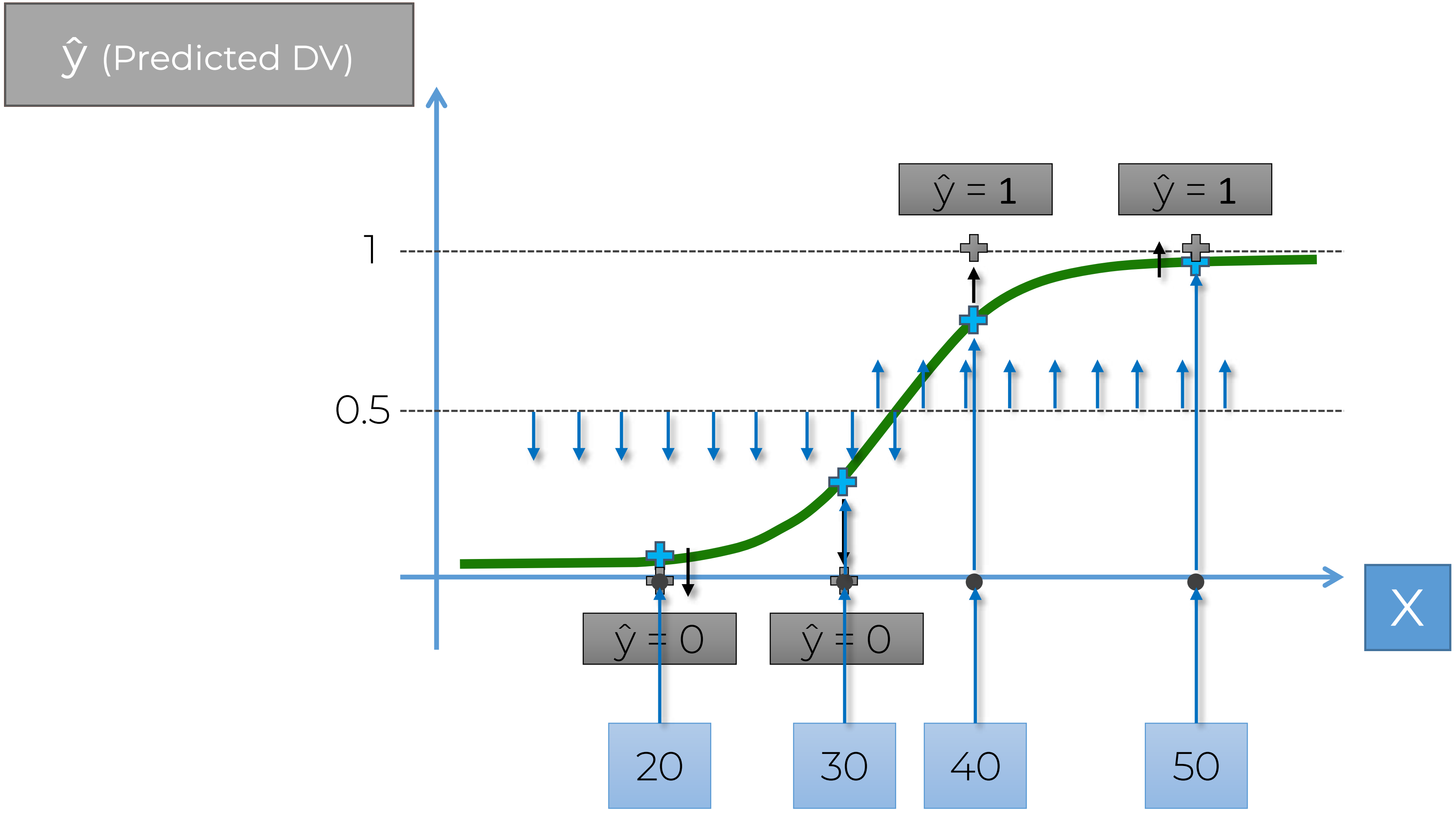
STEP 2: Build the Decision Tree associated to these K data points.

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

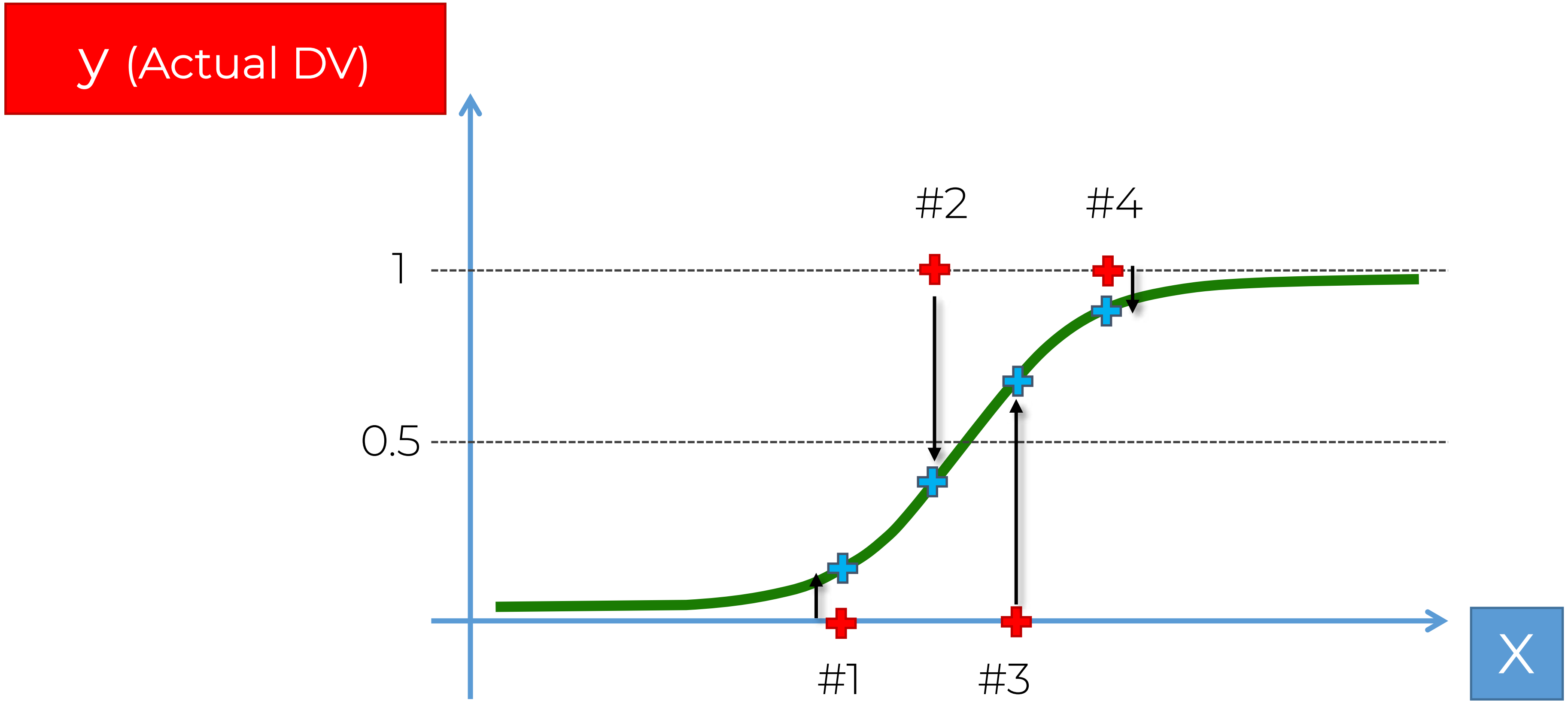
STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

False Positives
False Negatives

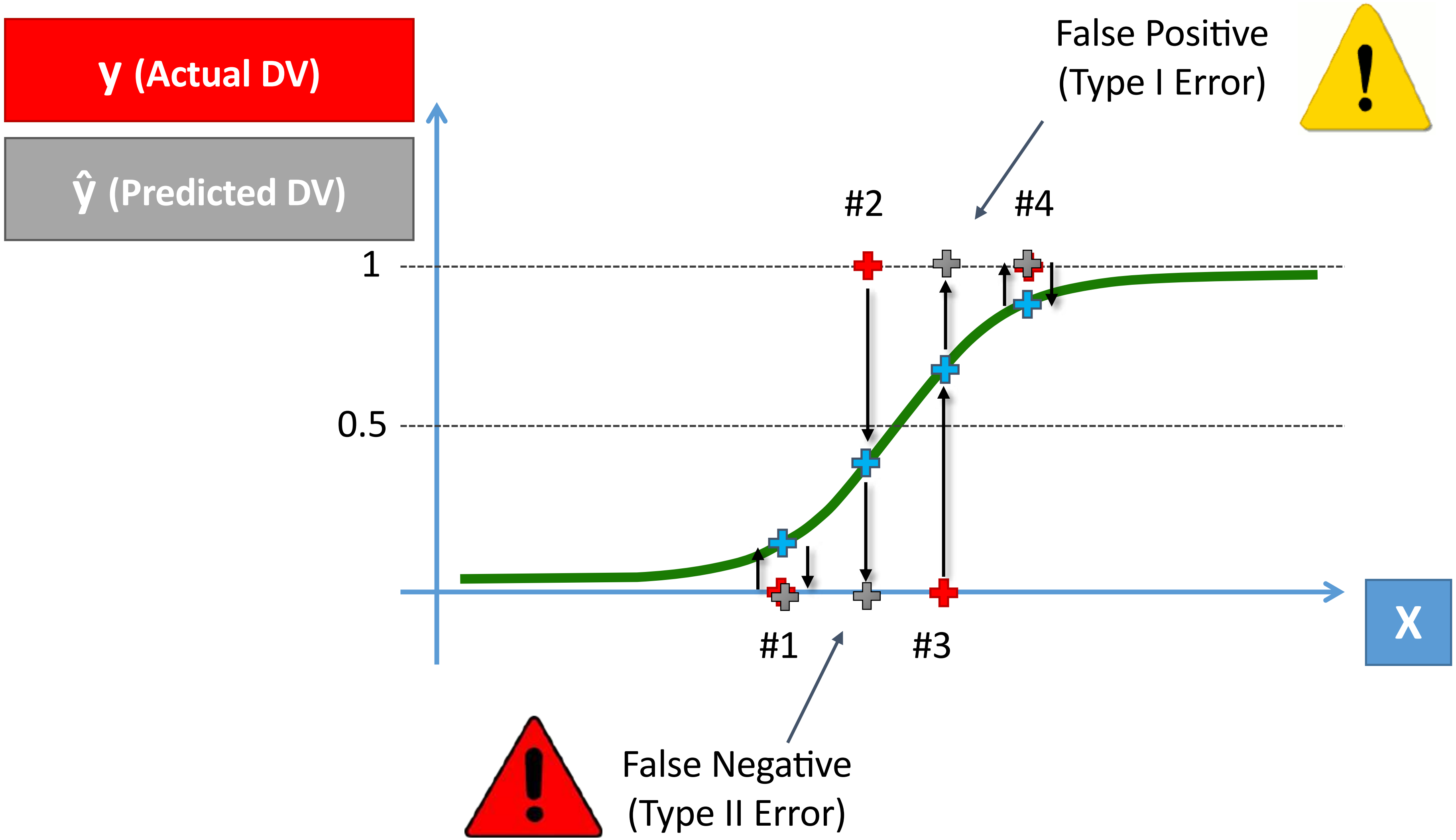
False Positives & Negatives



False Positives & Negatives



False Positives & Negatives



Fin.

Confusion Matrix & Accuracy

Confusion Matrix & Accuracy

		Prediction	
		NEG	POS
Actual	NEG	TRUE NEG	FALSE POS
	POS	FALSE NEG	TRUE POS

Type II Error
(False Negatives)

Type I Error
(False Positives)

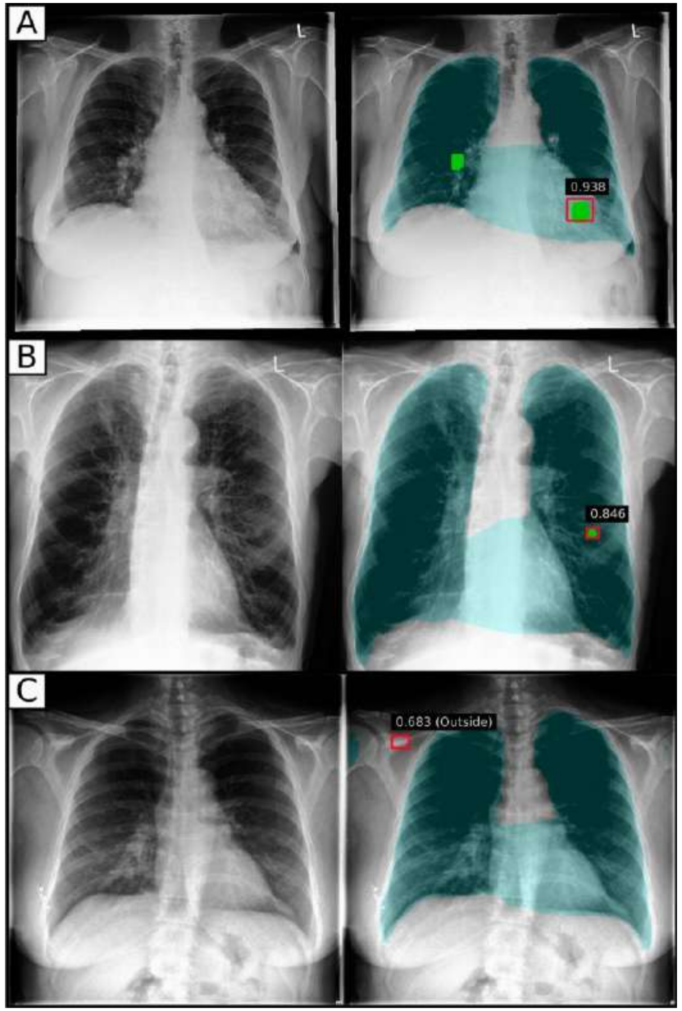


Image source: nature.com

Confusion Matrix & Accuracy

		Prediction	
		NEG	POS
Actual	NEG	43	12
	POS	4	41

Type II Error
(False Negatives)

Type I Error
(False Positives)

Accuracy Rate and Error Rate:

$$AR = \frac{Correct}{Total} = \frac{TN + TP}{Total} = \frac{84}{100} = 84\%$$

$$ER = \frac{Incorrect}{Total} = \frac{FP + FN}{Total} = \frac{16}{100} = 16\%$$

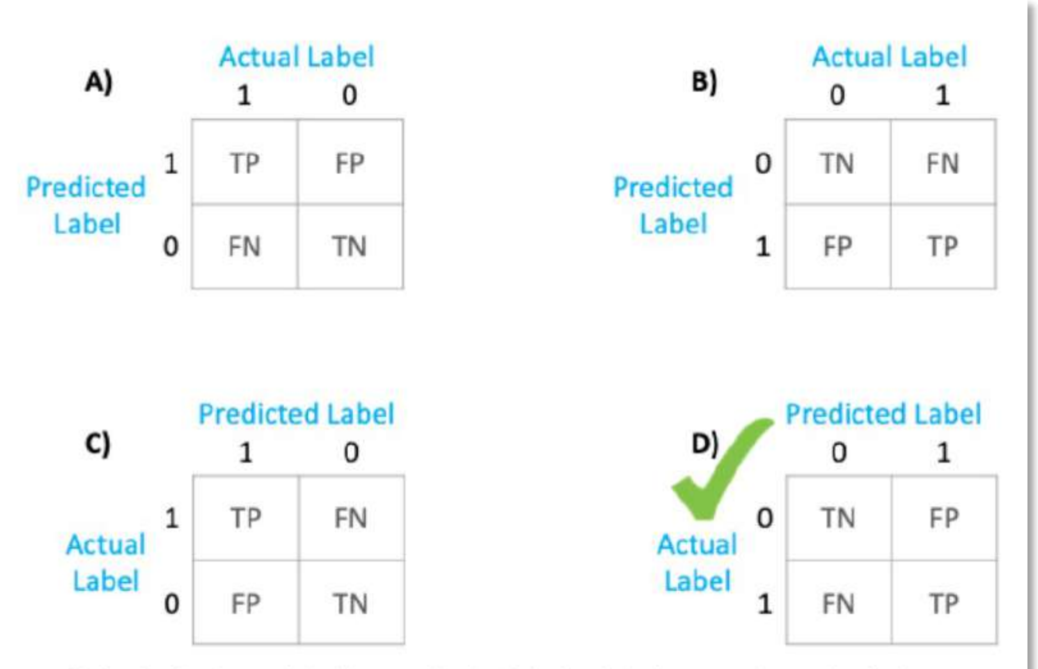
Additional Reading

Understanding the Confusion Matrix from Scikit learn

Samarth Agrawal (2021)

Link:

https://towardsdatascience.com/understanding-the-confusion-matrix-from-scikit-learn-c51d88929c79



Referring back to original image. Option D is the default output. Image by Author

Accuracy Paradox

Accuracy Paradox

		ŷ (Predicted DV)	
		0	1
y (Actual DV)	0	9,700	150
	1	50	100

Scenario 1:

Accuracy Rate = Correct / Total

AR = 9,800/10,000 = 98%

Accuracy Paradox

		ŷ (Predicted DV)	
		0	1
y (Actual DV)	0	9,850	0
	1	150	0

<u>Scenario 1:</u>

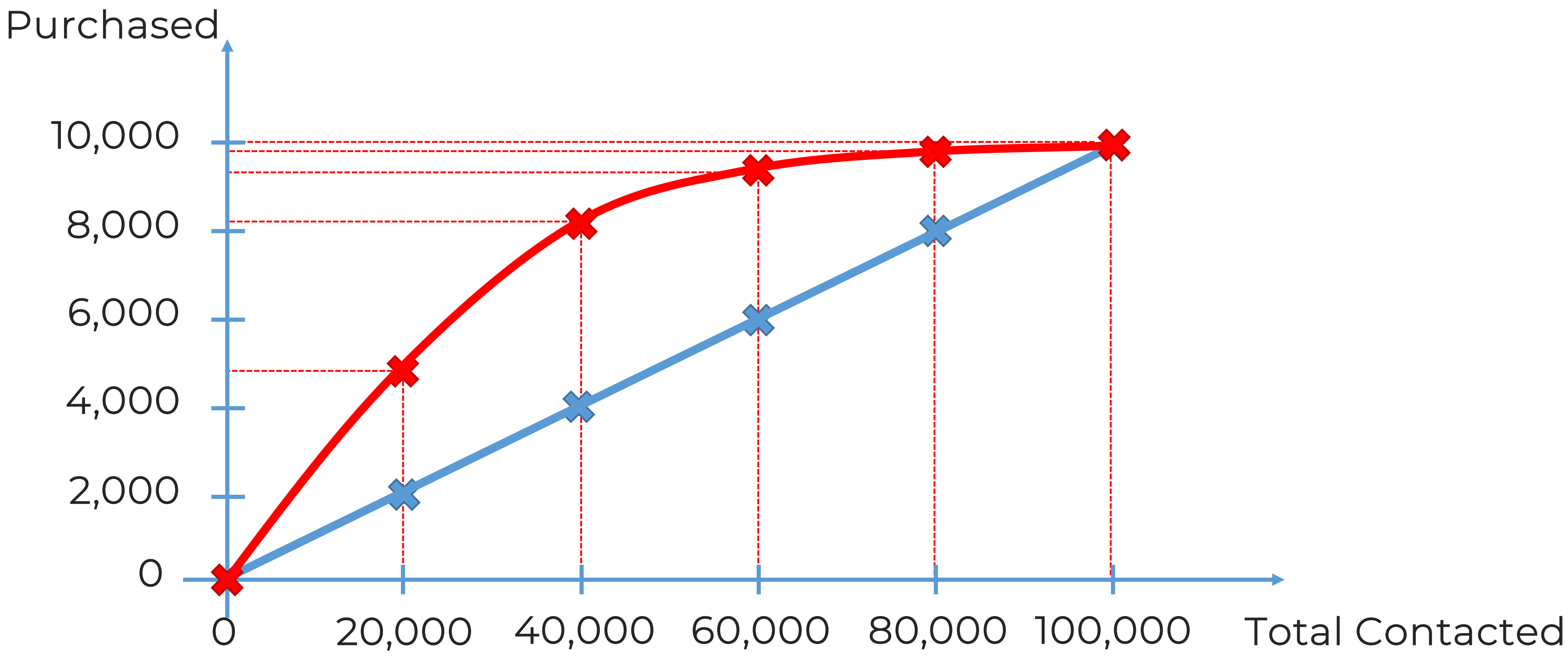
Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

<u>Scenario 2:</u>

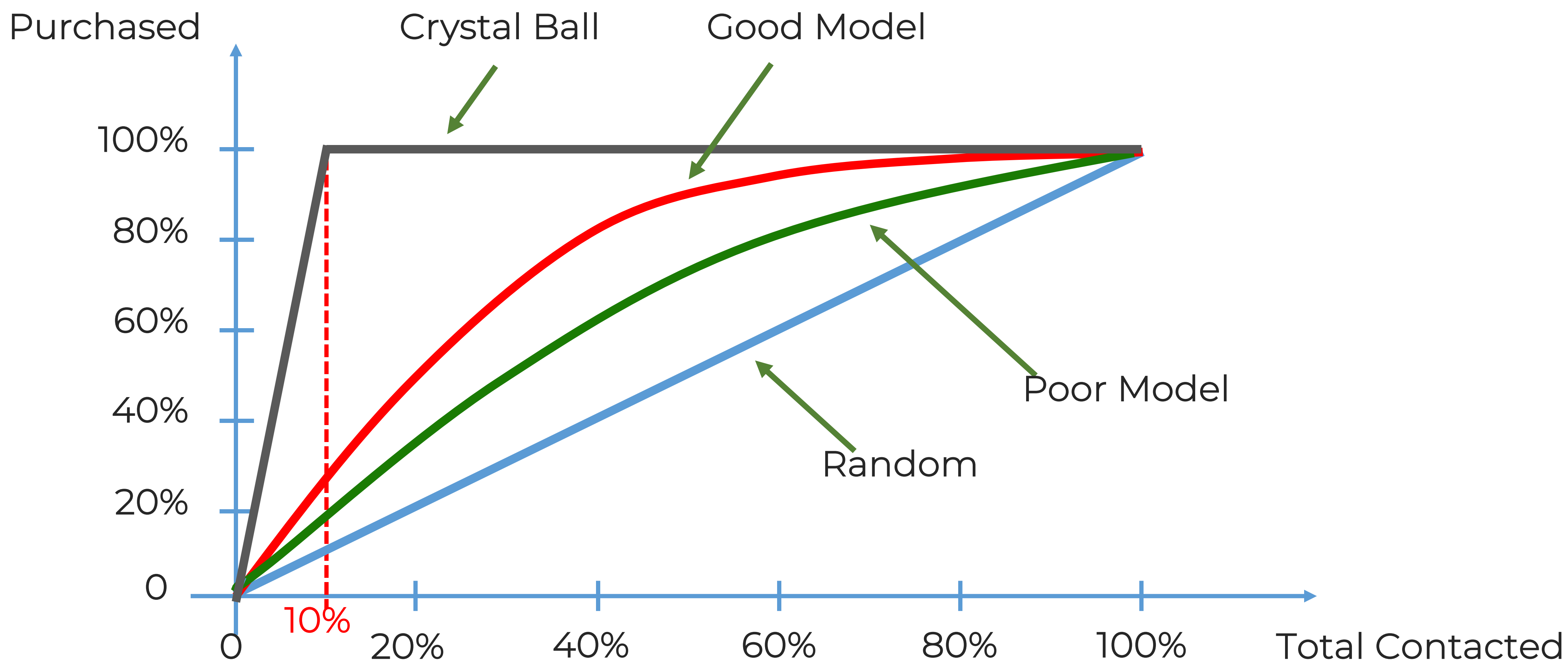
Accuracy Rate = Correct / Total
AR = 9,850/10,000 = 98.5%

Cumulative Accuracy Profile

CAP


Purchased
10,000
8,000
6,000
4,000
2,000
0
0
20,000
40,000
60,000
80,000
100,000
Total Contacted

CAP


Purchased
Crystal Ball
Good Model
100%
80%
60%
40%
20%
0
Poor Model
Random
10%
0
20%
40%
60%
80%
100%
Total Contacted

CAP

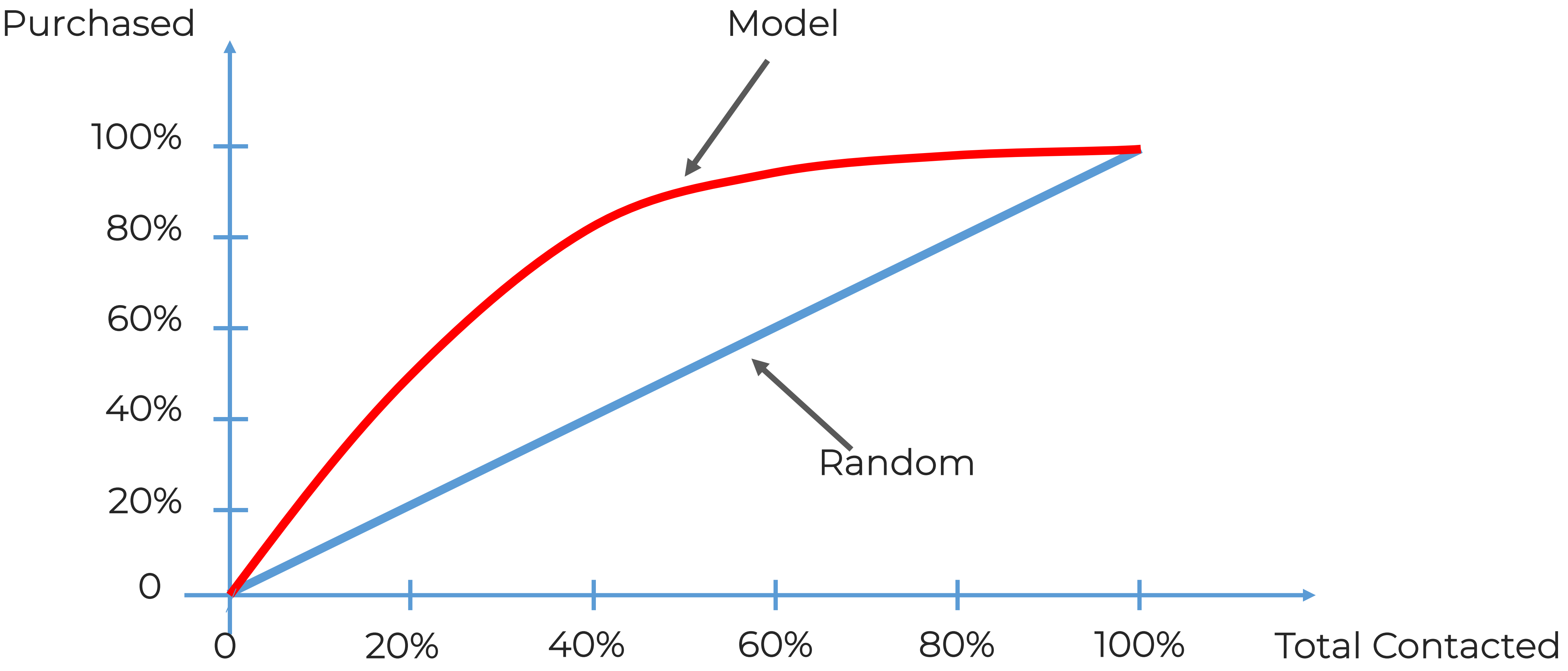
Note:

CAP = Cumulative Accuracy Profile

$\neq$

ROC = Receiver Operating Characteristic

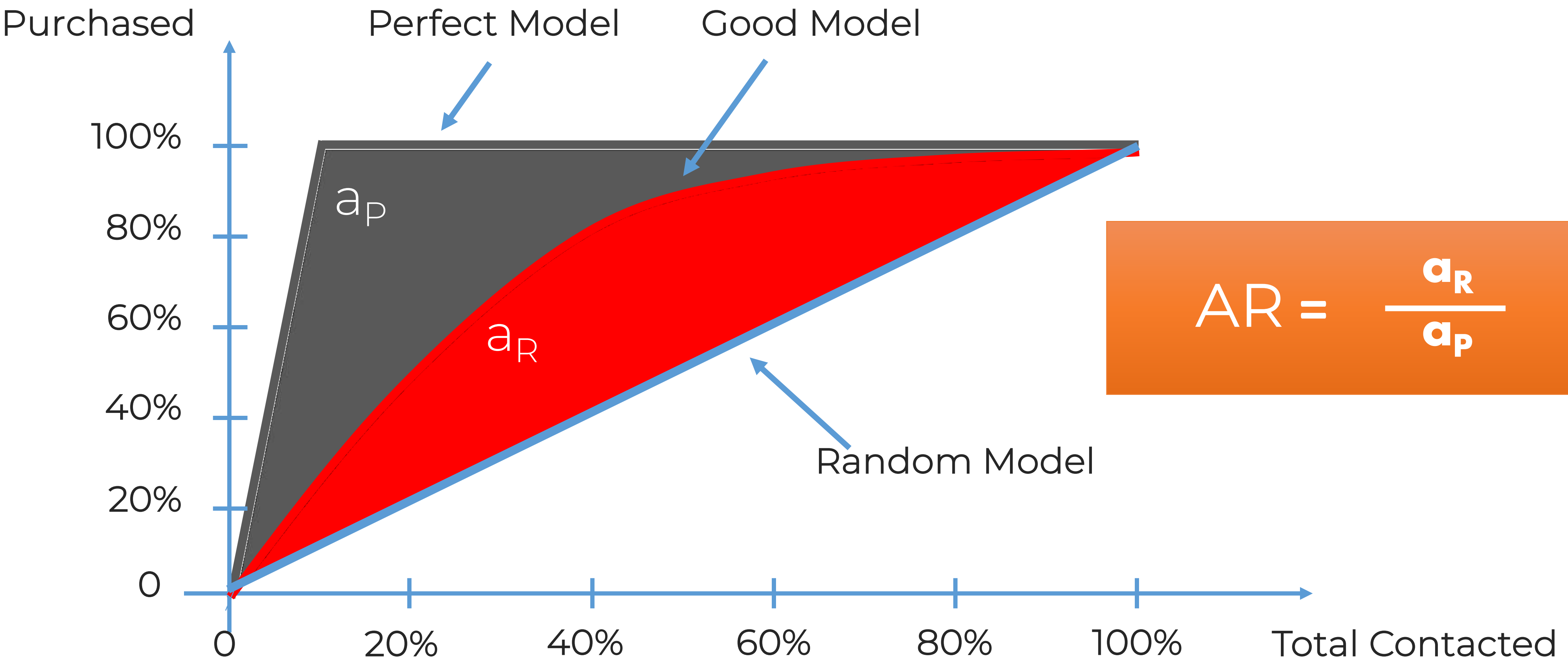
CAP



Purchased
Model
100%
80%
60%
40%
20%
0
Random
0
20%
40%
60%
80%
100%
Total Contacted

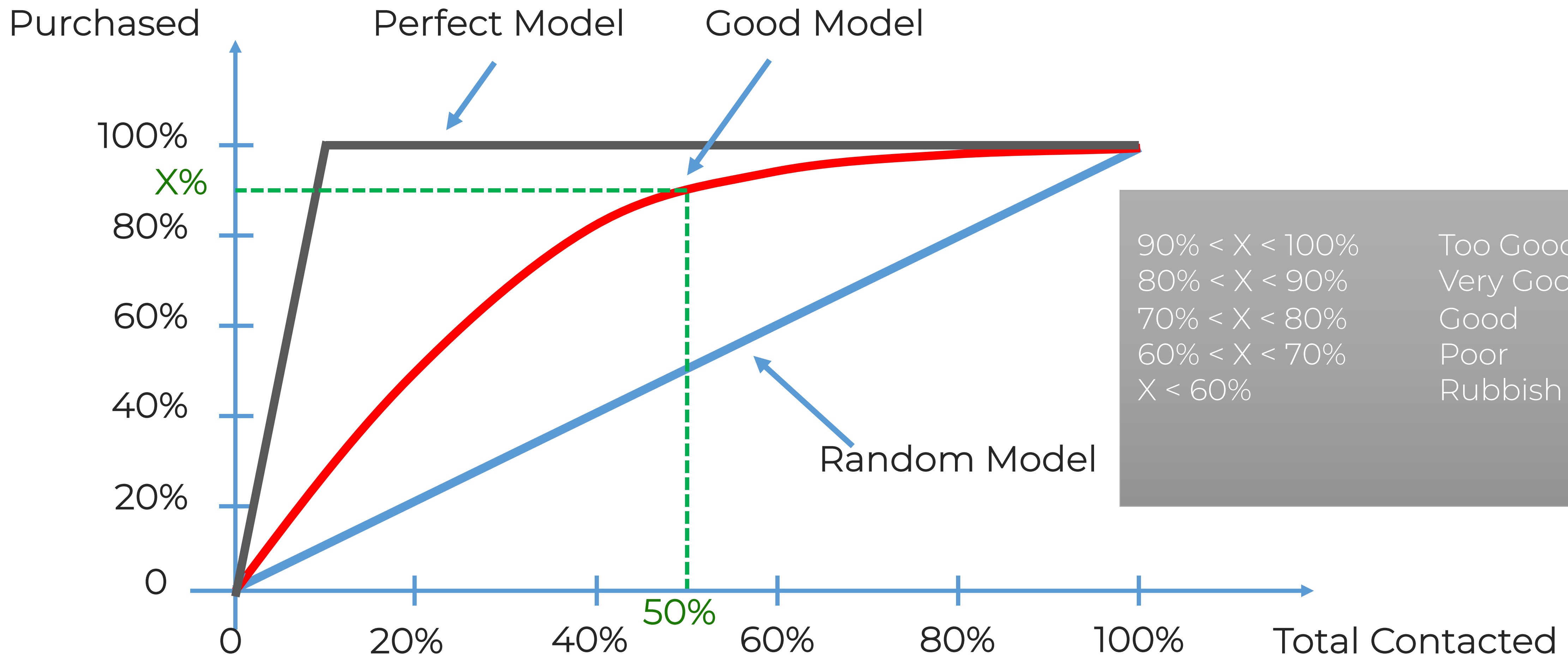
CAP Analysis

CAP Analysis



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

CAP Analysis



90% < X < 100%	Too Good
80% < X < 90%	Very Good
70% < X < 80%	Good
60% < X < 70%	Poor
X < 60%	Rubbish

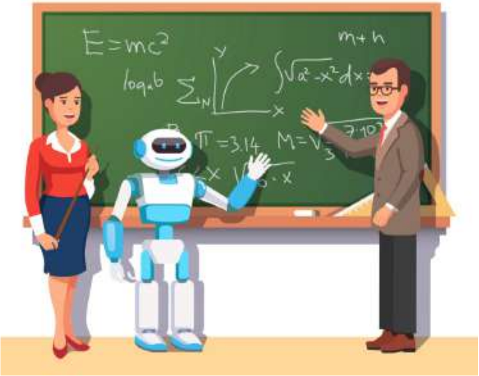
Clustering

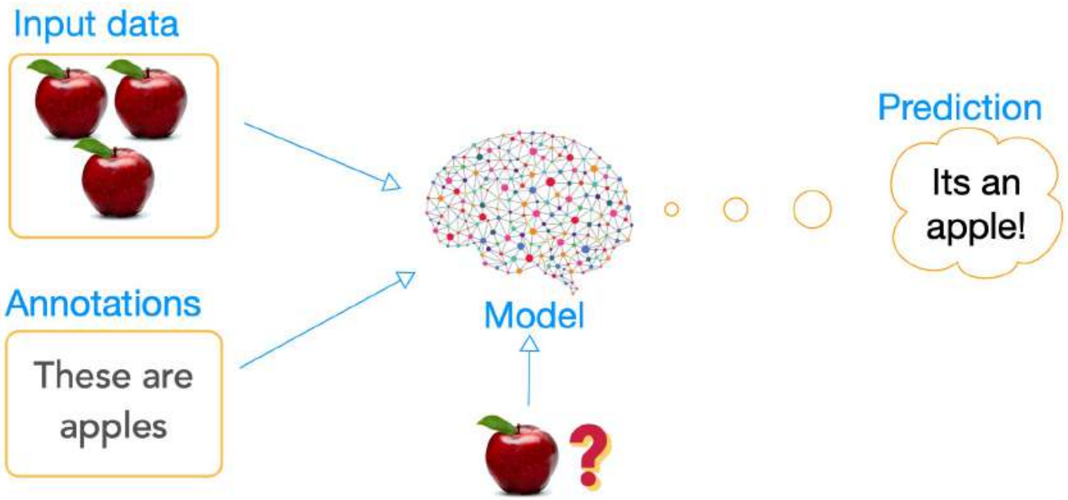
What is Clustering?

Clustering – grouping <u>unlabelled</u> data

What is Clustering?

Supervised Learning
(e.g. Regression, Classification)





Unsupervised Learning
(e.g. Clustering)

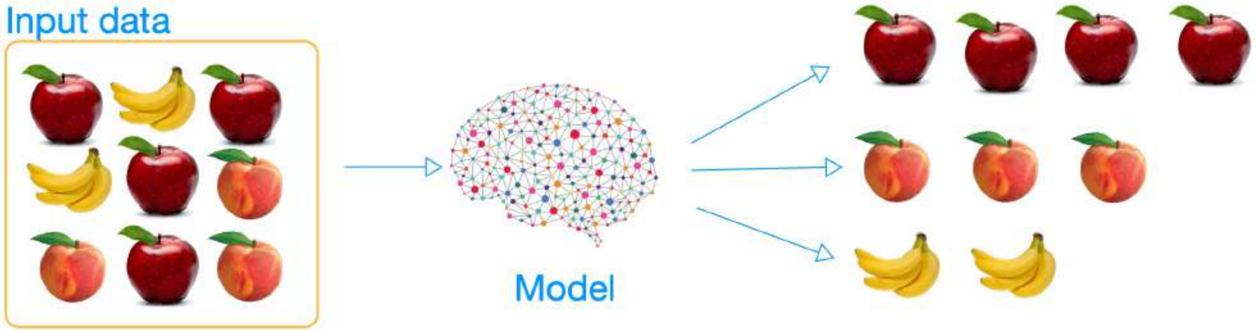
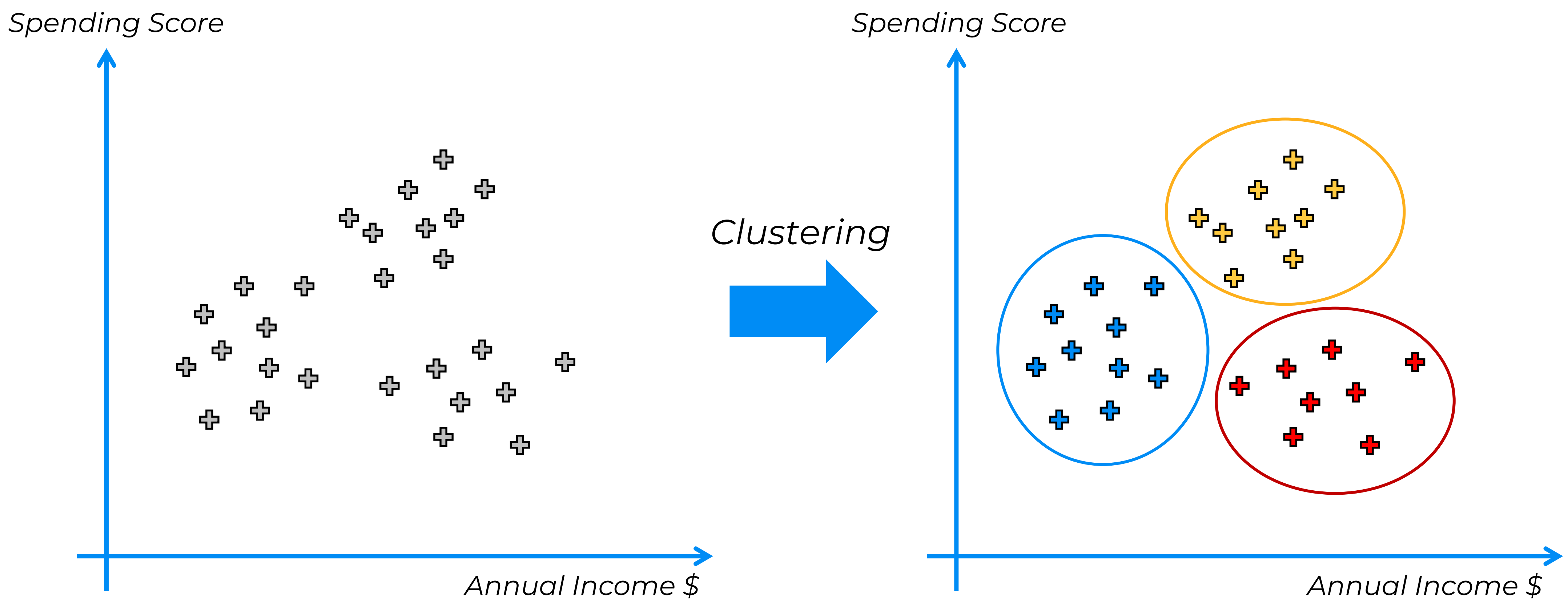


Image source: mdpi.com/2073-8994/10/12/734

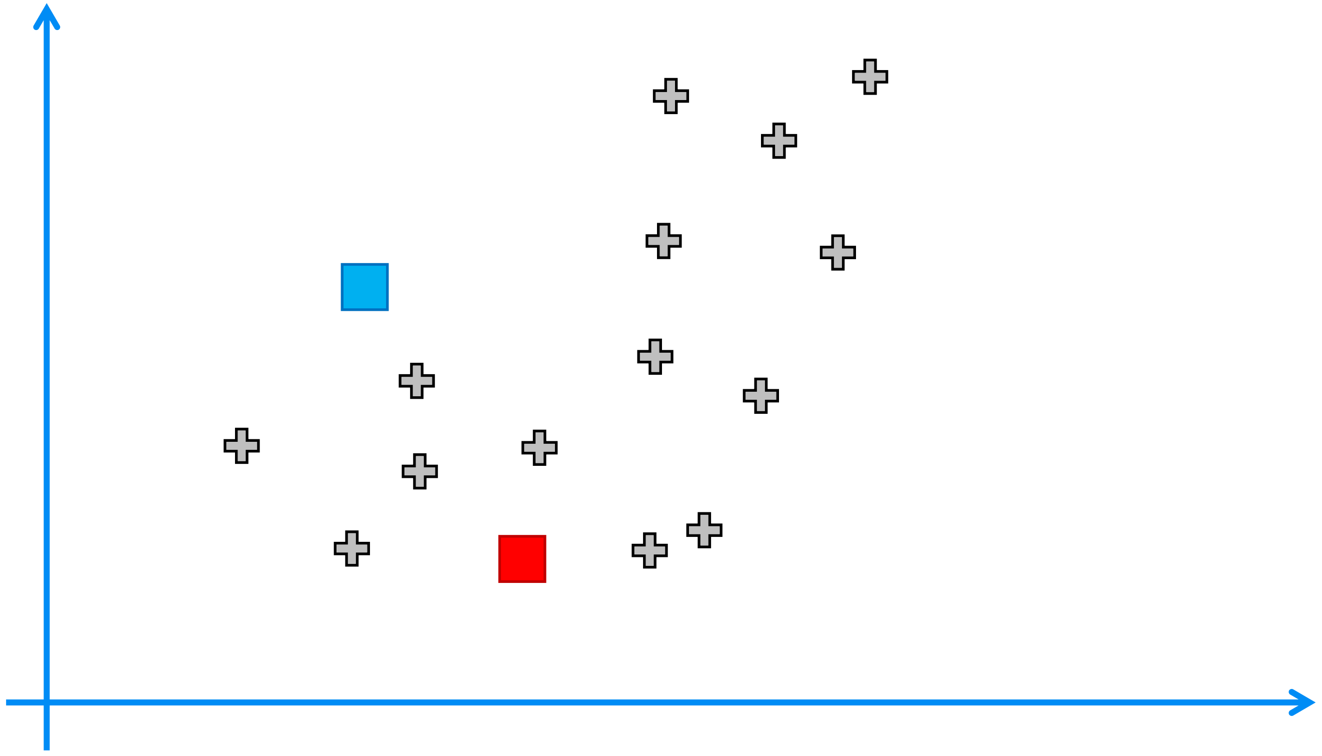
What is Clustering?



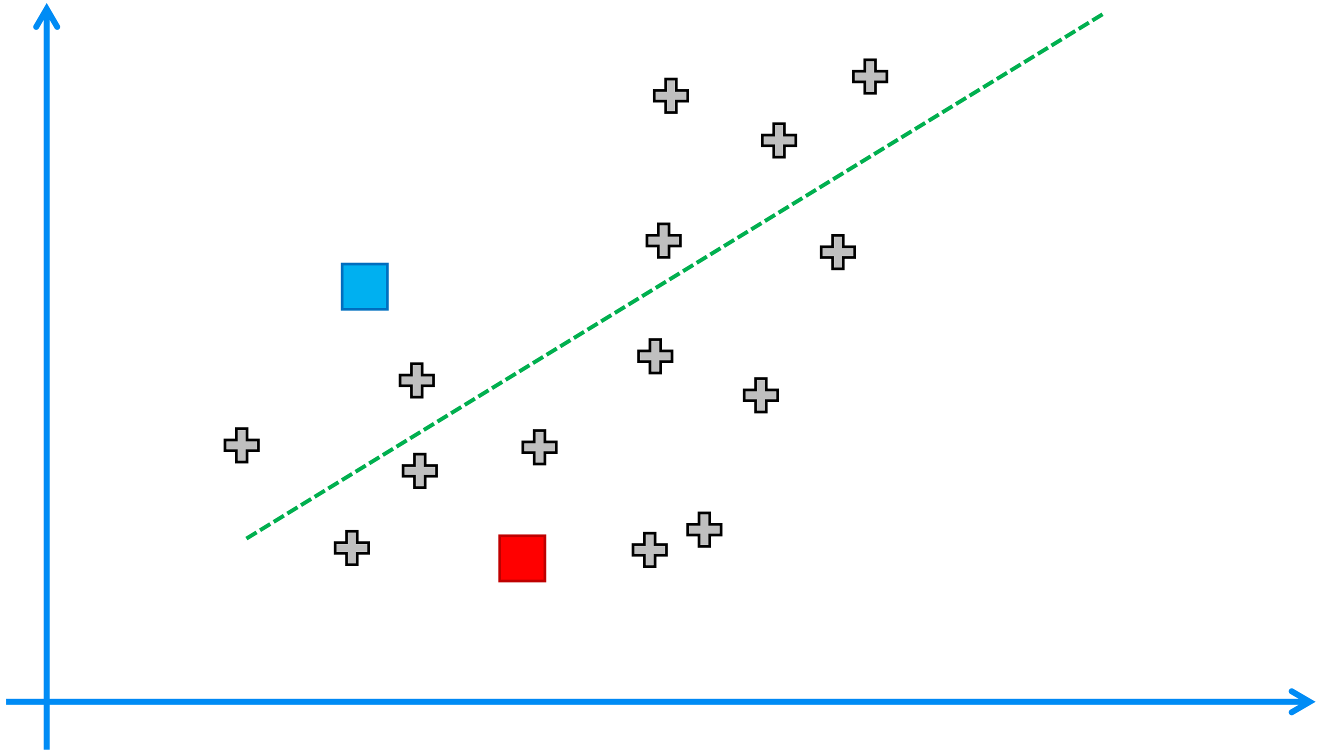
K-Means Clustering

K-Means Clustering

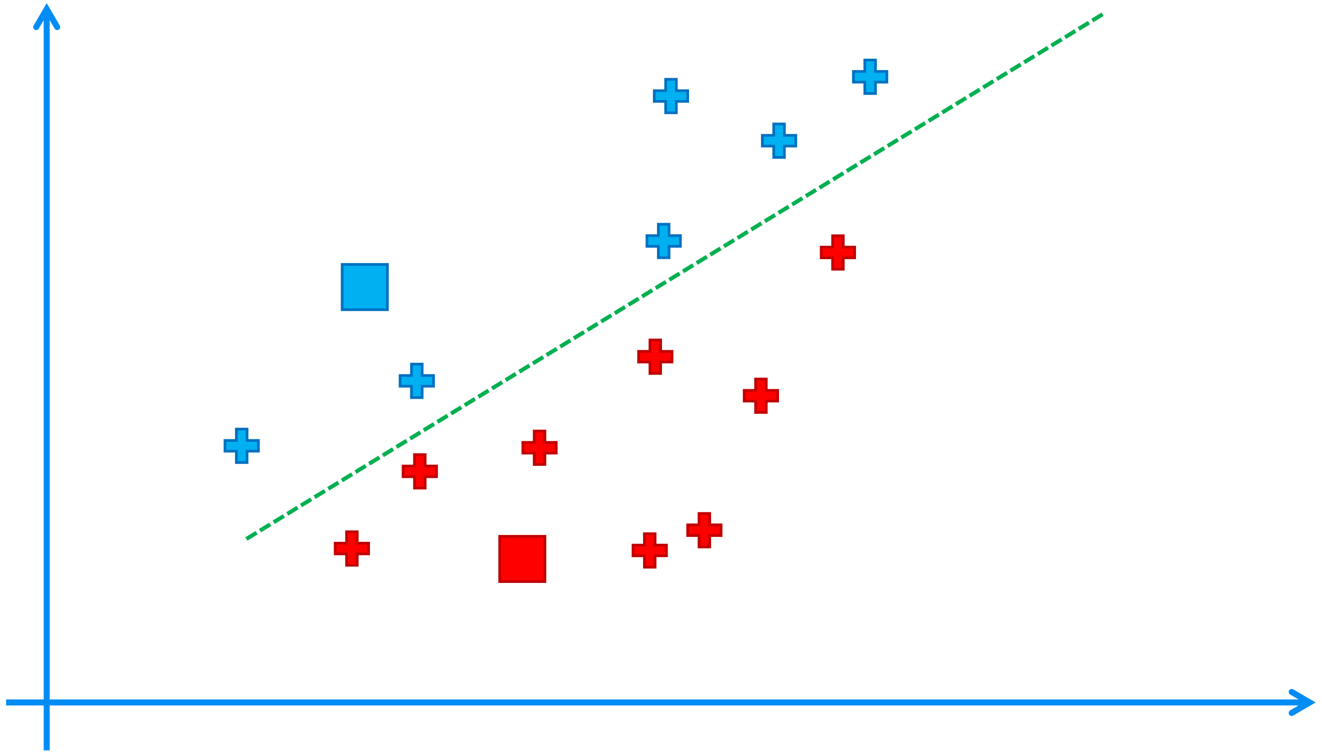
K-Means Clustering



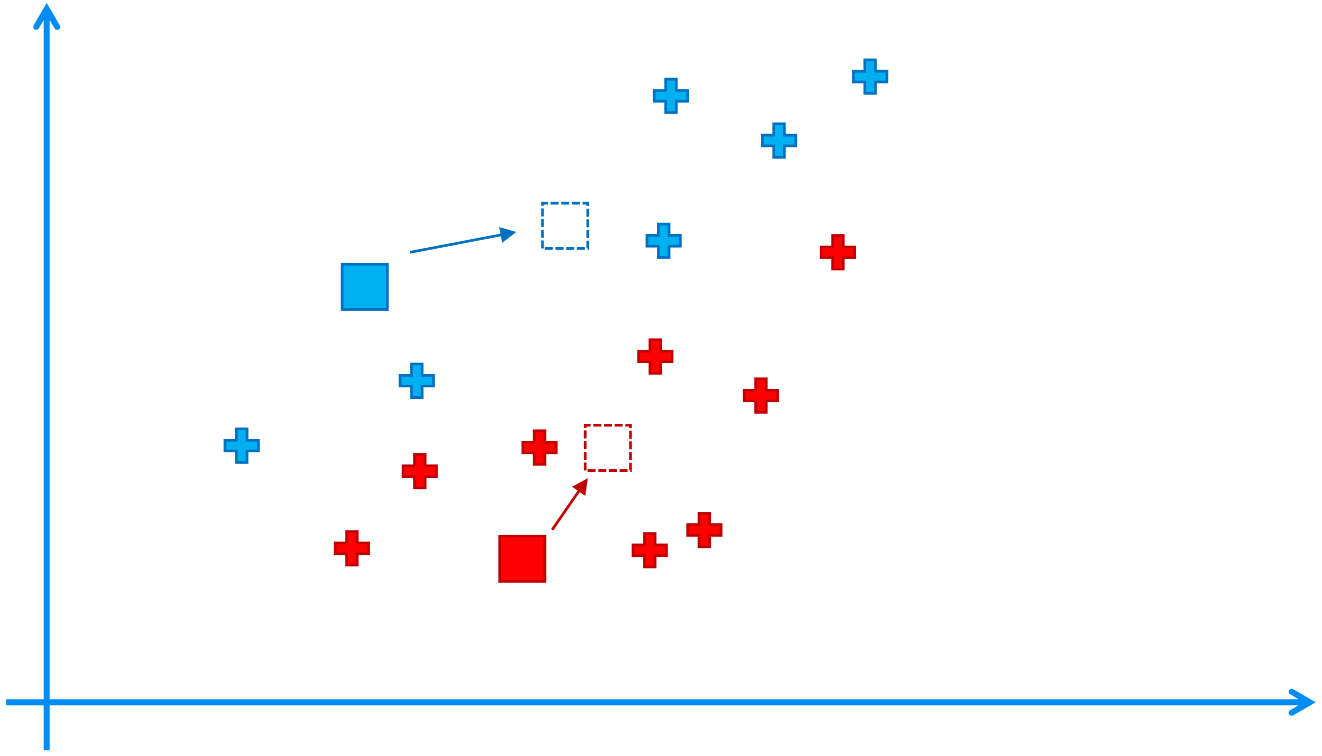
K-Means Clustering



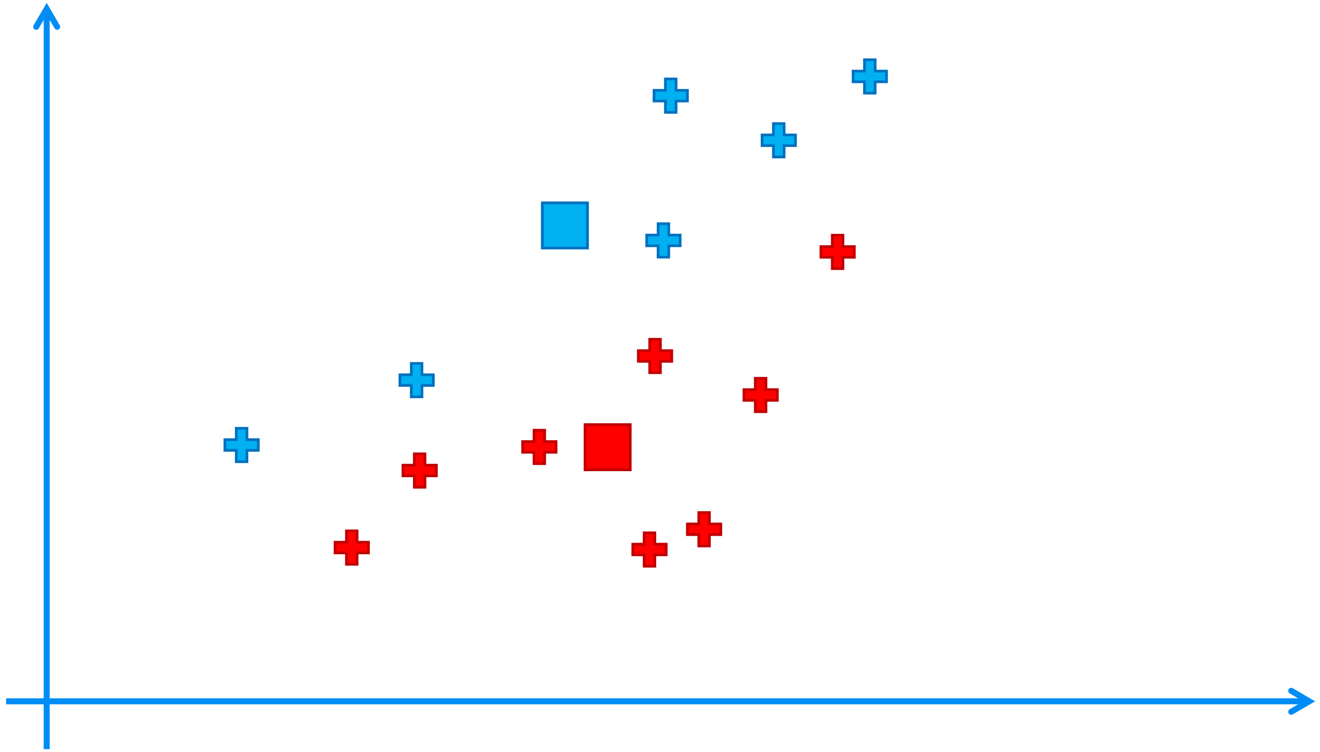
K-Means Clustering



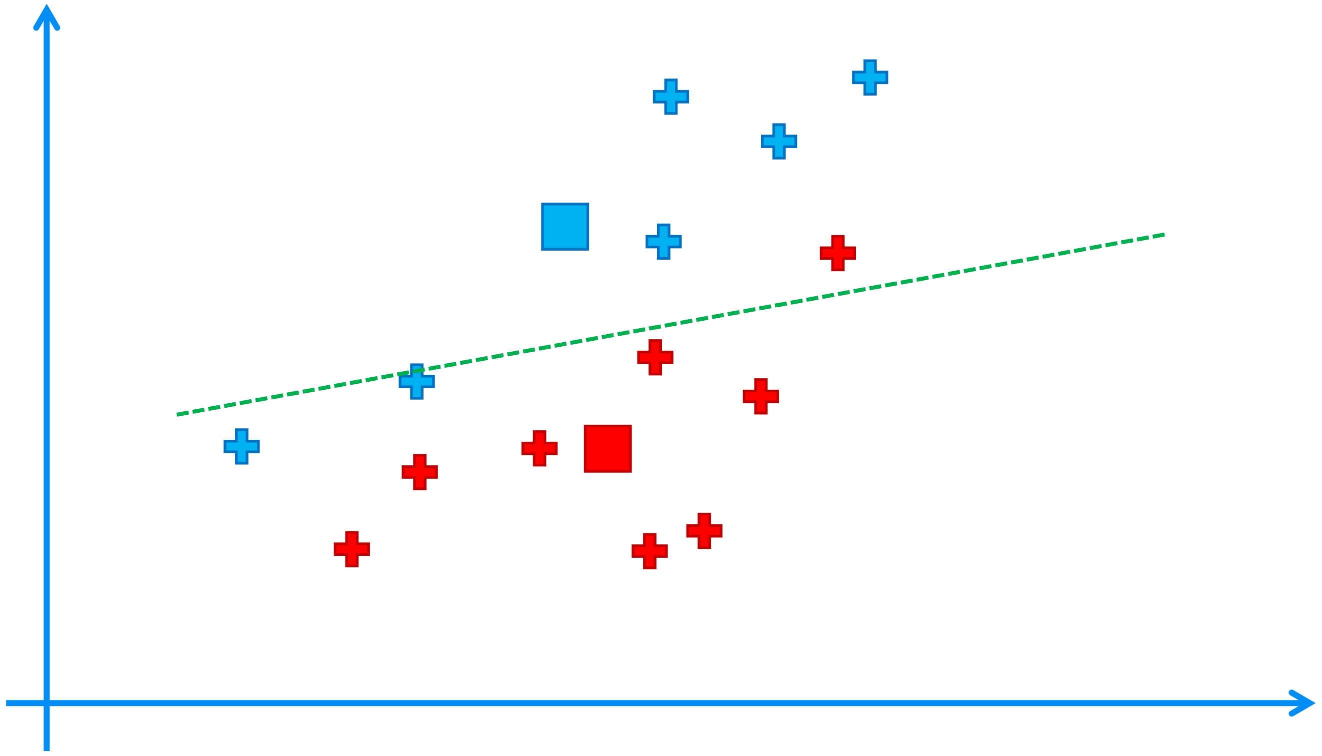
K-Means Clustering



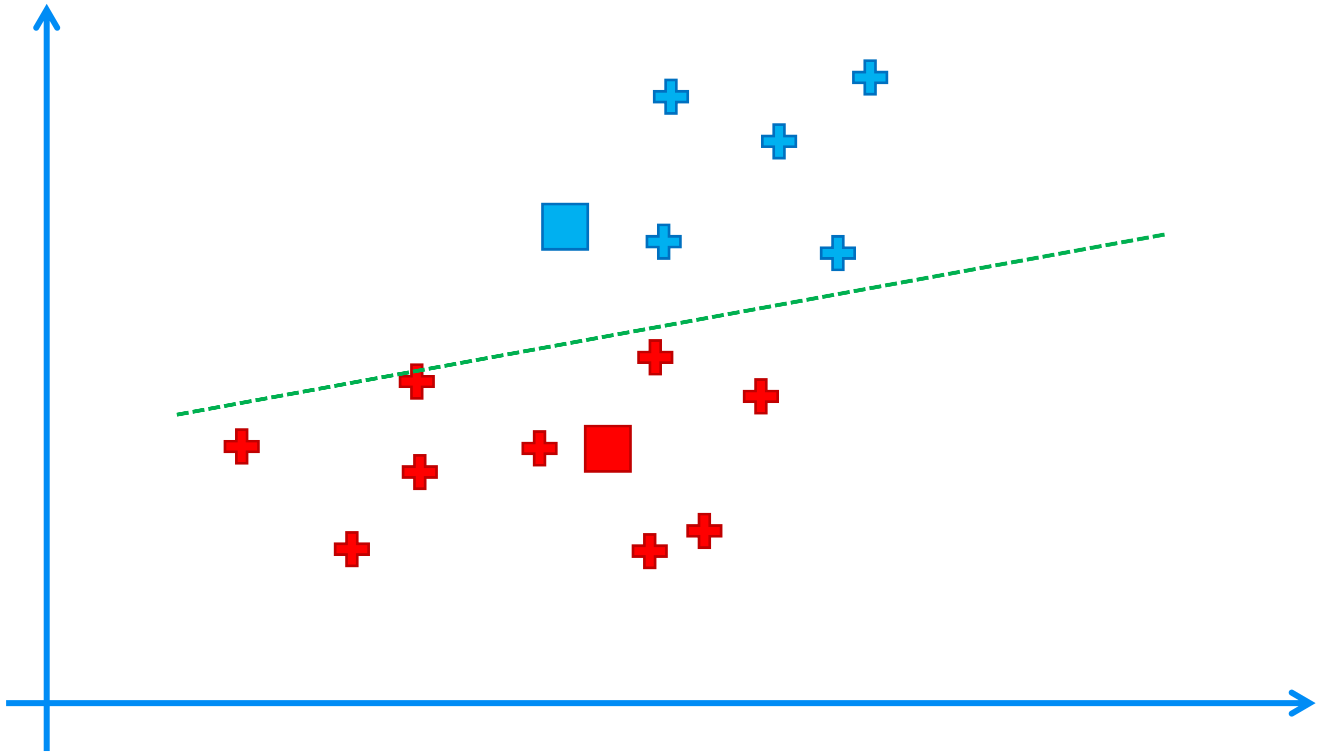
K-Means Clustering



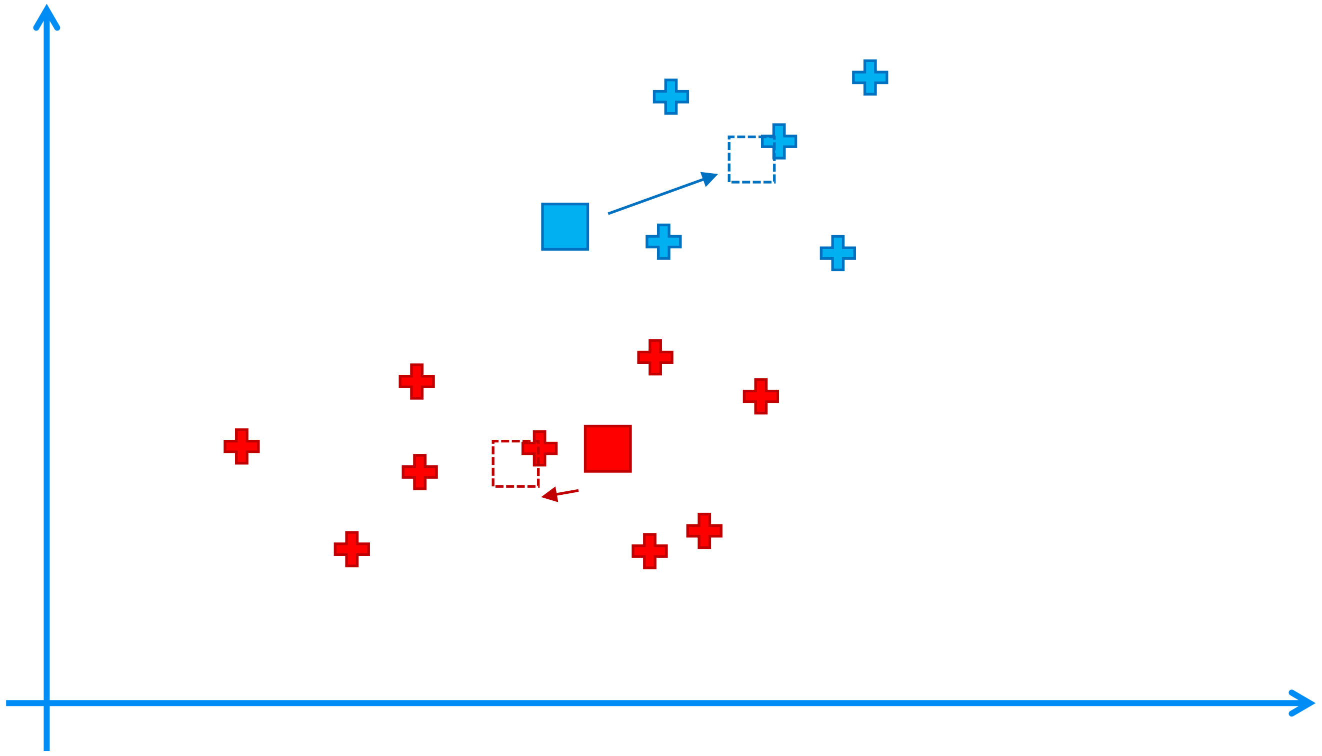
K-Means Clustering



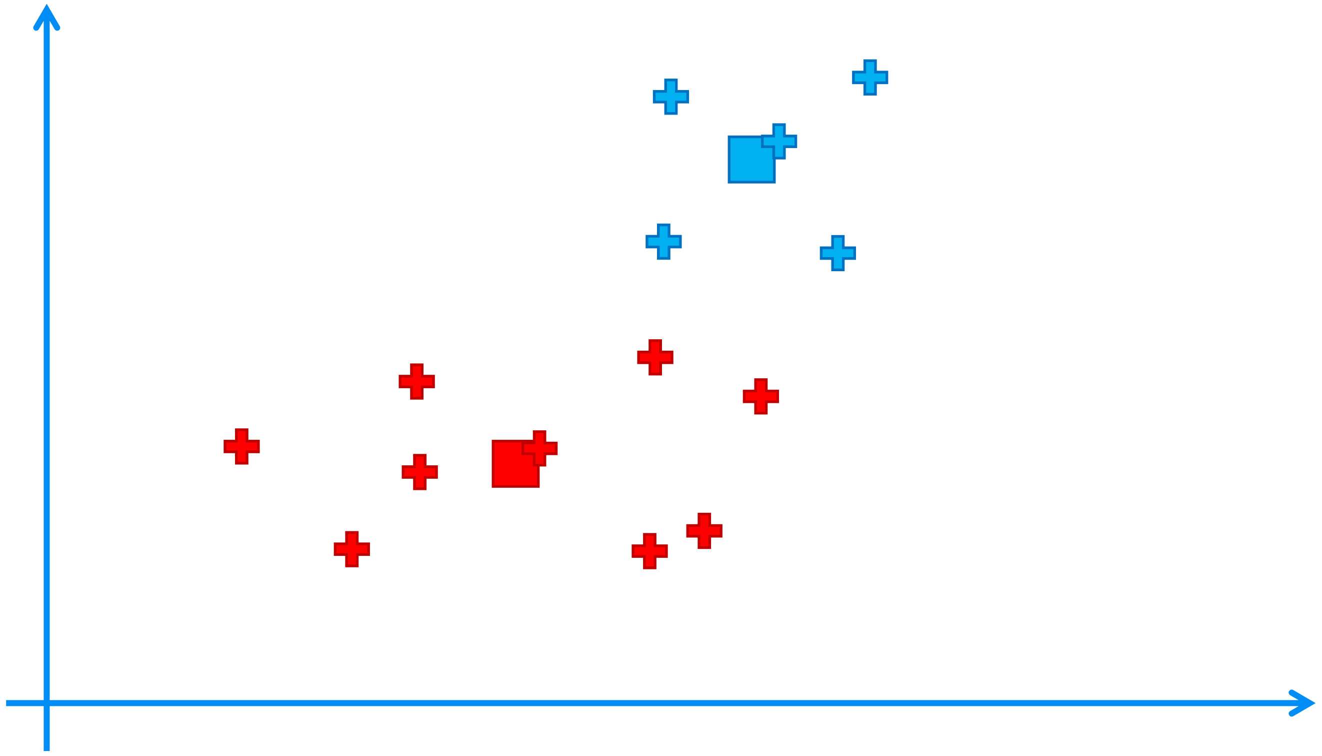
K-Means Clustering



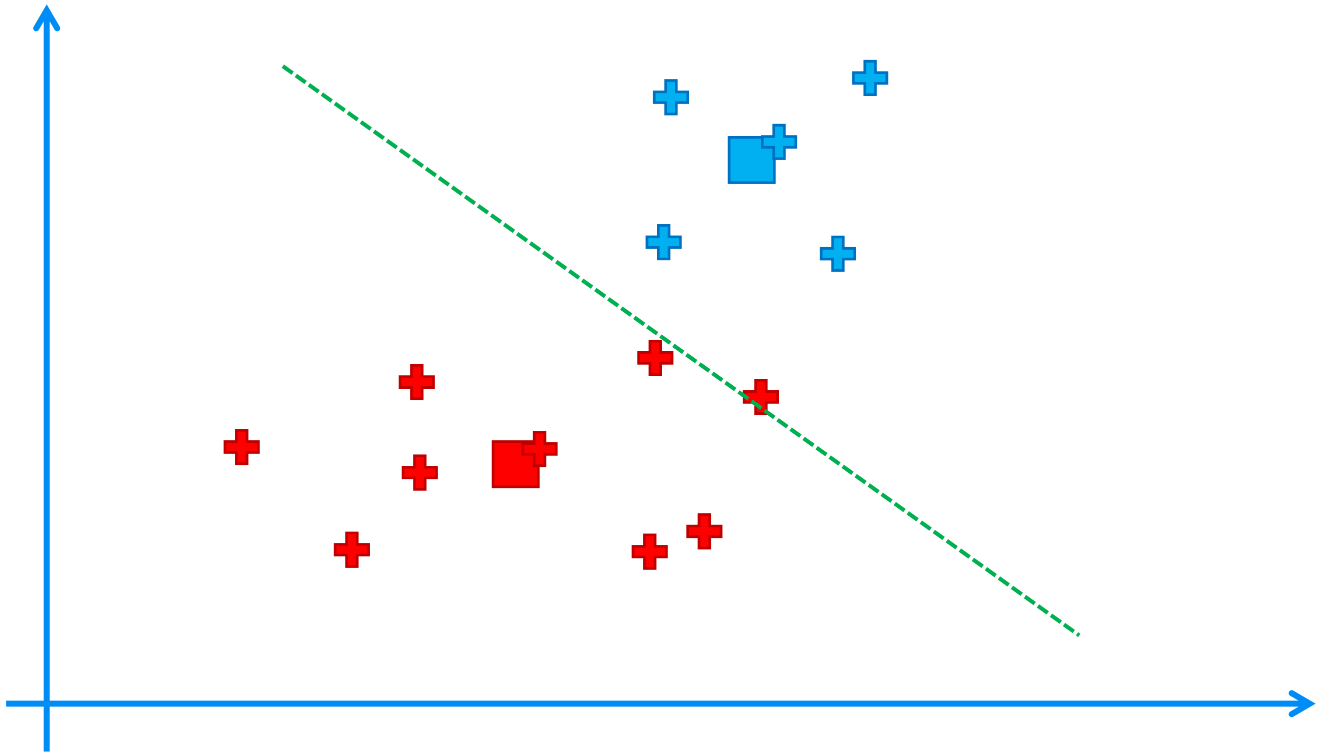
K-Means Clustering



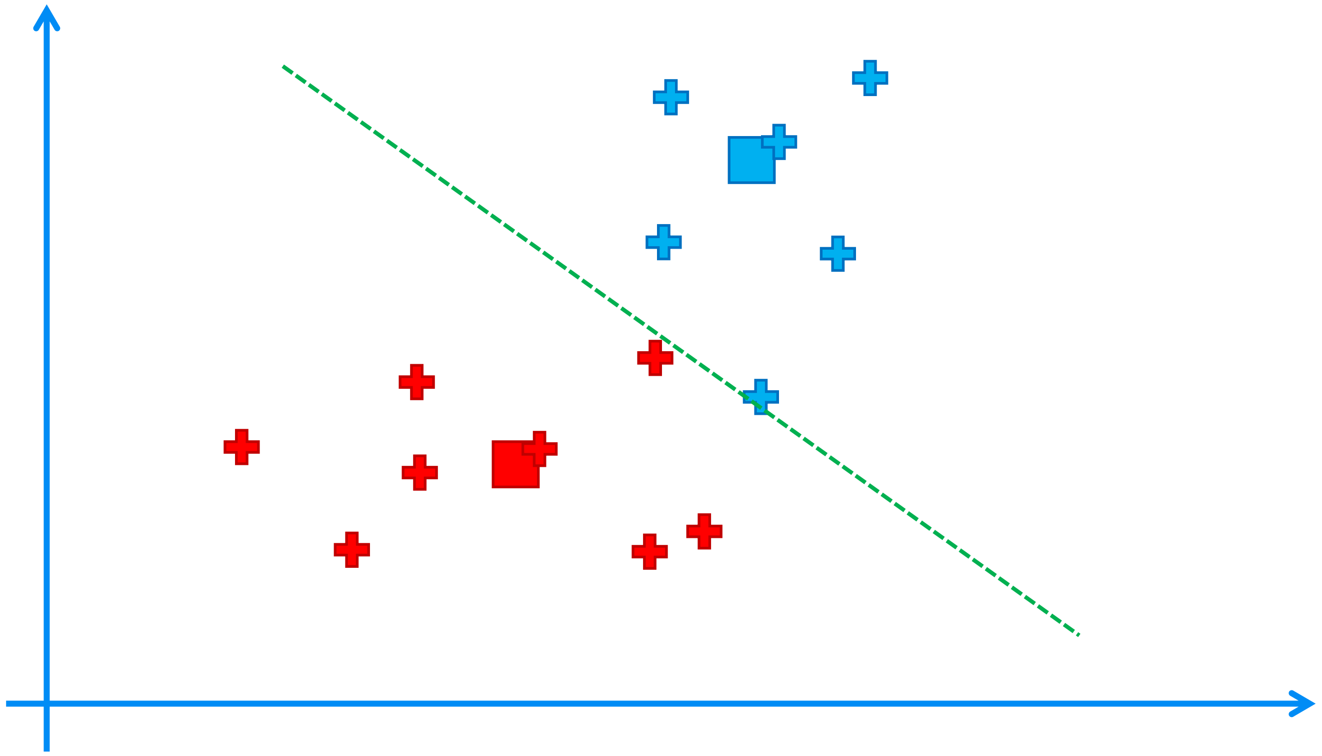
K-Means Clustering



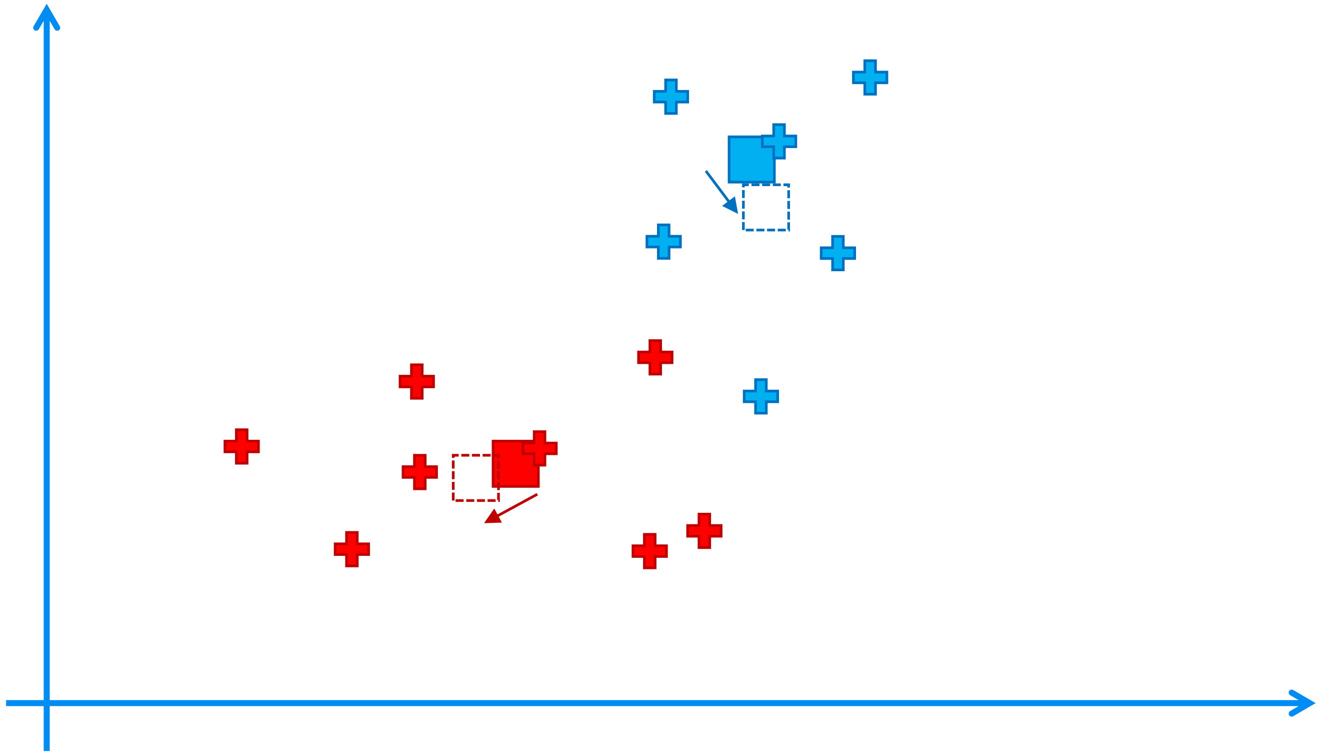
K-Means Clustering



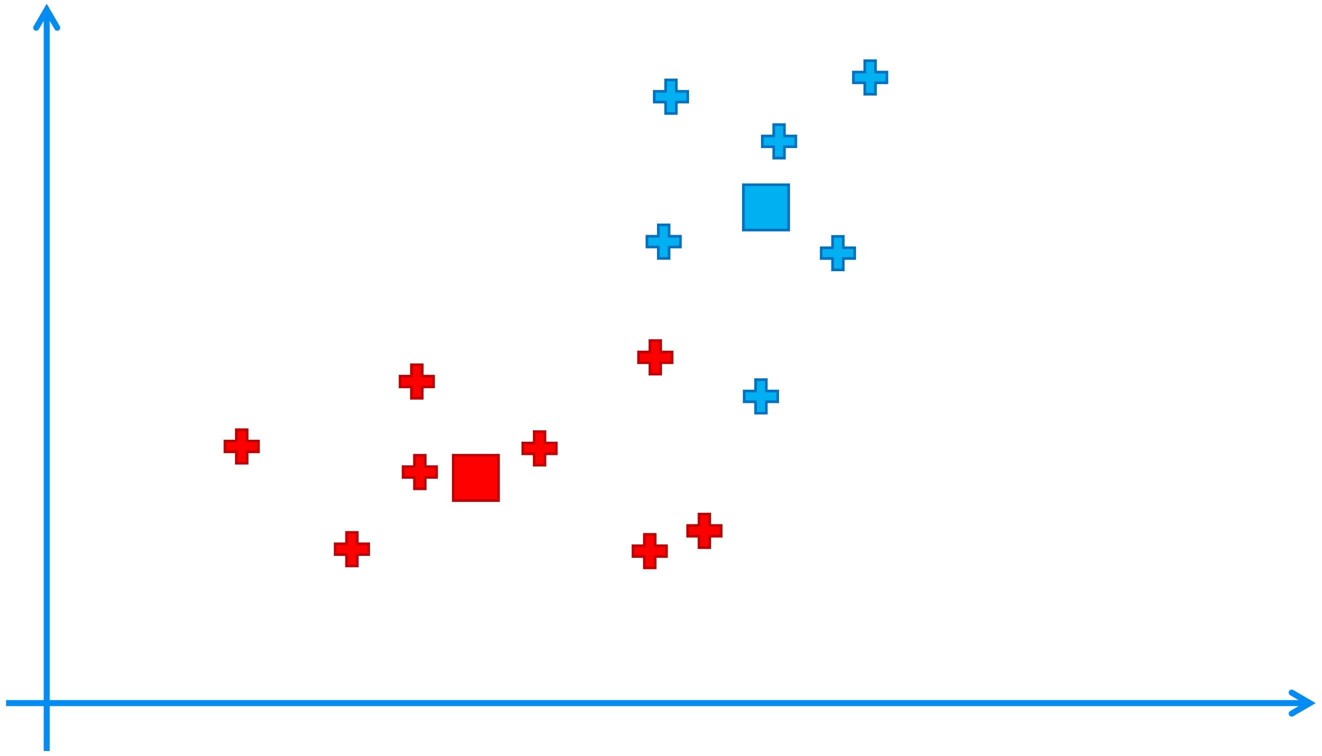
K-Means Clustering



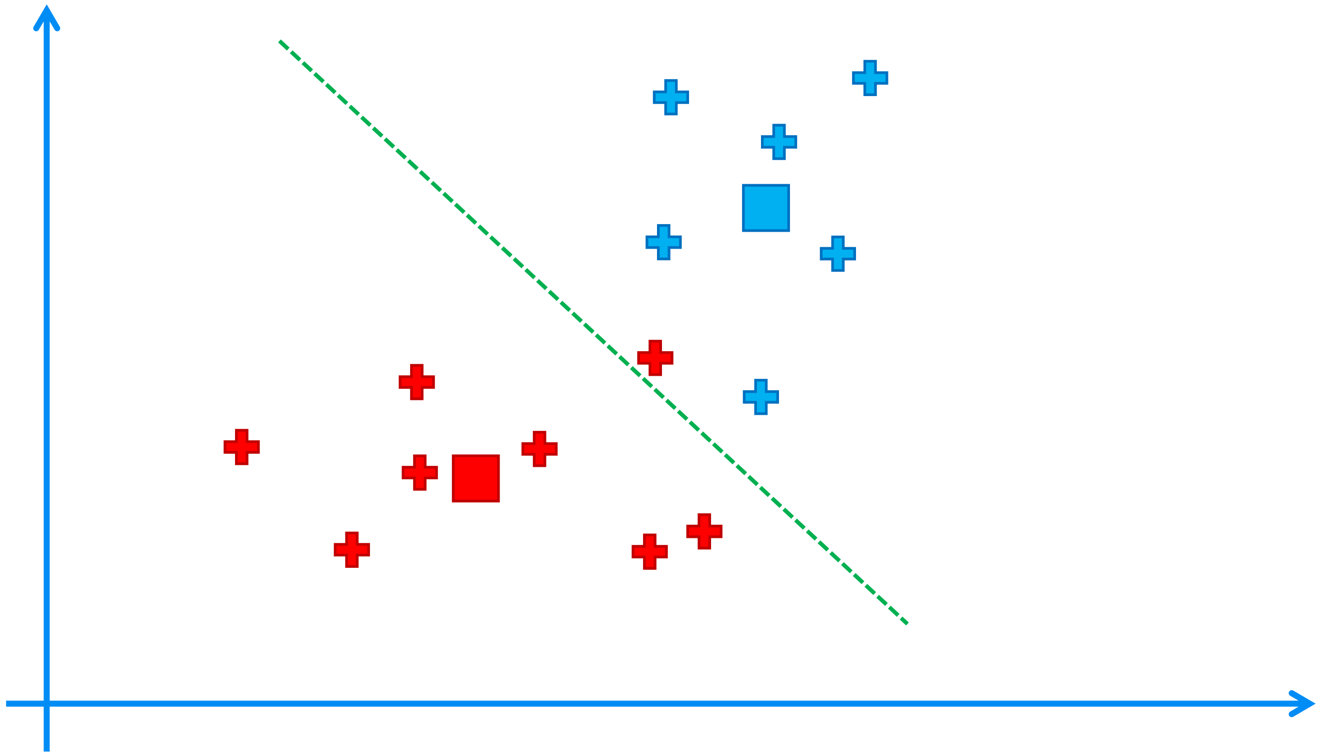
K-Means Clustering



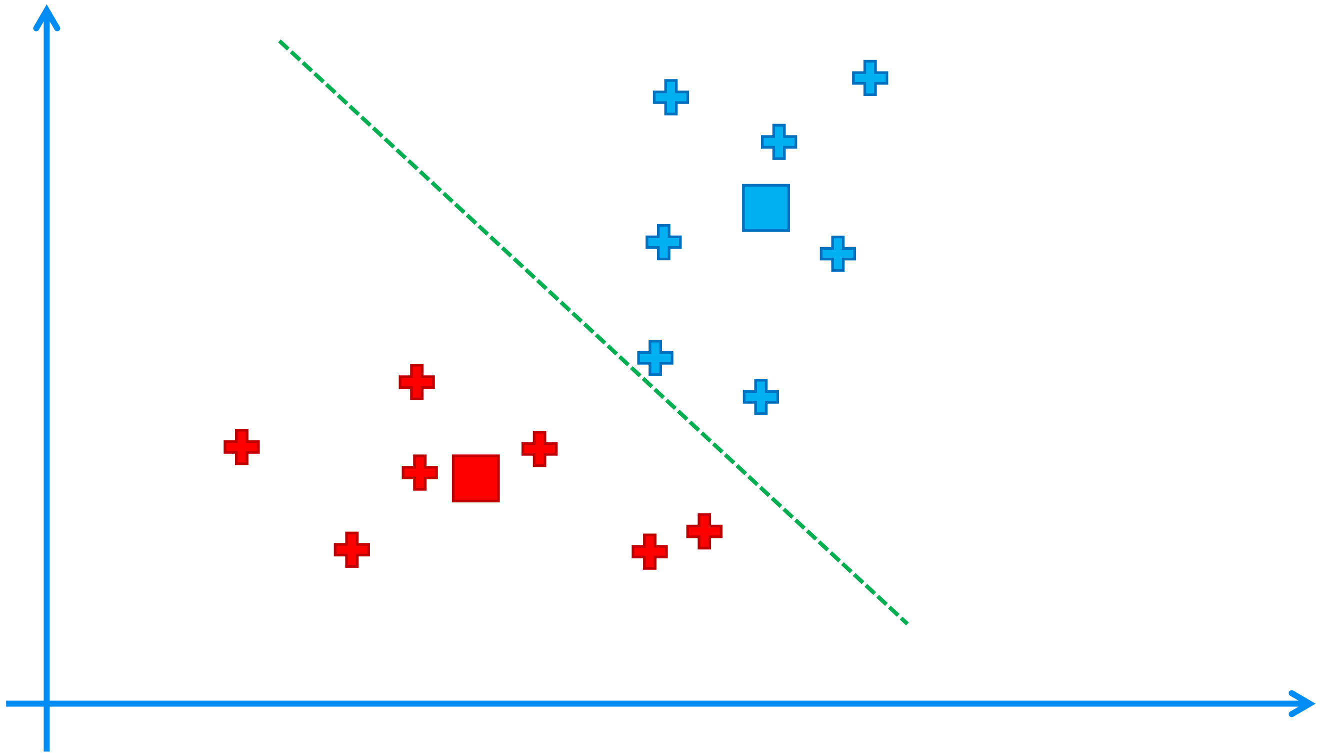
K-Means Clustering



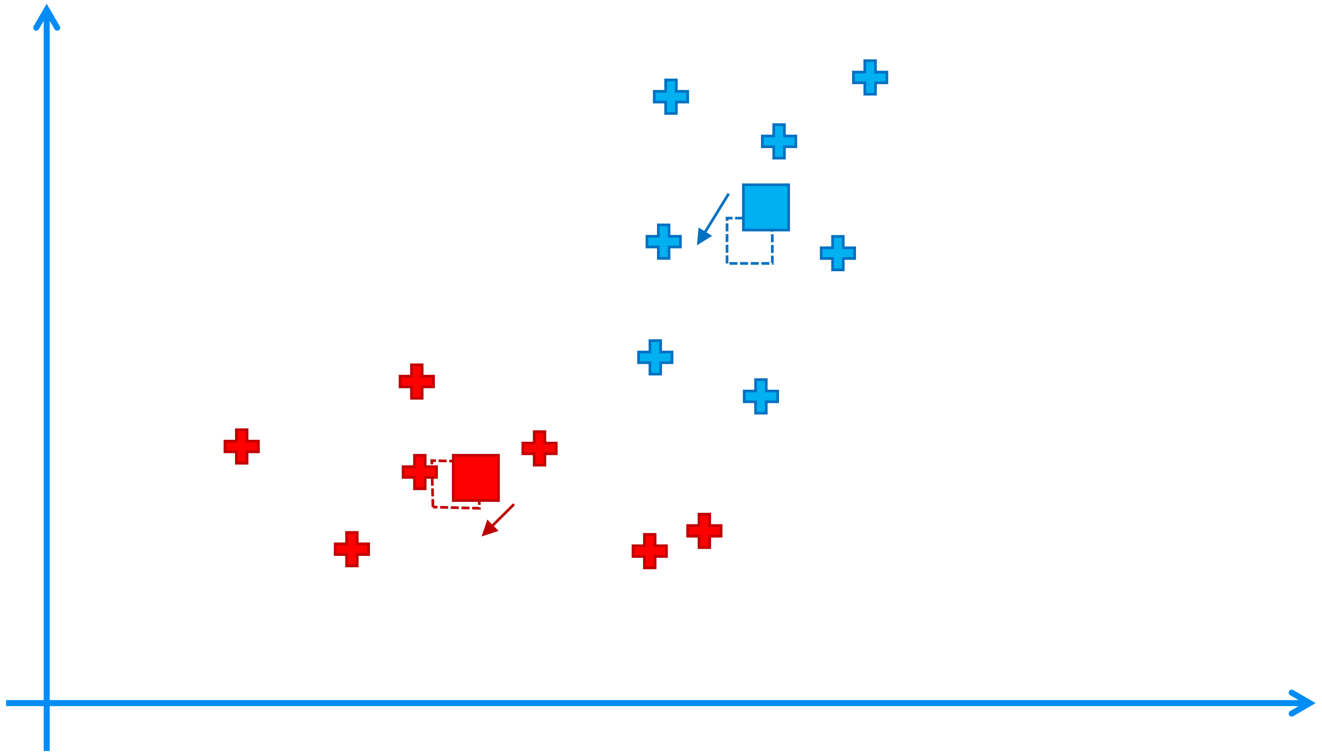
K-Means Clustering



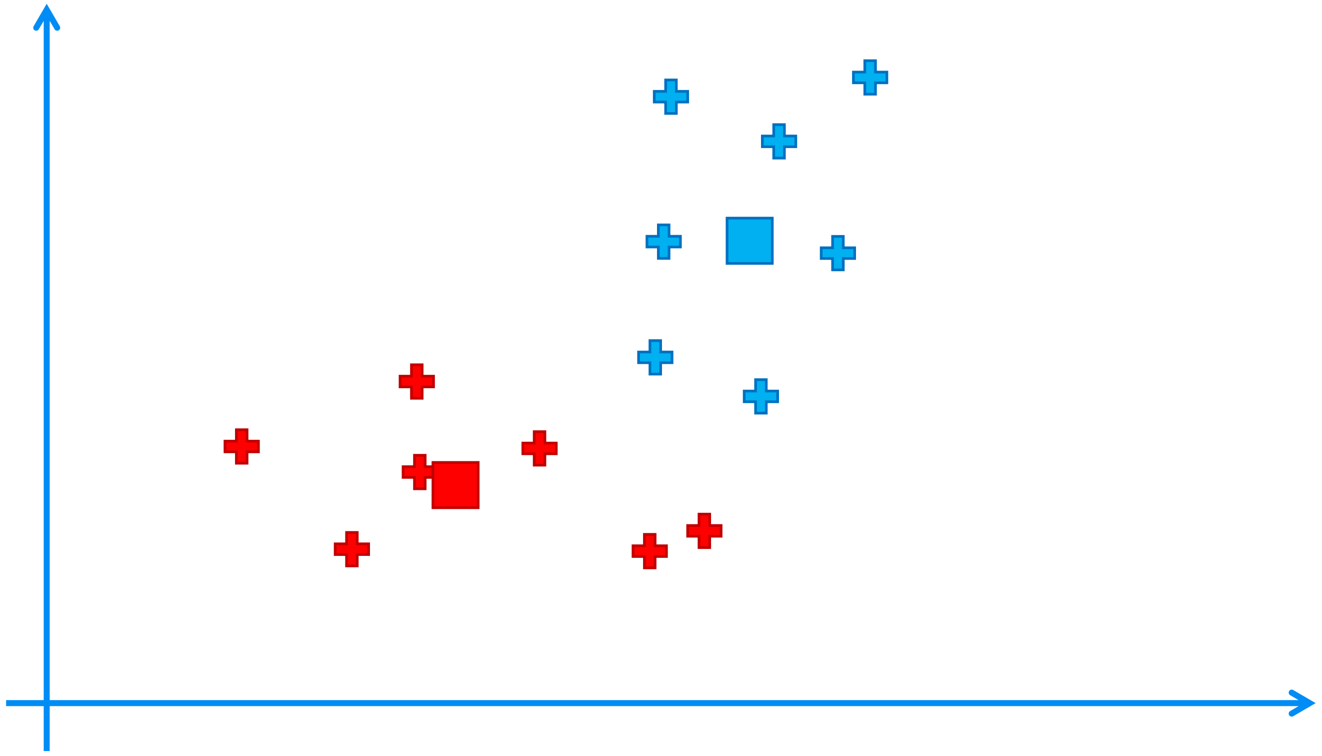
K-Means Clustering



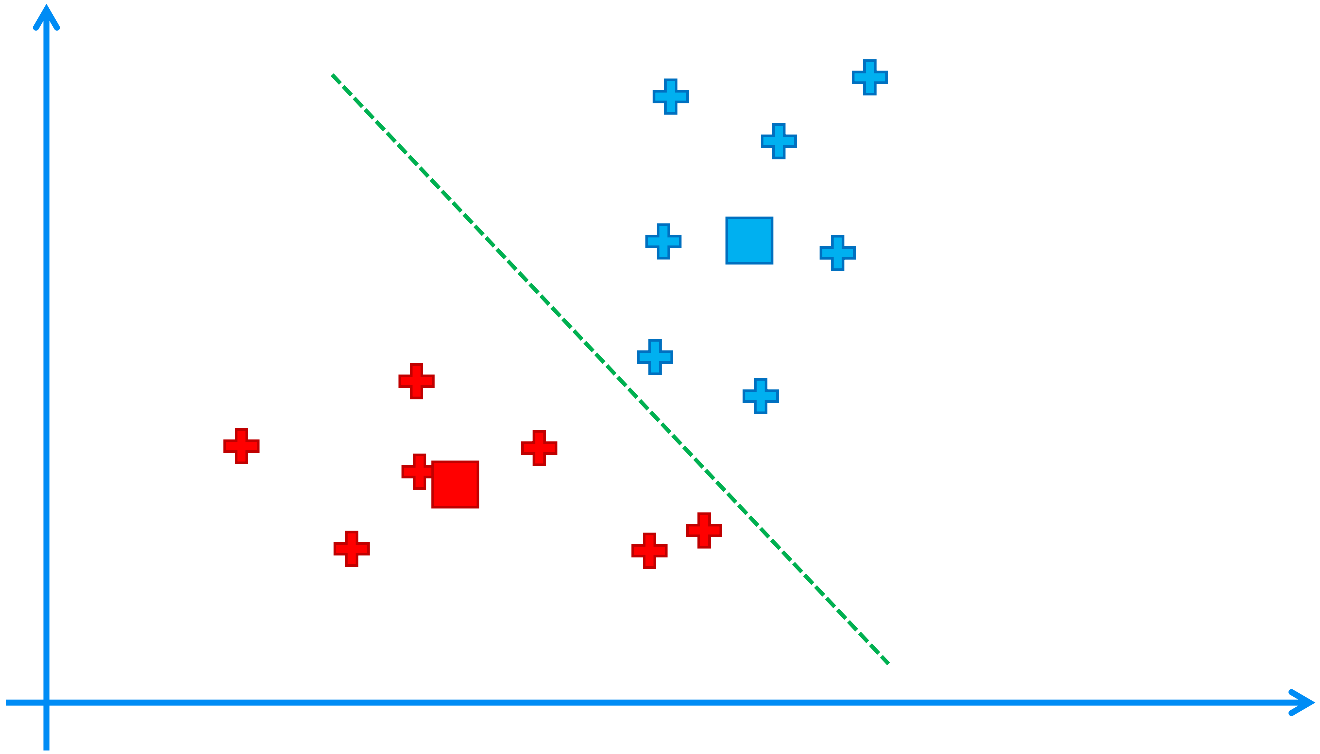
K-Means Clustering



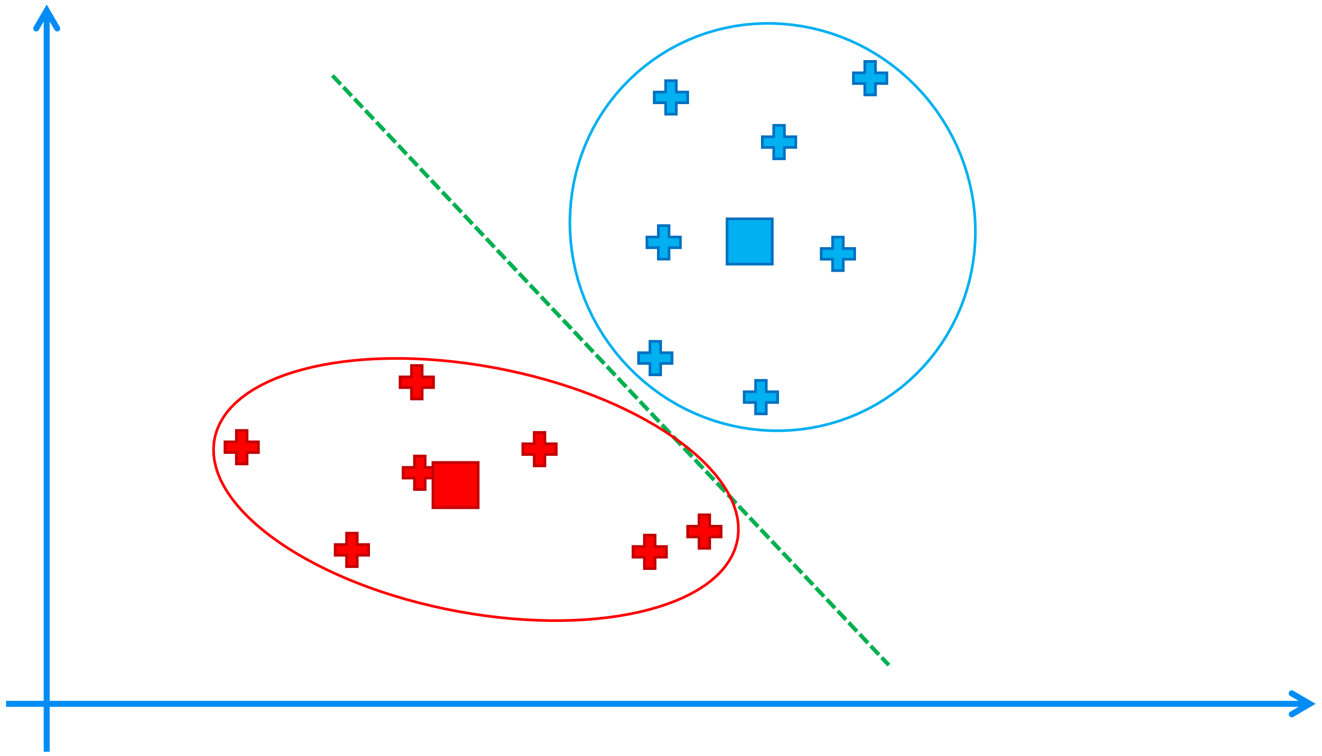
K-Means Clustering



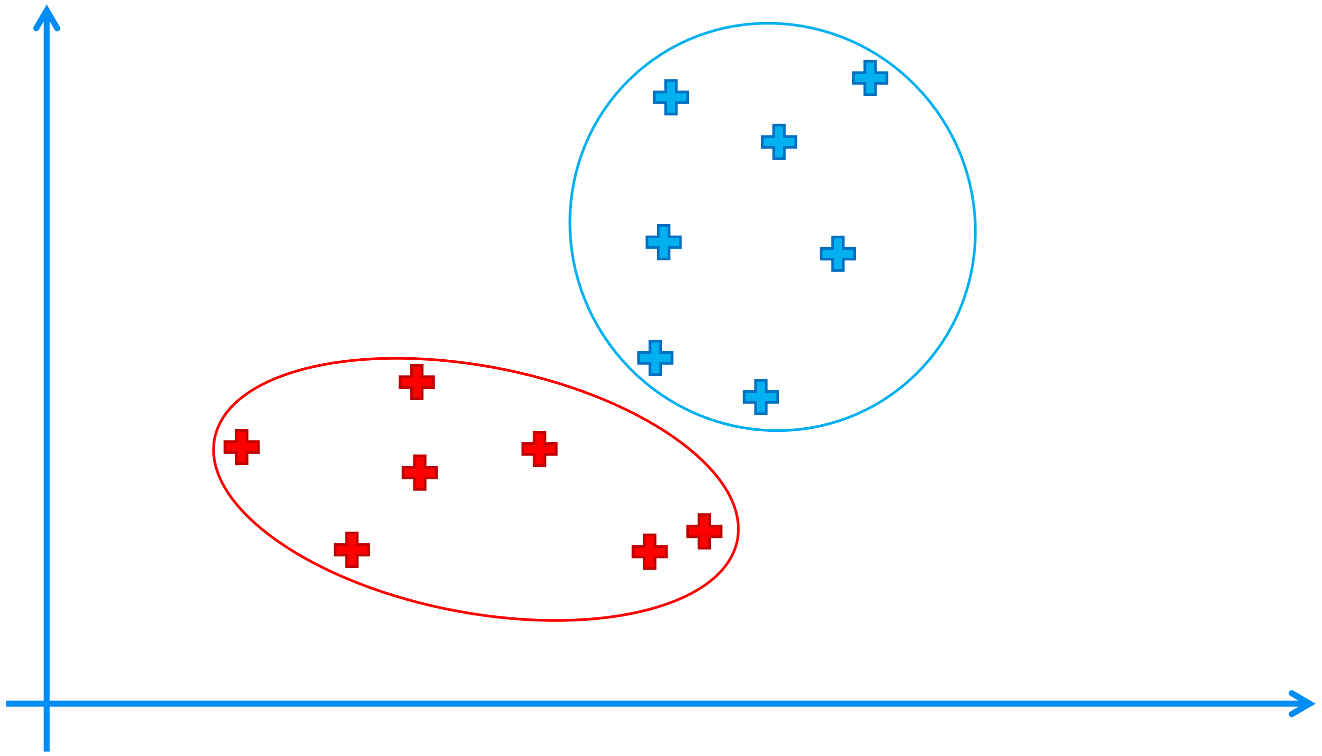
K-Means Clustering



K-Means Clustering

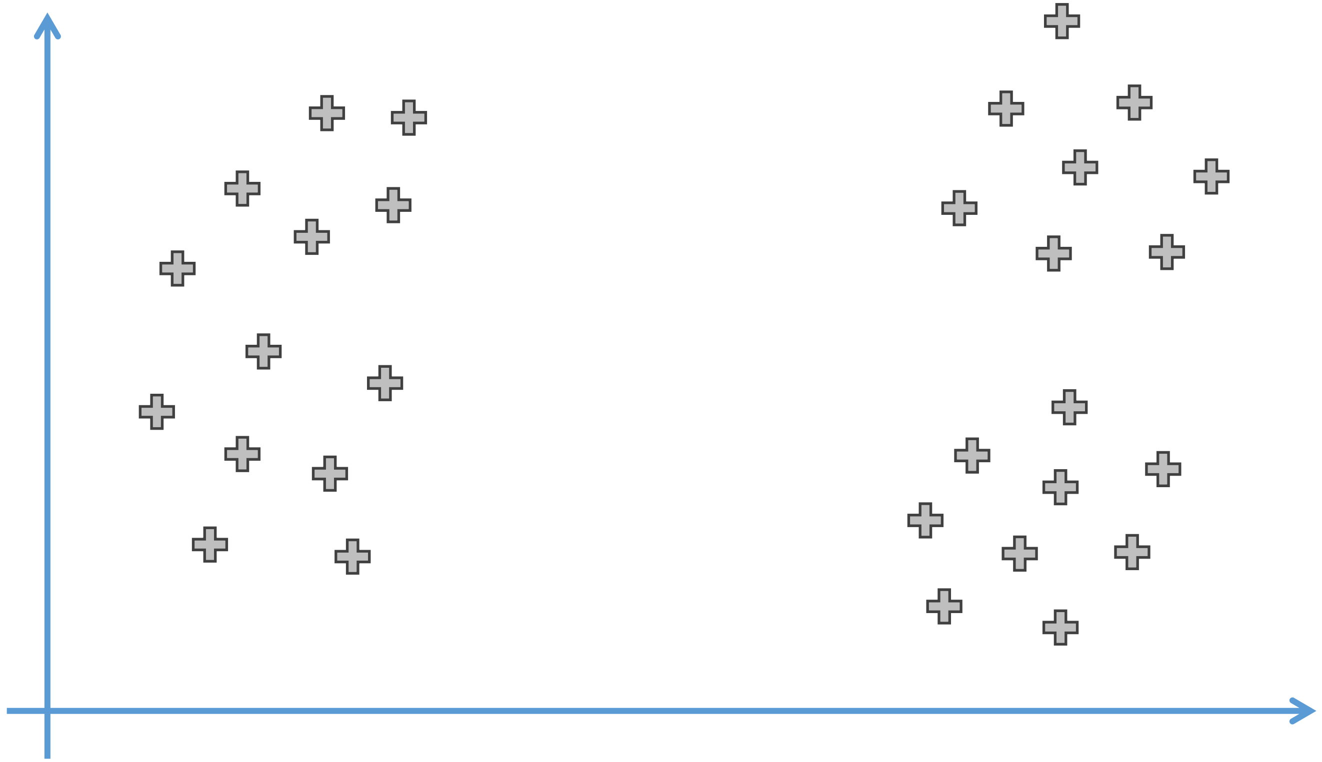


K-Means Clustering



The Elbow Method

The Elbow Method



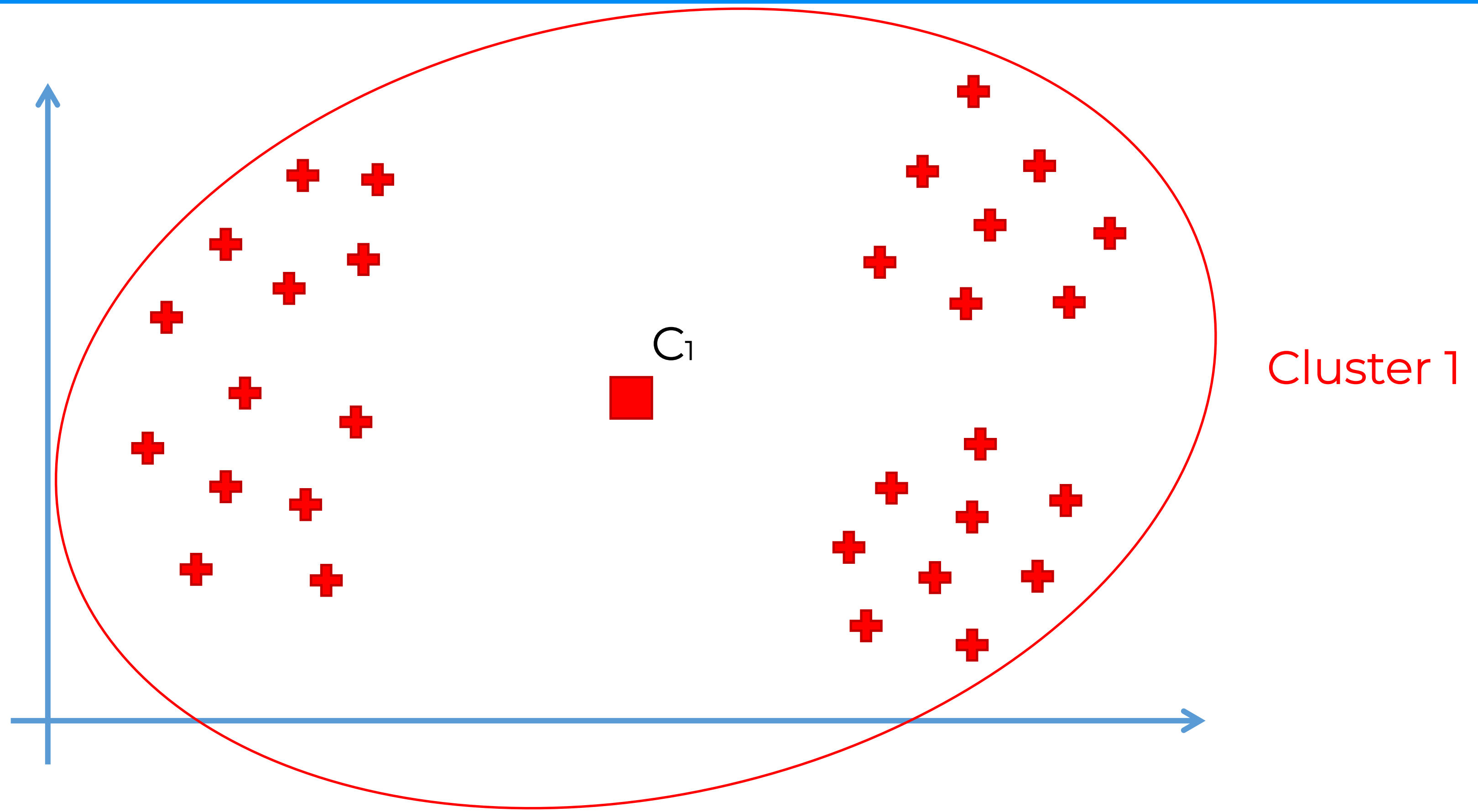
The Elbow Method

Within Cluster Sum of Squares:

$$\mathrm{WCSS} = \sum_{\mathrm{P_i\ in\ Cluster\ 1}} \mathrm{distance}(\mathrm{P_i}, \mathrm{C_1})^2 + \sum_{\mathrm{P_i\ in\ Cluster\ 2}} \mathrm{distance}(\mathrm{P_i}, \mathrm{C_2})^2 + \ \ldots$$

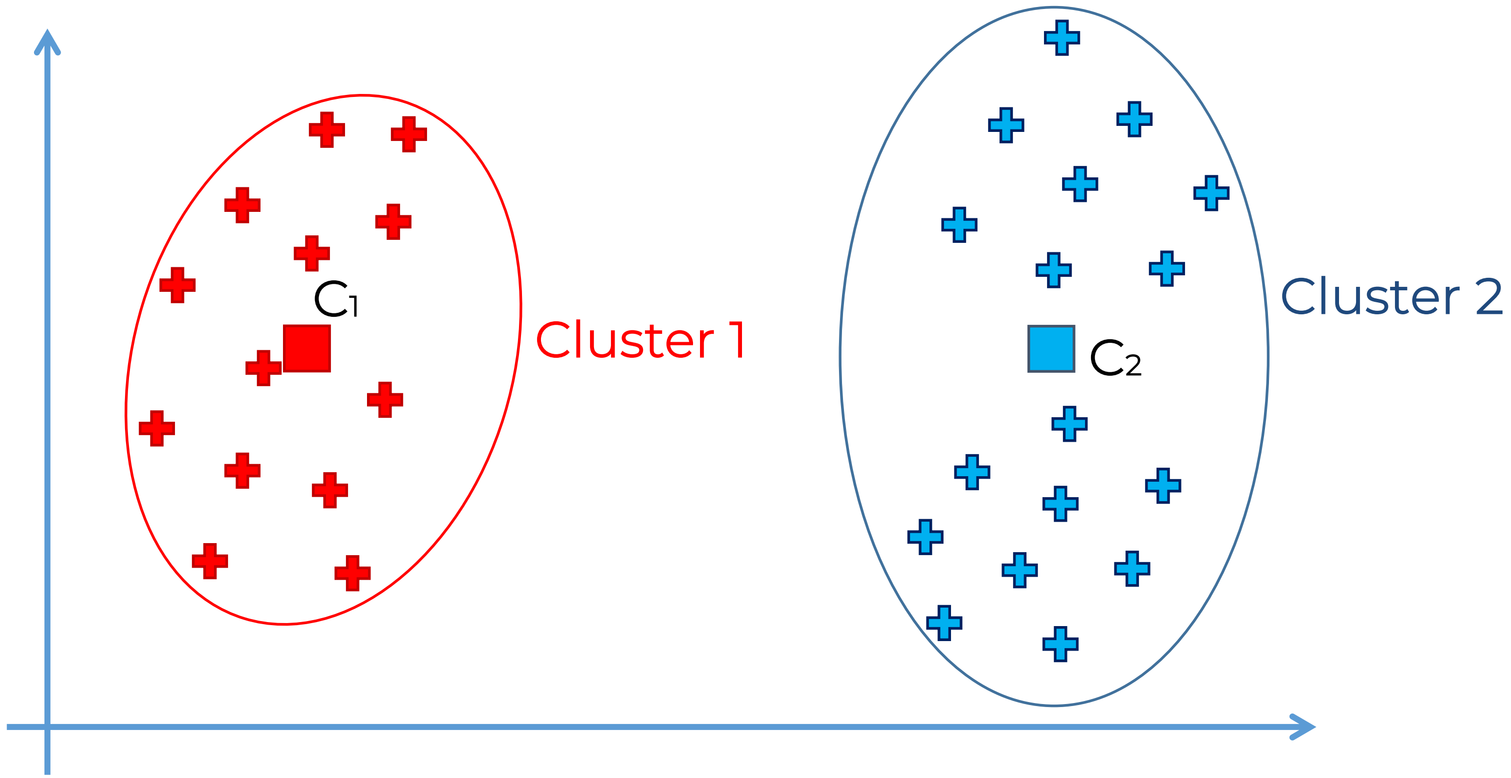
The Elbow Method

The Elbow Method



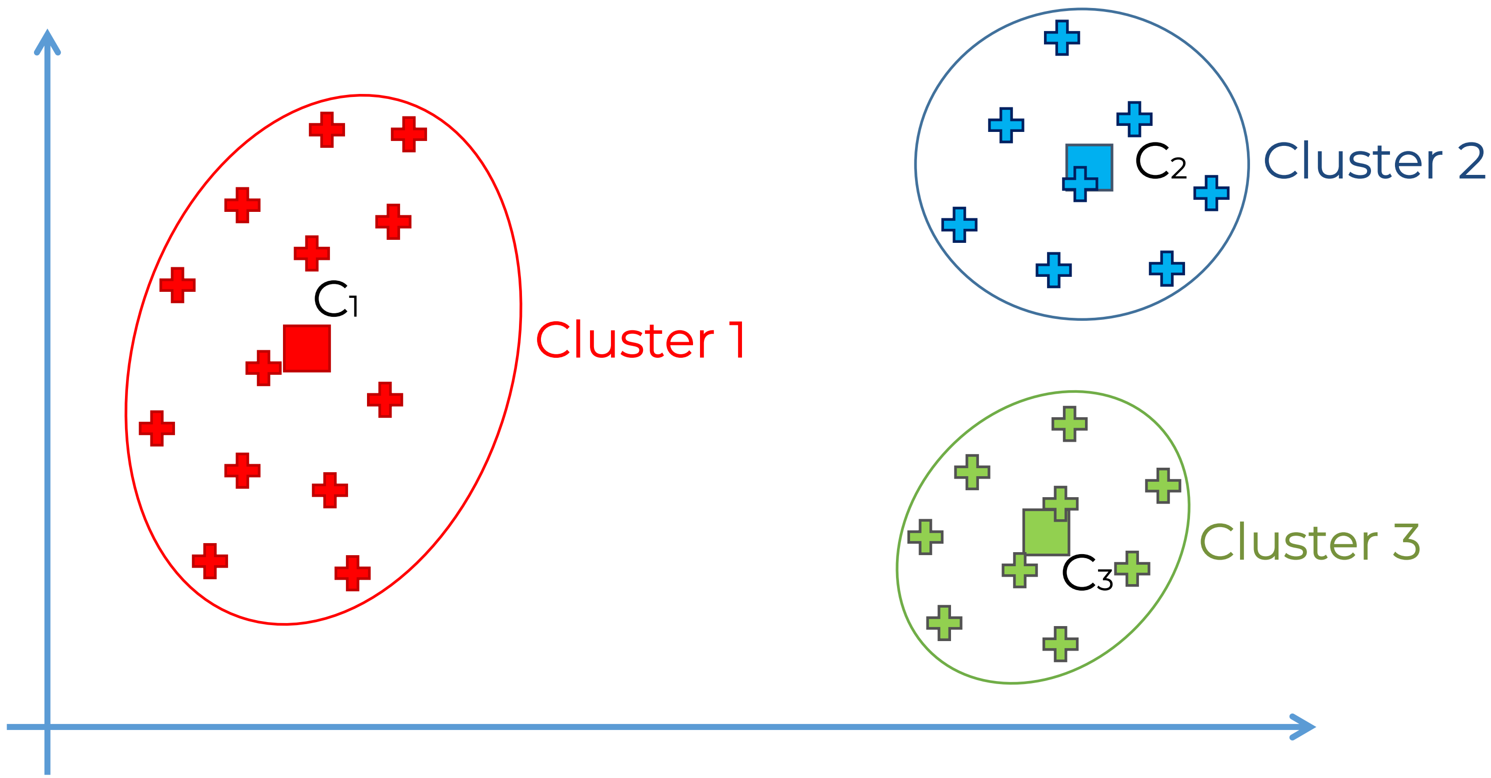
$$WCSS = \sum_{P_i \text{ in Cluster } 1} distance(P_i, C_1)^2$$

The Elbow Method



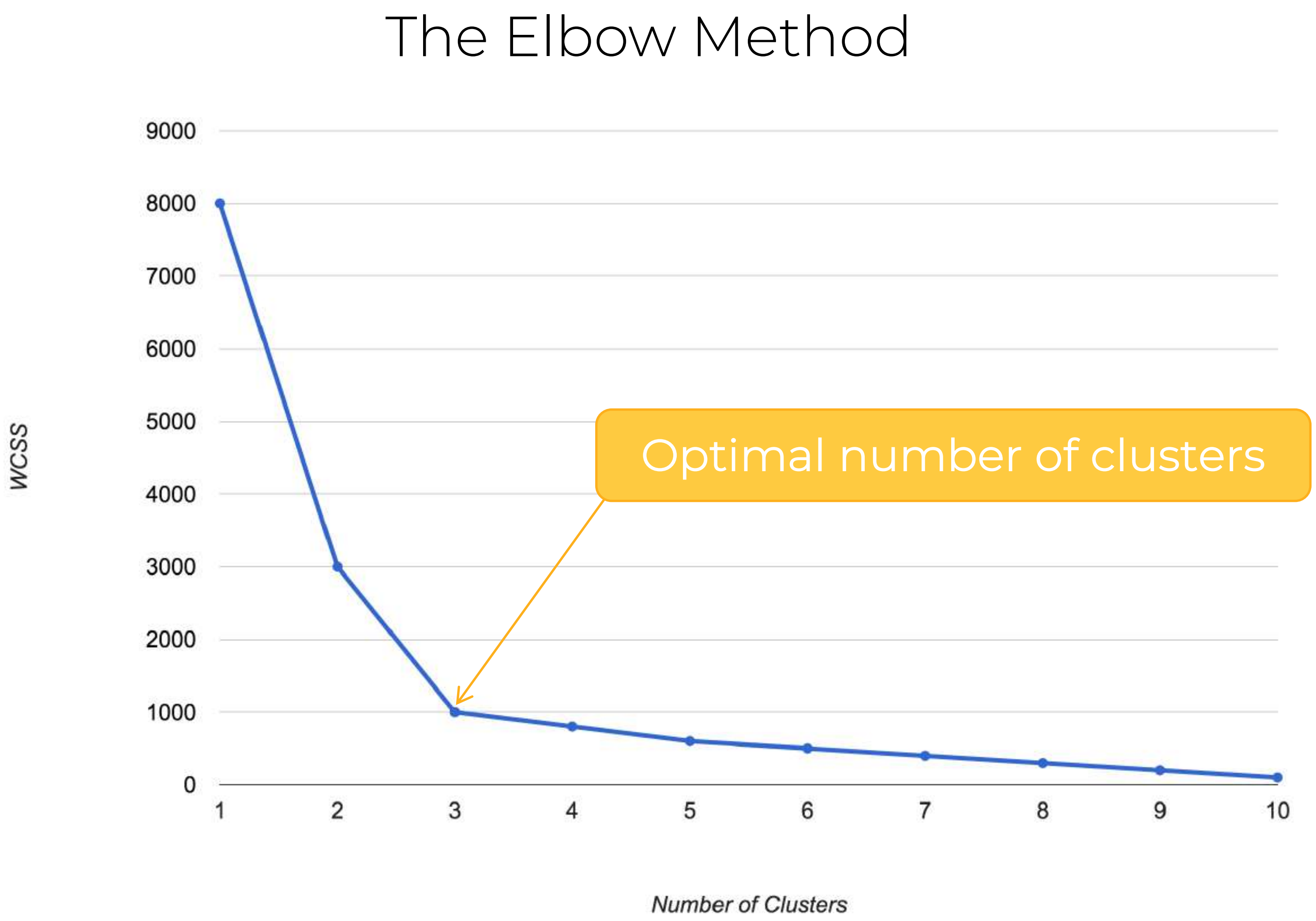
$$\text{WCSS} = \sum_{P_i \text{ in Cluster } 1} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster } 2} \text{distance}(P_i, C_2)^2$$

The Elbow Method



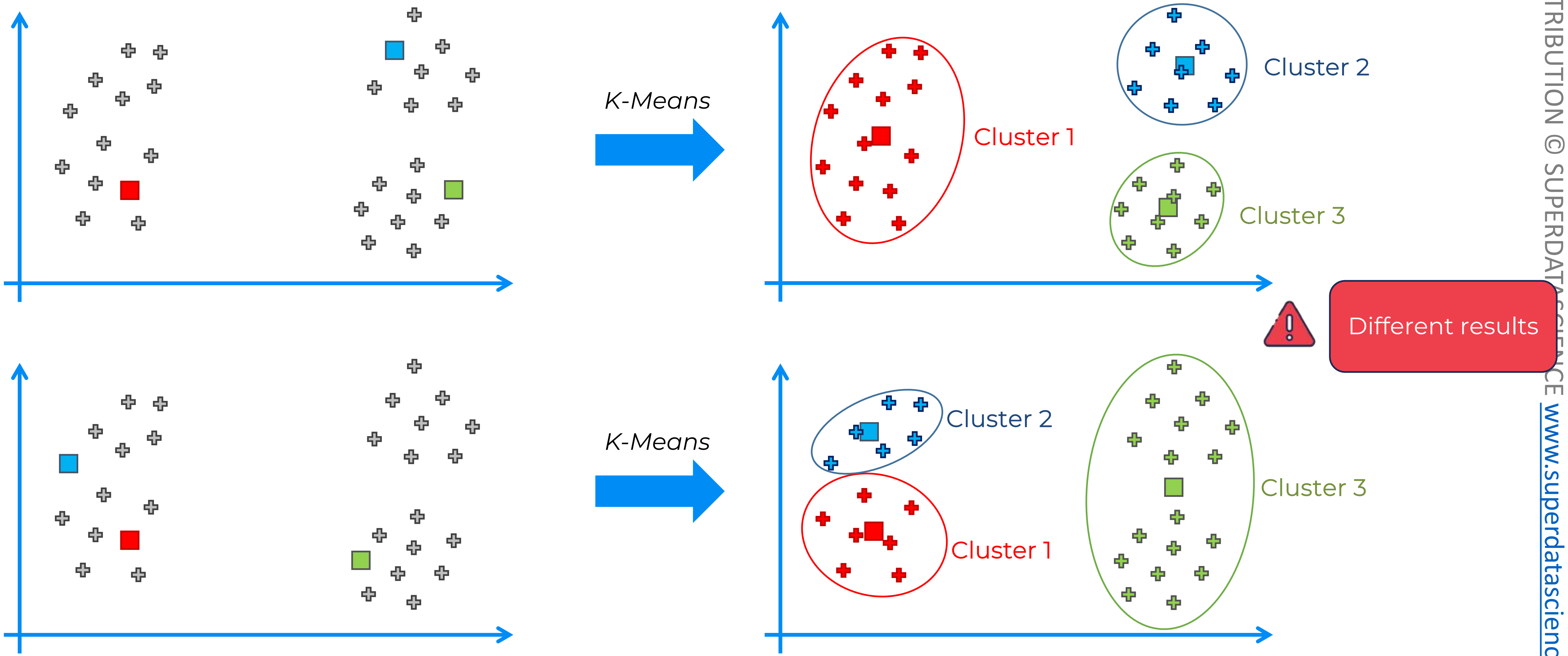
$$\mathrm{WCSS} = \sum_{P_i \text{ in Cluster } 1} \mathrm{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster } 2} \mathrm{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster } 3} \mathrm{distance}(P_i, C_3)^2$$

The Elbow Method



K-Means++

K-Means++


K-Means
Cluster 1
Cluster 2
Cluster 3
Different results
K-Means
Cluster 2
Cluster 1
Cluster 3

K-Means++

K-Means++ Initialization Algorithm:

Step 1: Choose first centroid at random among data points

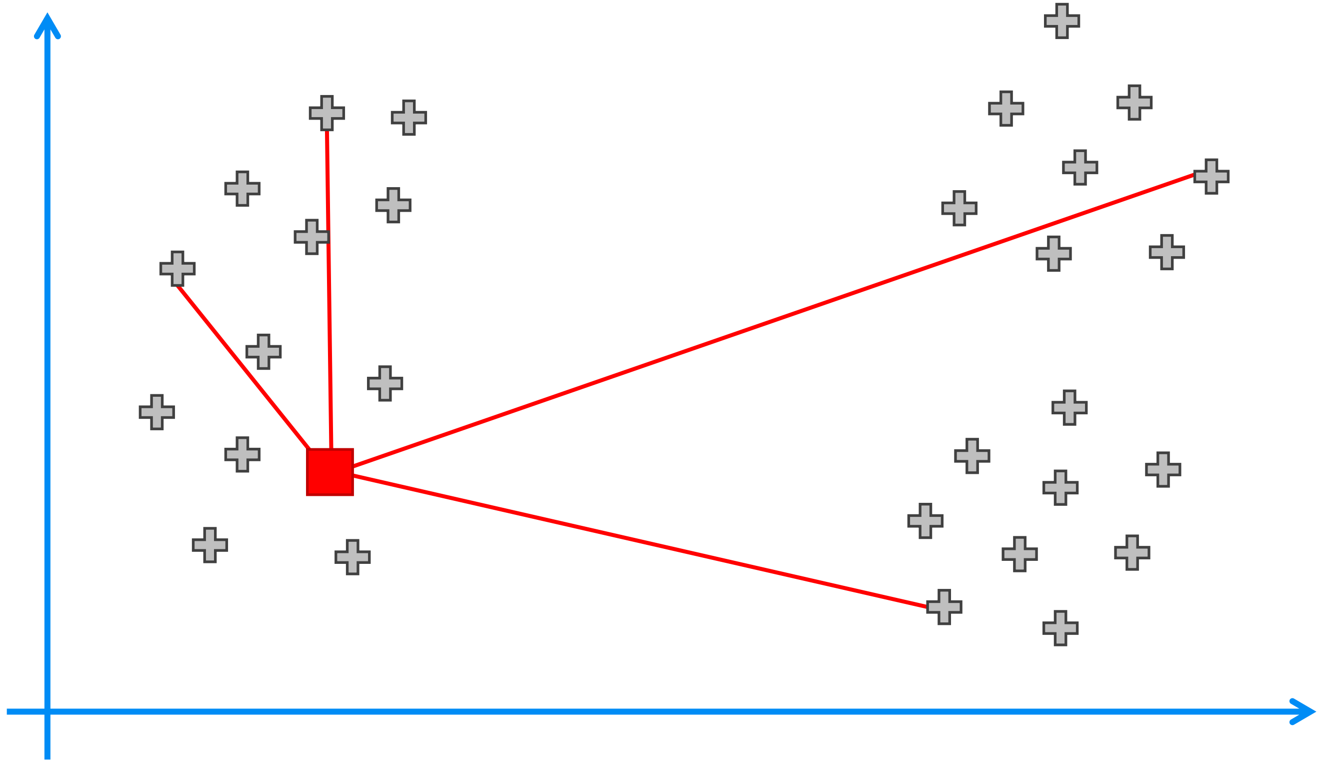
Step 2: For each of the remaining data points compute the distance (D) to the nearest out of already selected centroids

Step 3: Choose next centroid among remaining data points using weighted random selection – weighted by D^2

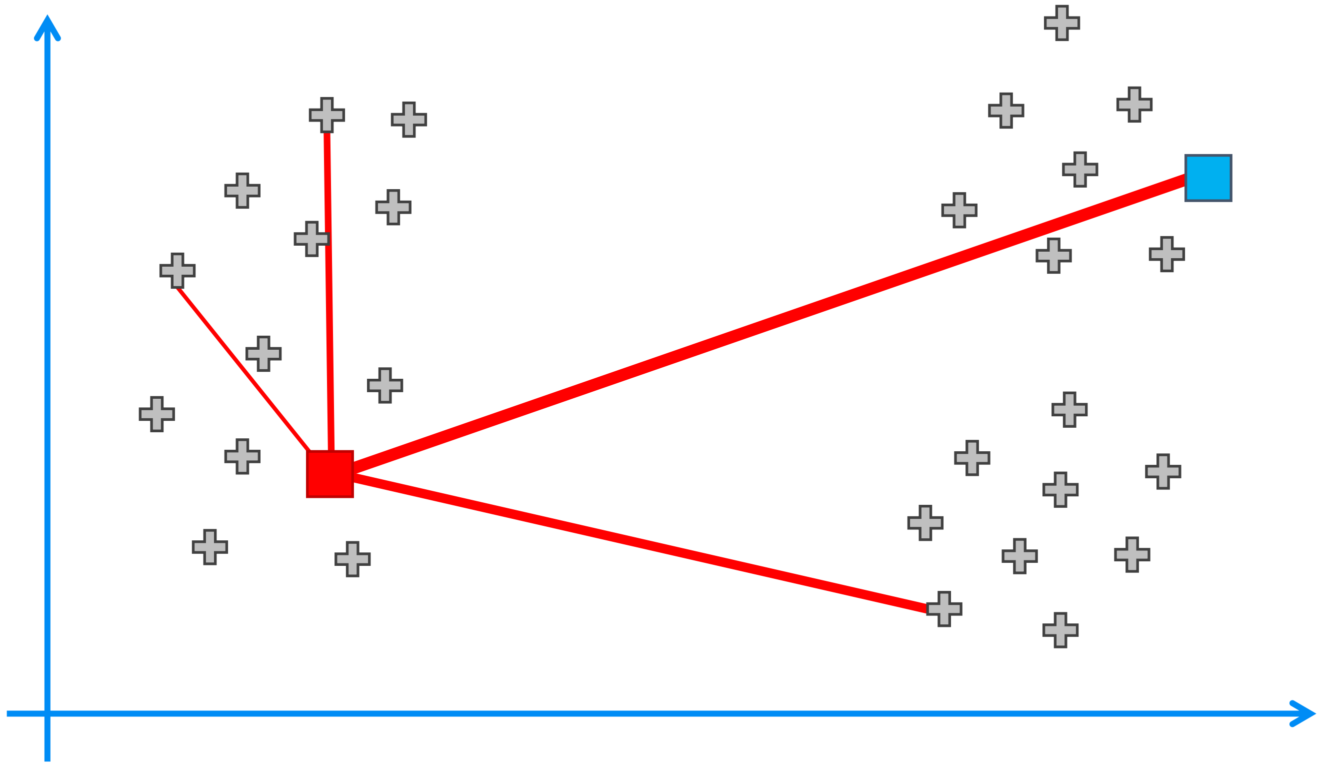
Step 4: Repeat Steps 2 and 3 until all k centroids have been selected

Step 5: Proceed with standard k-means clustering

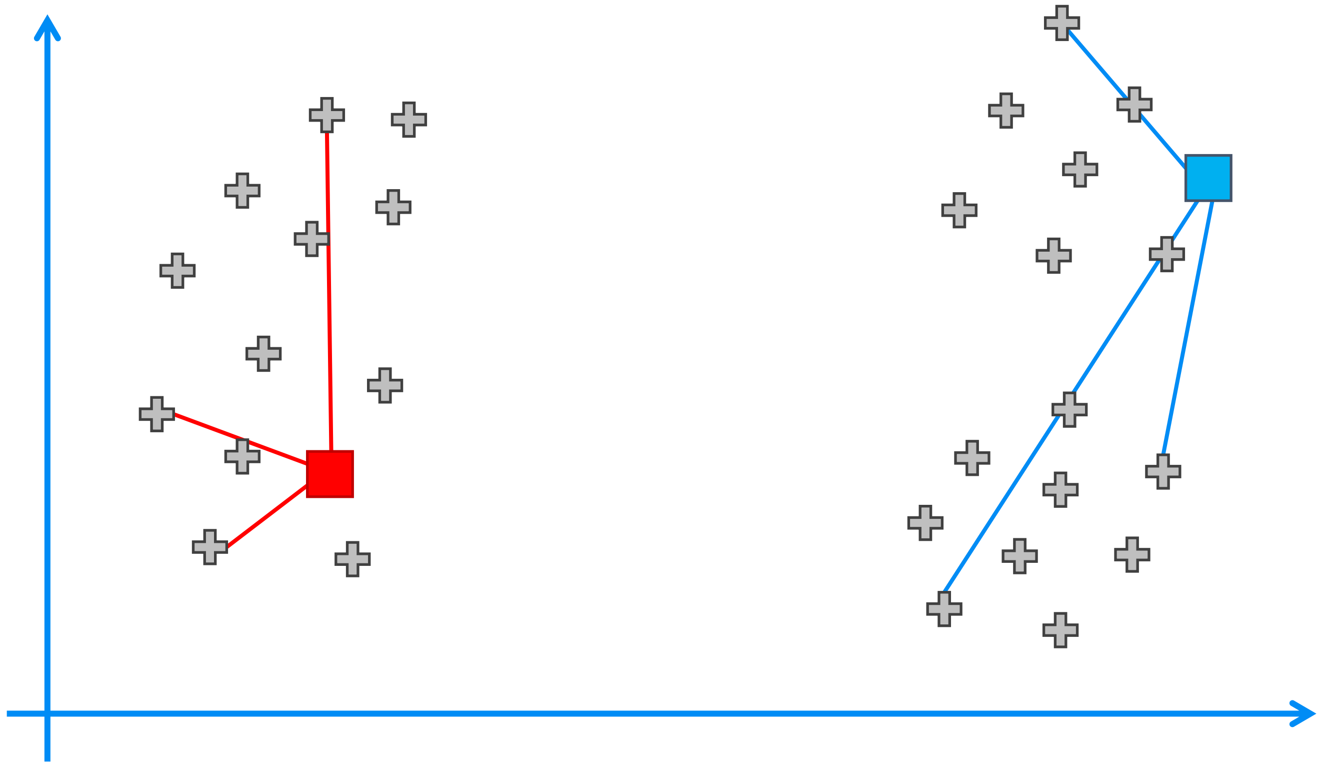
K-Means++



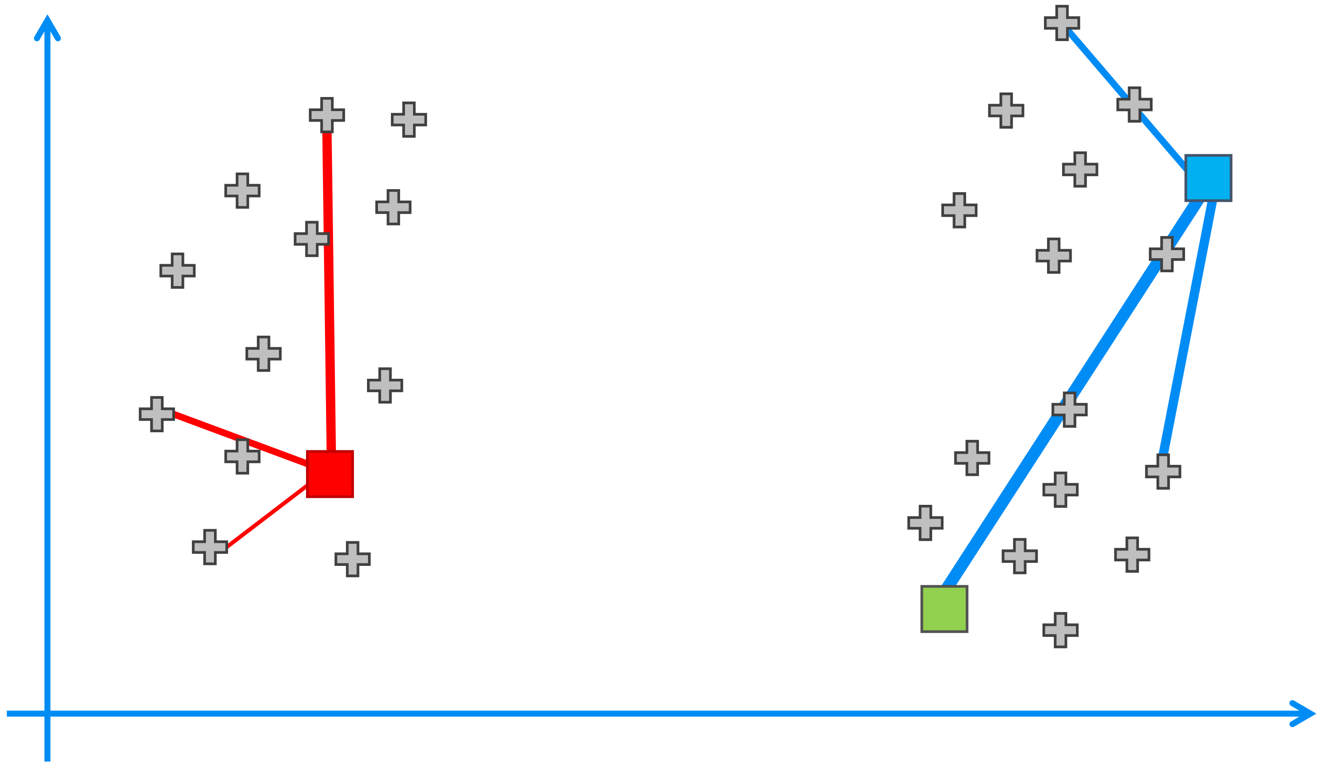
K-Means++



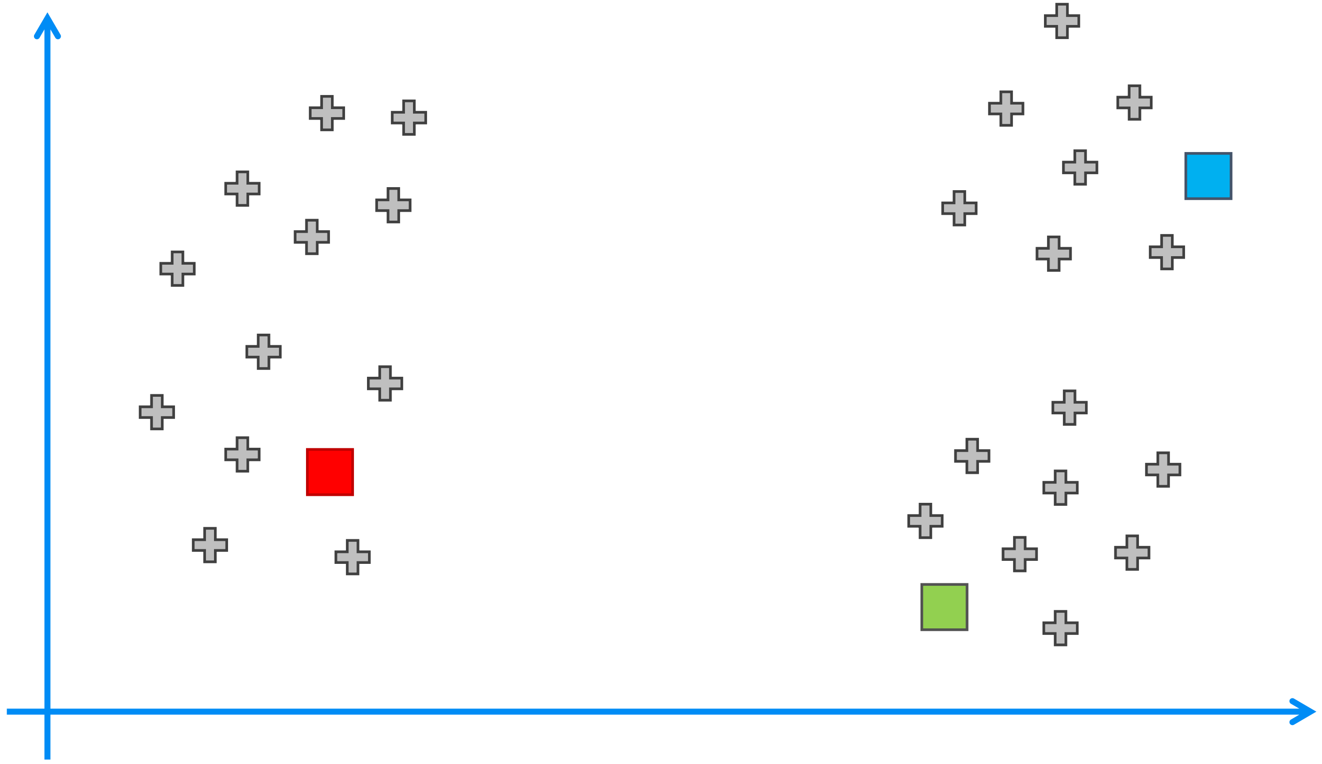
K-Means++



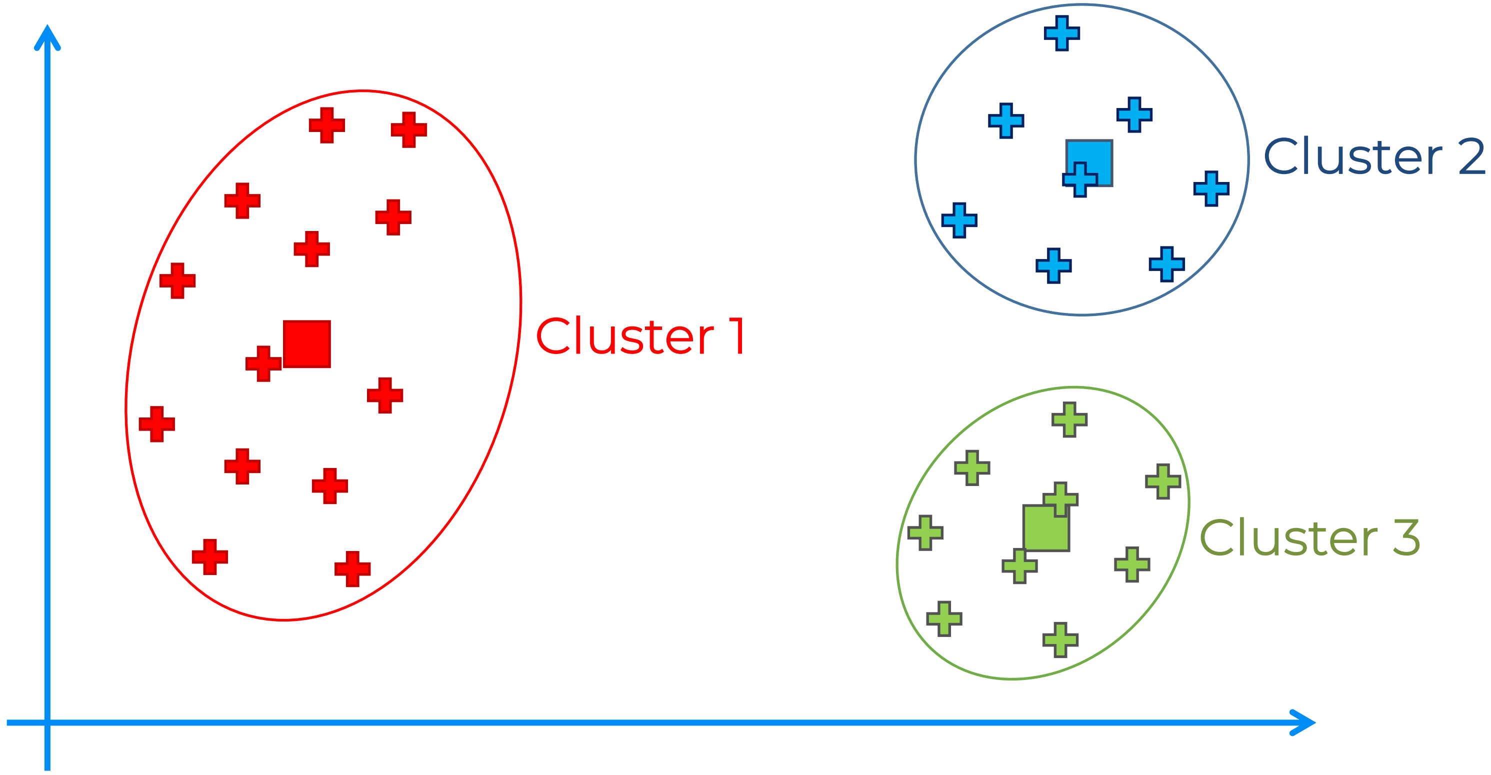
K-Means++



K-Means++

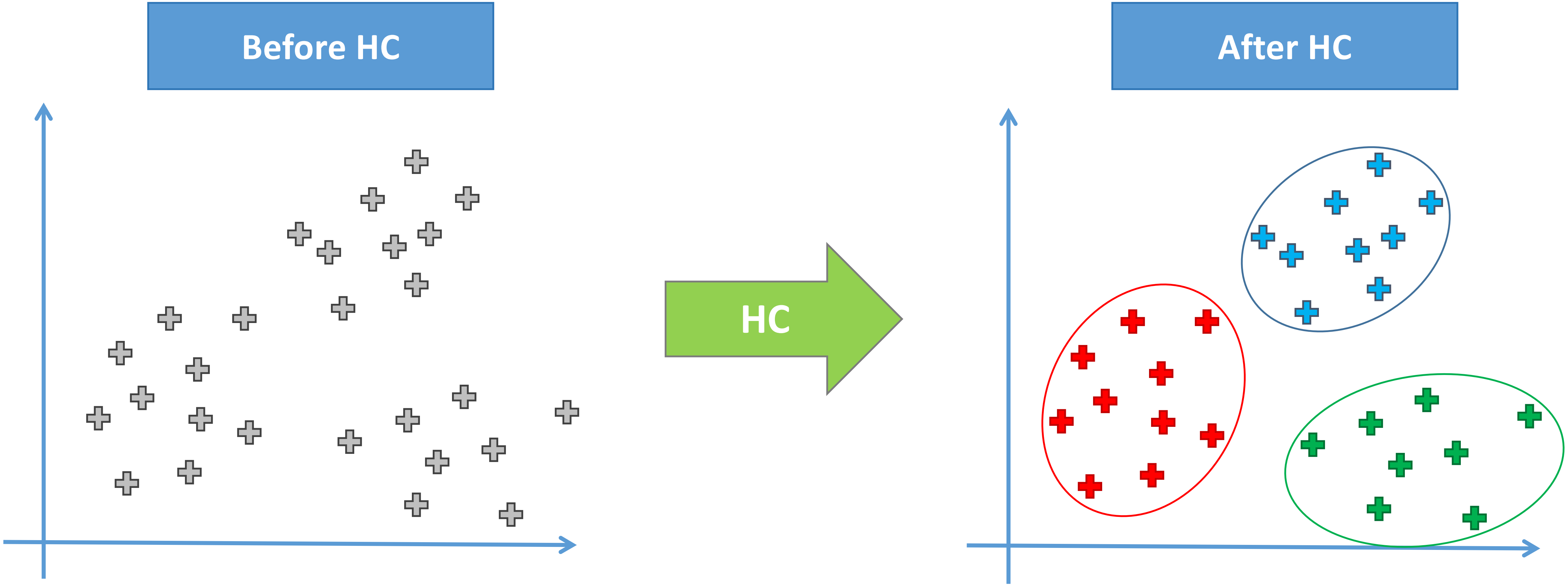


K-Means++



HC Intuition: Understanding HC

What HC does for you



Same as K-Means but different process

NOTE:
Agglomerative
&
Divisive

Agglomerative HC

STEP 1: Make each data point a single-point cluster ➡ That forms N clusters

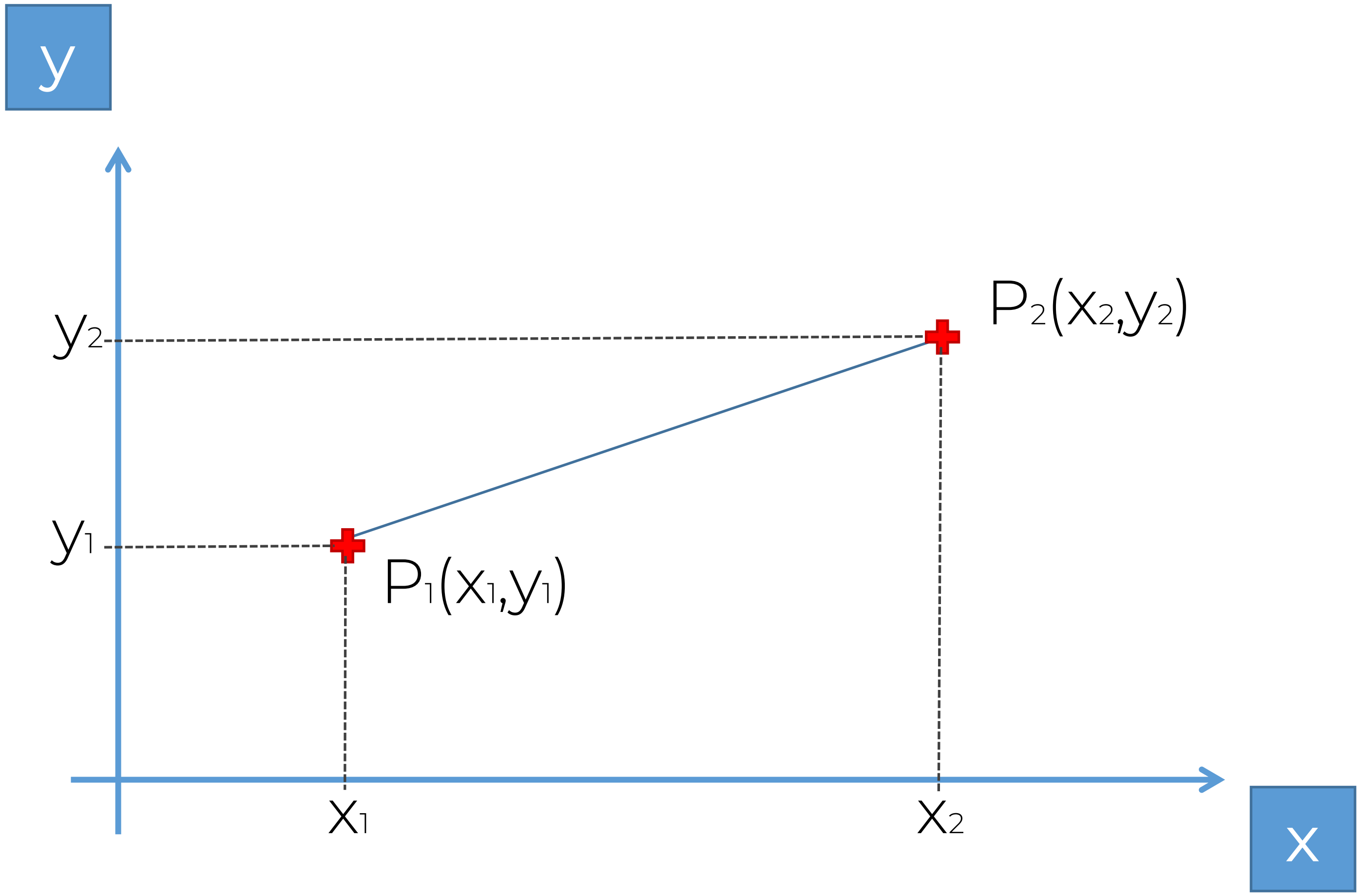
STEP 2: Take the two closest data points and make them one cluster ➡ That forms N-1 clusters

STEP 3: Take the two closest clusters and make them one cluster ➡ That forms N - 2 clusters

STEP 4: Repeat **STEP 3** until there is only one cluster

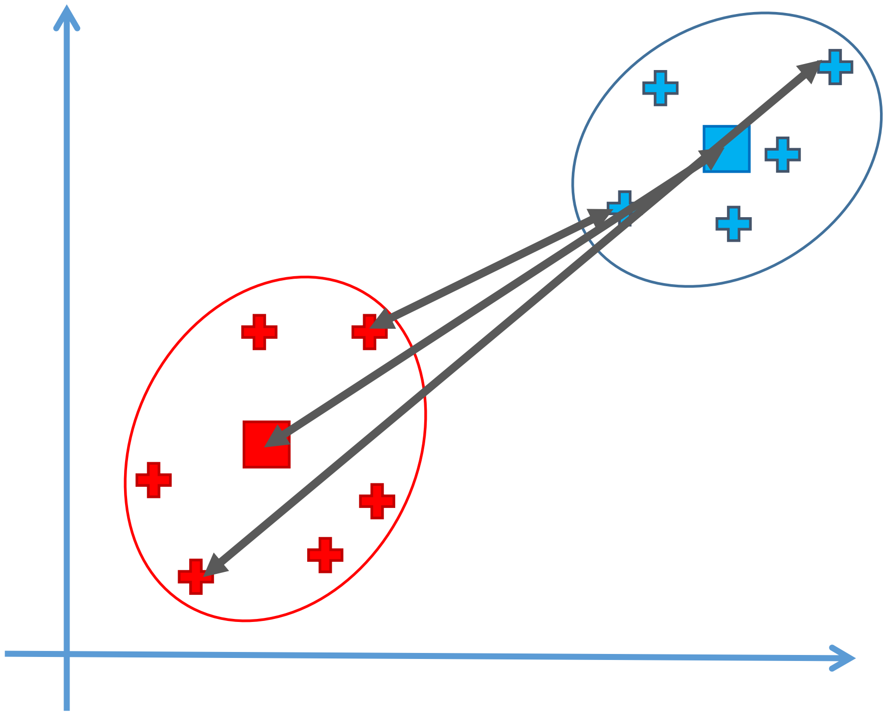
FIN

Euclidean Distance



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

Distance Between Clusters



Distance Between Two Clusters:

- Option 1: Closest Points
- Option 2: Furthest Points
- Option 3: Average Distance
- Option 4: Distance Between Centroids

Agglomerative HC

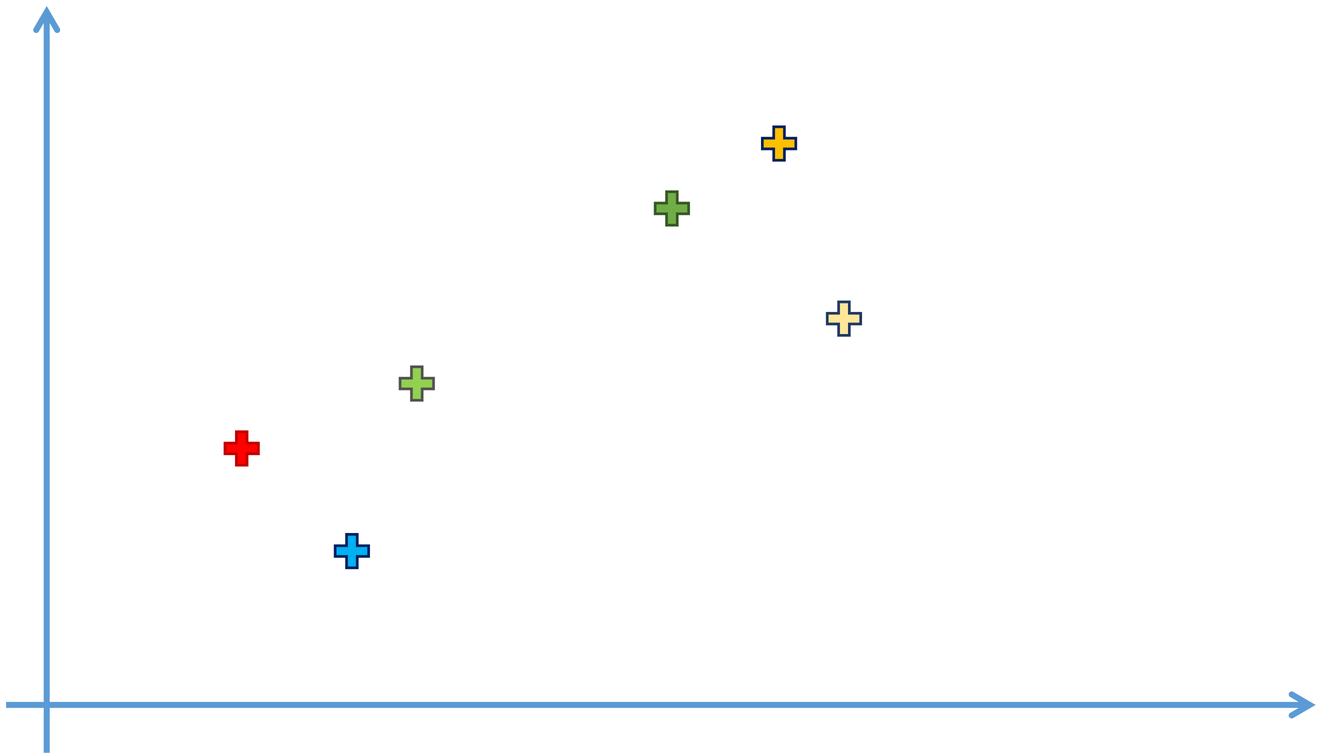
Consider the following dataset of N = 6 data points

Agglomerative HC

STEP 1: Make each data point a single-point cluster ➡ That forms 6 clusters

Agglomerative HC

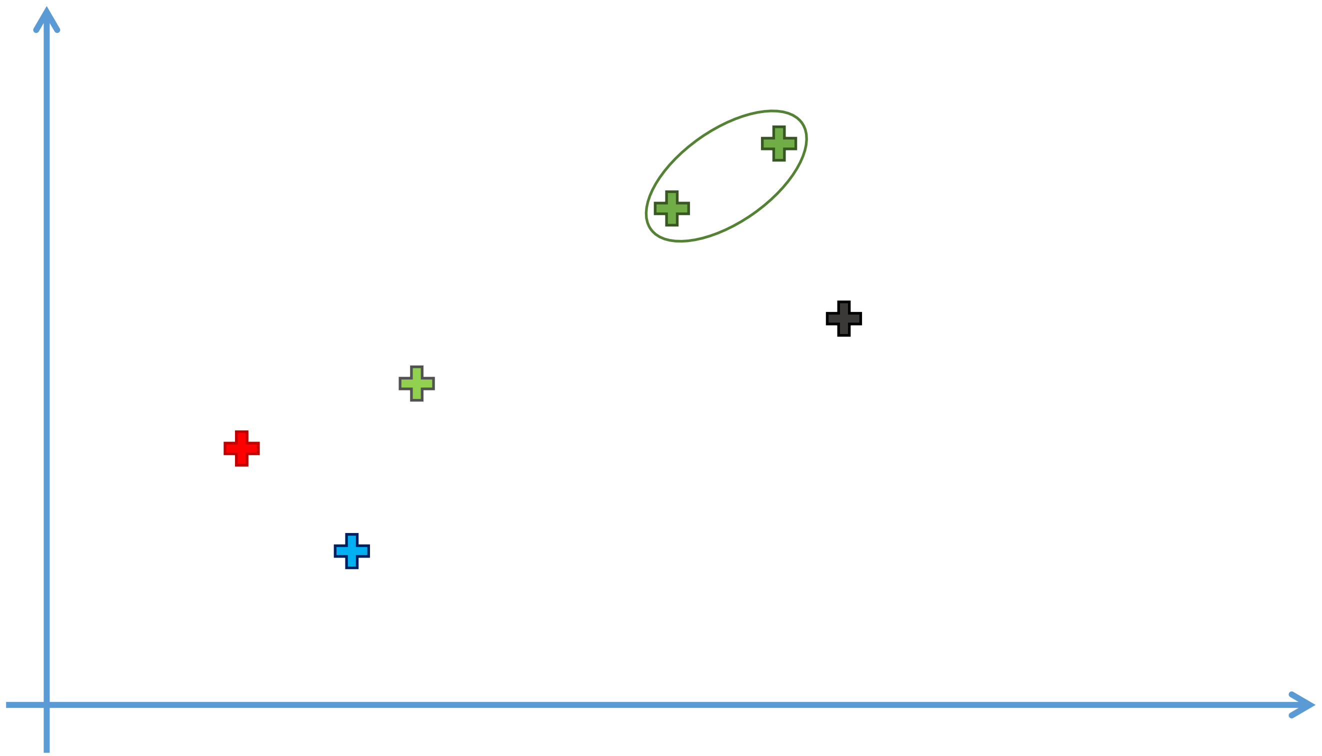
STEP 1: Make each data point a single-point cluster ➡ That forms 6 clusters



Agglomerative HC

STEP 2: Take the two closest data points and make them one cluster

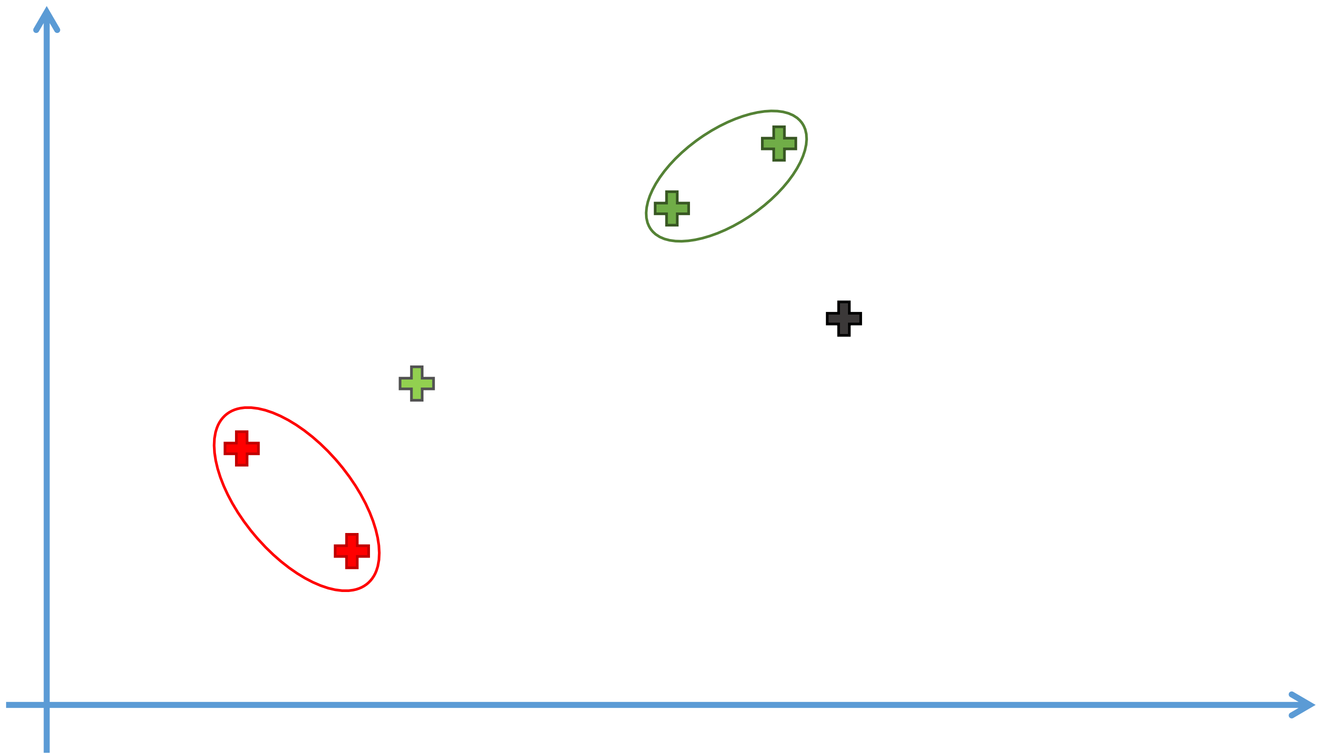
➡ That forms 5 clusters



Agglomerative HC

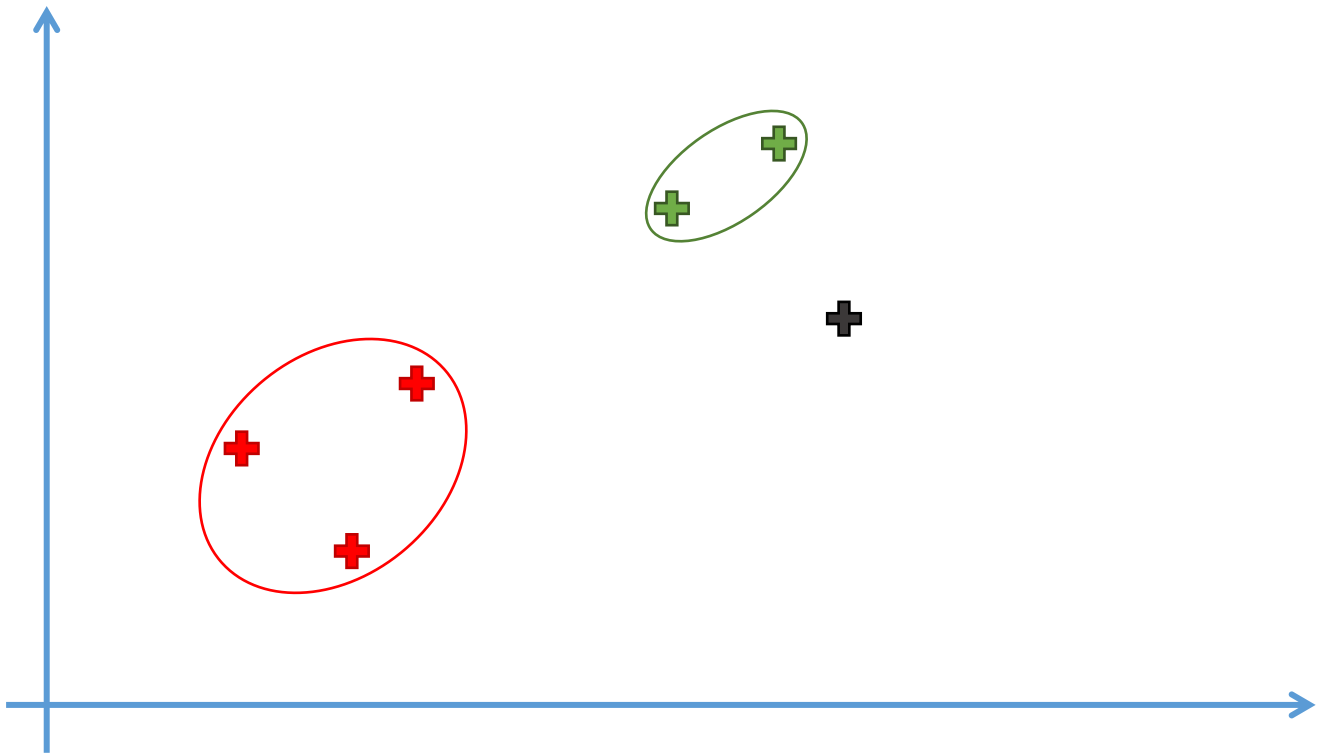
STEP 3: Take the two closest clusters and make them one cluster

➡That forms 4 clusters



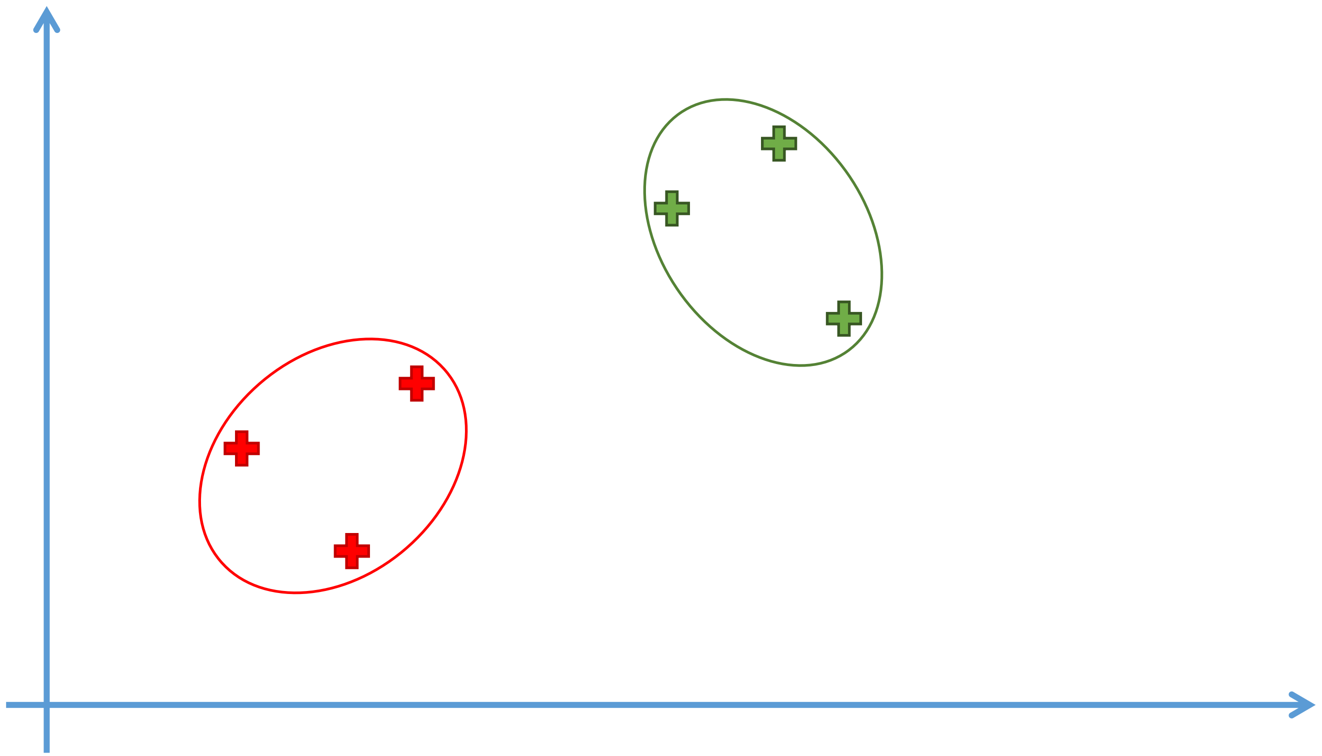
Agglomerative HC

STEP 4: Repeat STEP 3 until there is only one cluster



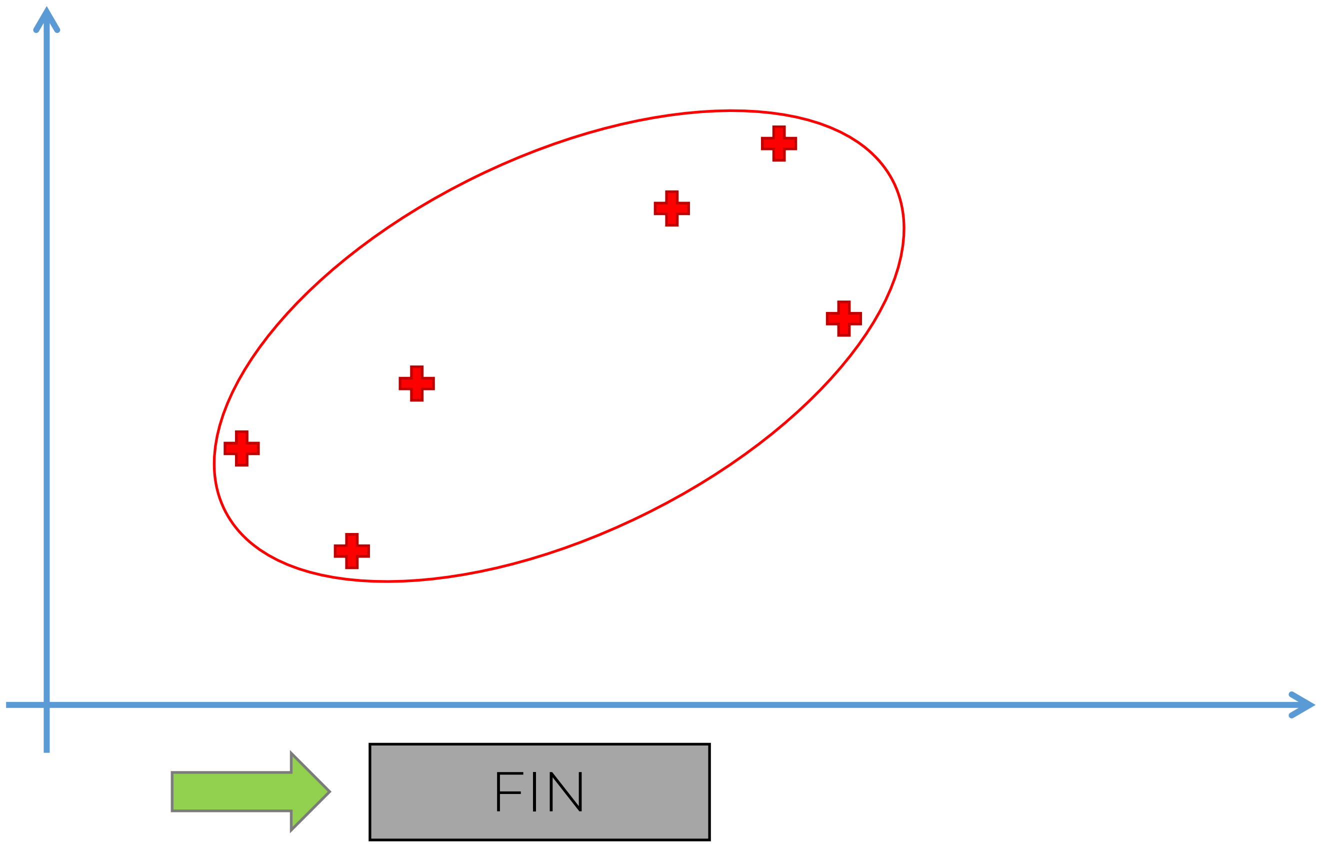
Agglomerative HC

STEP 4: Repeat STEP 3 until there is only one cluster



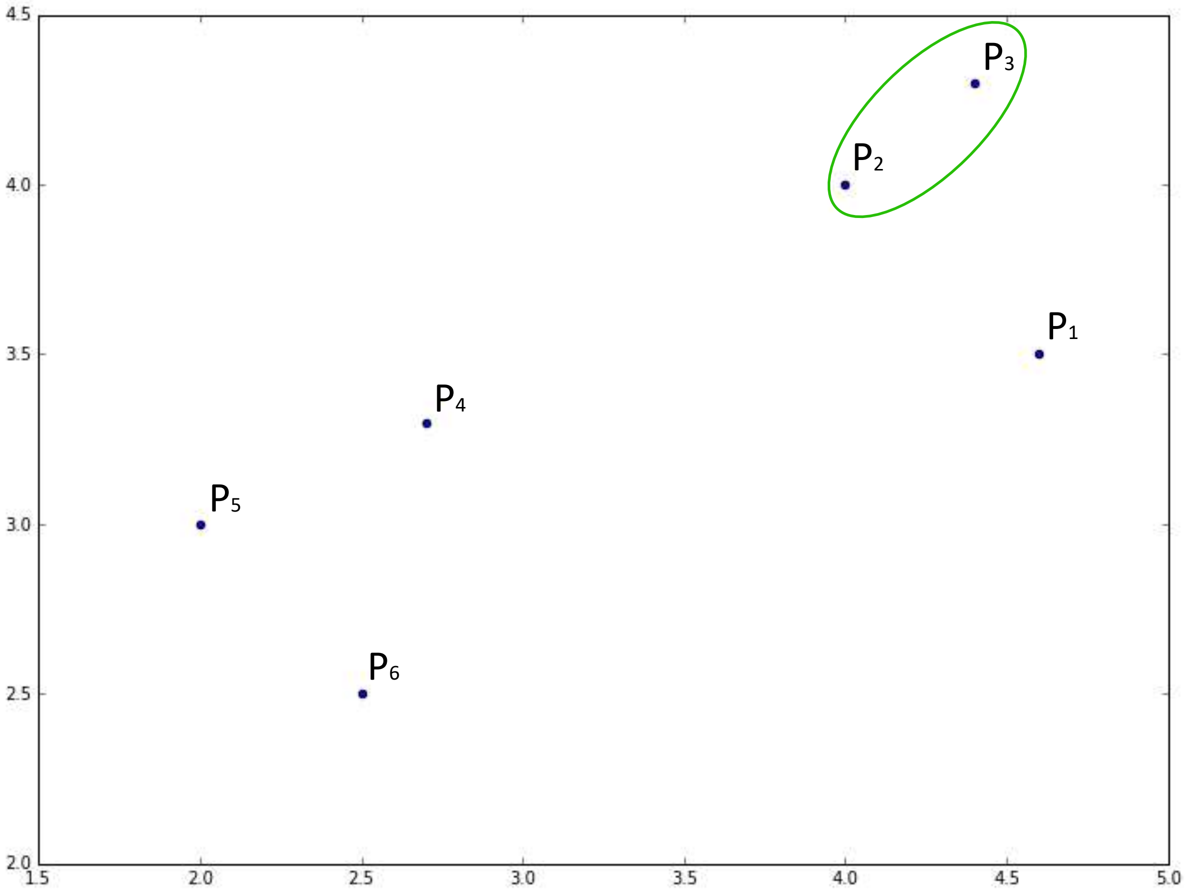
Agglomerative HC

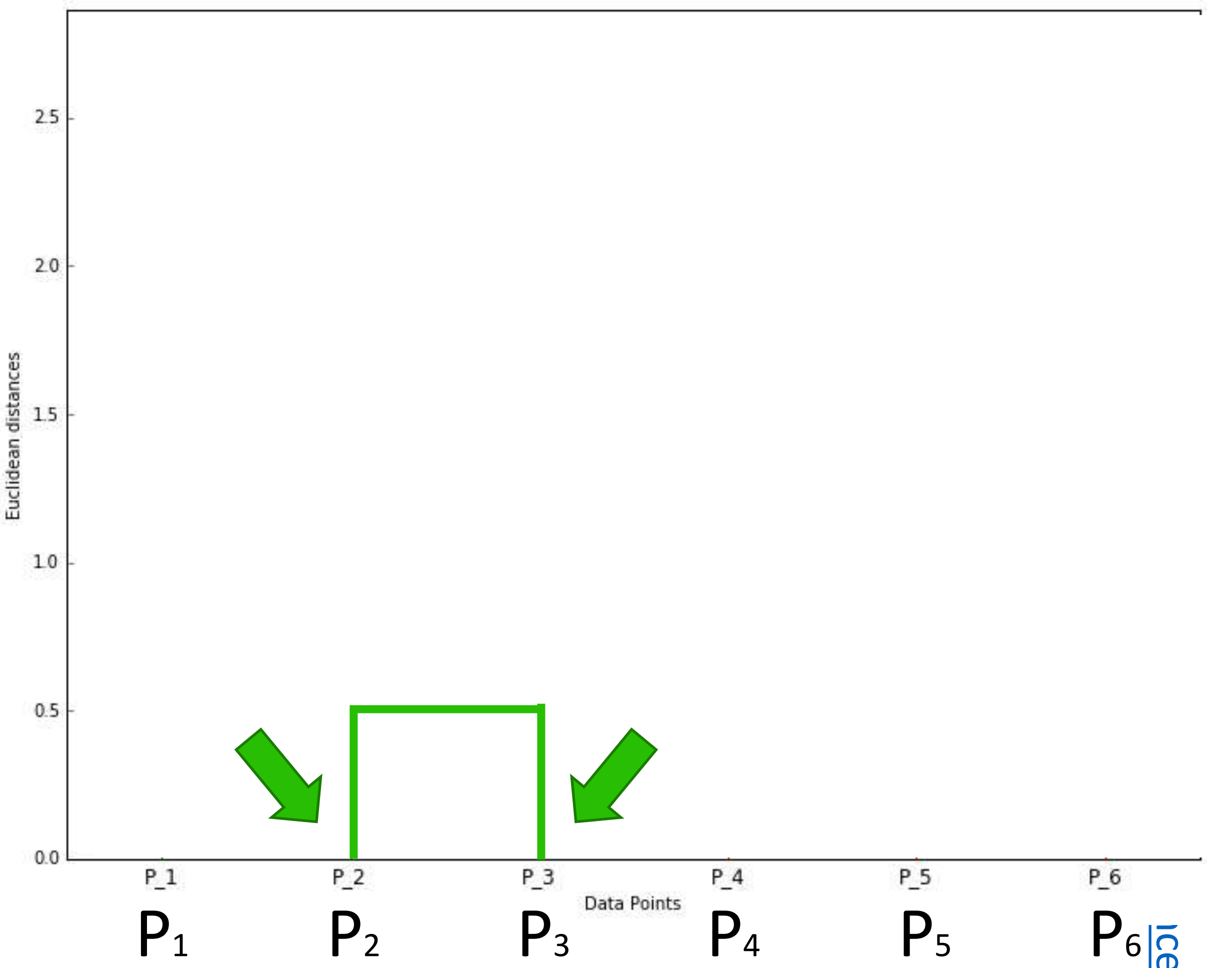
STEP 4: Repeat STEP 3 until there is only one cluster



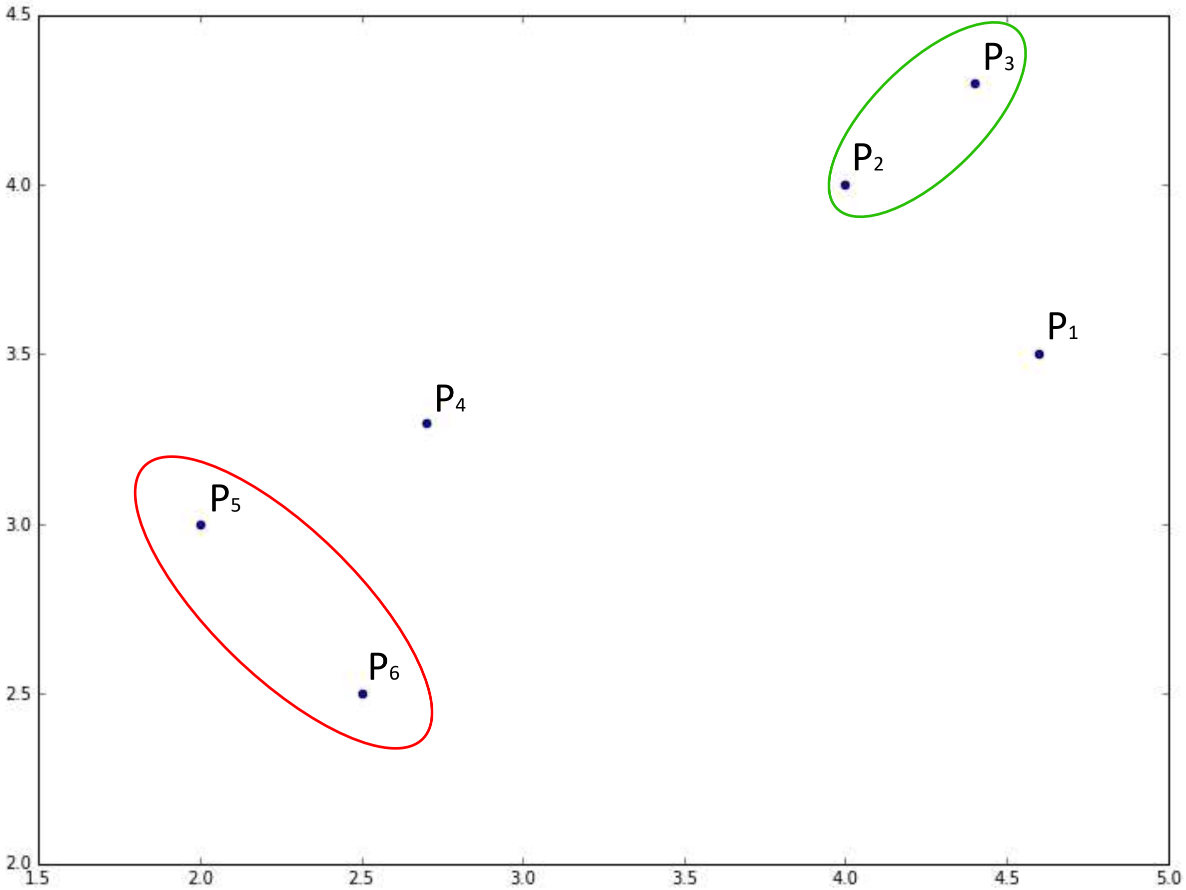
HC Intuition: How Do Dendograms Work?

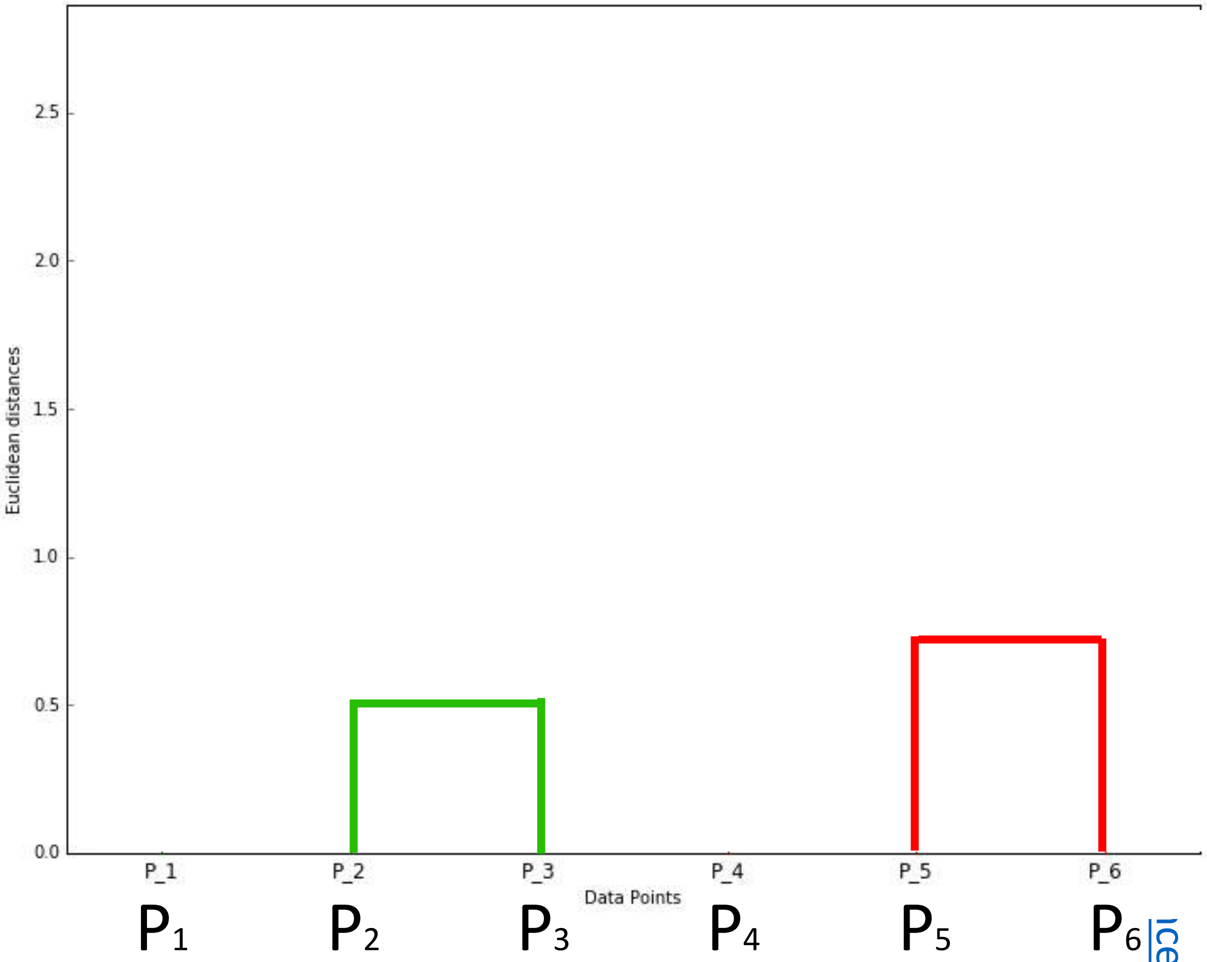
How Do Dendograms Work?



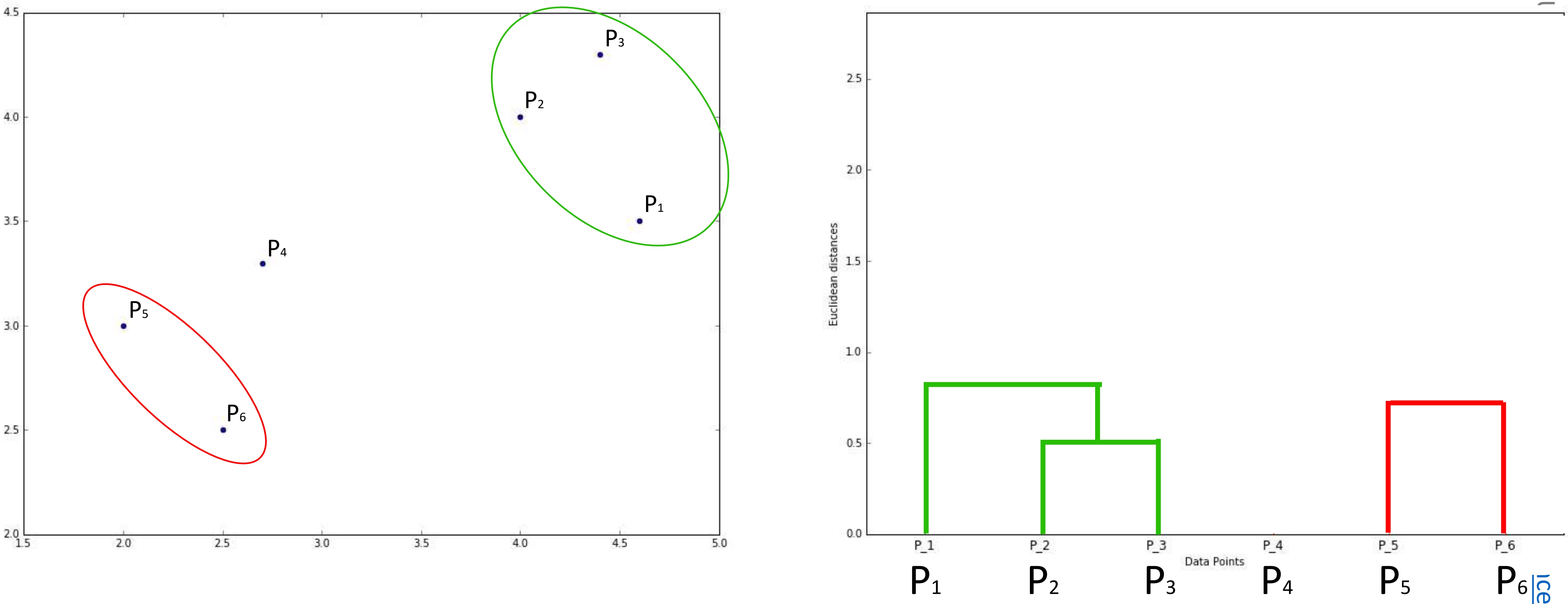


How Do Dendograms Work?

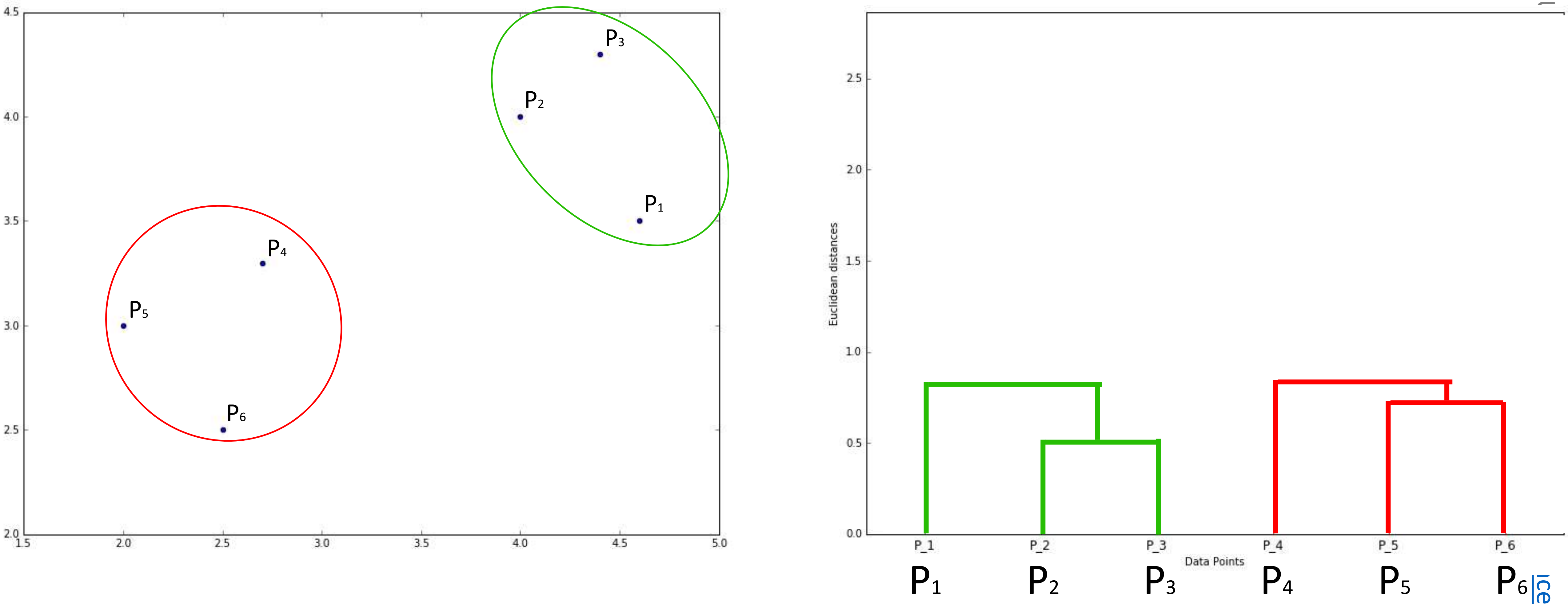




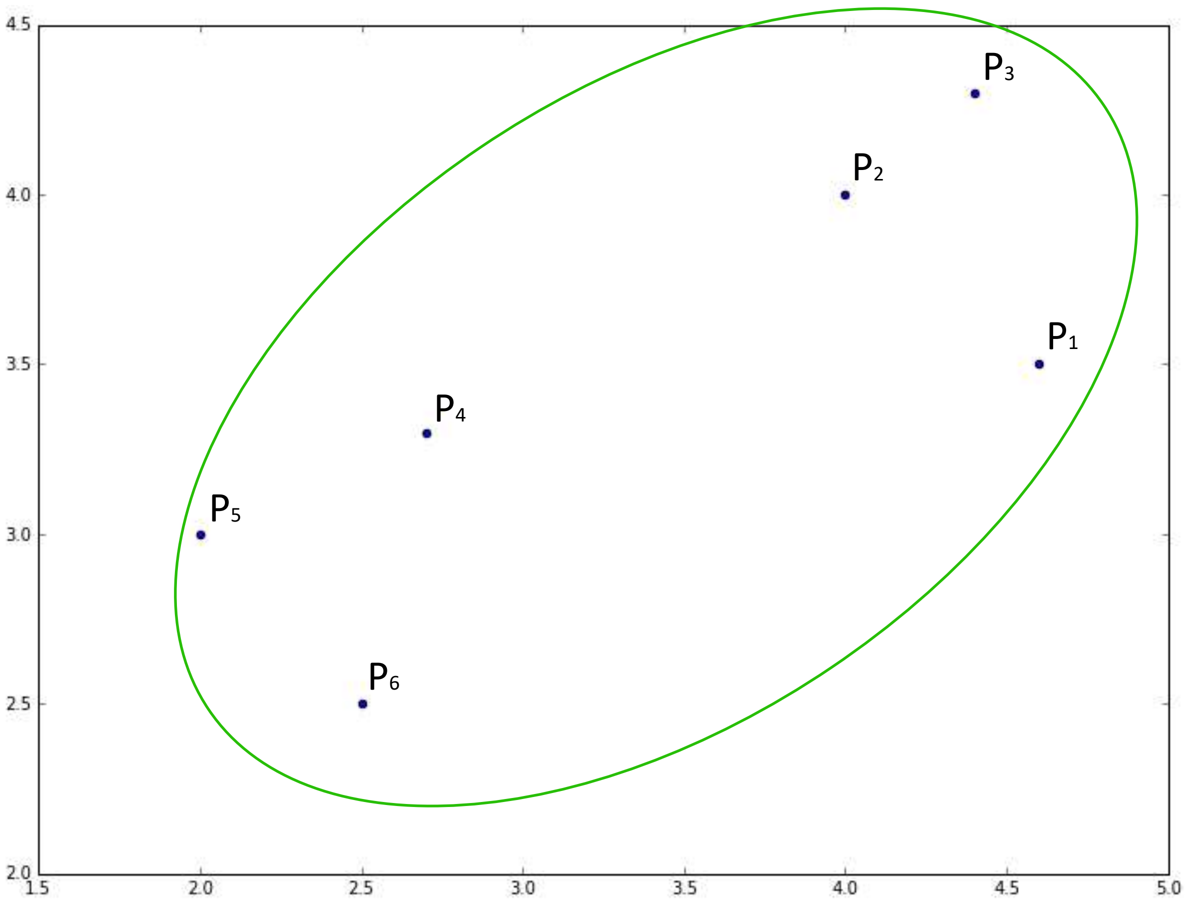
How Do Dendograms Work?

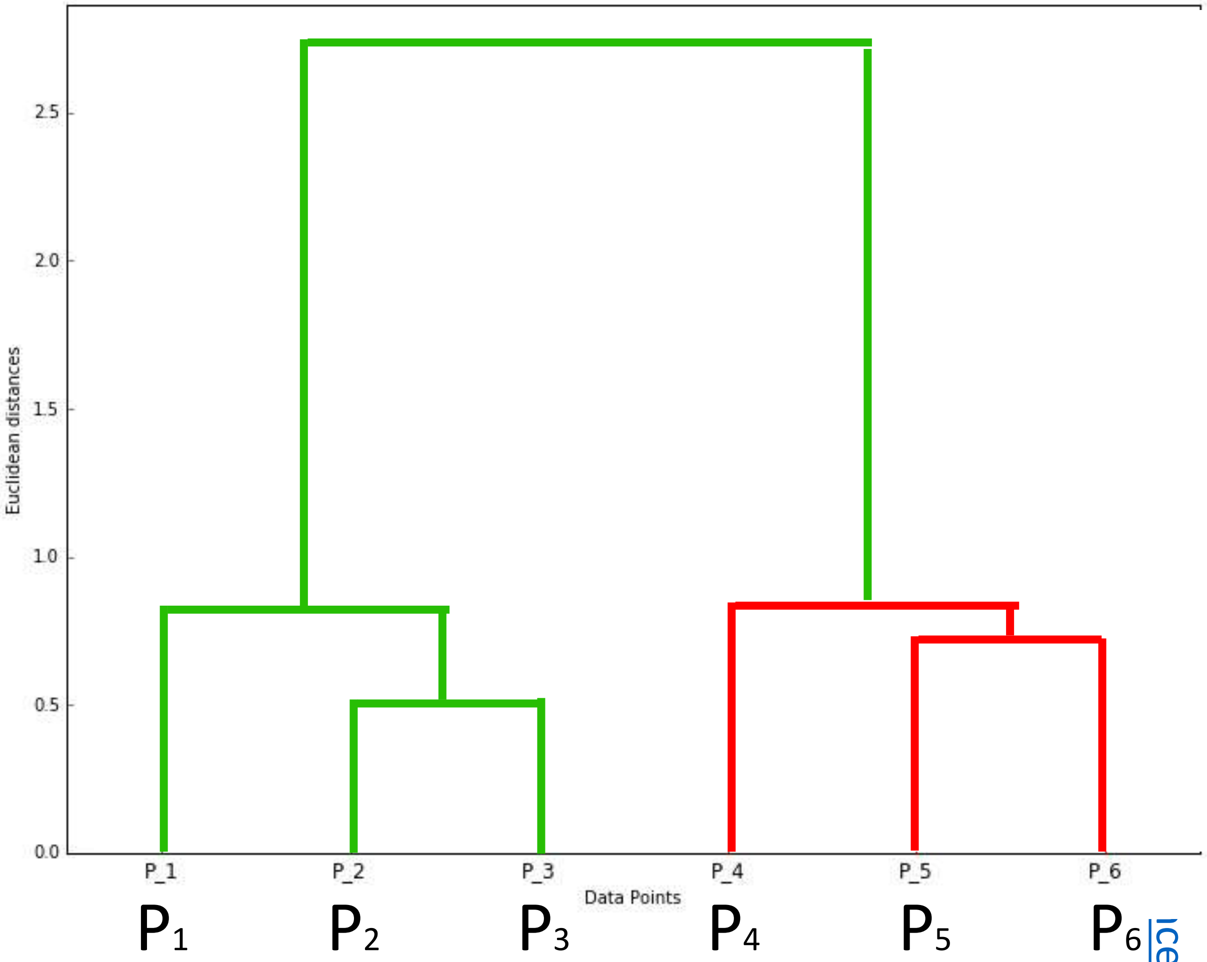


How Do Dendograms Work?

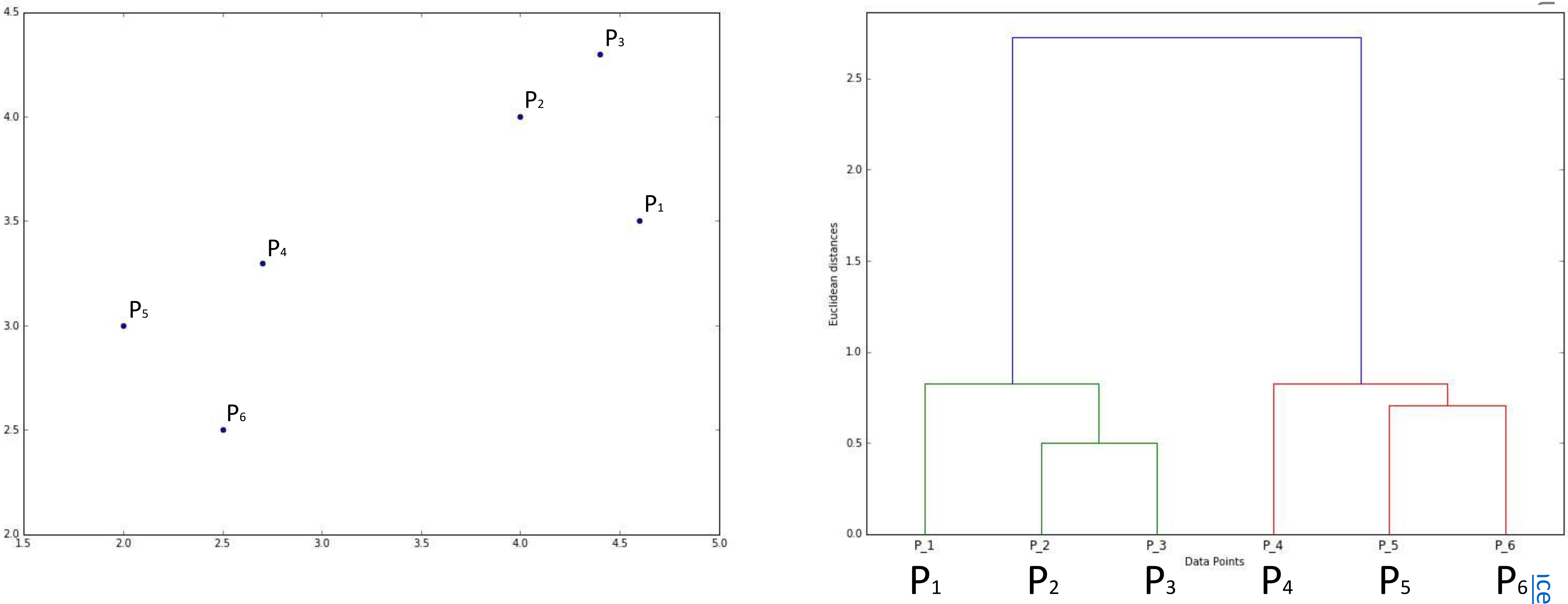


How Do Dendograms Work?



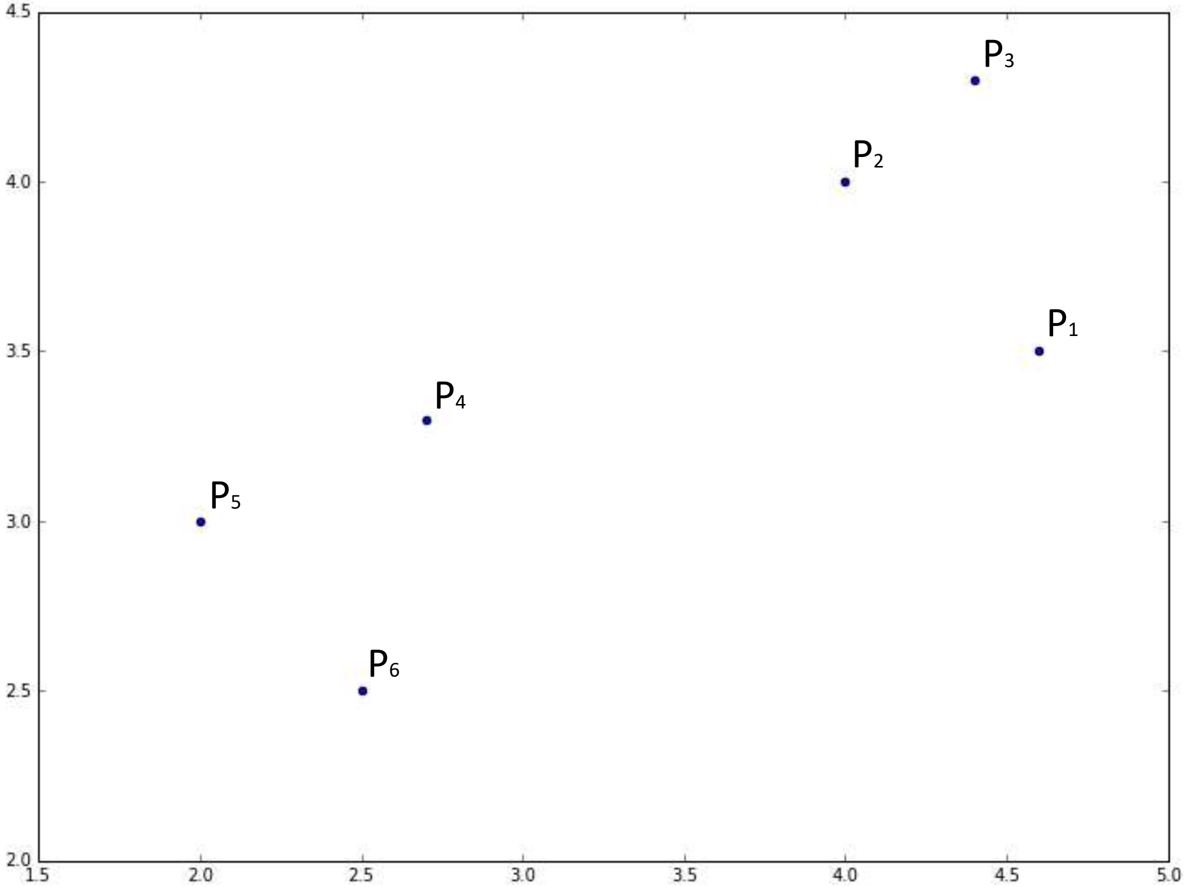


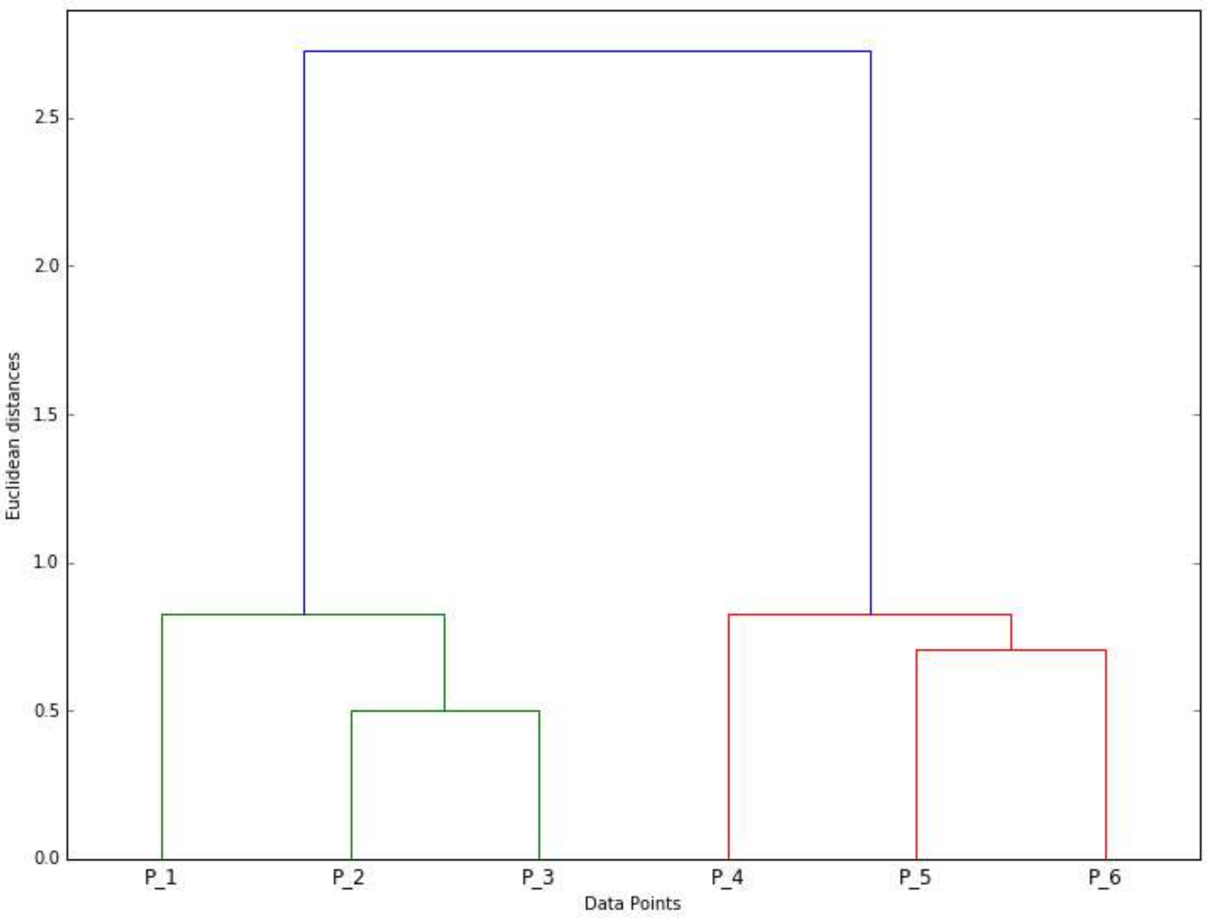
How Do Dendograms Work?



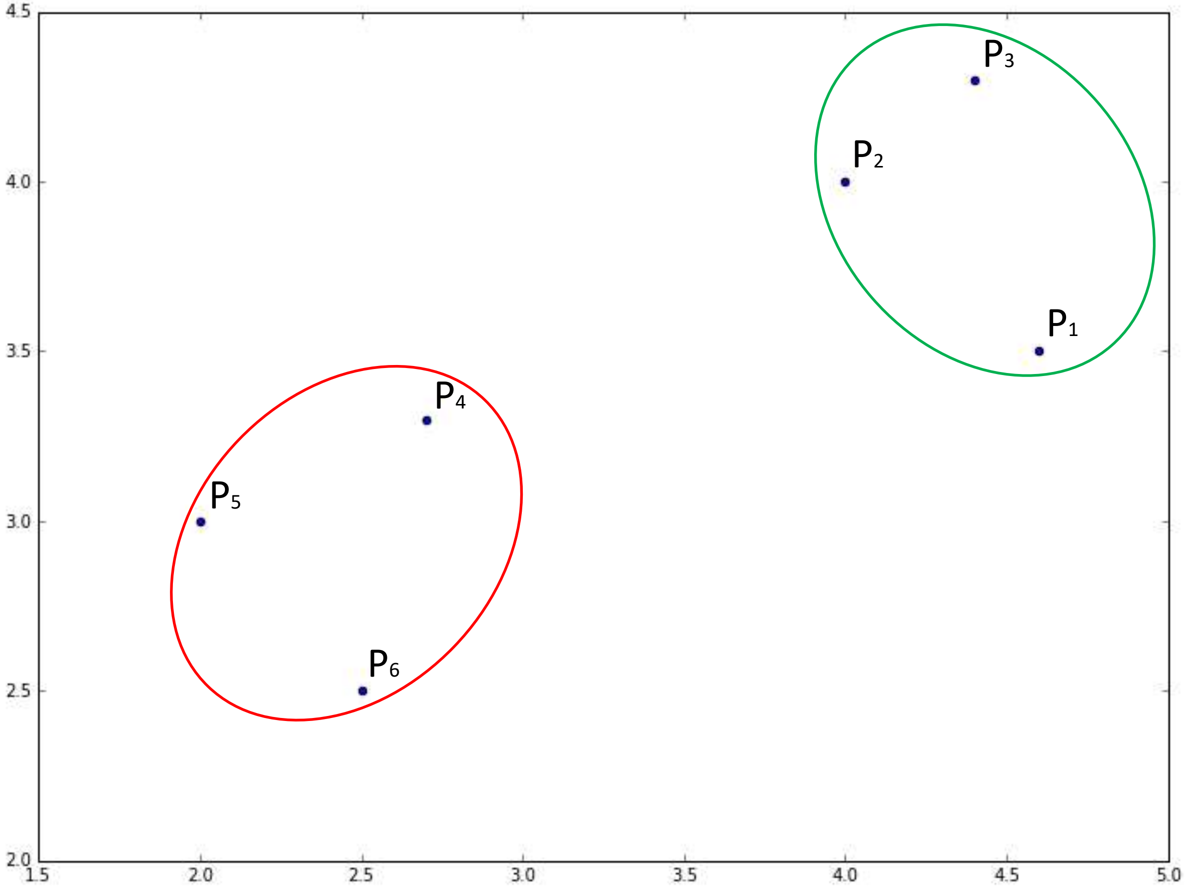
HC Intuition: Using Dendrograms

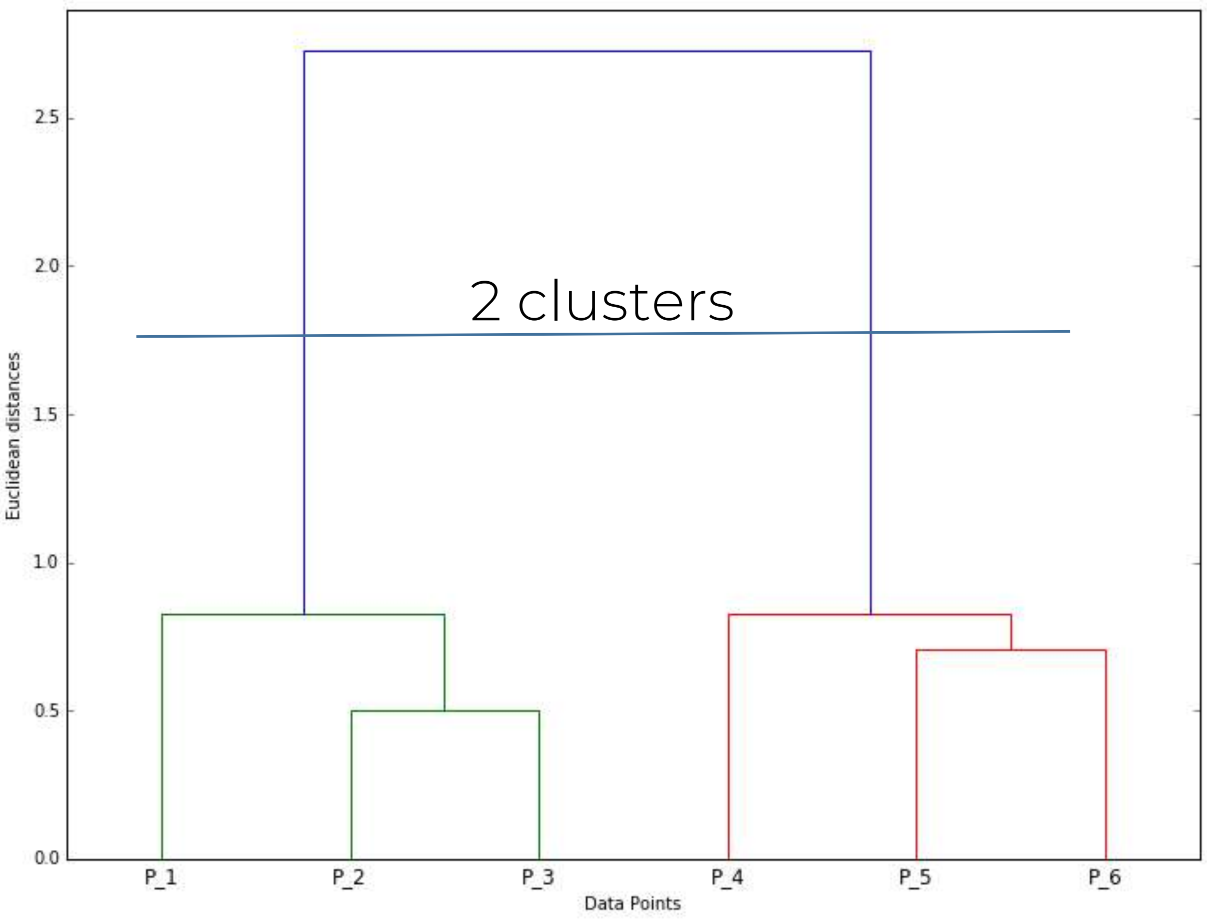
Dendrograms – Two Clusters



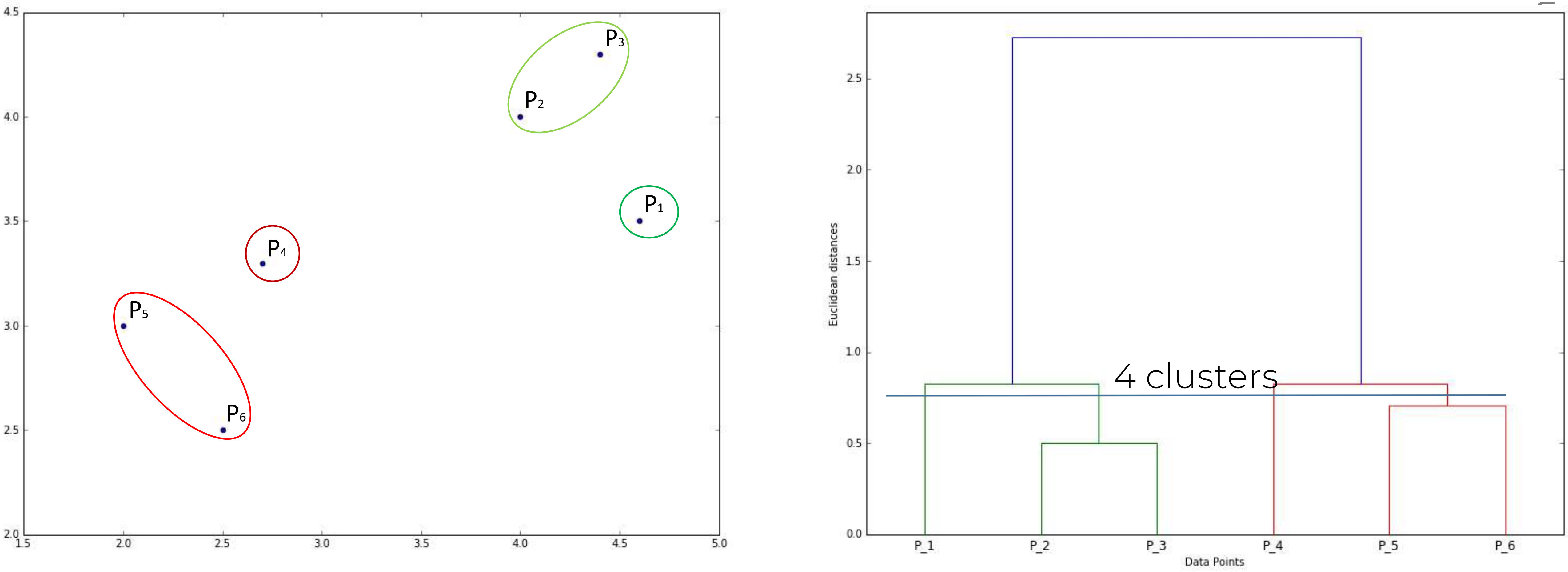


Dendrograms – Two Clusters

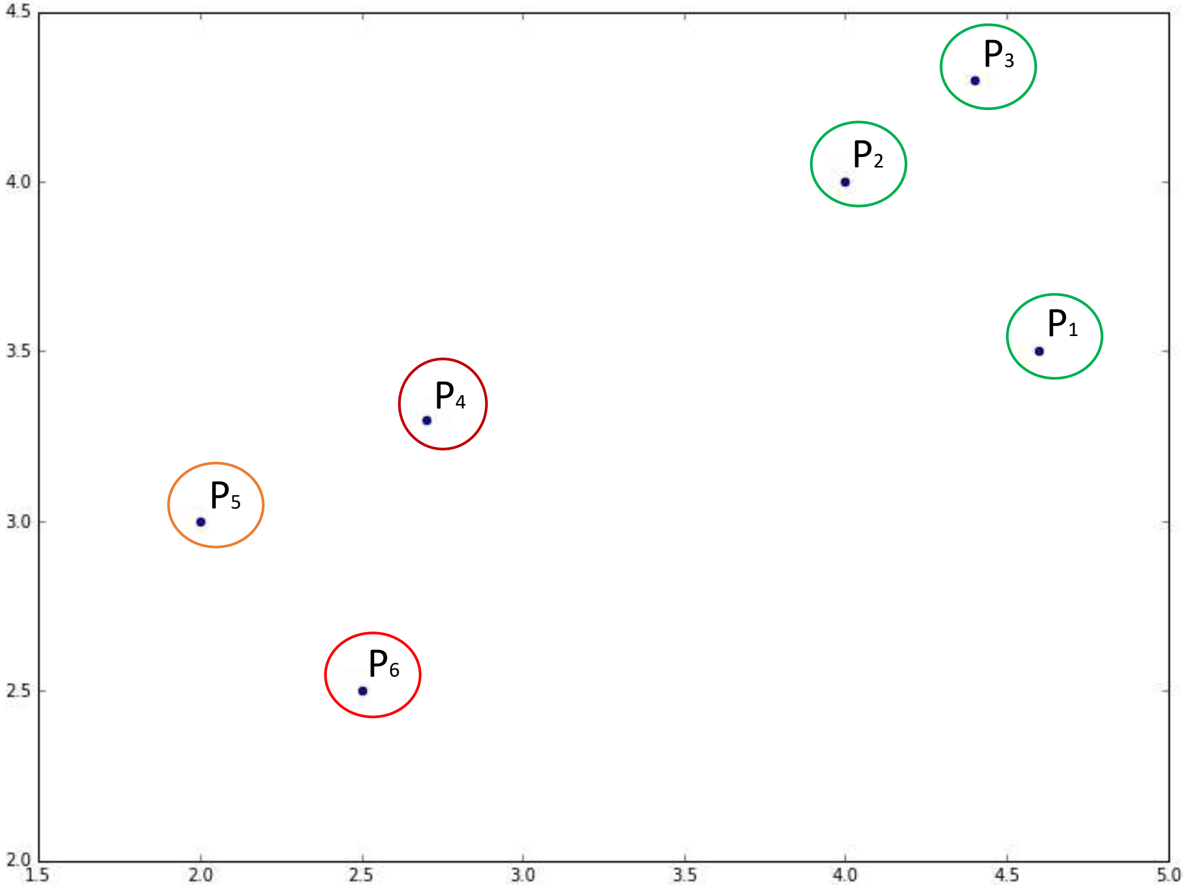


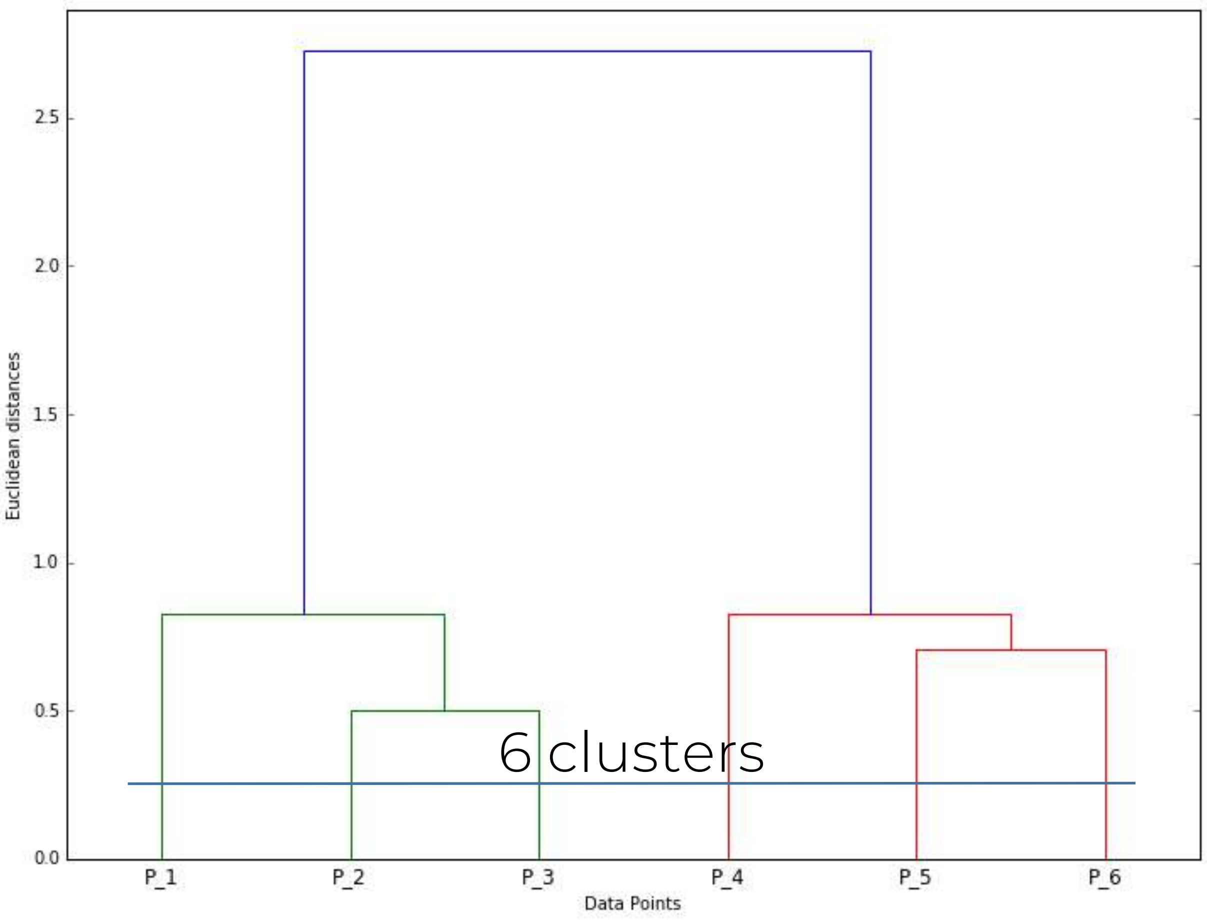


Dendrograms – Four Clusters

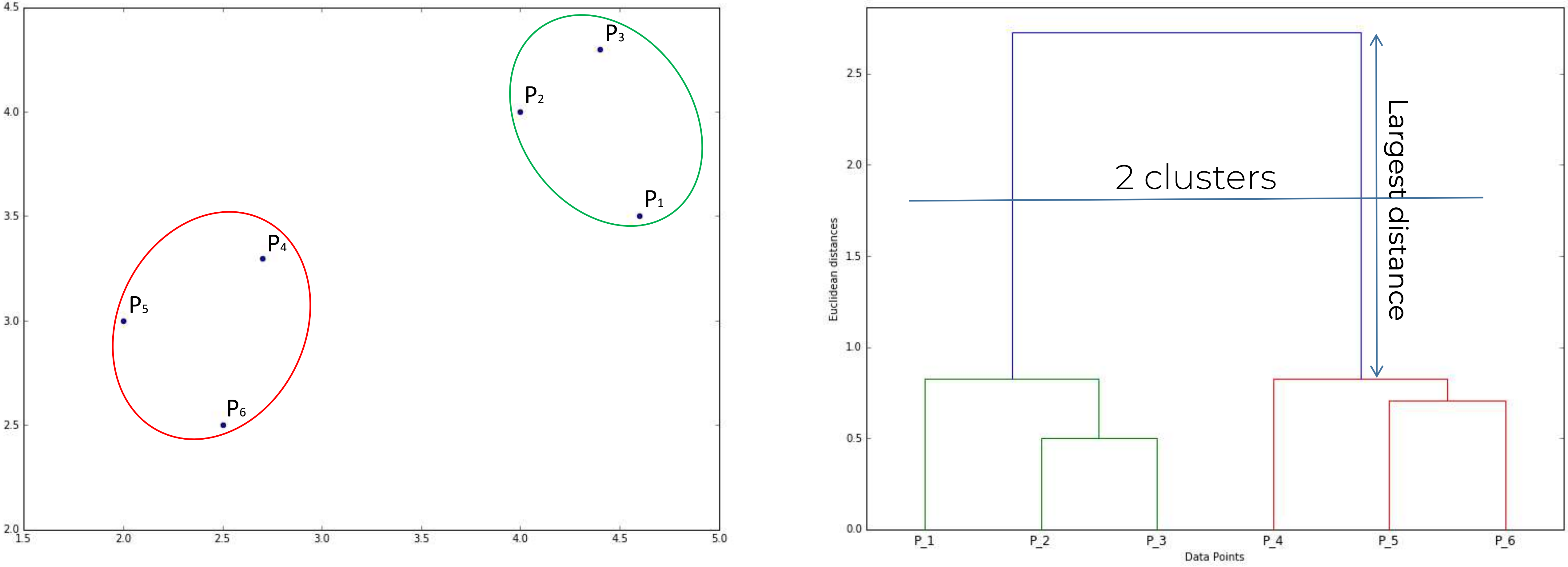


Dendrograms – Six Clusters



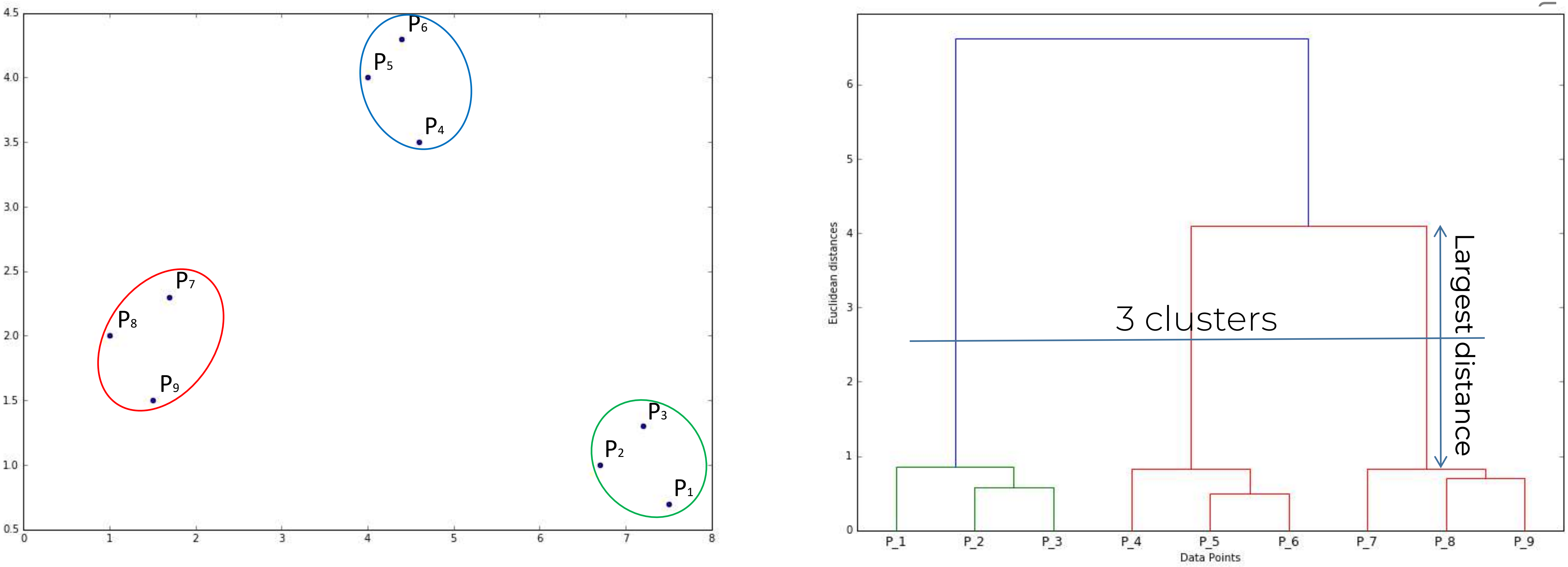


Dendrograms – Optimal # of Clusters



Knowledge Test

Dendrograms – Knowledge Test



Association Rule Learning
Apriori Intuition

ARL - What is it all about ?

ARL - What is it all about ?

People who bought also bought …

ARL- Movie Recommendation

User ID	Movies liked
46578	Movie1, Movie2, Movie3, Movie4
98989	Movie1, Movie2
71527	Movie1, Movie2, Movie4
78981	Movie1, Movie2
89192	Movie2, Movie4
61557	Movie1, Movie3

Potential Rules:

Movie1 ⇨ Movie2

Movie2 ⇨ Movie4

Movie1 ⇨ Movie3

ARL- Market Basket Optimisation

Transaction ID	Products purchased
46578	Burgers, French Fries, Vegetables
98989	Burgers, French Fries, Ketchup
71527	Vegetables, Fruits
78981	Pasta, Fruits, Butter, Vegetables
89192	Burgers, Pasta, French Fries
61557	Fruits, Orange Juice, Vegetables
87923	Burgers, French Fries, Ketchup, Mayo

Potential Rules:

Burgers → French Fries

Vegetables → Fruits

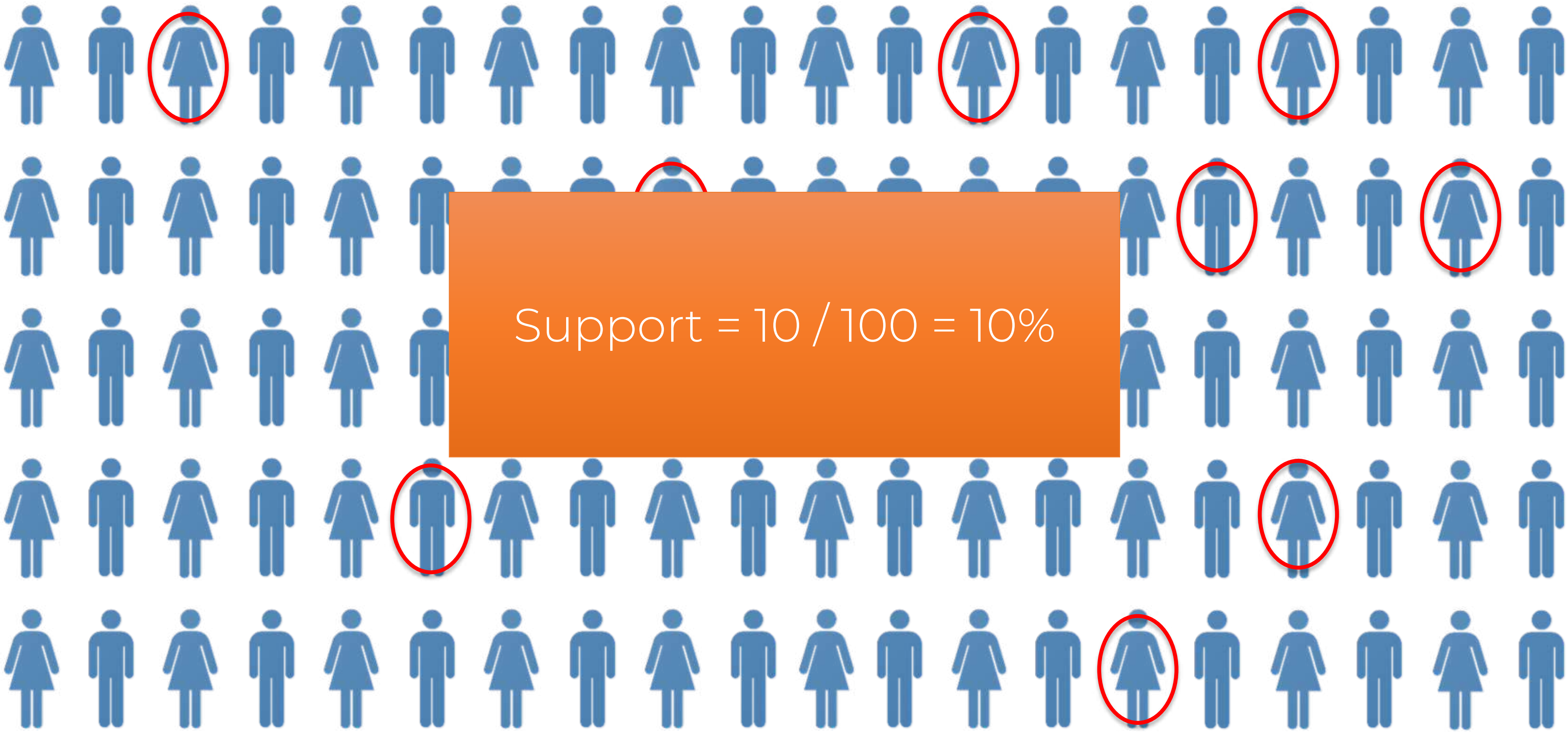
Burgers, French Fries → Ketchup

Apriori - Support

Movie Recommendation: $$\text{support}(\boldsymbol{M}) = \frac{\#\ \text{user watchlists containing } \boldsymbol{M}}{\#\ \text{user watchlists}}$$

Market Basket Optimisation: $$\text{support}(\boldsymbol{I}) = \frac{\#\ \text{transactions containing } \boldsymbol{I}}{\#\ \text{transactions}}$$

Apriori - Support



Apriori- Confidence

Movie Recommendation: $$\text{confidence}(\boldsymbol{M_1} \rightarrow \boldsymbol{M_2}) = \frac{\#\ \text{user watchlists containing } \boldsymbol{M_1} \text{ and } \boldsymbol{M_2}}{\#\ \text{user watchlists containing } \boldsymbol{M_1}}$$

Market Basket Optimisation: $$\text{confidence}(\boldsymbol{I_1} \rightarrow \boldsymbol{I_2}) = \frac{\#\ \text{transactions containing } \boldsymbol{I_1} \text{ and } \boldsymbol{I_2}}{\#\ \text{transactions containing } \boldsymbol{I_1}}$$

Apriori - Confidence

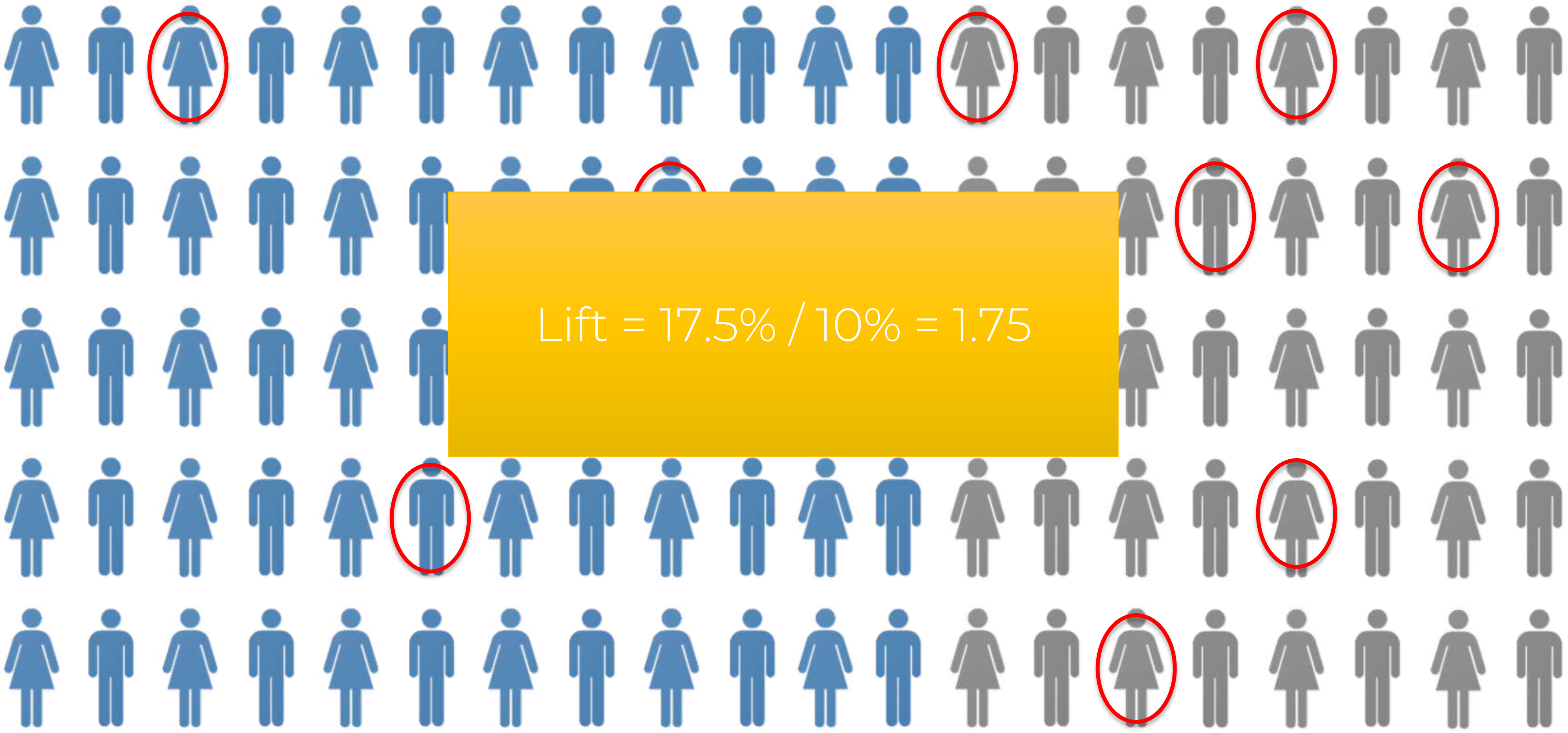
Apriori - Confidence

Apriori - Lift

Movie Recommendation: $$\mathrm{lift}(\boldsymbol{M_1} \rightarrow \boldsymbol{M_2}) = \frac{\mathrm{confidence}(\boldsymbol{M_1} \rightarrow \boldsymbol{M_2})}{\mathrm{support}(\boldsymbol{M_2})}$$

Market Basket Optimisation: $$\mathrm{lift}(\boldsymbol{I_1} \rightarrow \boldsymbol{I_2}) = \frac{\mathrm{confidence}(\boldsymbol{I_1} \rightarrow \boldsymbol{I_2})}{\mathrm{support}(\boldsymbol{I_2})}$$

Apriori- Lift



Apriori - Algorithm

Step 1: Set a minimum support and confidence

Step 2: Take all the subsets in transactions having higher support than minimum support

Step 3: Take all the rules of these subsets having higher confidence than minimum confidence

Step 4: Sort the rules by decreasing lift

Association Rule Learning
Eclat Intuition

ARL- What is it all about ?

People who bought also bought …

ARL - Movie Recommendation

User ID	Movies liked
46578	Movie1, Movie2, Movie3, Movie4
98989	Movie1, Movie2
71527	Movie1, Movie2, Movie4
78981	Movie1, Movie2
89192	Movie2, Movie4
61557	Movie1, Movie3

Potential Rules:

Movie1 ⇒ Movie2

Movie2 ⇒ Movie4

Movie1 ⇒ Movie3

ARL - Market Basket Optimisation

Transaction ID	Products purchased
46578	Burgers, French Fries, Vegetables
98989	Burgers, French Fries, Ketchup
71527	Vegetables, Fruits
78981	Pasta, Fruits, Butter, Vegetables
89192	Burgers, Pasta, French Fries
61557	Fruits, Orange Juice, Vegetables
87923	Burgers, French Fries, Ketchup, Mayo

Potential Rules:

Burgers $\Rightarrow$ French Fries

Vegetables $\Rightarrow$ Fruits

Burgers, French Fries $\Rightarrow$ Ketchup

Eclat - Support

Movie Recommendation: $$\text{support}(\boldsymbol{M}) = \frac{\#\ \text{user watchlists containing}\ \boldsymbol{M}}{\#\ \text{user watchlists}}$$

Market Basket Optimisation: $$\text{support}(\boldsymbol{I}) = \frac{\#\ \text{transactions containing}\ \boldsymbol{I}}{\#\ \text{transactions}}$$

Eclat- Algorithm

Step 1: Set a minimum support

Step 2: Take all the subsets in transactions having higher support than minimum support

Step 3: Sort these subsets by decreasing support

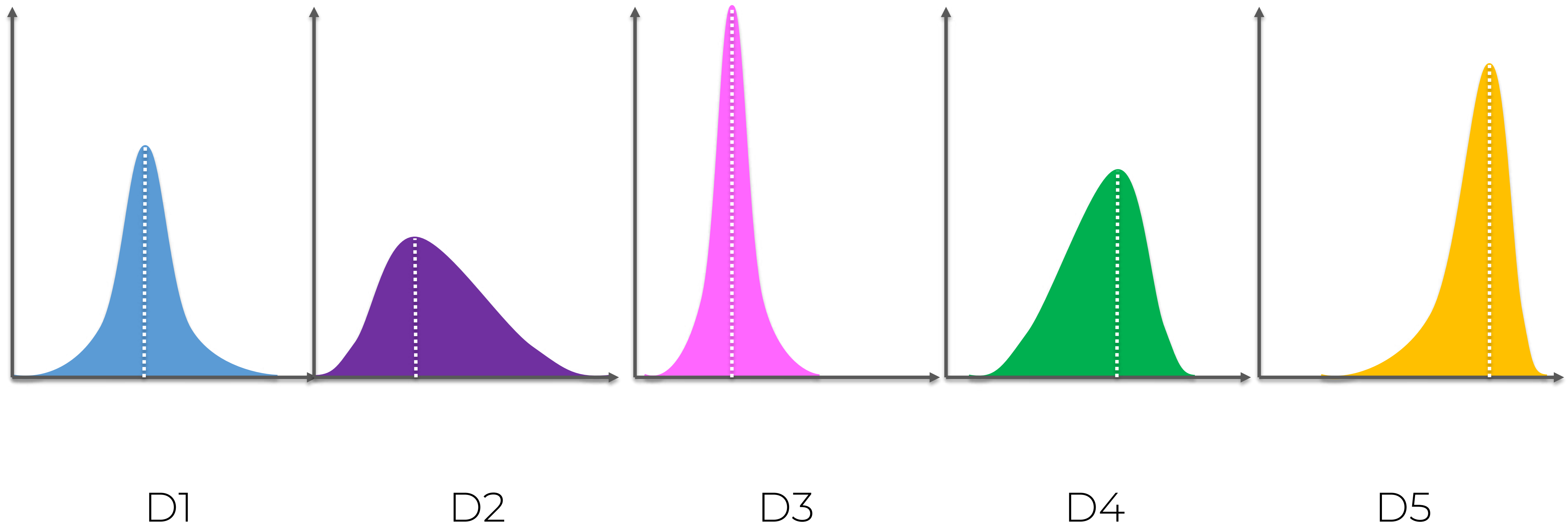
Reinforcement Learning

The Multi-Armed Bandit Problem

The Multi-Armed Bandit Problem

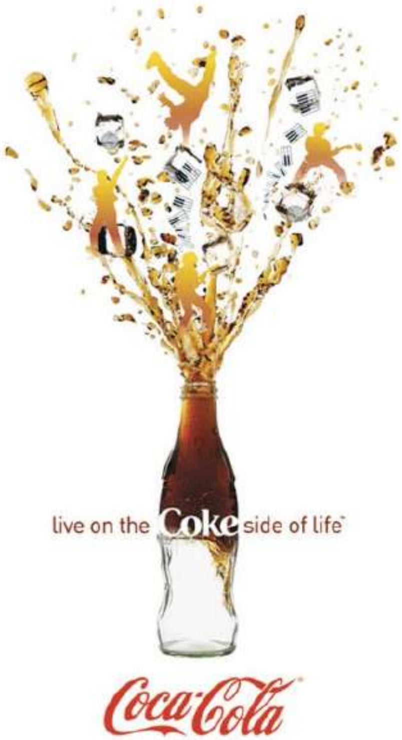
D1 D2 D3 D4 D5

The Multi-Armed Bandit Problem



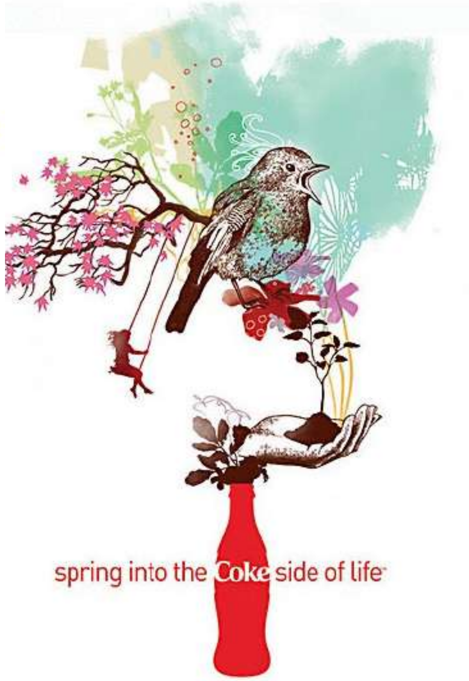
The Multi-Armed Bandit Problem

D1

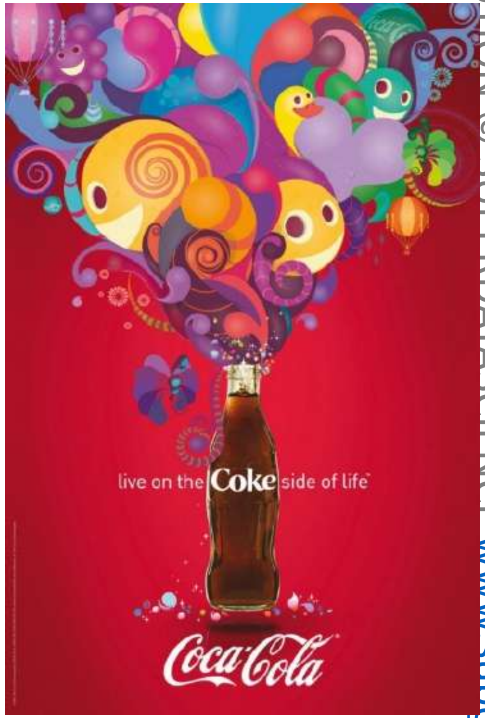


D2

D3



D4



D5

Examples used for educational purposes. No affiliation with Coca-Cola

Upper Confidence Bound Intuition (UCB)

The Multi-Armed Bandit Problem

D1

D2

D3

D4

D5

The Multi-Armed Bandit Problem

- We have d arms. For example, arms are ads that we display to users each time they connect to a web page.
- Each time a user connects to this web page, that makes a round.
- At each round n, we choose one ad to display to the user.
- At each round n, ad i gives reward $r_i(n) \in \{0, 1\}$: $r_i(n) = 1$ if the user clicked on the ad i, 0 if the user didn't.
- Our goal is to maximize the total reward we get over many rounds.

Upper Confidence Bound Algorithm

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i(n)$ - the number of times the ad i was selected up to round n,
- $R_i(n)$ - the sum of rewards of the ad i up to round n.

Step 2. From these two numbers we compute:

- the average reward of ad i up to round n

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

- the confidence interval $[\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)]$ at round n with

$$\Delta_i(n) = \sqrt{\frac{3}{2}\frac{\log(n)}{N_i(n)}}$$

Step 3. We select the ad i that has the maximum UCB $\bar{r}_i(n) + \Delta_i(n)$.

Upper Confidence Bound Algorithm

D1

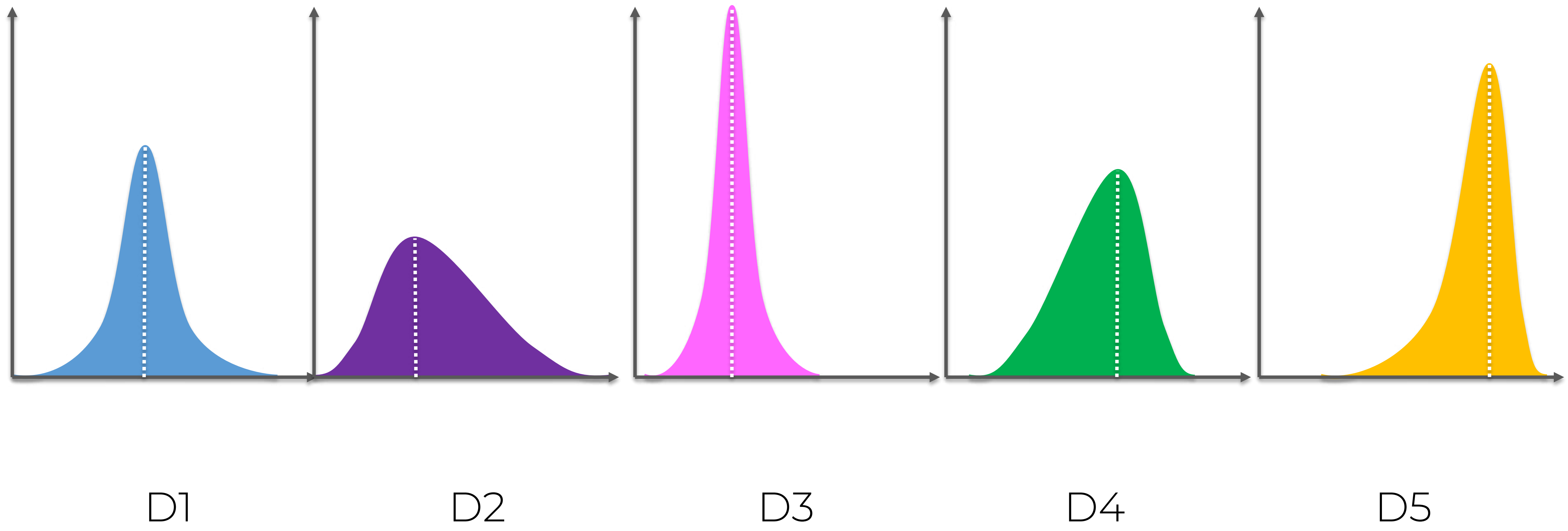
D2

D3

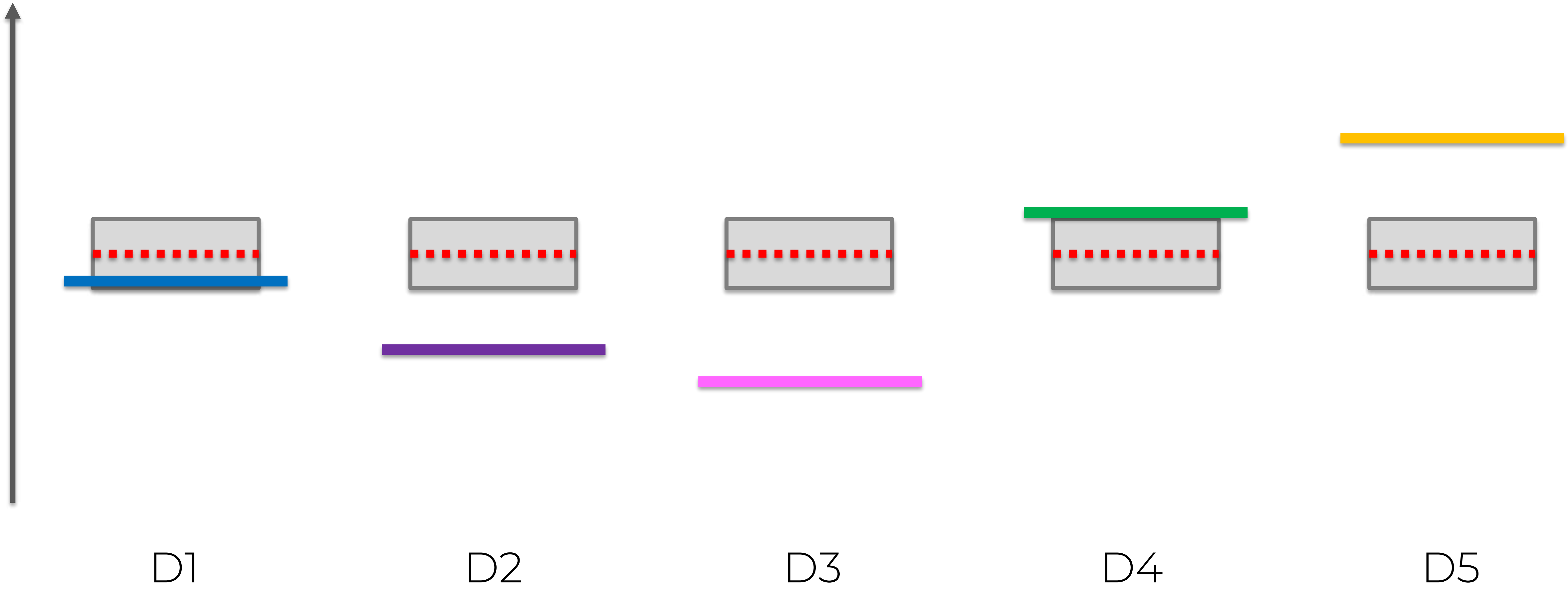
D4

D5

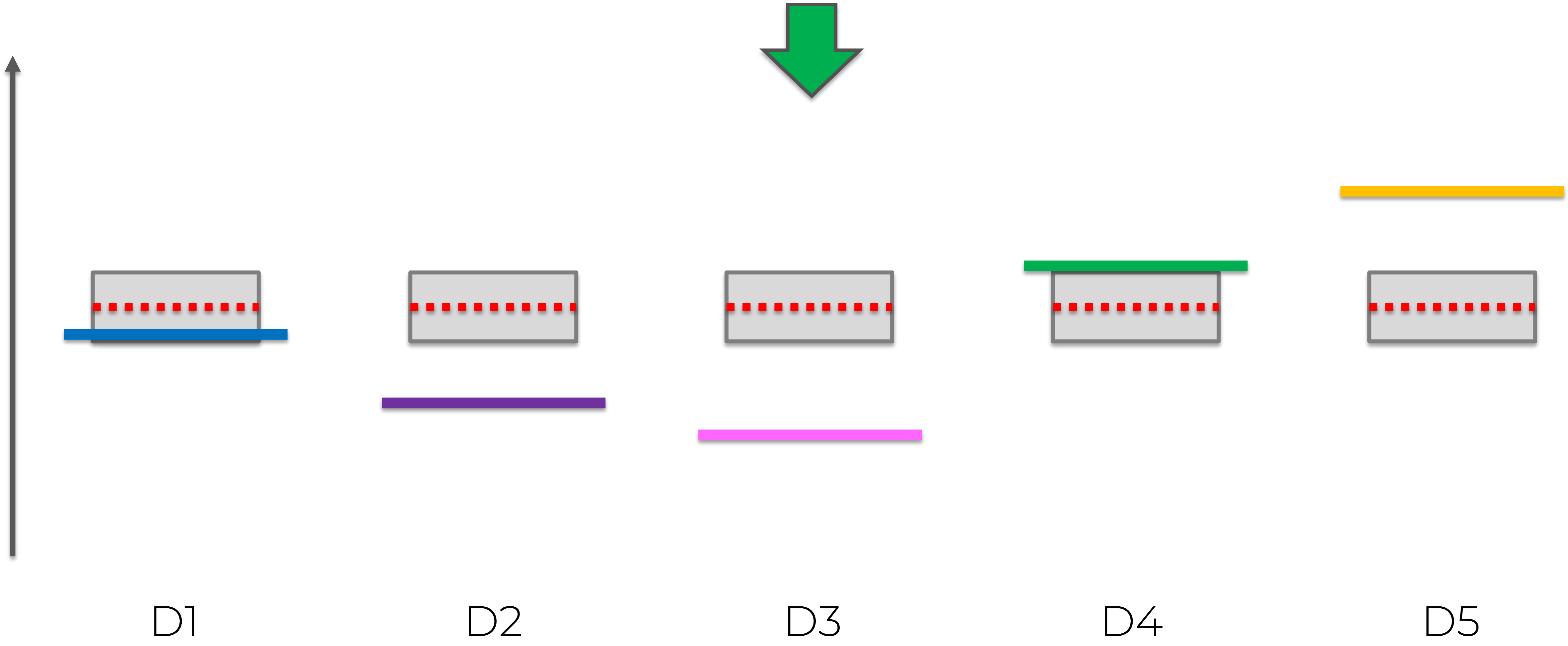
Upper Confidence Bound Algorithm



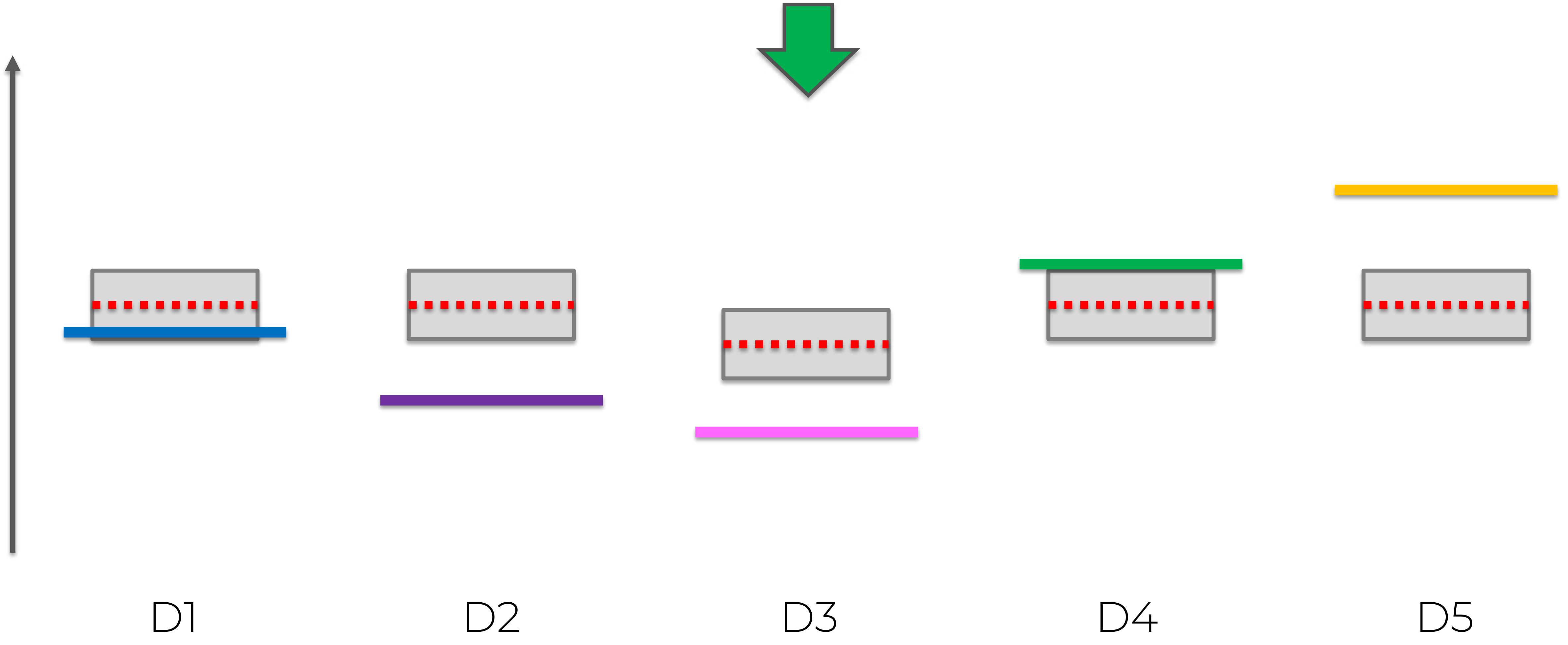
Upper Confidence Bound Algorithm



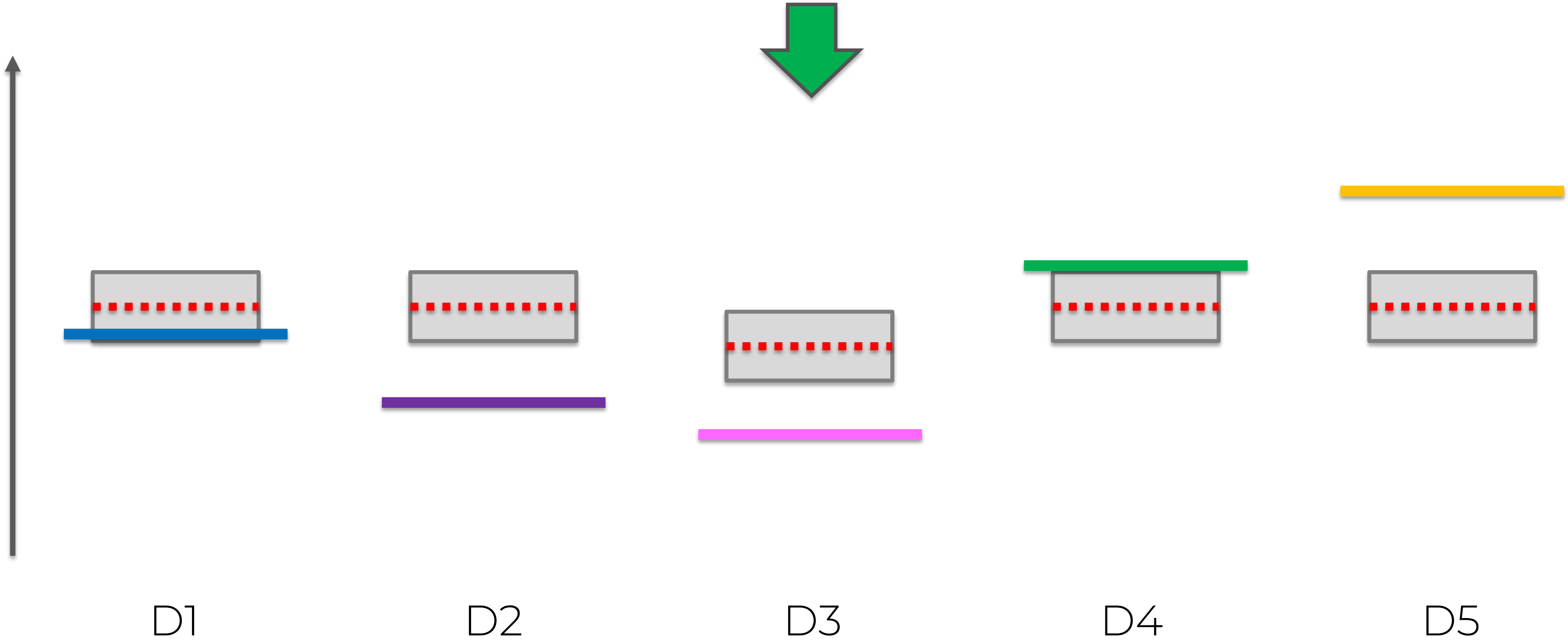
Upper Confidence Bound Algorithm



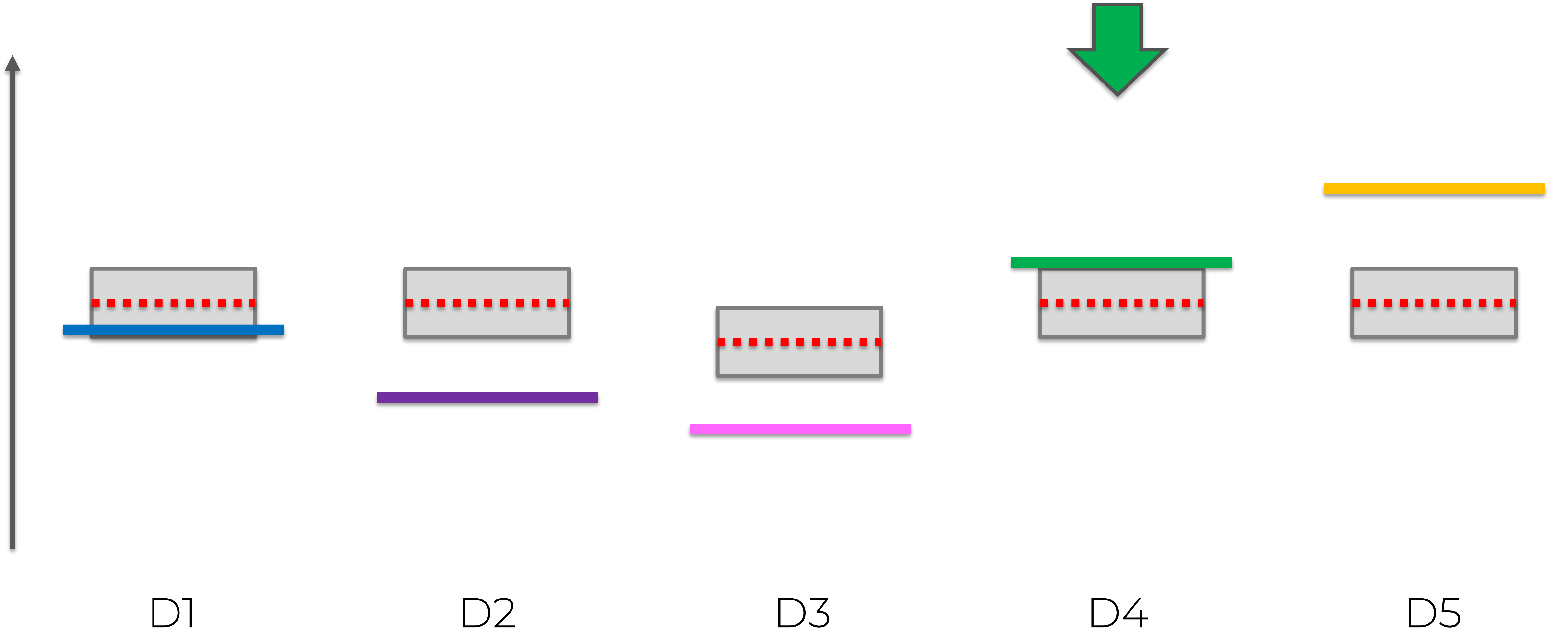
Upper Confidence Bound Algorithm



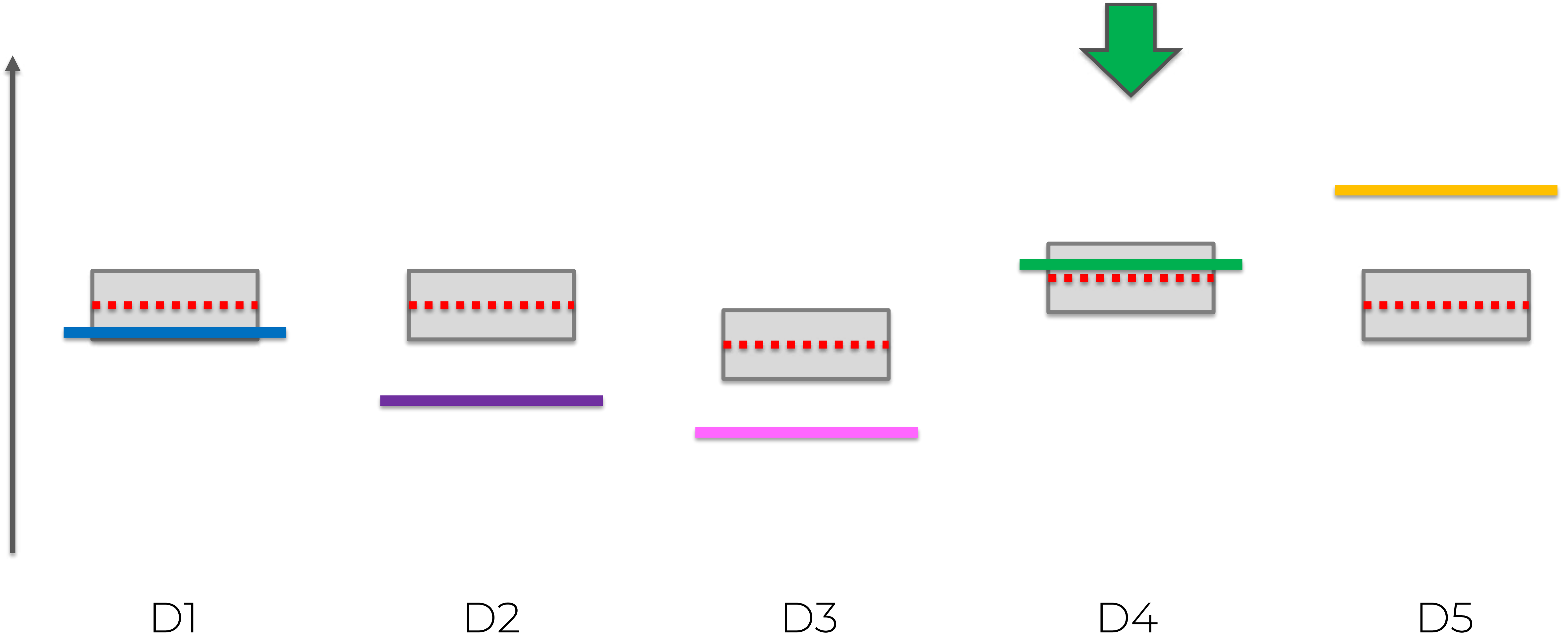
Upper Confidence Bound Algorithm



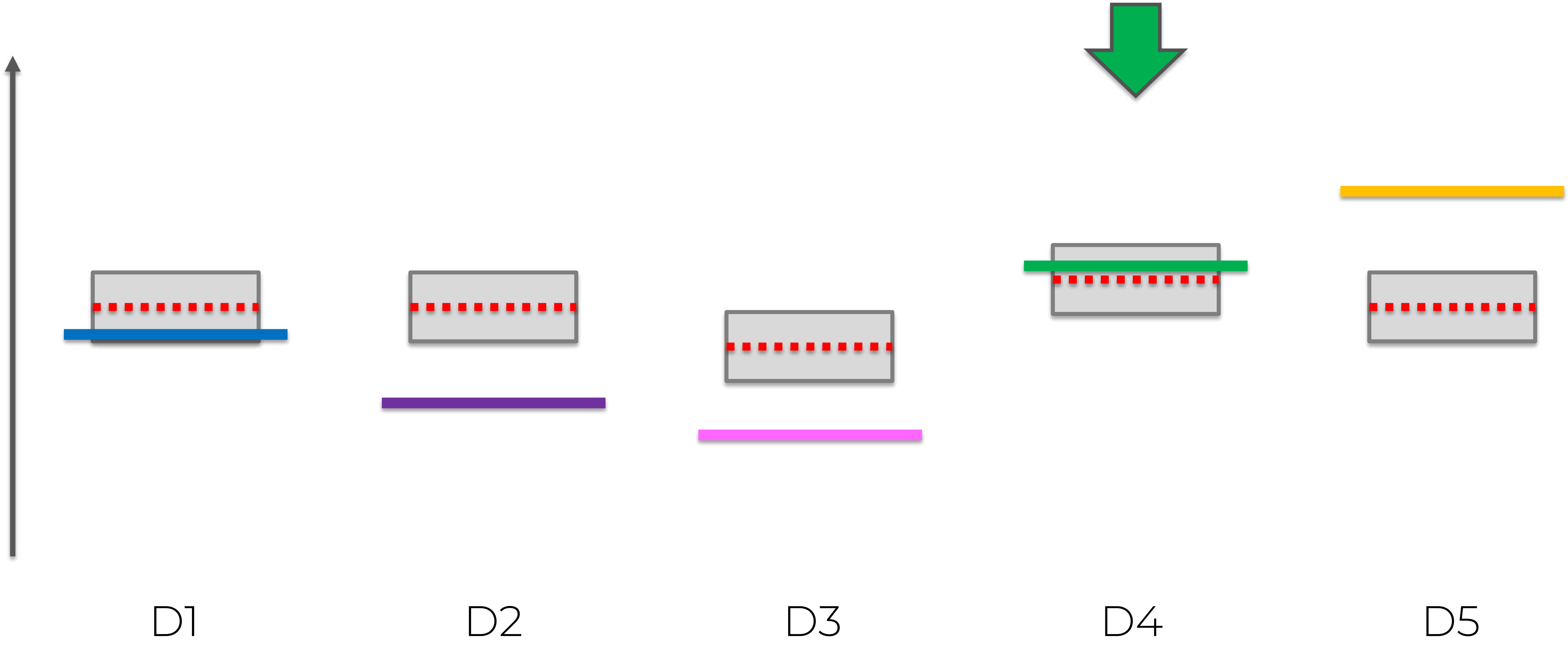
Upper Confidence Bound Algorithm



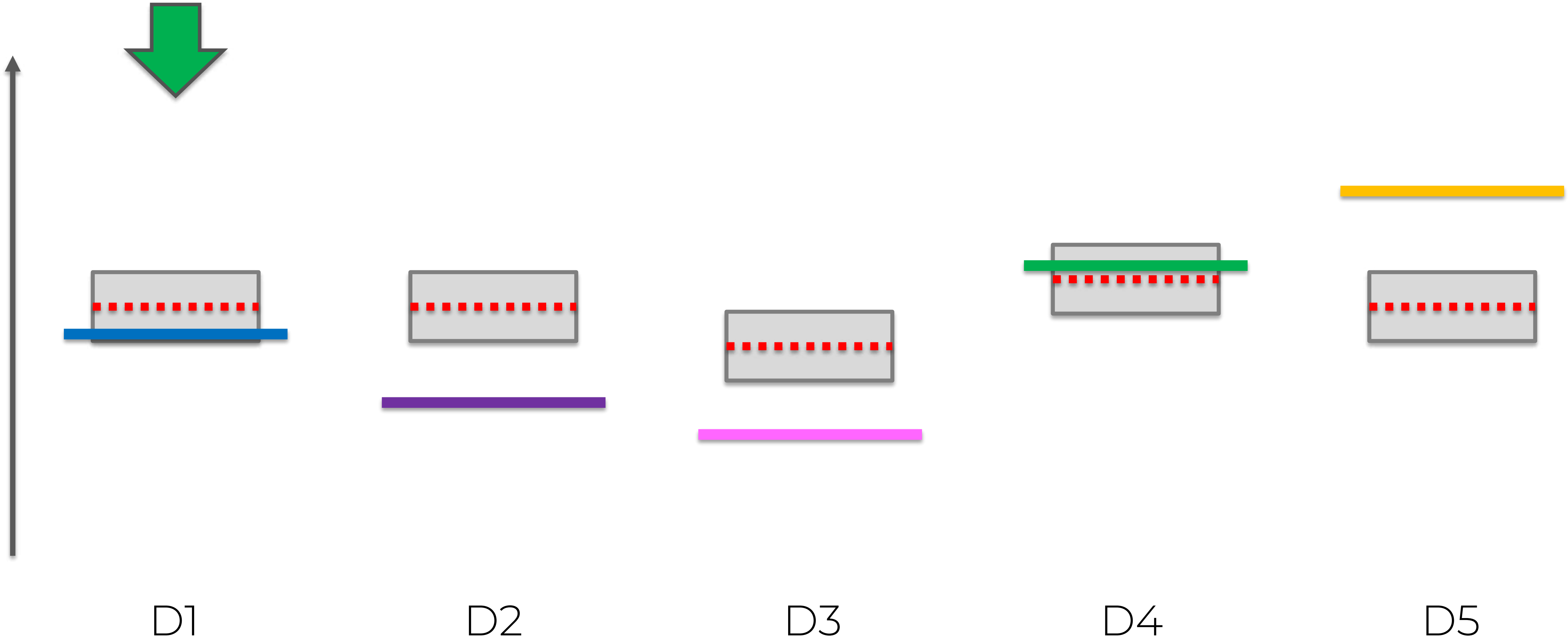
Upper Confidence Bound Algorithm



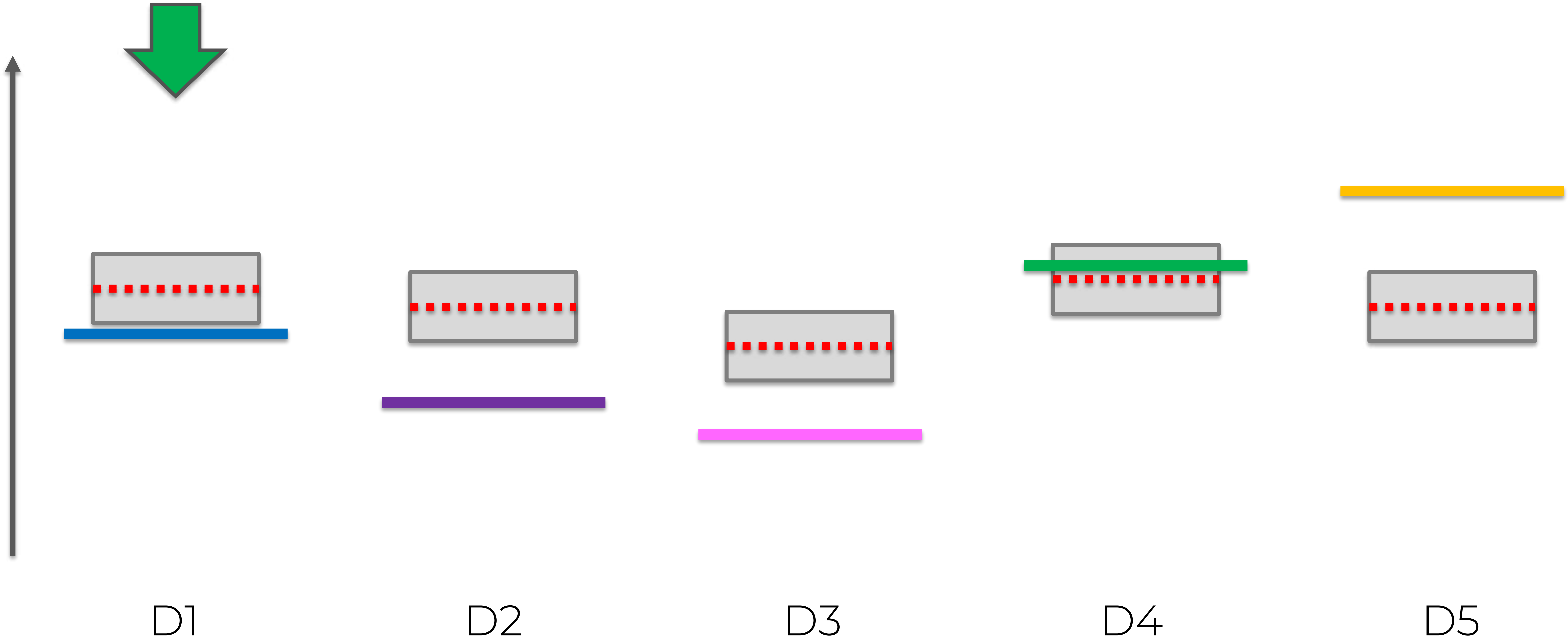
Upper Confidence Bound Algorithm



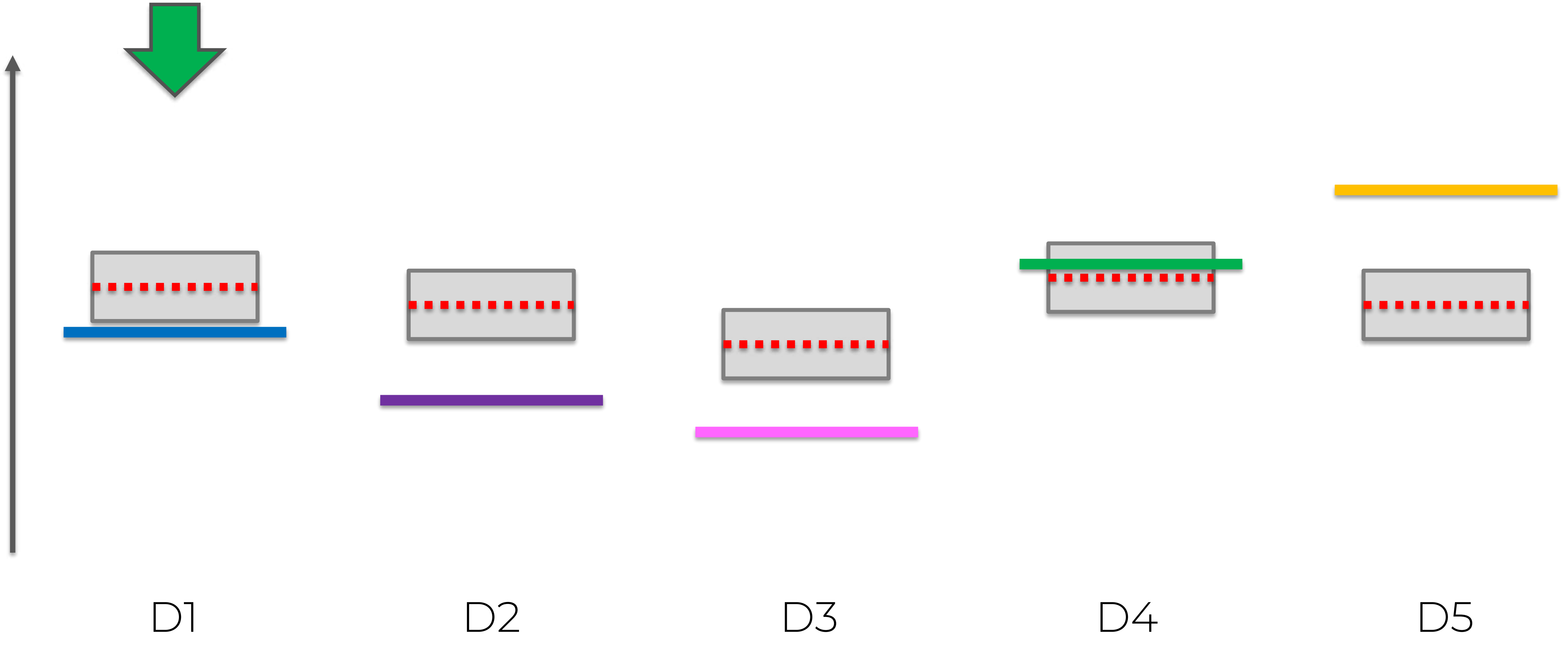
Upper Confidence Bound Algorithm



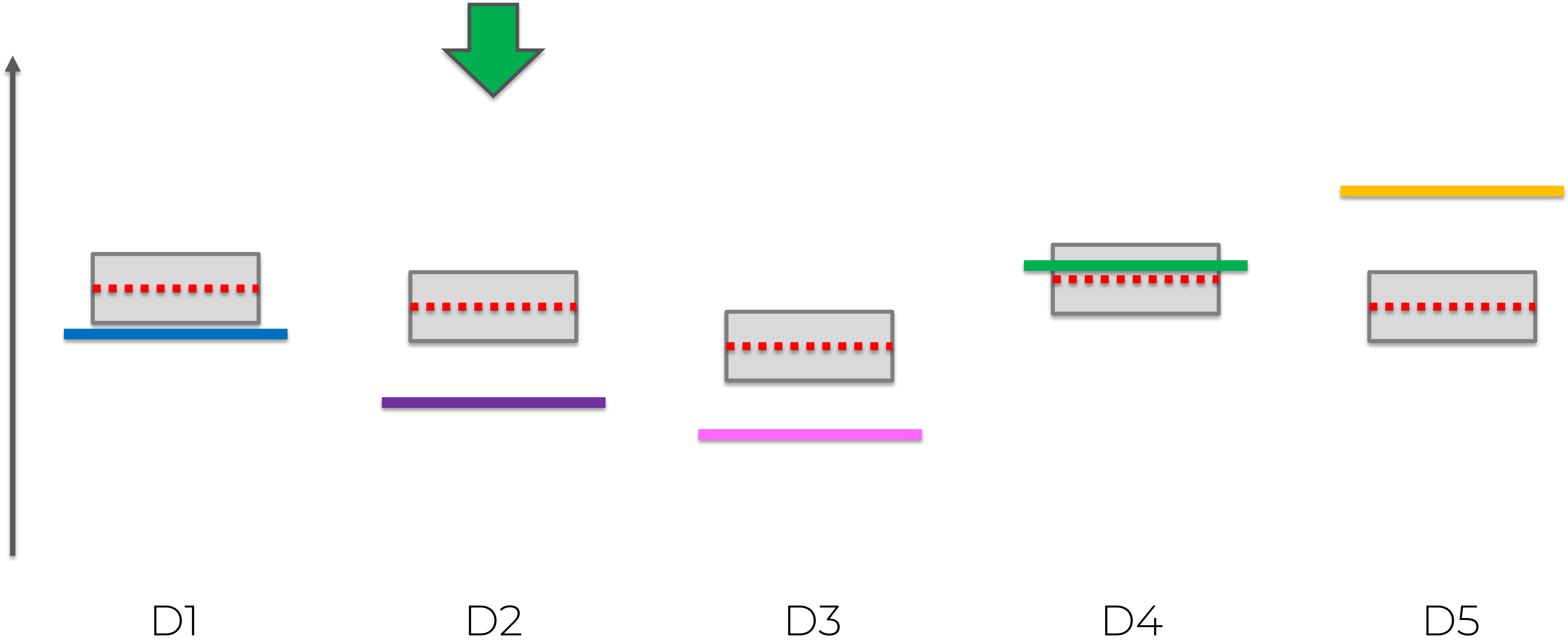
Upper Confidence Bound Algorithm



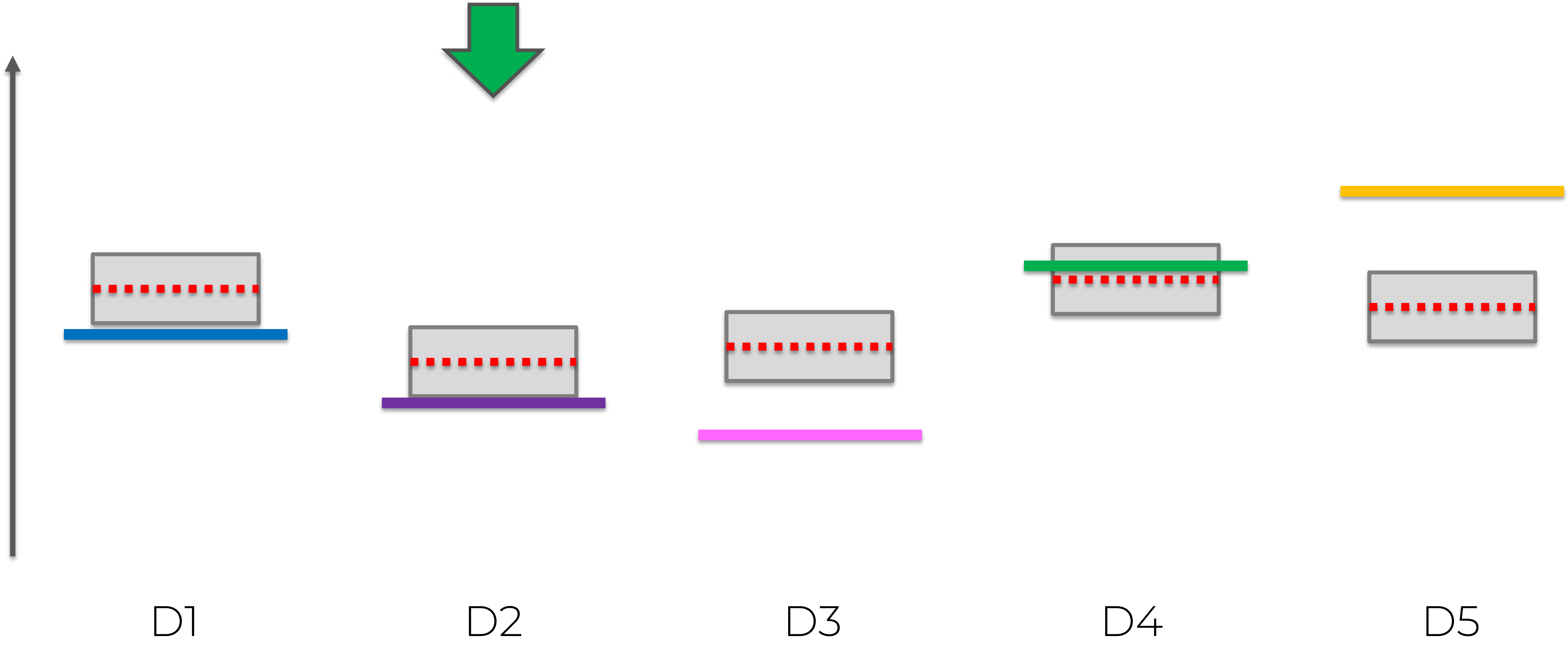
Upper Confidence Bound Algorithm



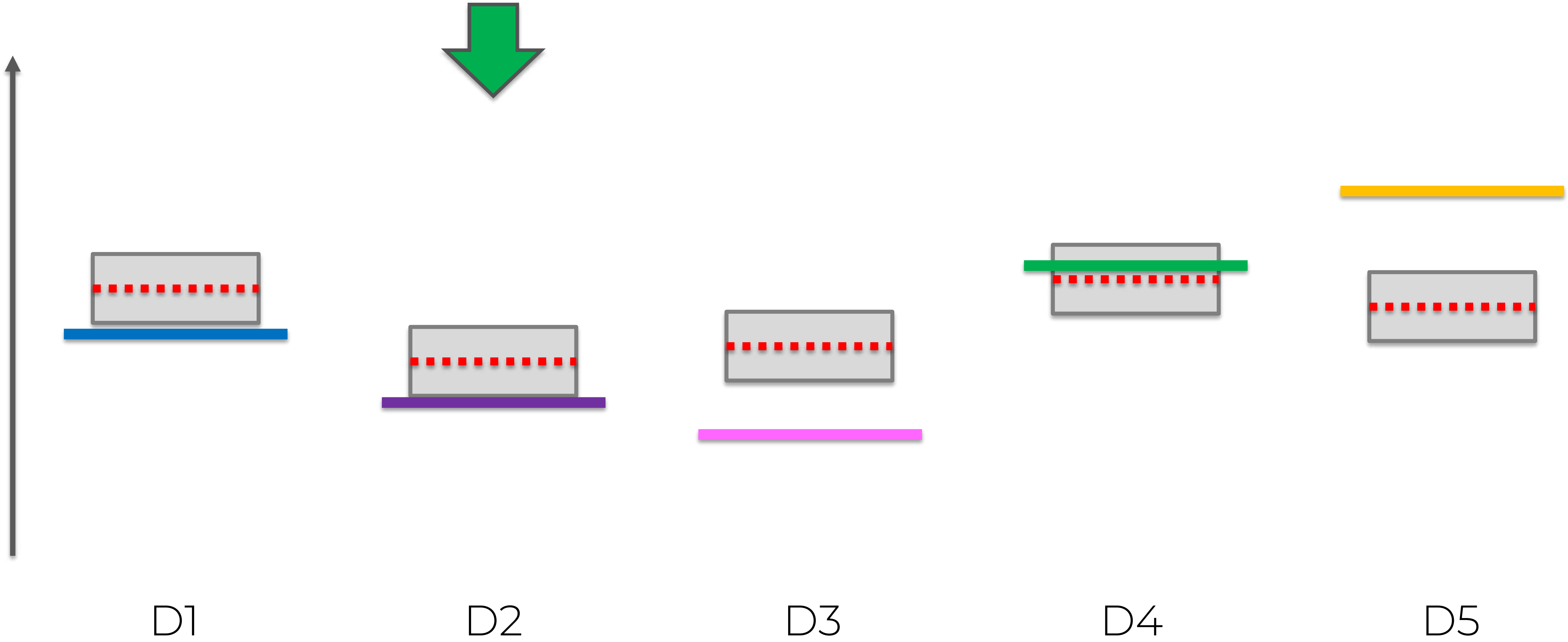
Upper Confidence Bound Algorithm



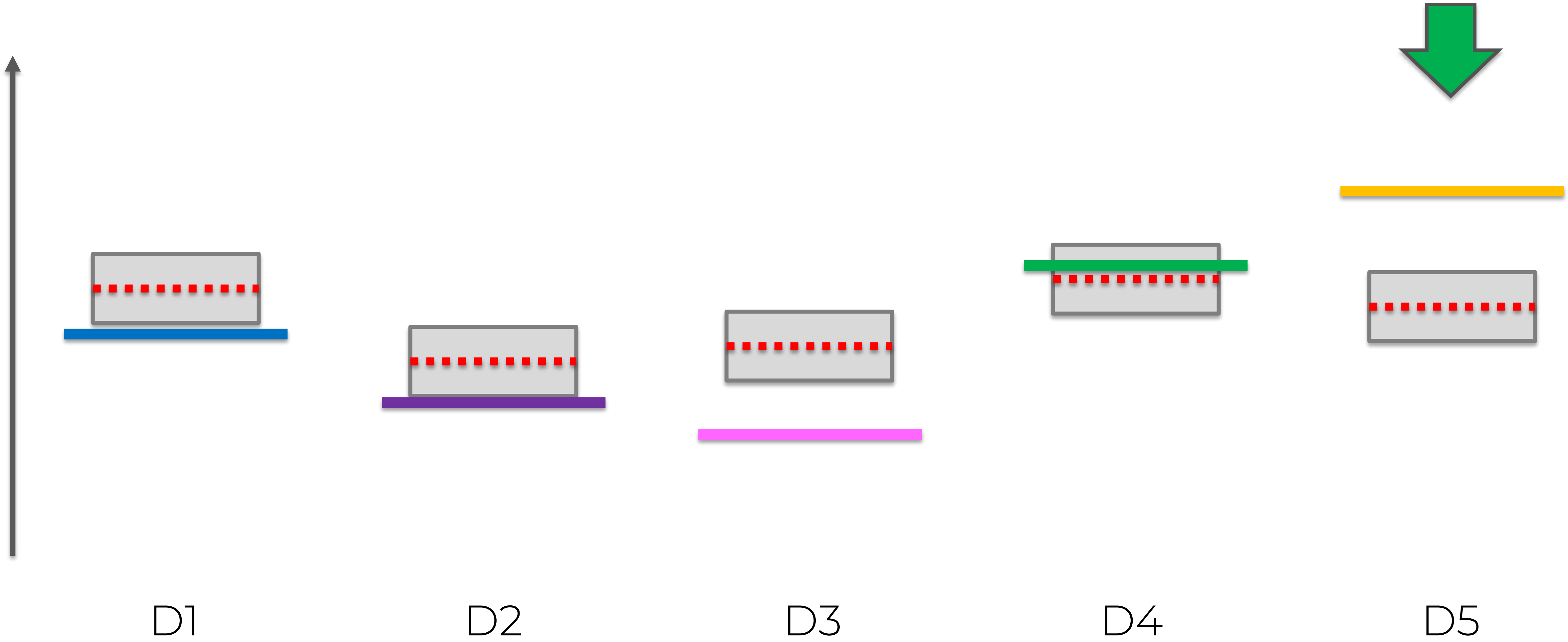
Upper Confidence Bound Algorithm



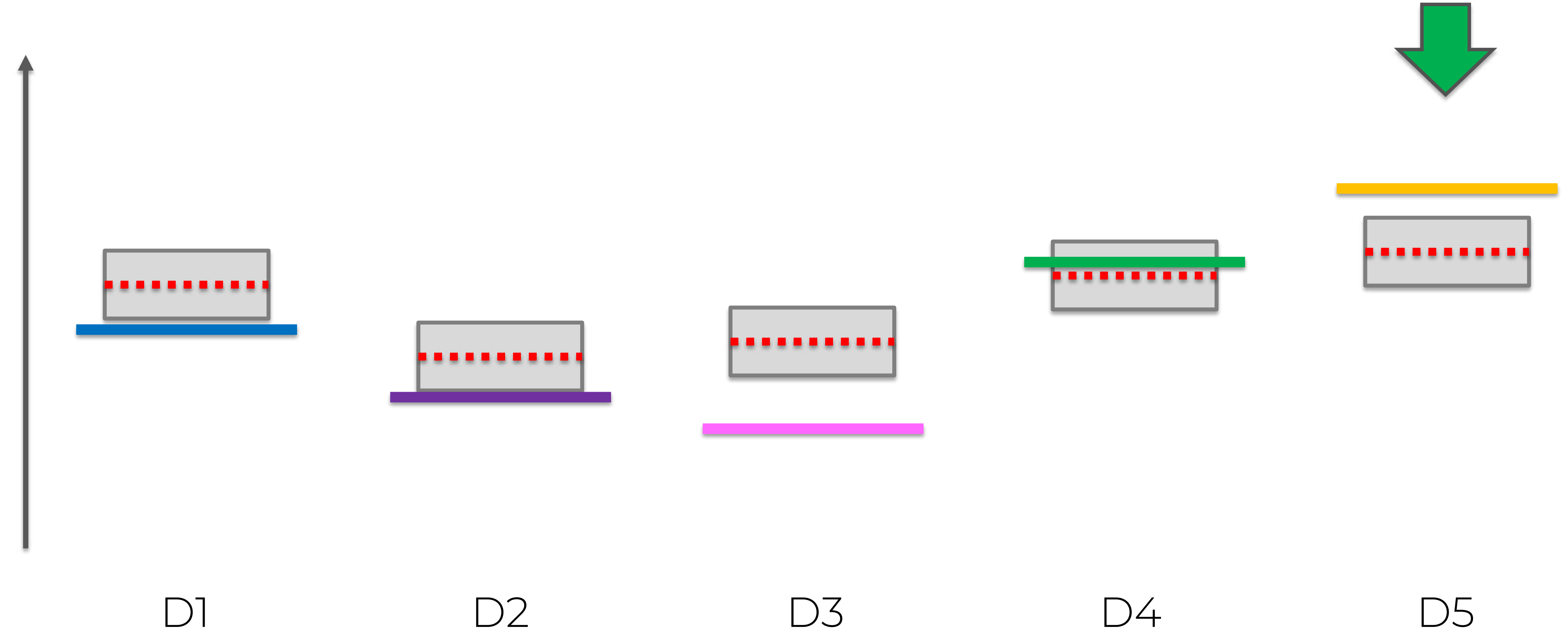
Upper Confidence Bound Algorithm



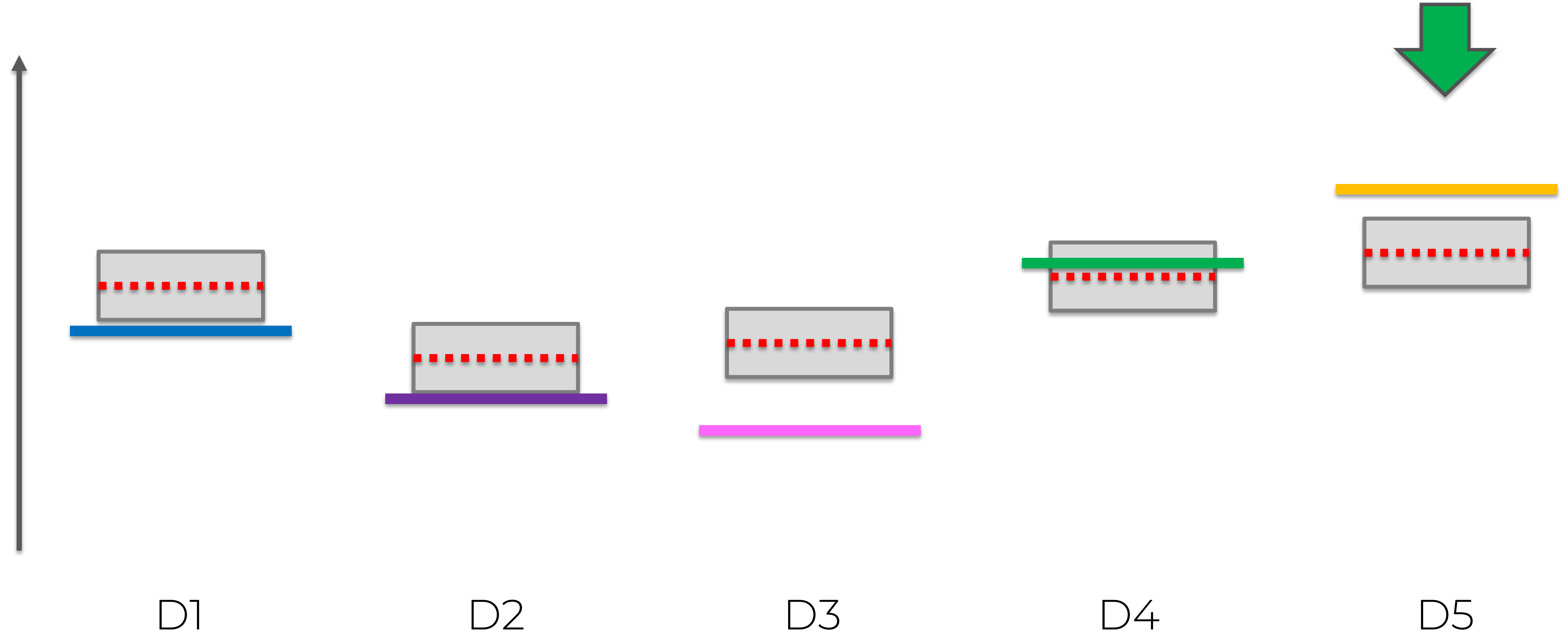
Upper Confidence Bound Algorithm



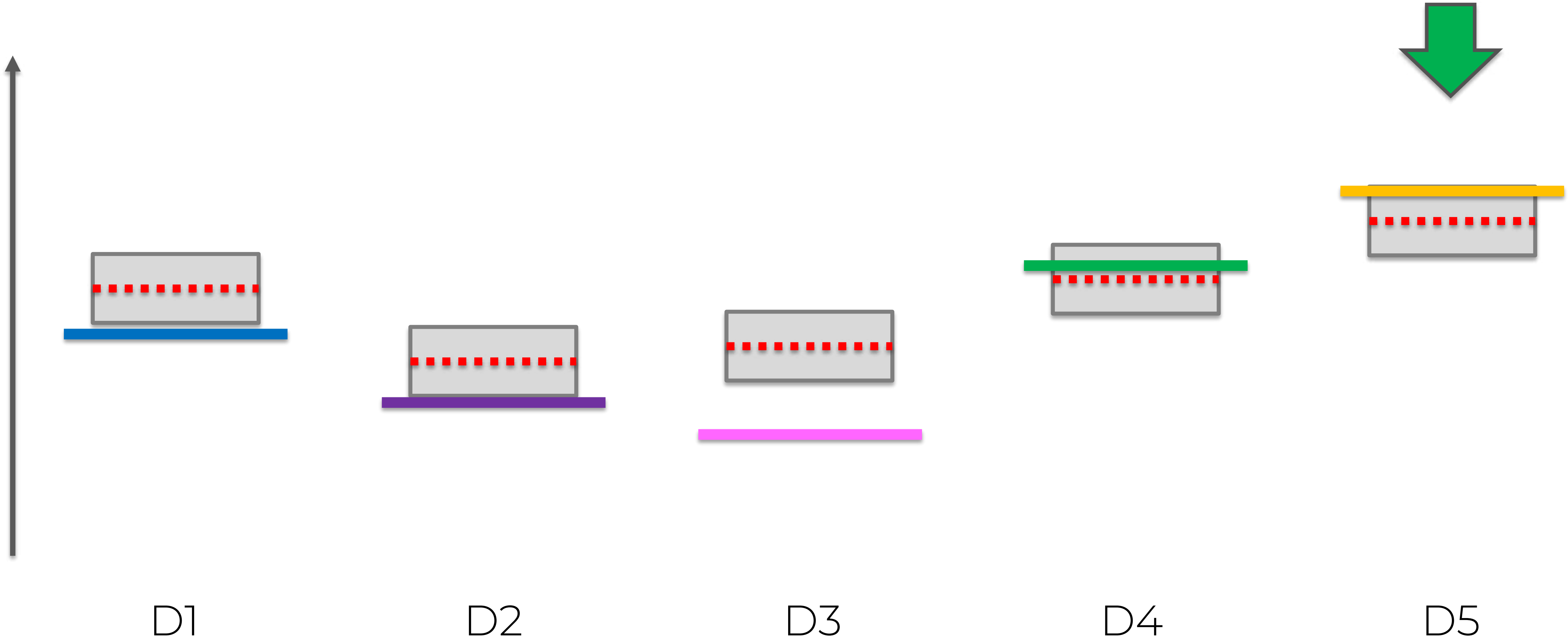
Upper Confidence Bound Algorithm



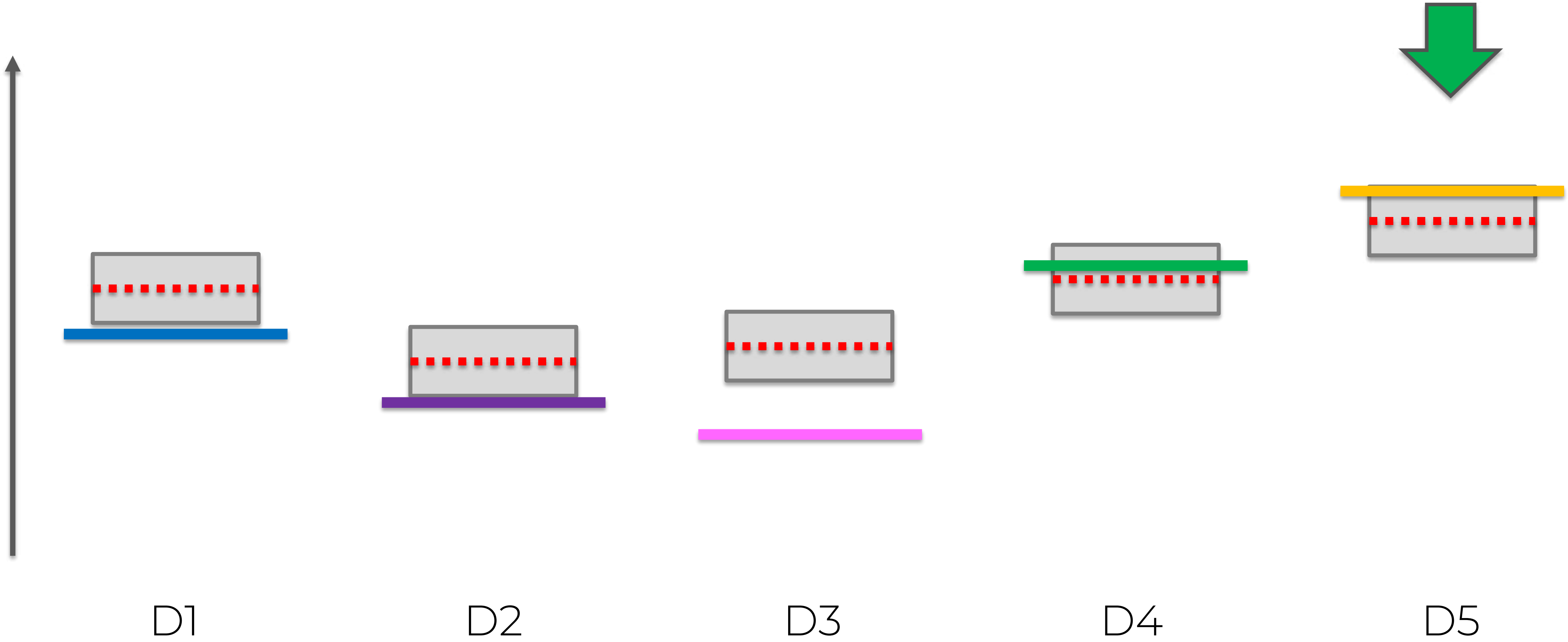
Upper Confidence Bound Algorithm



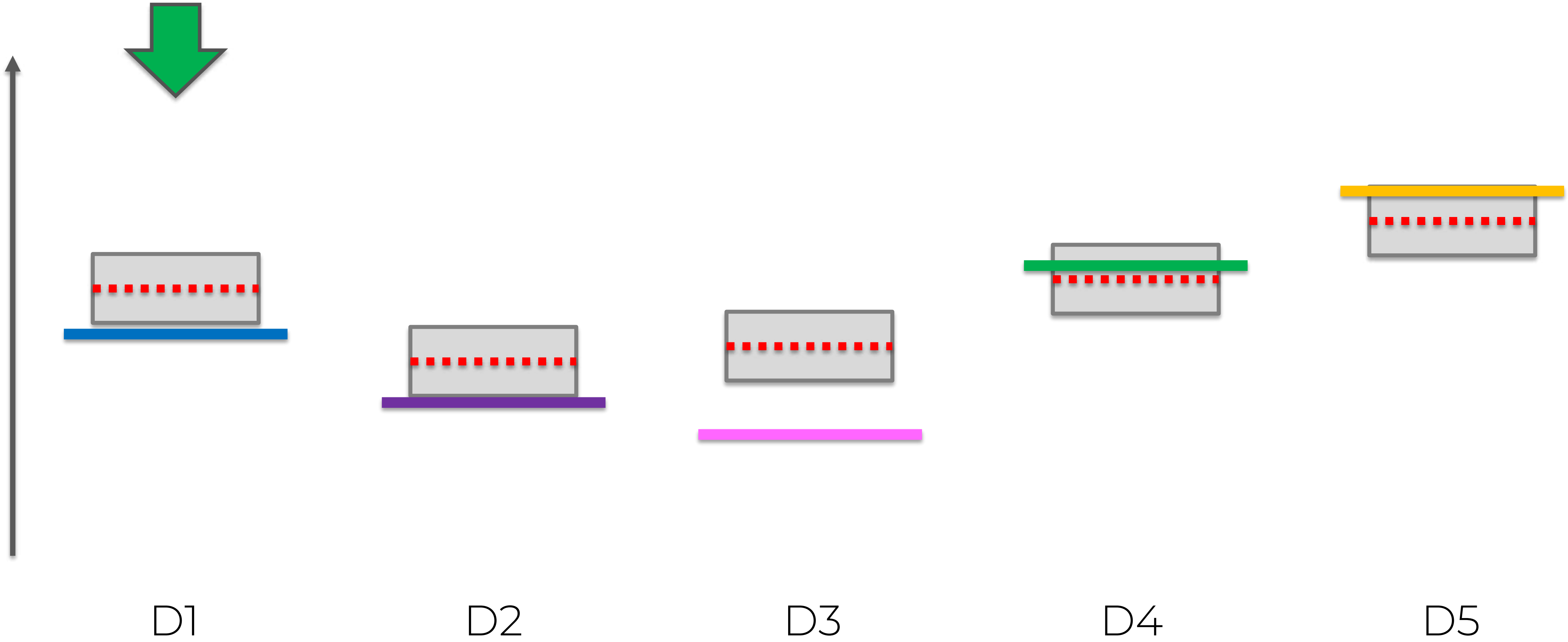
Upper Confidence Bound Algorithm



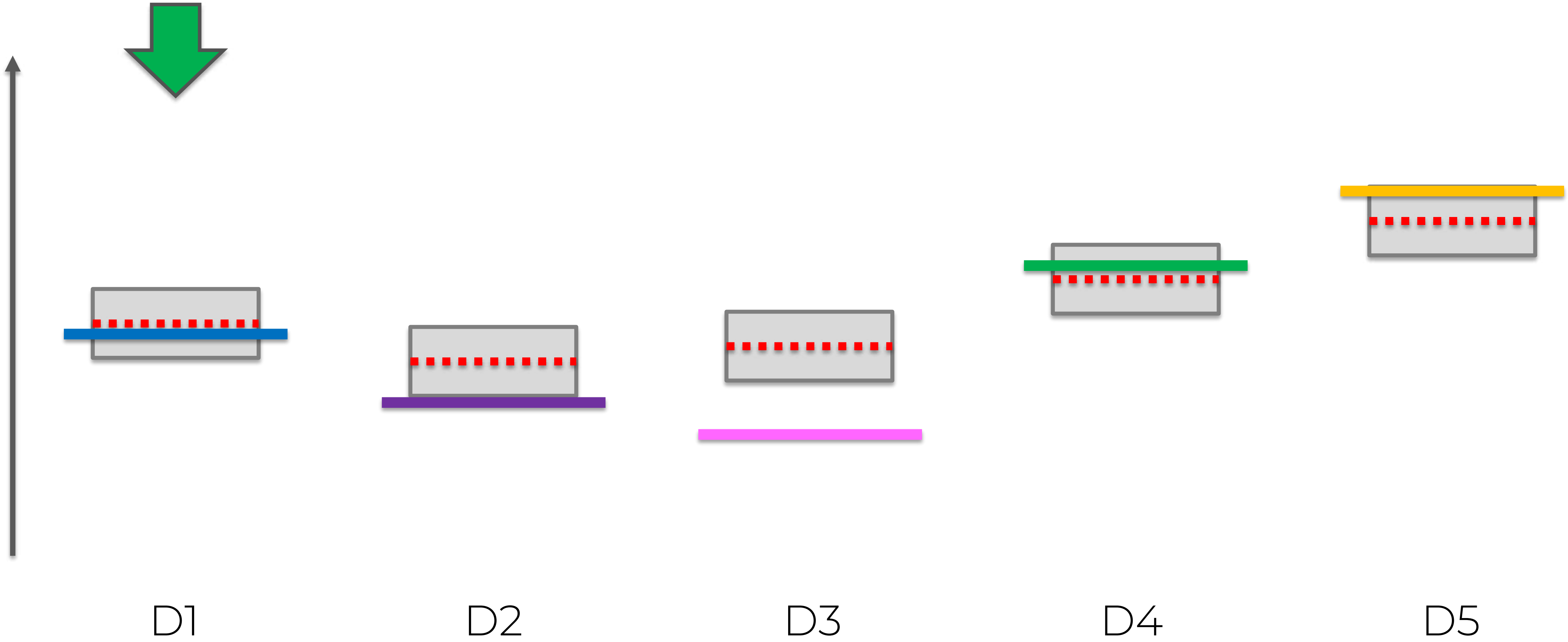
Upper Confidence Bound Algorithm



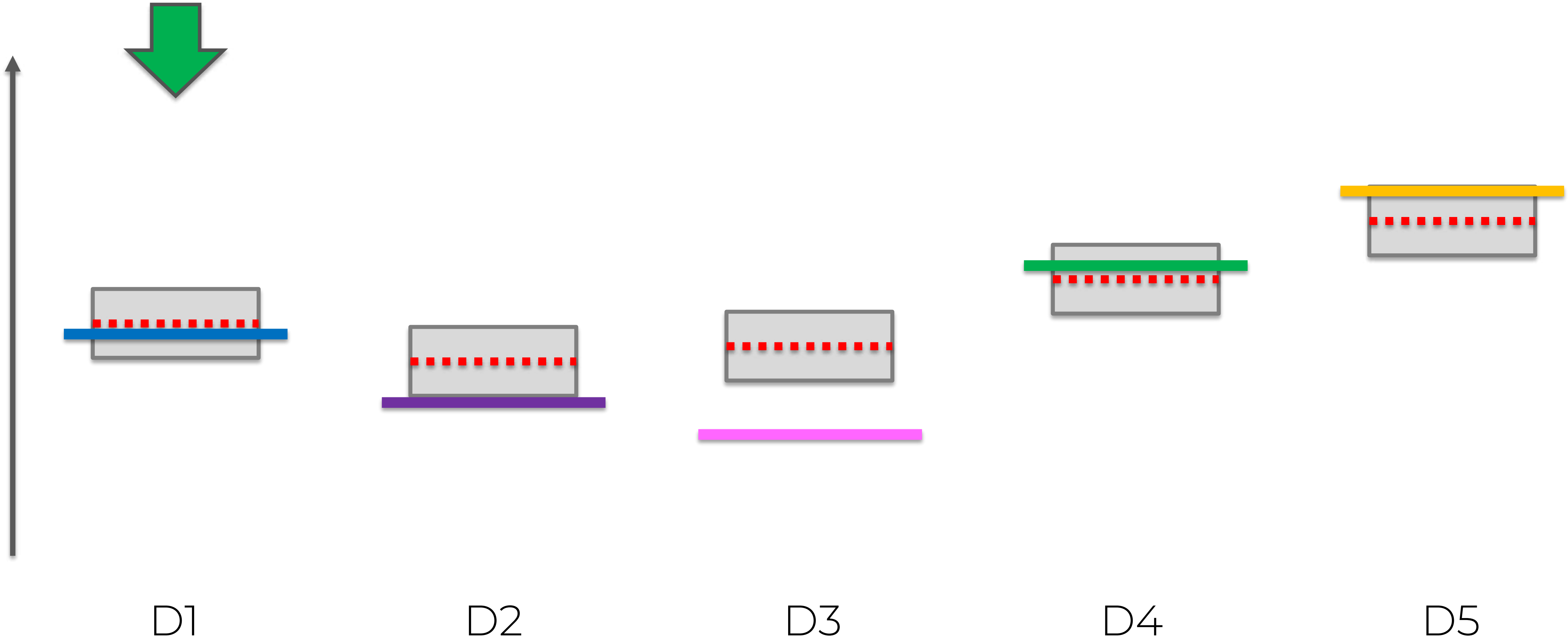
Upper Confidence Bound Algorithm



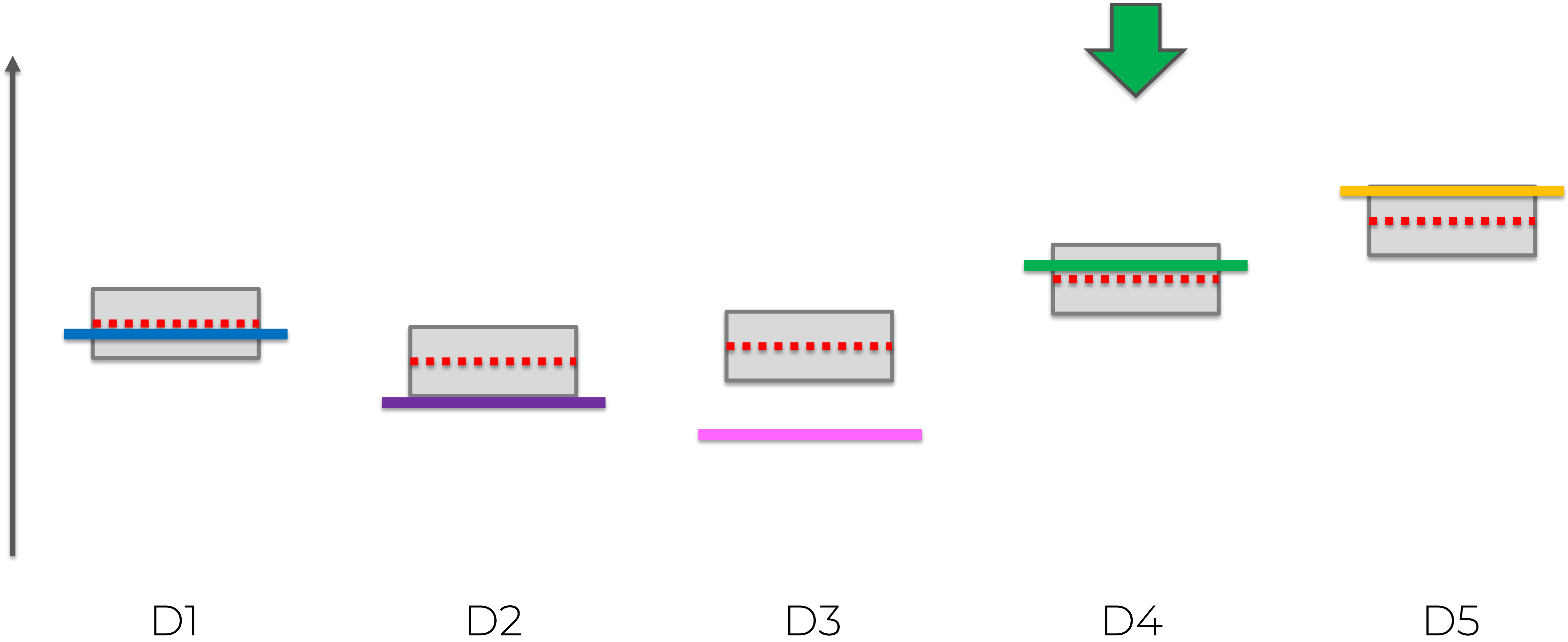
Upper Confidence Bound Algorithm



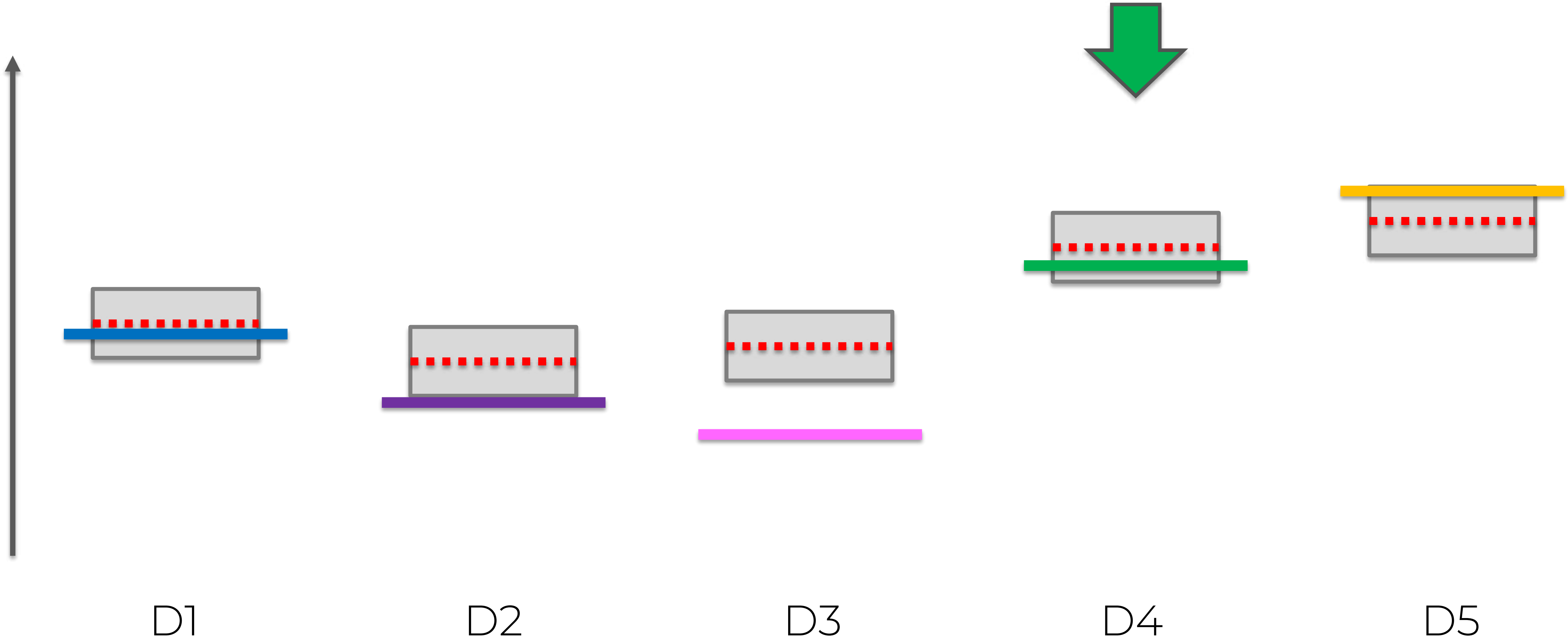
Upper Confidence Bound Algorithm



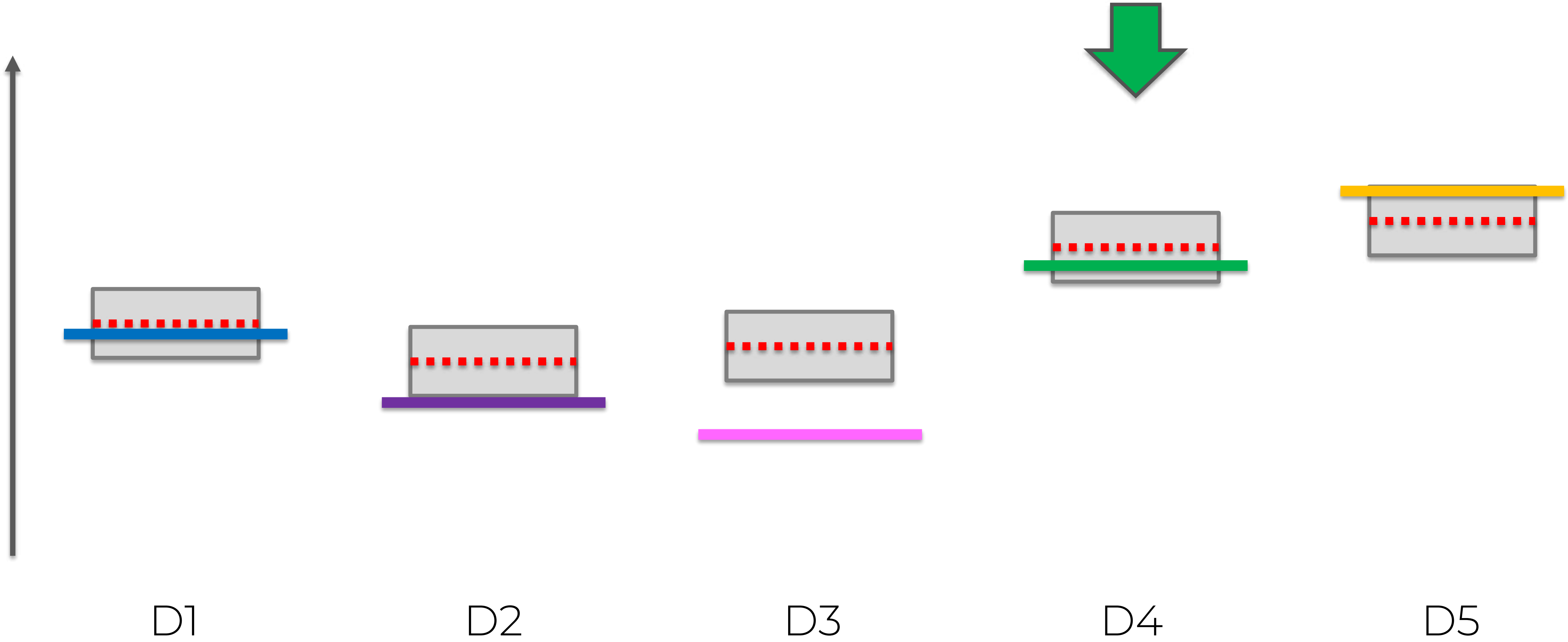
Upper Confidence Bound Algorithm



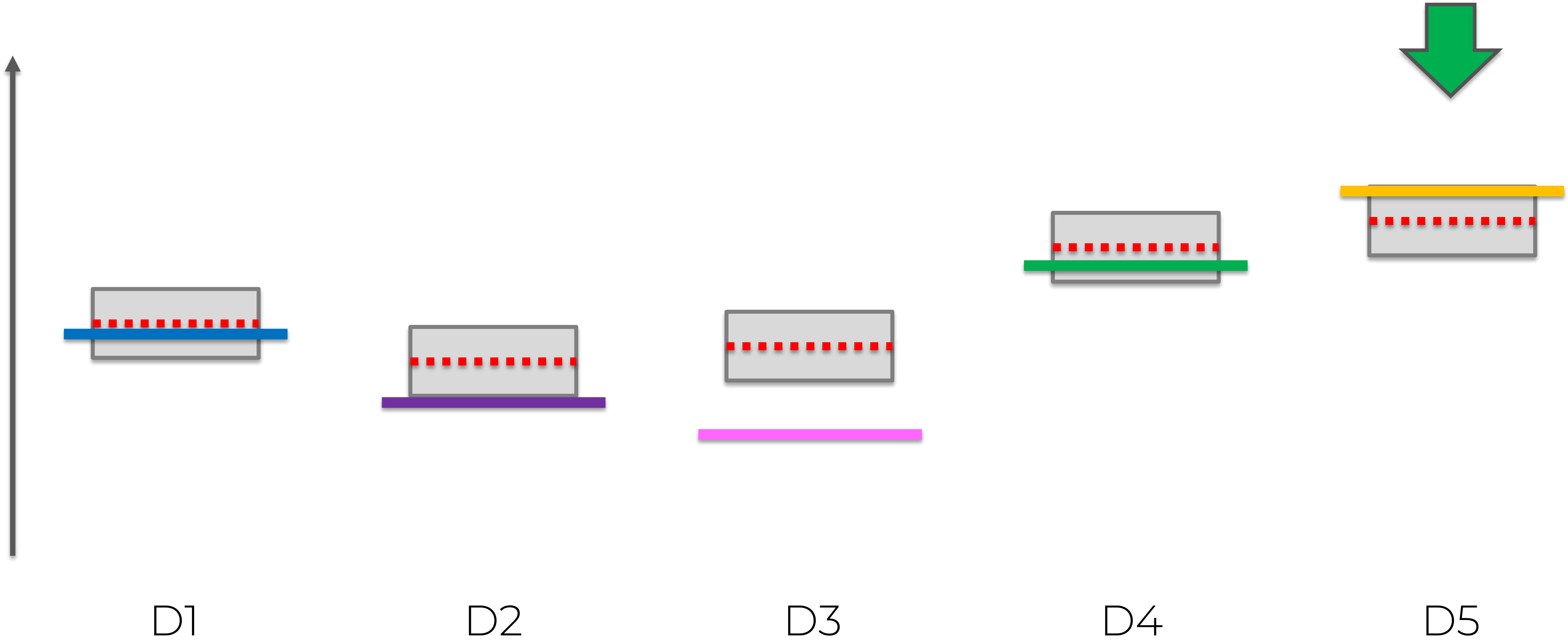
Upper Confidence Bound Algorithm



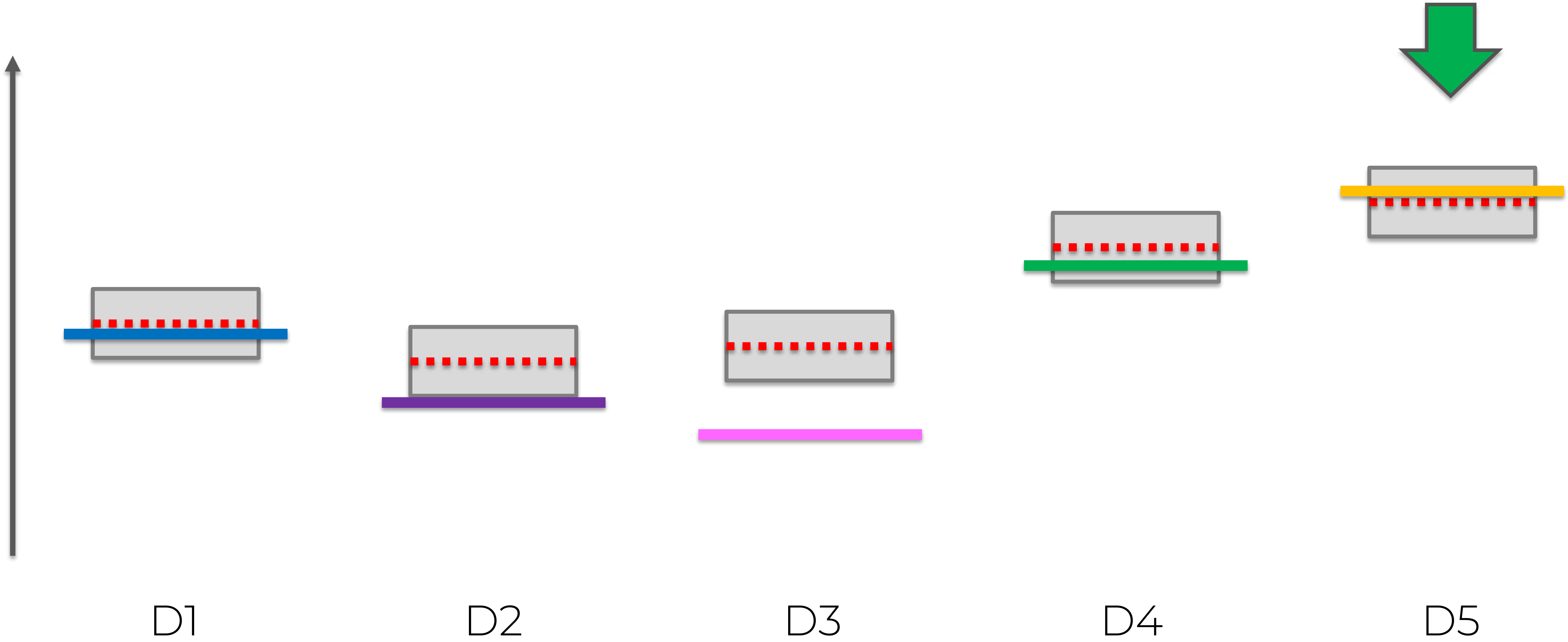
Upper Confidence Bound Algorithm



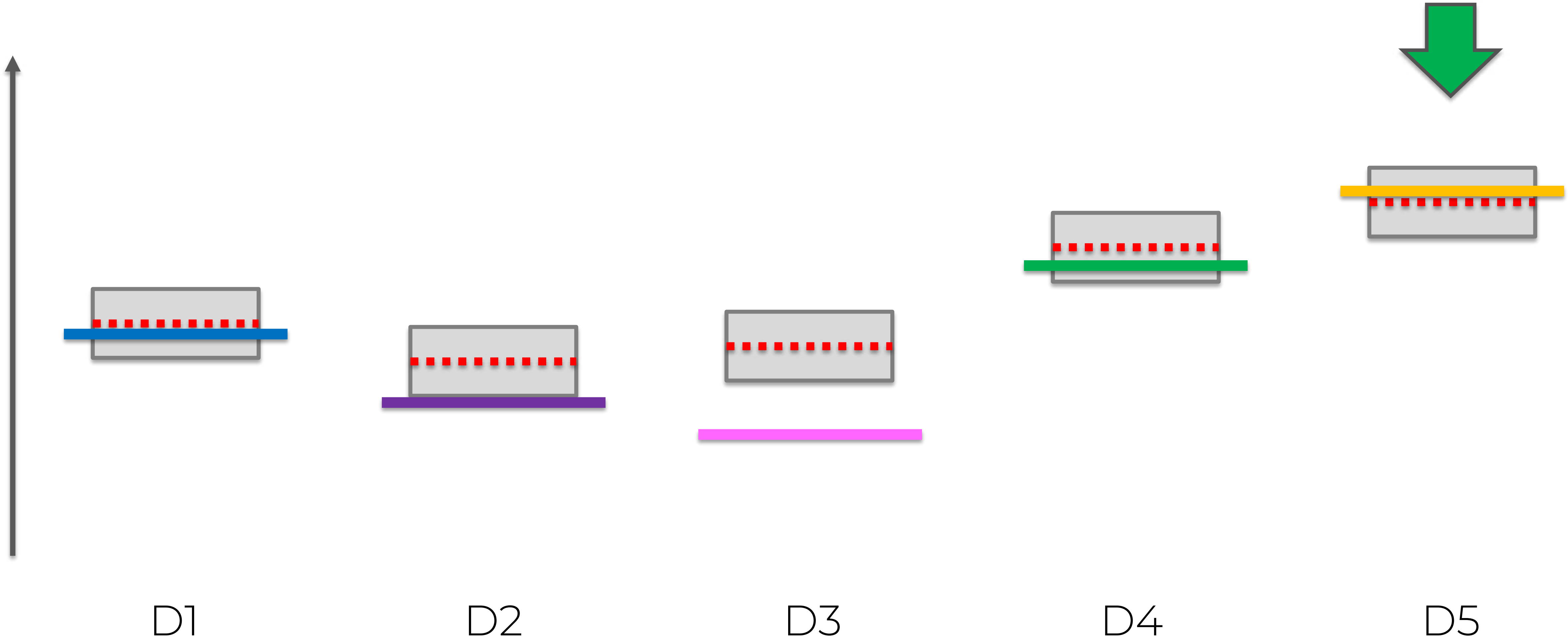
Upper Confidence Bound Algorithm



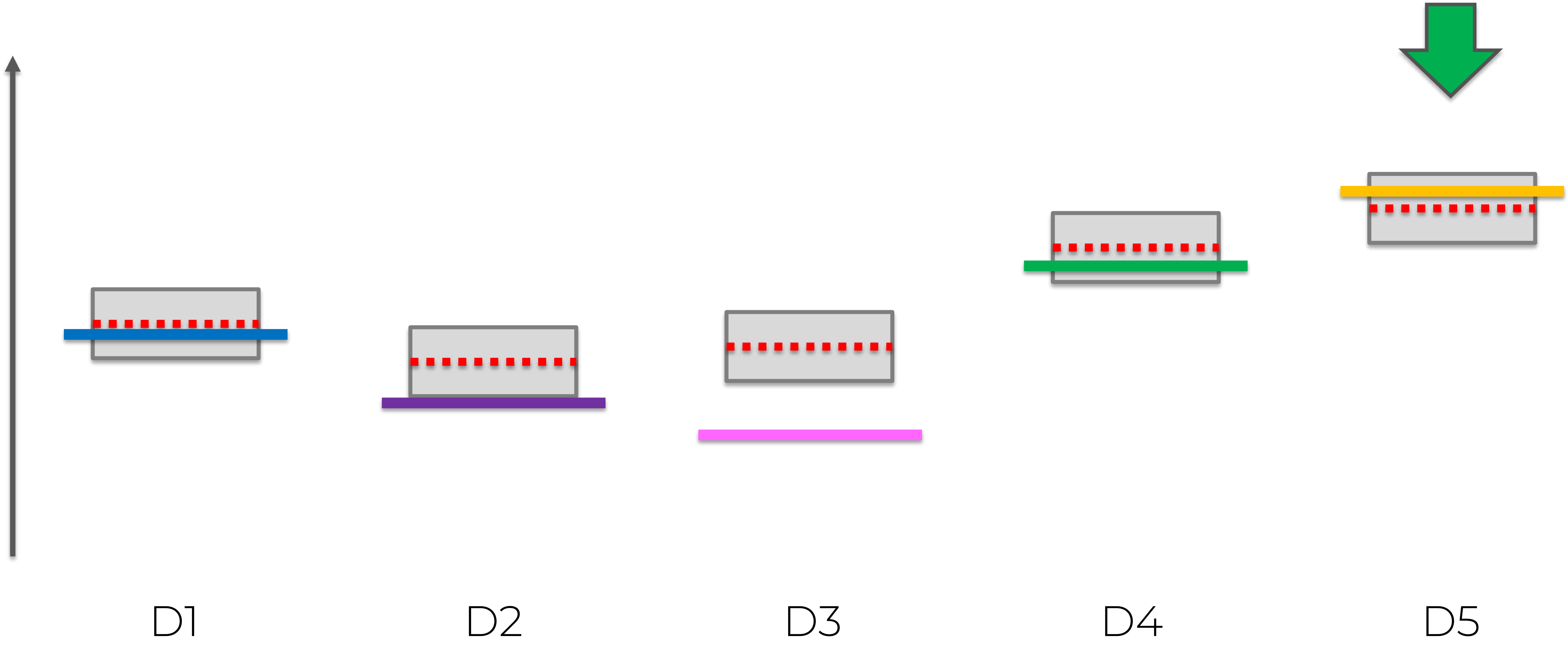
Upper Confidence Bound Algorithm



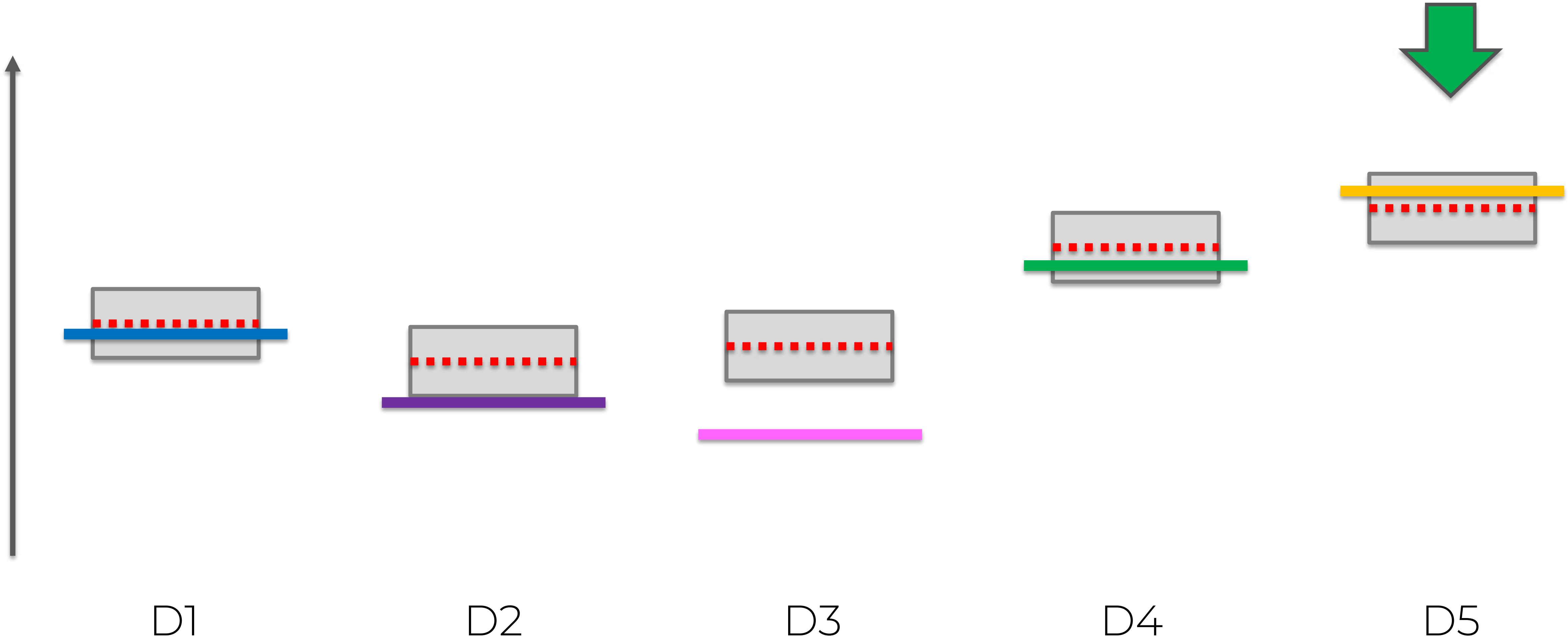
Upper Confidence Bound Algorithm



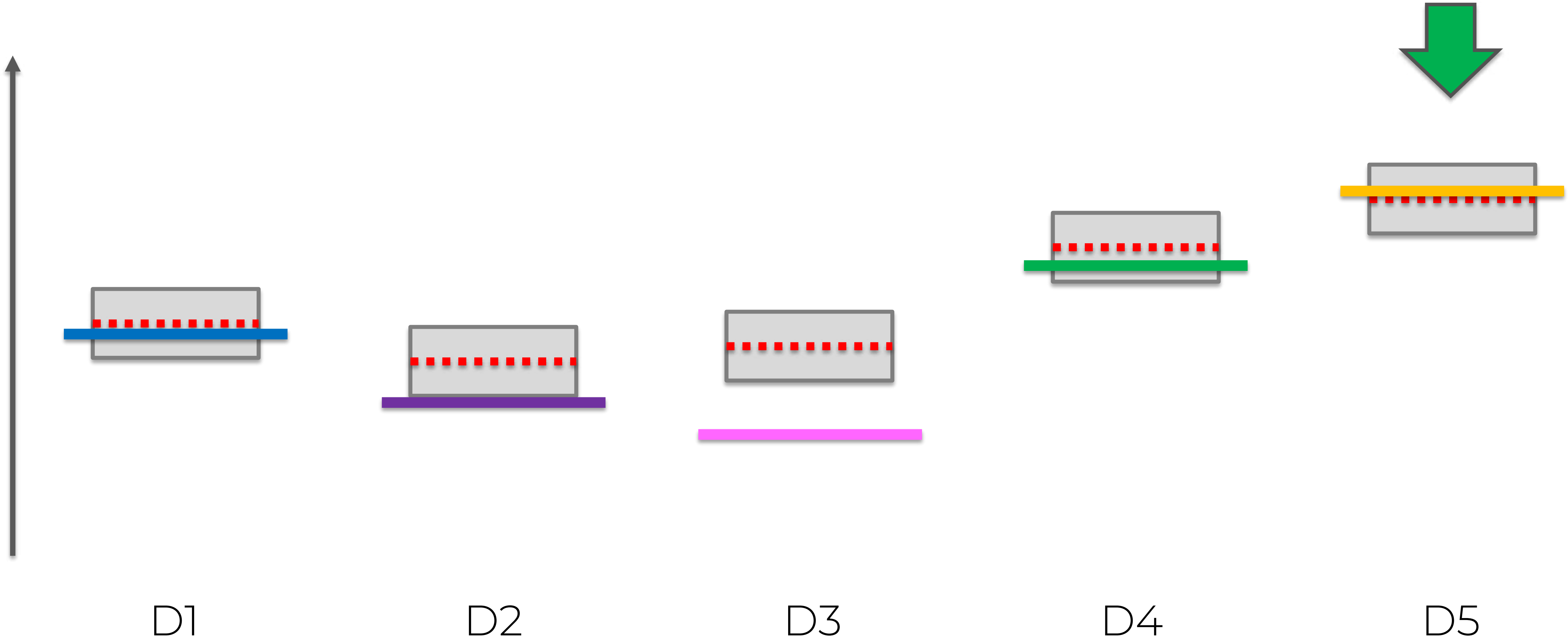
Upper Confidence Bound Algorithm



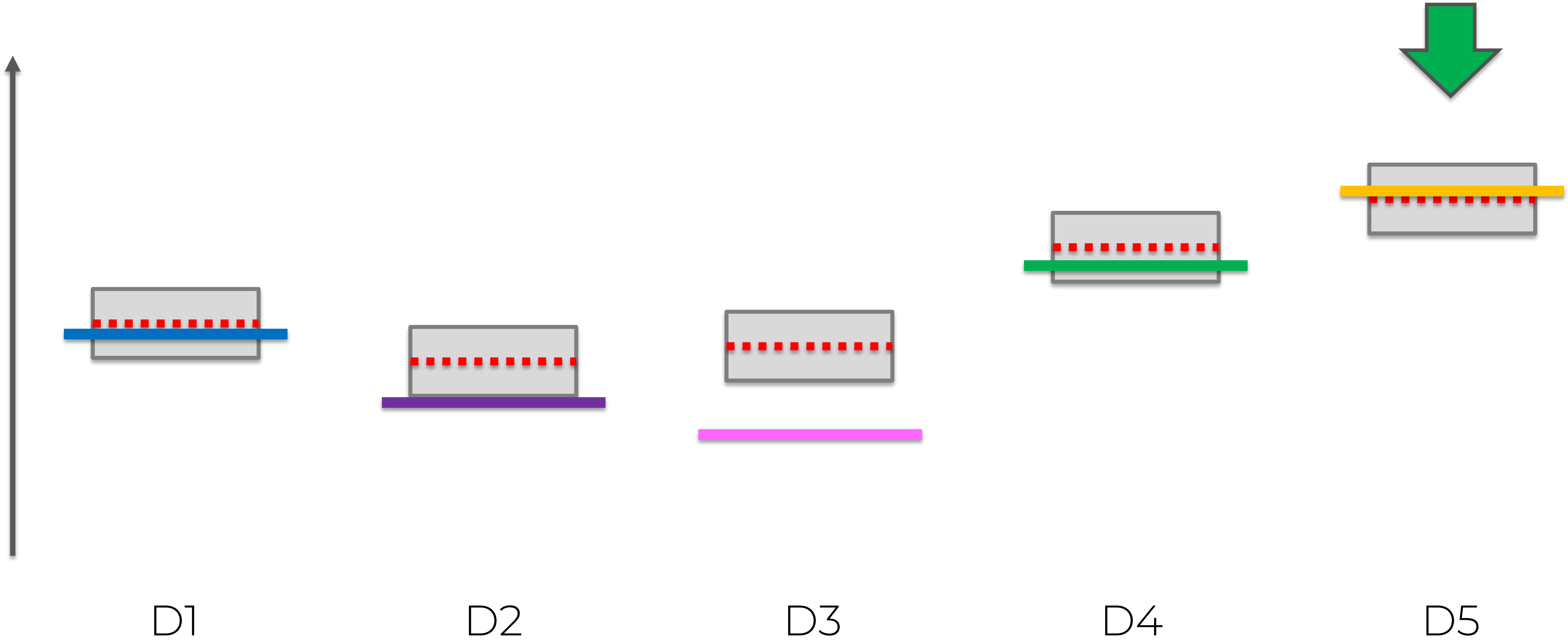
Upper Confidence Bound Algorithm



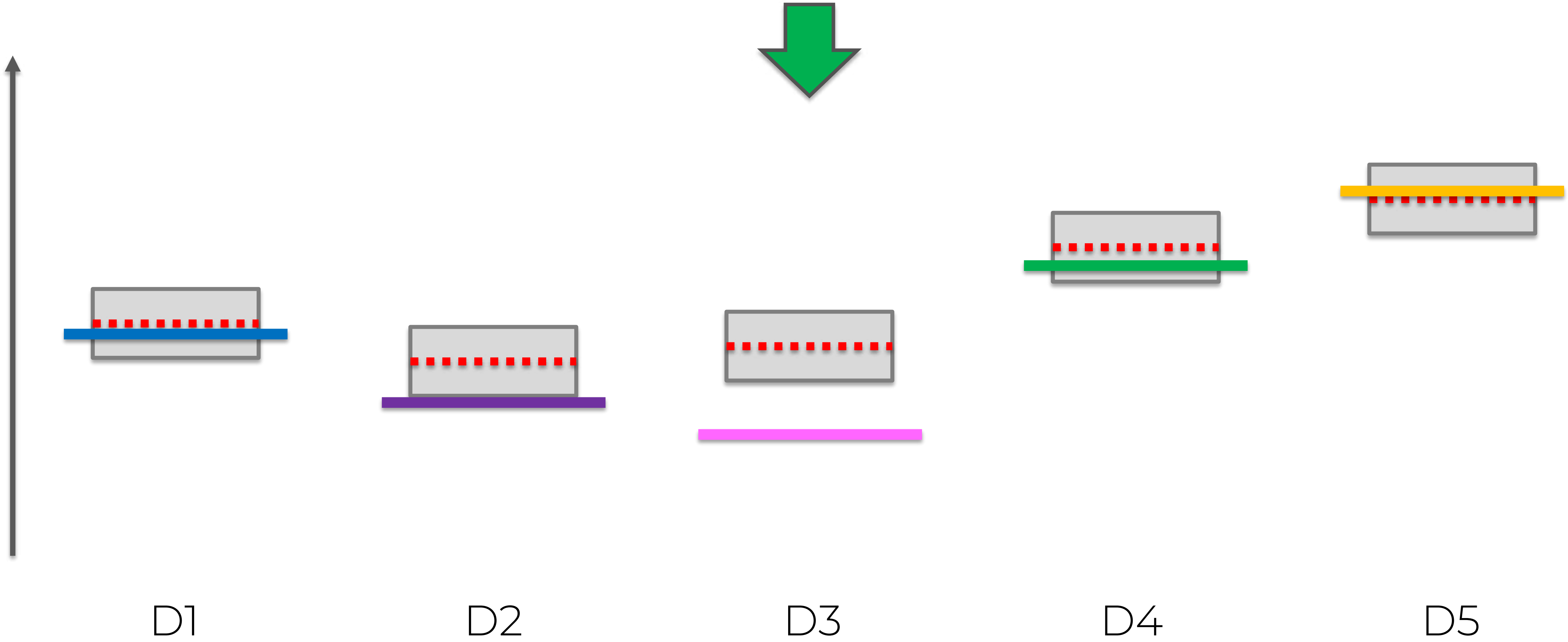
Upper Confidence Bound Algorithm



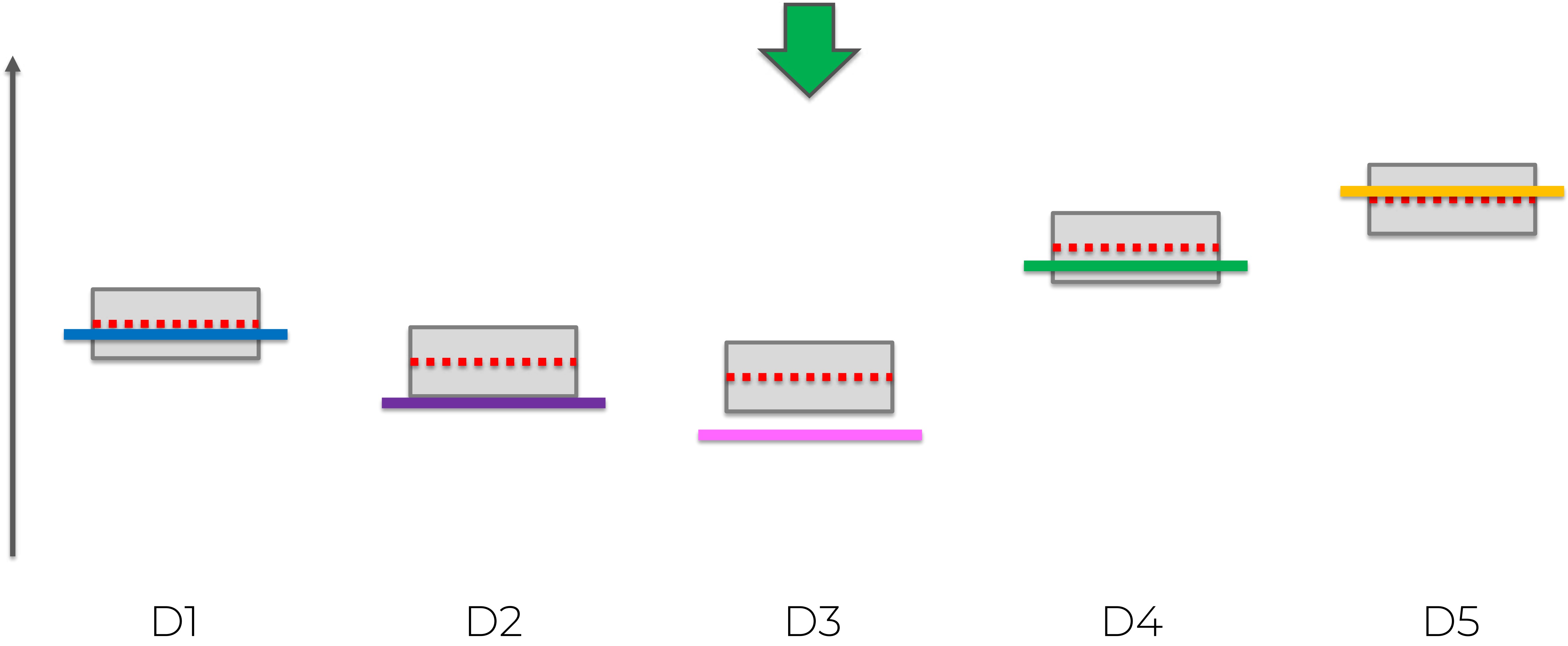
Upper Confidence Bound Algorithm



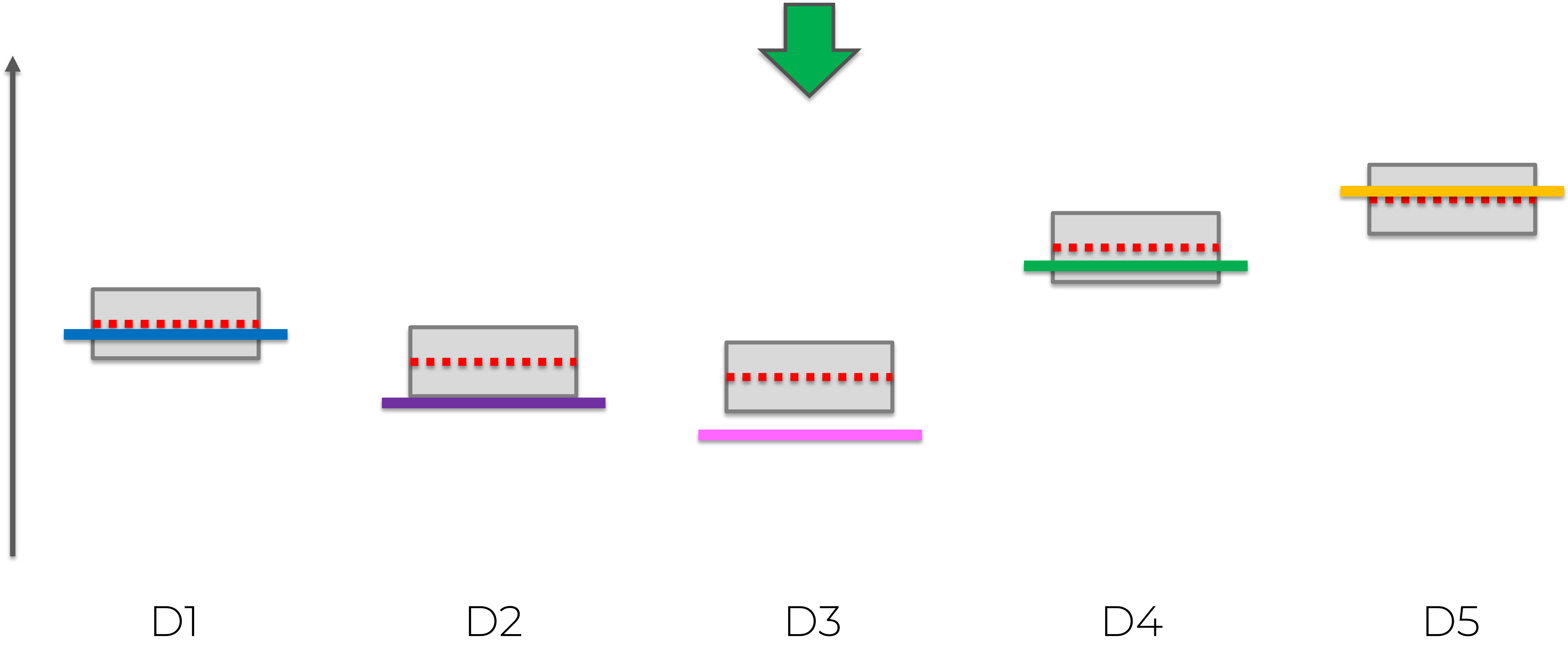
Upper Confidence Bound Algorithm



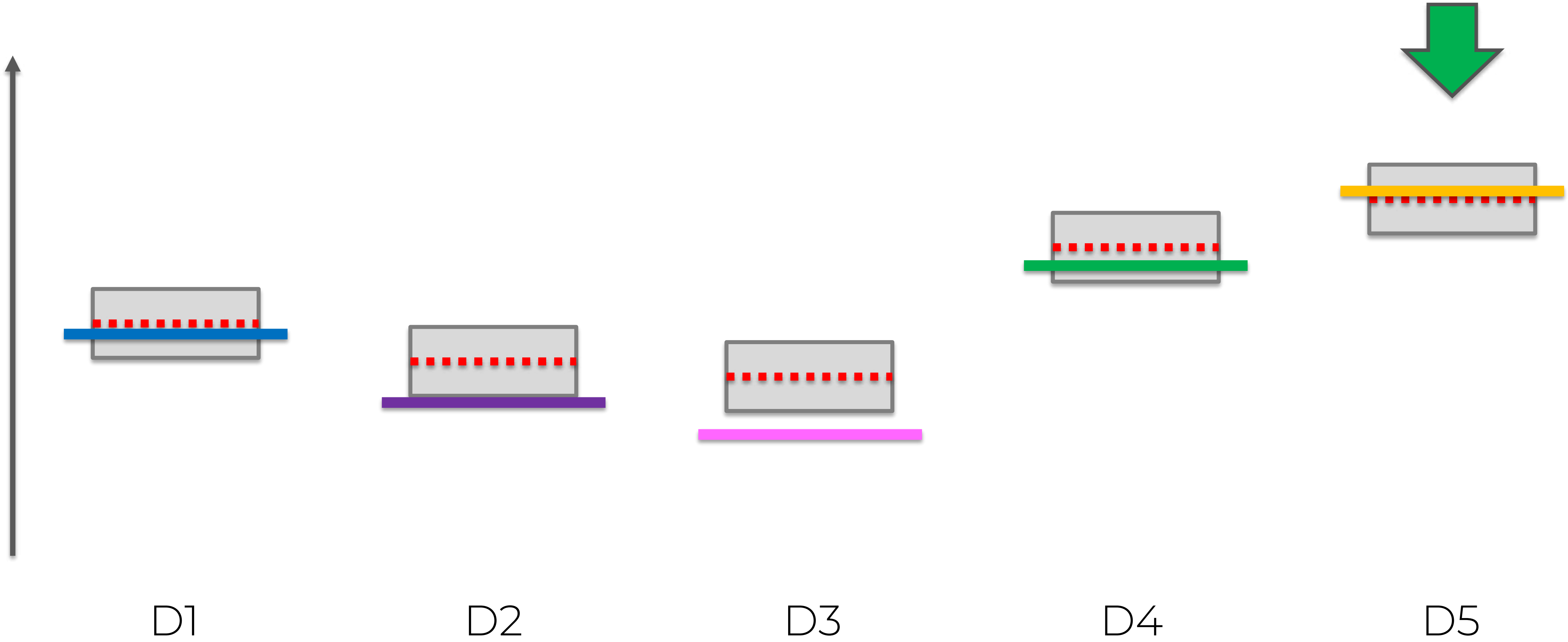
Upper Confidence Bound Algorithm



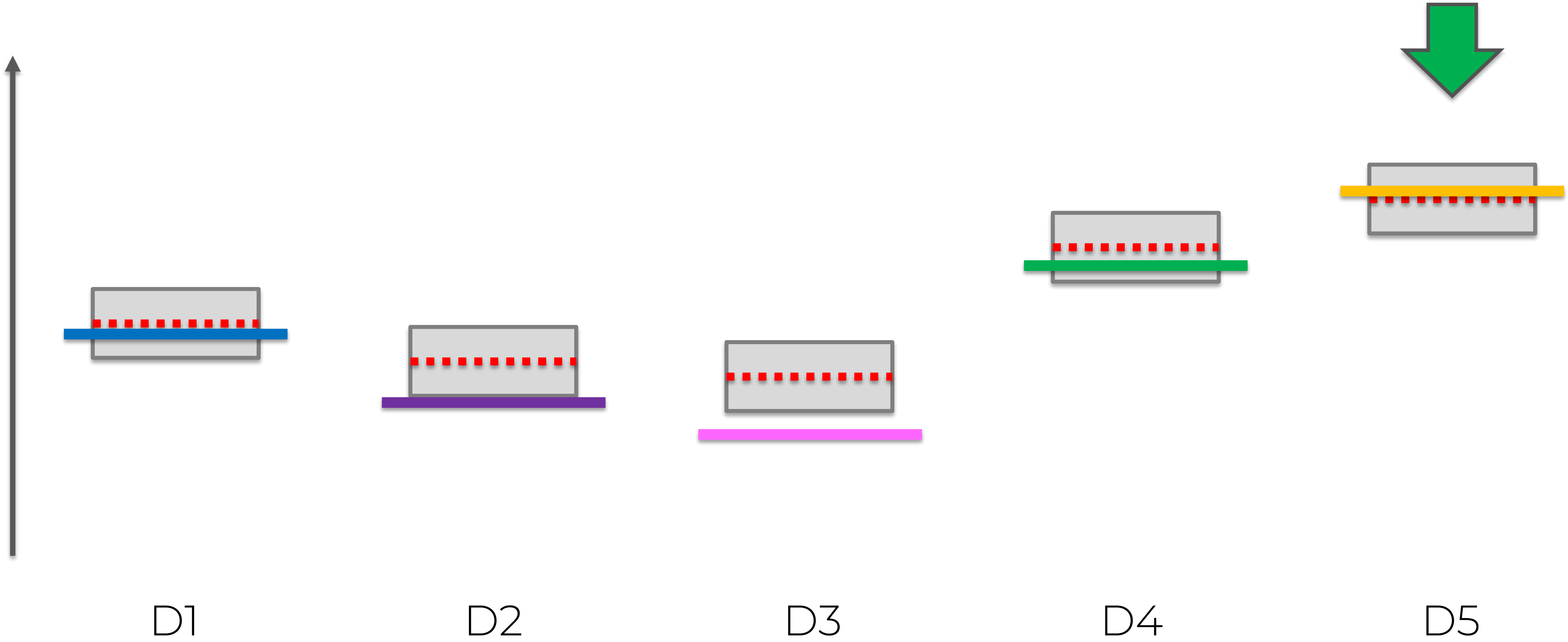
Upper Confidence Bound Algorithm



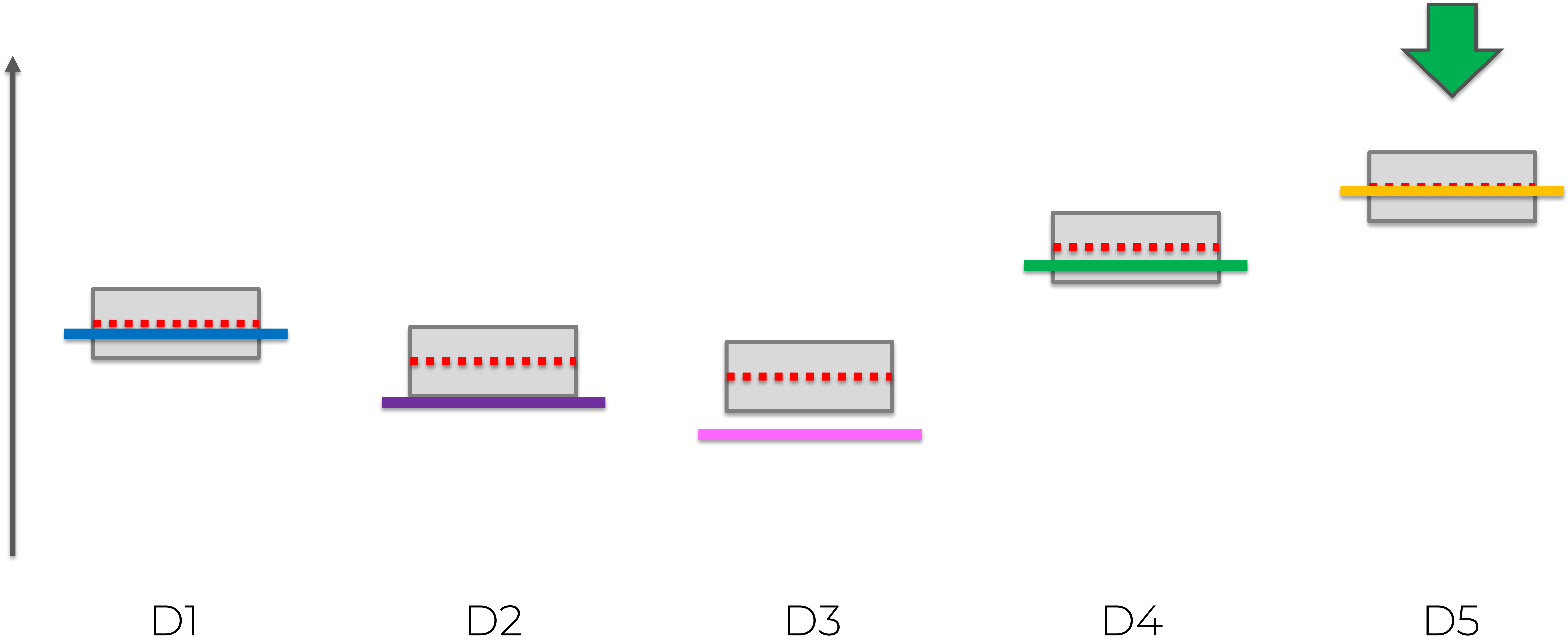
Upper Confidence Bound Algorithm



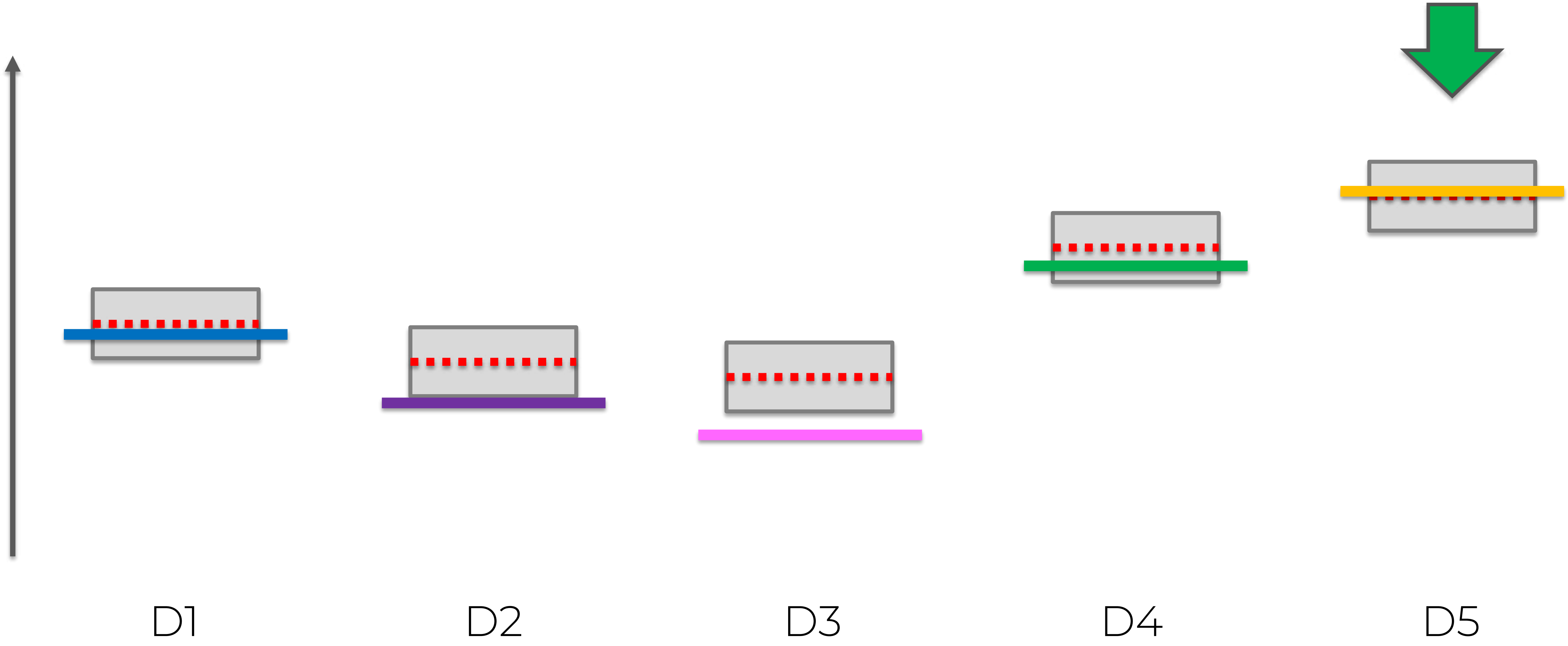
Upper Confidence Bound Algorithm



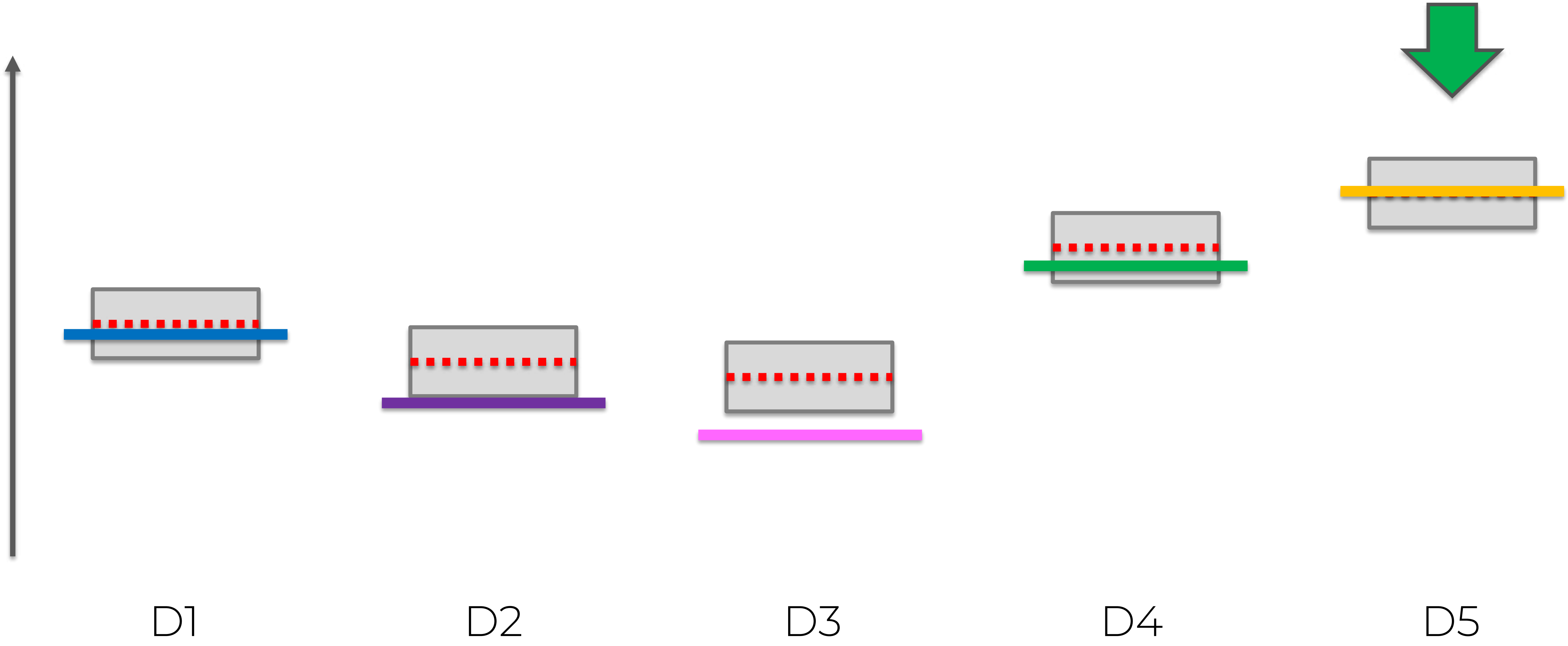
Upper Confidence Bound Algorithm



Upper Confidence Bound Algorithm



Upper Confidence Bound Algorithm



Thompson Sampling Algorithm Intuition

The Multi-Armed Bandit Problem

D1

D2

D3

D4

D5

The Multi-Armed Bandit Problem

- We have d arms. For example, arms are ads that we display to users each time they connect to a web page.
- Each time a user connects to this web page, that makes a round.
- At each round n, we choose one ad to display to the user.
- At each round n, ad i gives reward $r_i(n) \in \{0, 1\}$: $r_i(n) = 1$ if the user clicked on the ad i, 0 if the user didn't.
- Our goal is to maximize the total reward we get over many rounds.

Bayesian Inference

- Ad i gets rewards $\boldsymbol{y}$ from Bernoulli distribution $p(\boldsymbol{y}|\theta_i) \sim \mathcal{B}(\theta_i)$.
- θ_i is unknown but we set its uncertainty by assuming it has a uniform distribution $p(\theta_i) \sim \mathcal{U}([0,1])$, which is the prior distribution.
- Bayes Rule: we approach θ_i by the posterior distribution

$$\underbrace{p(\theta_i|\mathbf{y})}_{\text{posterior distribution}} = \frac{p(\mathbf{y}|\theta_i)p(\theta_i)}{\int p(\mathbf{y}|\theta_i)p(\theta_i)d\theta_i} \propto \underbrace{p(\mathbf{y}|\theta_i)}_{\text{likelihood function}} \times \underbrace{p(\theta_i)}_{\text{prior distribution}}$$

- We get $p(\theta_i|\mathbf{y}) \sim \beta(\text{number of successes} + 1, \text{number of failures} + 1)$
- At each round n we take a random draw $\theta_i(n)$ from this posterior distribution $p(\theta_i|\boldsymbol{y})$, for each ad i.
- At each round n we select the ad i that has the highest $\theta_i(n)$.

Thompson Sampling Algorithm

Step 1. At each round n, we consider two numbers for each ad i:

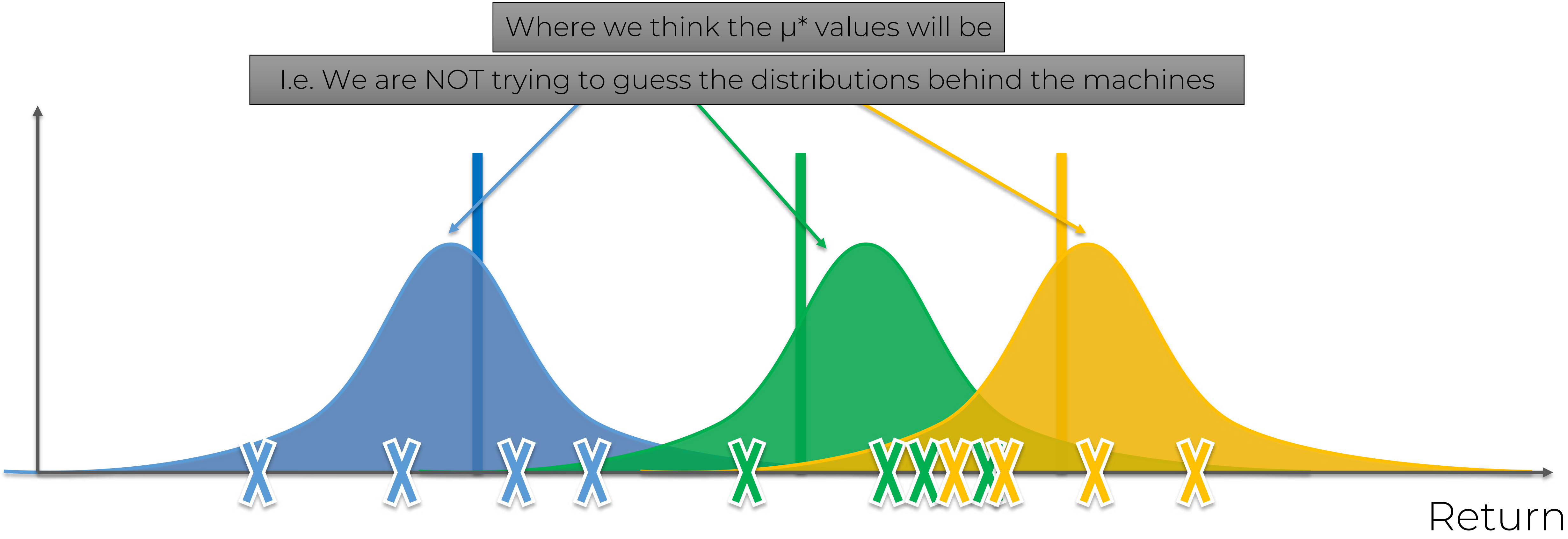
- $N_i^1(n)$ - the number of times the ad i got reward 1 up to round n,
- $N_i^0(n)$ - the number of times the ad i got reward 0 up to round n.

Step 2. For each ad i, we take a random draw from the distribution below:

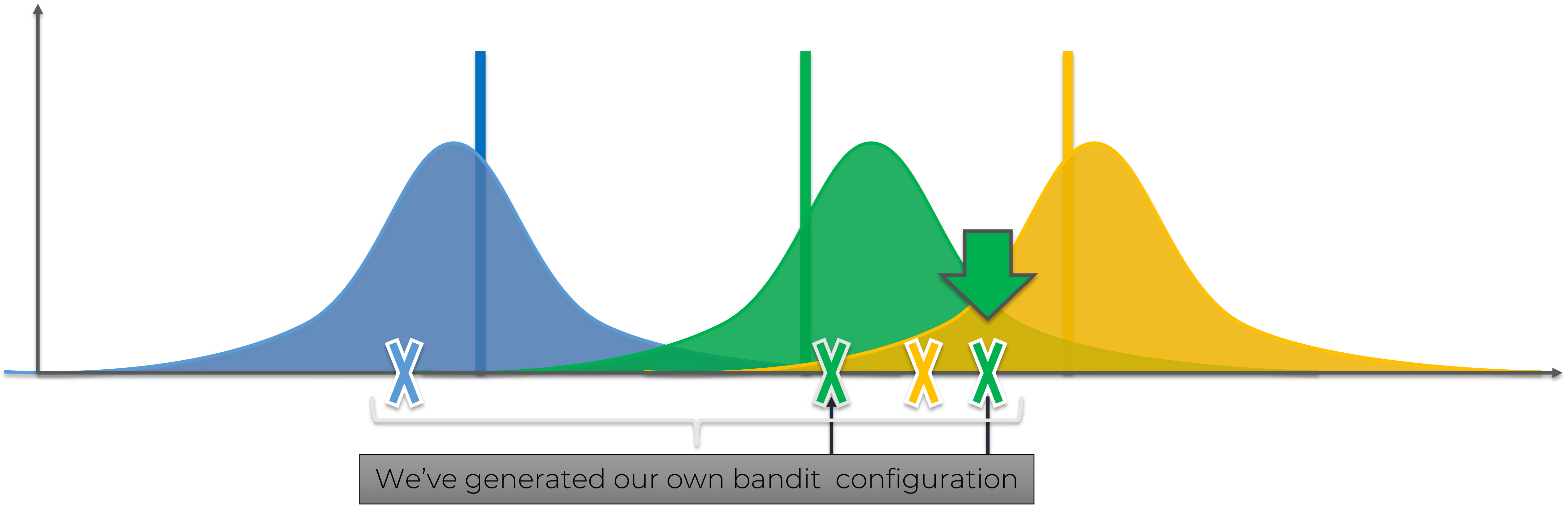
$$\theta_i(n) = \beta(N_i^1(n) + 1, N_i^0(n) + 1)$$

Step 3. We select the ad that has the highest $\theta_i(n)$.

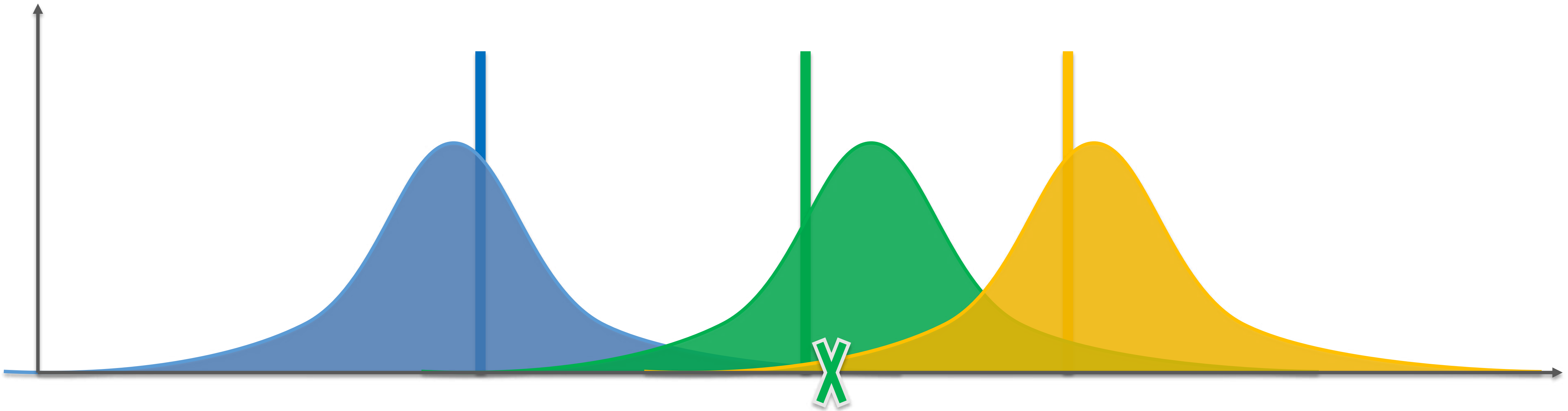
Thompson Sampling Algorithm



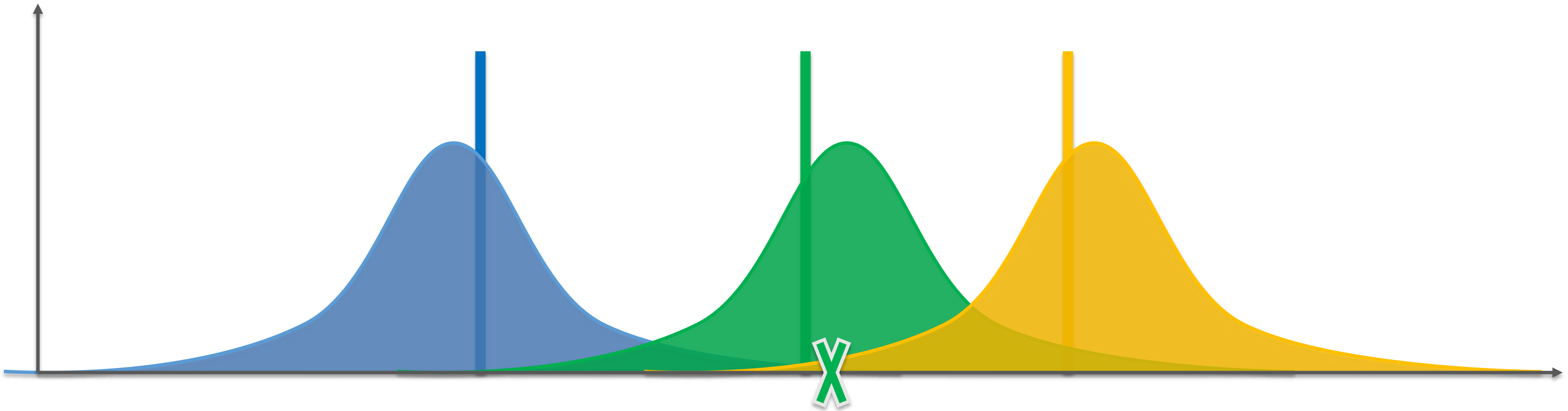
Thompson Sampling Algorithm



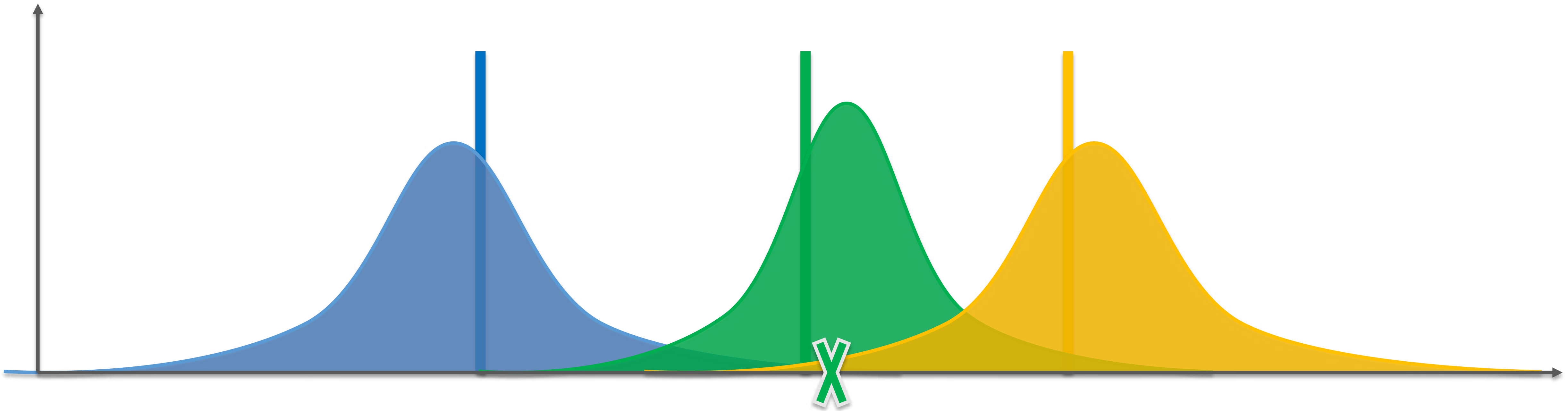
Thompson Sampling Algorithm



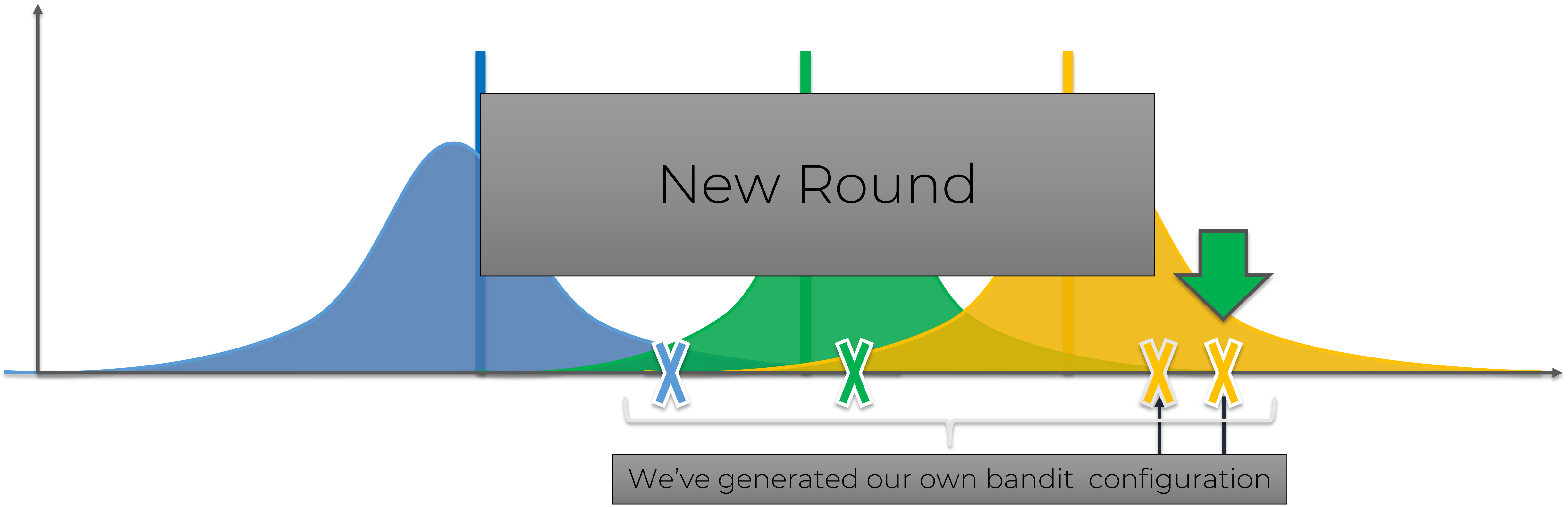
Thompson Sampling Algorithm



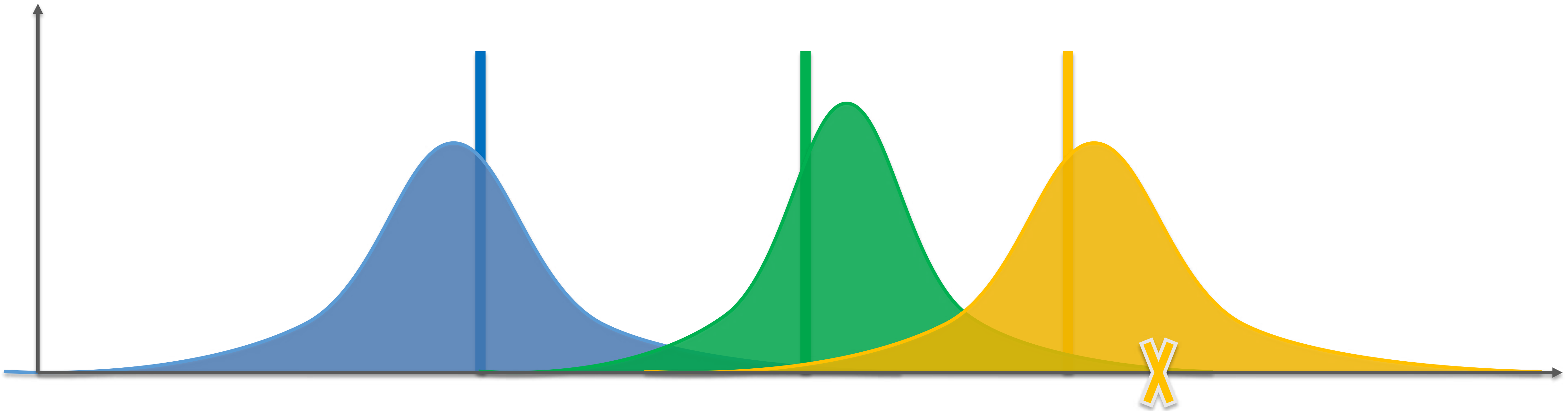
Thompson Sampling Algorithm



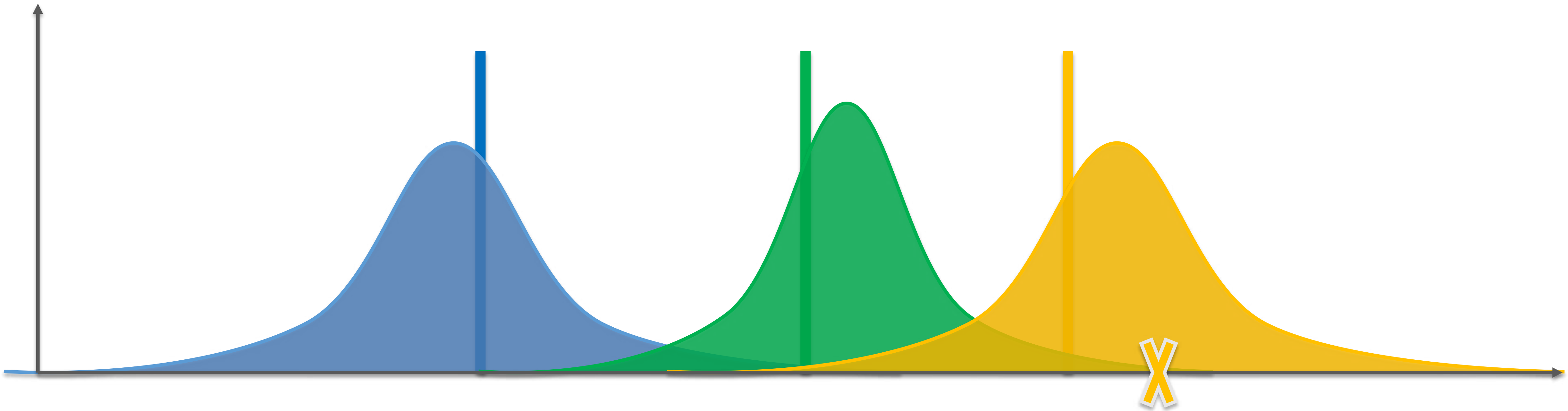
Thompson Sampling Algorithm



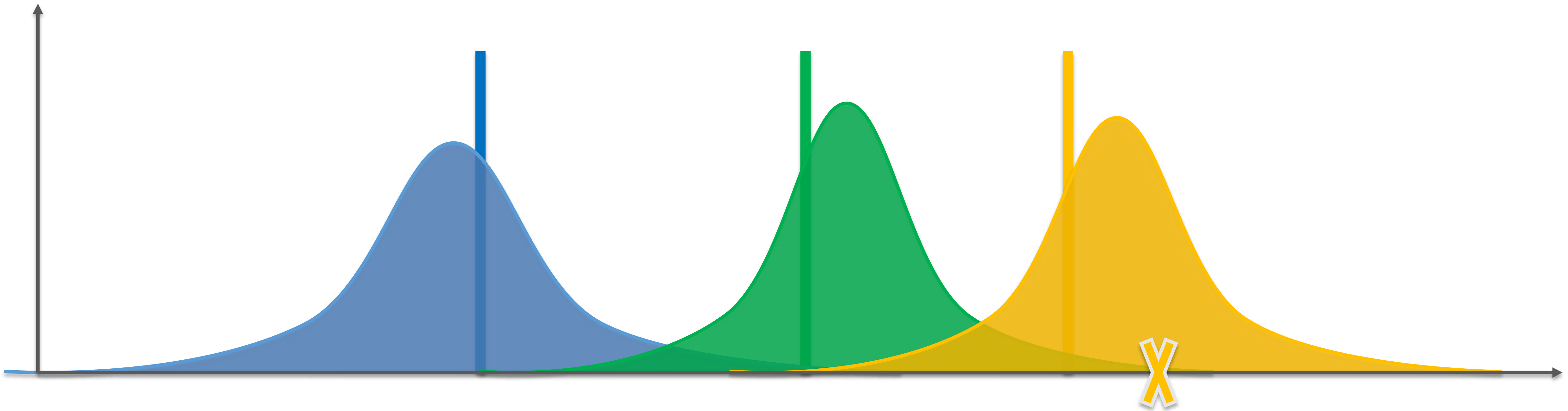
Thompson Sampling Algorithm



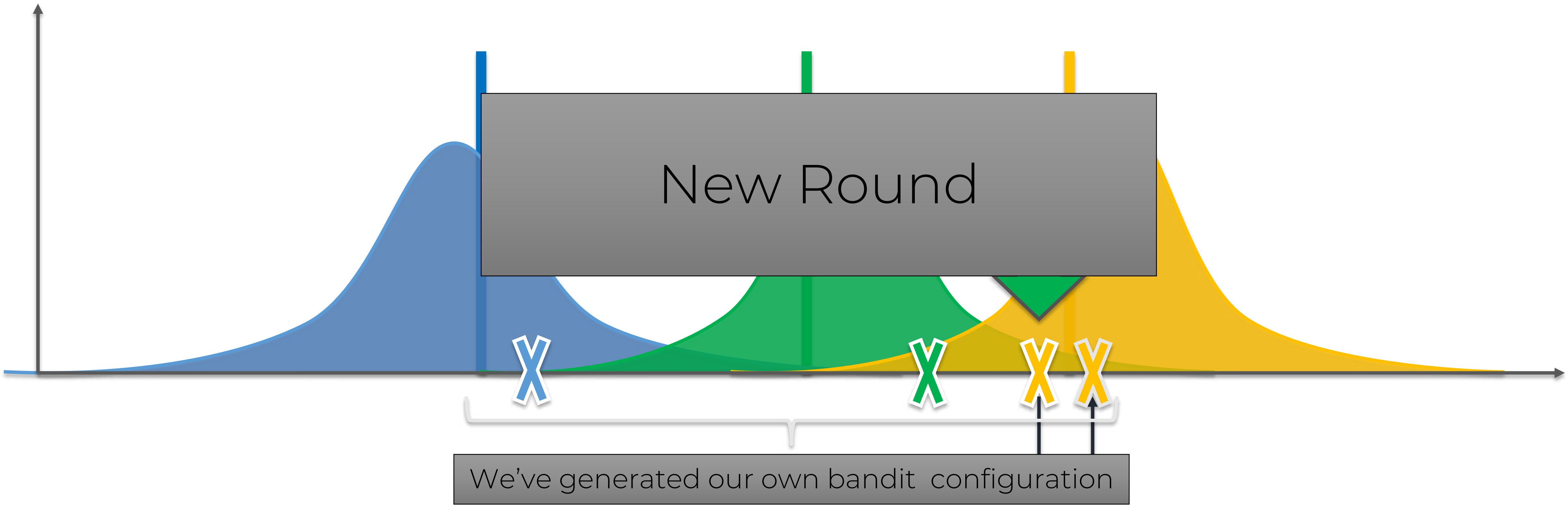
Thompson Sampling Algorithm



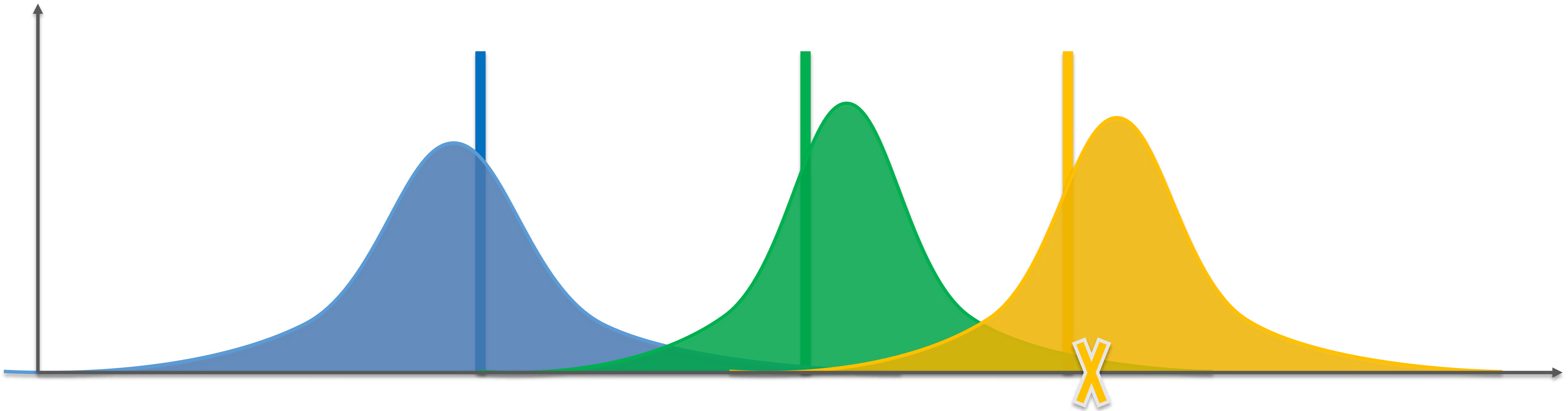
Thompson Sampling Algorithm



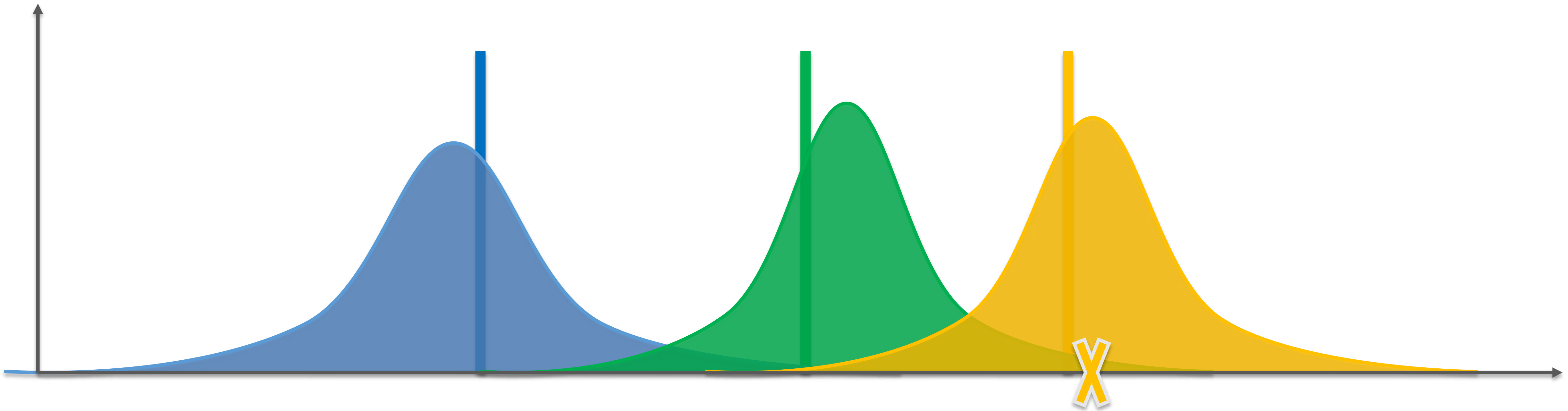
Thompson Sampling Algorithm



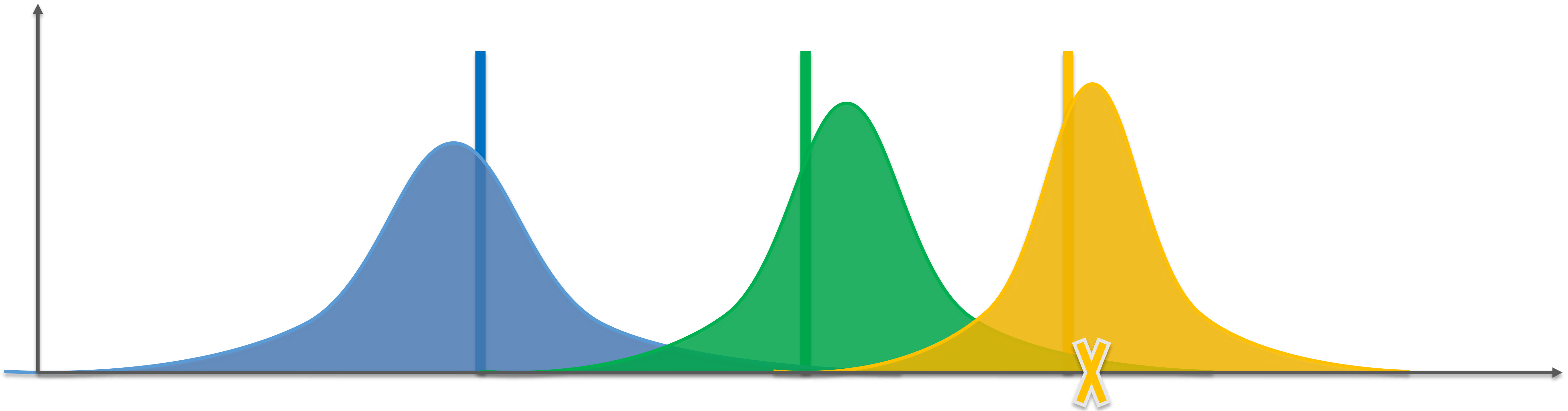
Thompson Sampling Algorithm



Thompson Sampling Algorithm



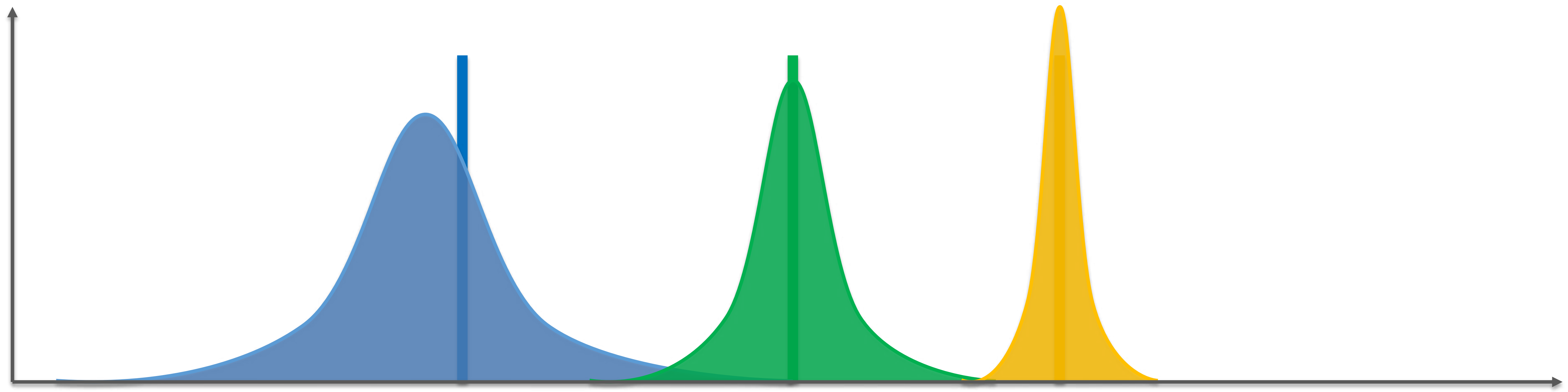
Thompson Sampling Algorithm



Thompson Sampling Algorithm

And so on…

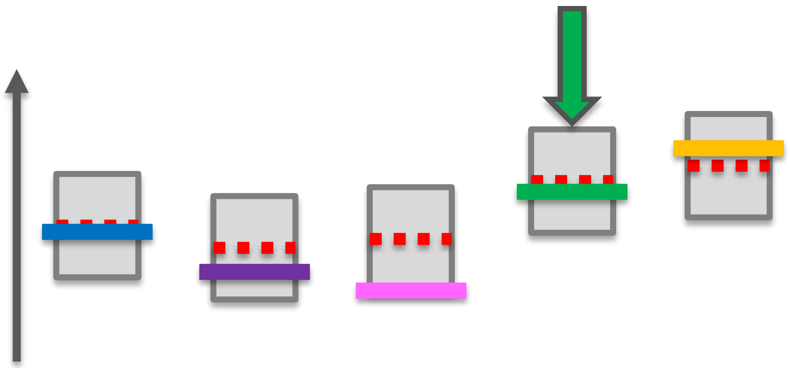
Thompson Sampling Algorithm



UCB vs Thompson Sampling

Thompson Sampling Algorithm

UCB



- Deterministic
- Requires update at every round

Thompson Sampling

- Probabilistic
- Can accommodate delayed feedback
- Better empirical evidence

Natural Language Processing (NLP)

Plan of Attack

Plan of Attack

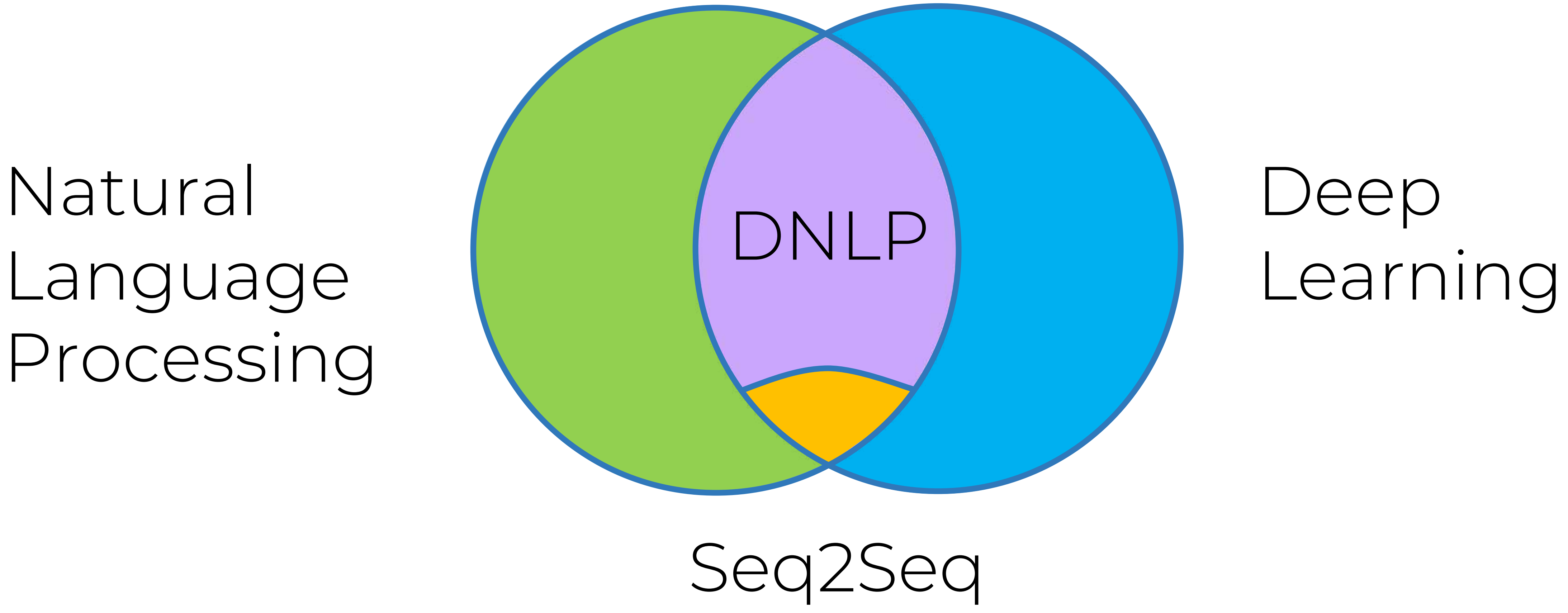
Here's what we will learn:

- Types of Natural Language Processing
- Classical vs Deep Learning Models
- End-to-end Deep Learning Models
- Bag-Of-Words

- Note: Seq2Seq and Chatbots are outside the scope of this course

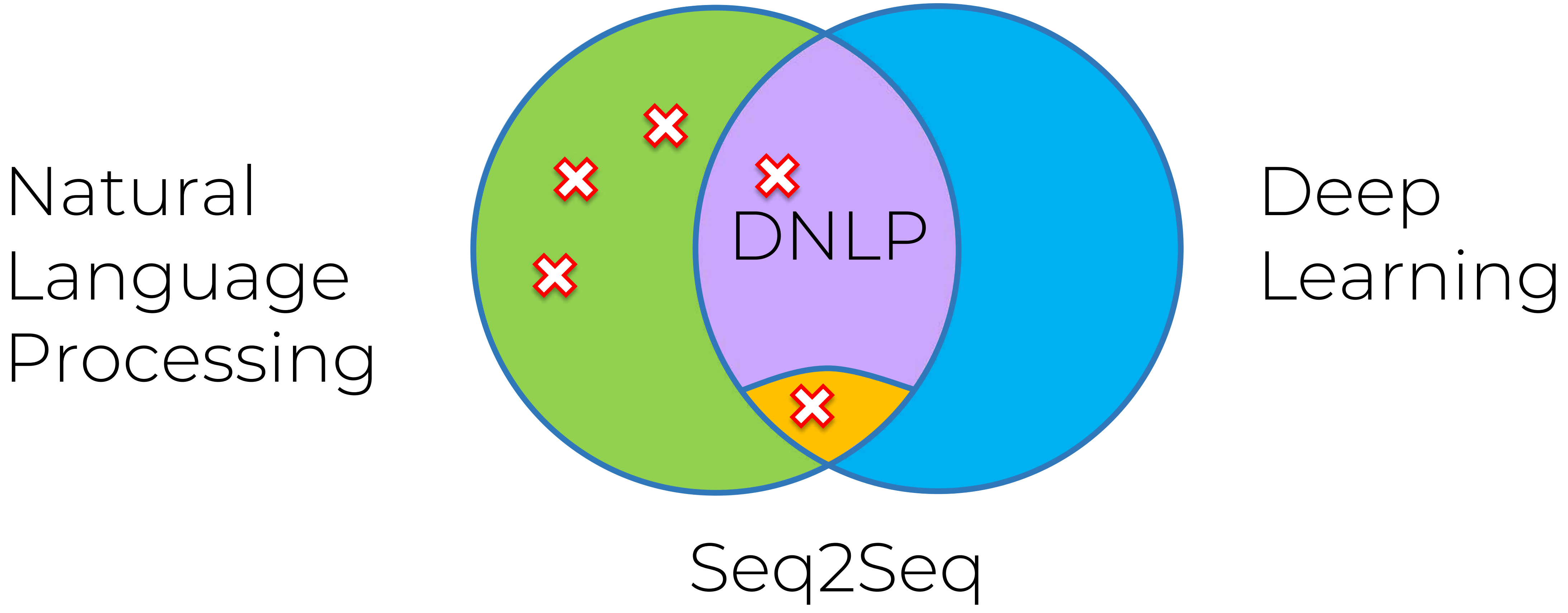
Types of NLP

Types of NLP



Classical vs Deep Learning Models

Classical vs Deep Learning Models



Classical vs Deep Learning Models

Some examples:

1. If / Else Rules (Chatbot)
2. Audio frequency components analysis (Speech Recognition)
3. Bag-of-words model (Classification)
4. CNN for text Recognition (Classification)

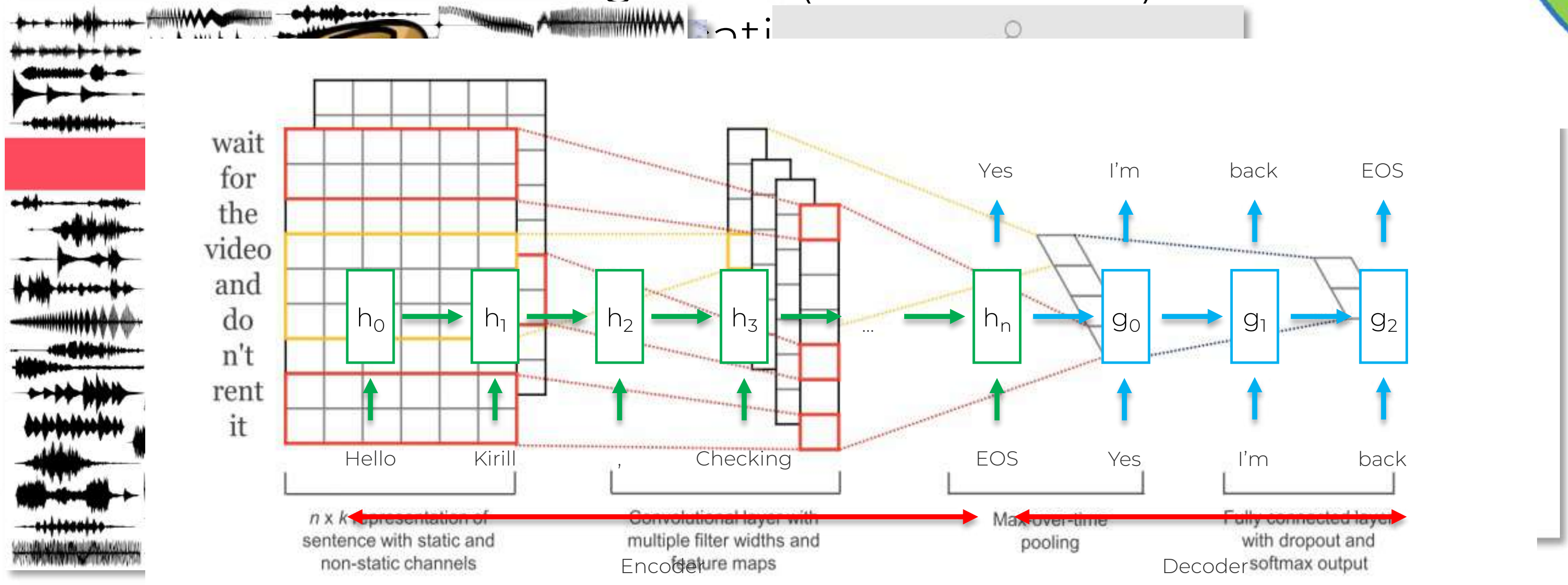
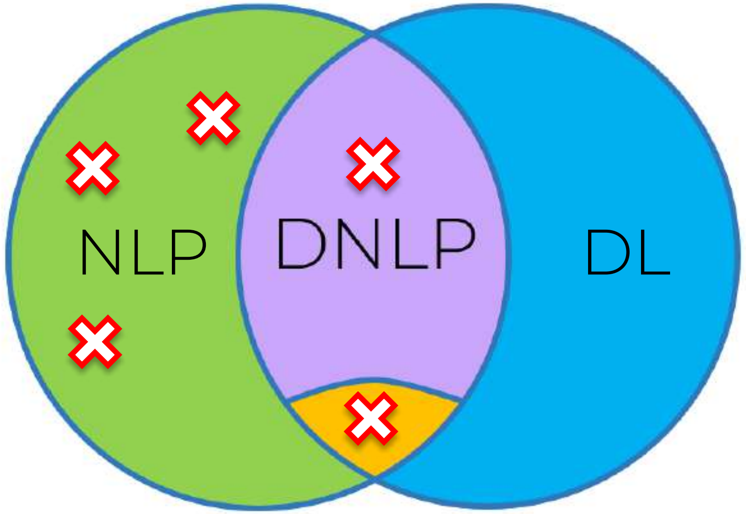


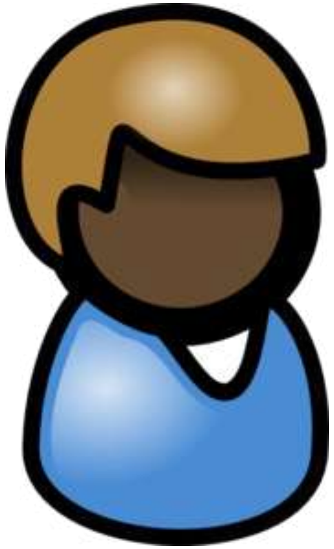
Image Source: www.wildml.com

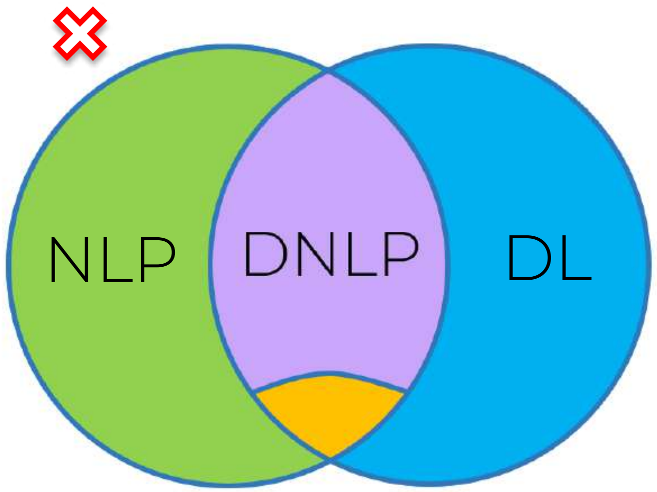


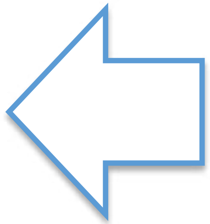
Seq2Seq

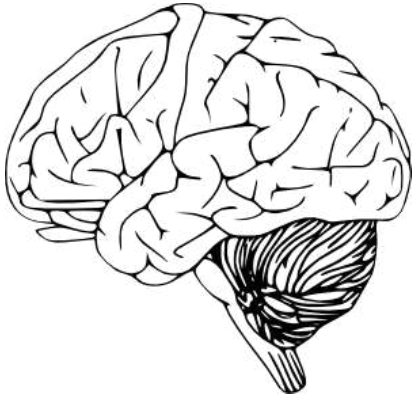
End-to-end Deep Learning Models

End-to-end Deep Learning Models



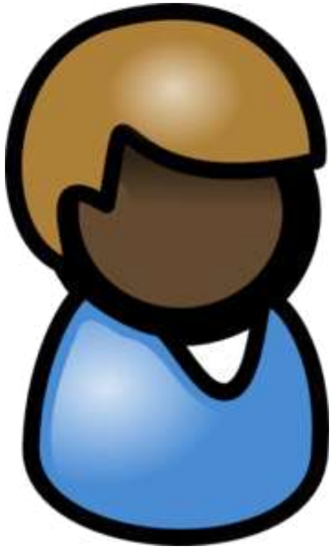


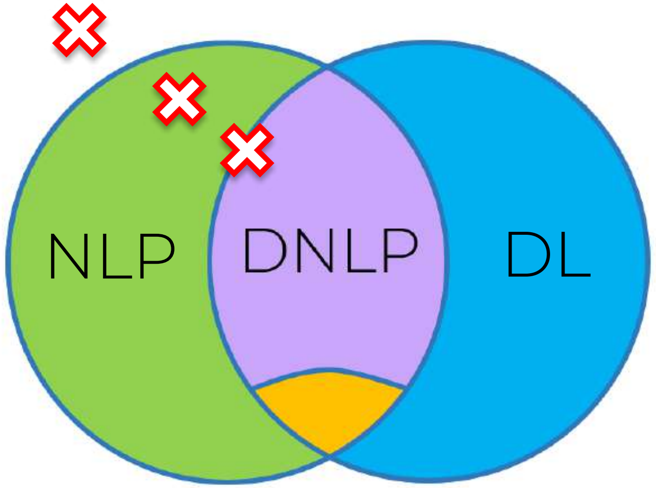




End-to-end Deep Learning Models

End-to-end Deep Learning Models

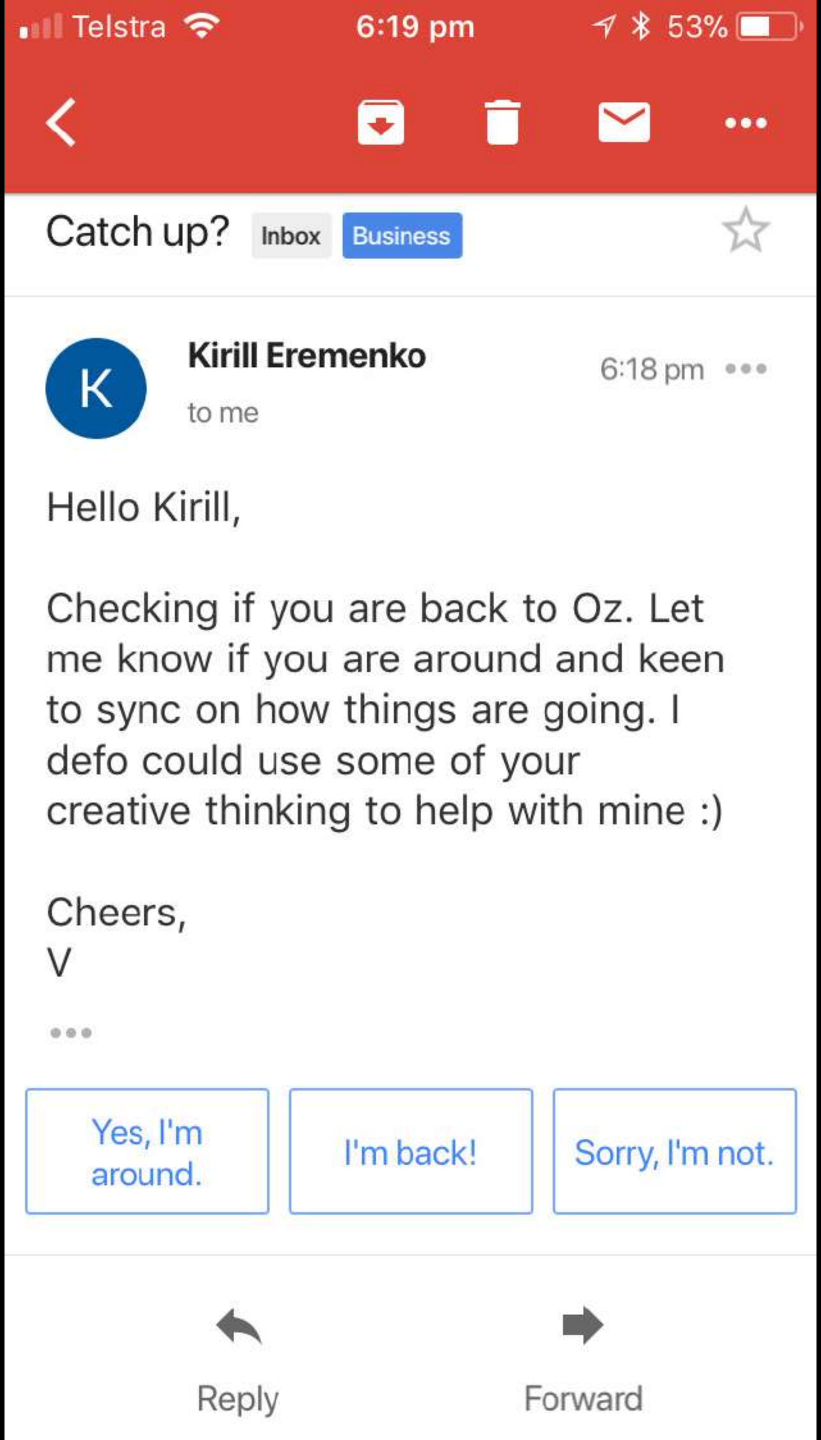




End-to-end Deep Learning Models

End-to-end Deep Learning Models

Bag-Of-Words


Telstra
6:19 pm
53%
Catch up?
Inbox
Business
K
Kirill Eremenko
6:18 pm
to me
Hello Kirill,
Checking if you are back to Oz. Let me know if you are around and keen to sync on how things are going. I defo could use some of your creative thinking to help with mine :)
Cheers,
V
Yes, I'm around.
I'm back!
Sorry, I'm not.
Reply
Forward

Bag-Of-Words

Yes / No

Bag-Of-Words

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, … , 0]

20,000 elements long

if badminton table

171,476 words

The Second Edition of the 20-volume Oxford English Dictionary contains full entries for **171,476 words** in current use, and **47,156** obsolete words. To this may be added around **9,500** derivative words included as subentries.

How many words are there in the English language?
https://en.oxforddictionaries.com/.../how-many-words-are-there-in-the-english-language

About this result Feedback

People also ask

How many words in the English language does the average person know?

Most adult native test-takers range from **20,000–35,000 words**. Average native test-takers of age 8 already know **10,000 words**. Average native test-takers of age 4 already know **5,000 words**. Adult native test-takers learn almost 1 new word a day until middle age. May 29, 2013

Lexical facts - The Economist
https://www.economist.com/blogs/johnson/2013/05/vocabulary-size

We have seen that the Oxford English Dictionary contains **171,476 words** in current use, whereas a vocabulary of just **3000 words** provides coverage for around 95% of common texts. If you do the math, that's 1.75% of the total number of words in use! Mar 14, 2013

How many words in the english language ? How many do i need to ...
https://www.lingholic.com/how-many-words-do-i-need-to-know-the-955-rule-in-langua...

Bag-Of-Words

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, … , 0]

20,000 elements long

SOS

EOS

Special Words

Bag-Of-Words

Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, … , 0]

20,000 elements long

Bag-Of-Words

Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, … , 3]

20,000 elements long

Bag-Of-Words

Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, … , 3] → **Yes / No ?**

20,000 elements long

Bag-Of-Words

Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, … , 3] ⟶ Yes / No ?

20,000 elements long

Training Data:

Hey mate, have you read about Hinton's capsule networks?	⟶	No
Did you like that recipe I sent you last week?	⟶	Yes
Hi Kirill, are you coming to dinner tonight?	⟶	Yes
Dear Kirill, would you like to service your car with us again?	⟶	No
Are you coming to Australia in December?	⟶	Yes
…	⟶	…

Bag-Of-Words

Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, … , 3] ⟶ **Yes / No ?**

20,000 elements long

Training Data:

[1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, … , 2] ⟶ **No**

[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, … , 0] ⟶ **Yes**

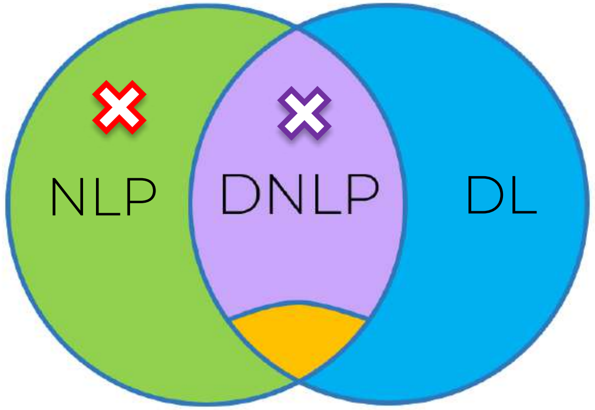
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, … , 1] ⟶ **Yes**

[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, … , 1] ⟶ **No**

[1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, … , 1] ⟶ **Yes**

… ⟶ …

Bag-Of-Words



Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, … , 3] → Yes / No ?

20,000 elements long

Training Data:

[1, 1, 0, 0, 0 … 0, 1, 0, 1, 0, 1,
[1, 1, 0, 0, 0 … 0, 2, 0, 0, 0
[1, 1, 0, 0, 0 … 0, 1, 0, 0, 0,
[1, 1, 0, 0, 0 … 0, 1, 1, 0, 1, 0
[1, 1, 0, 0, 0 … 0, 1, 0, 0, 1,
…

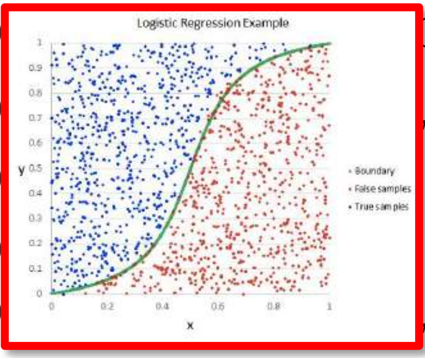
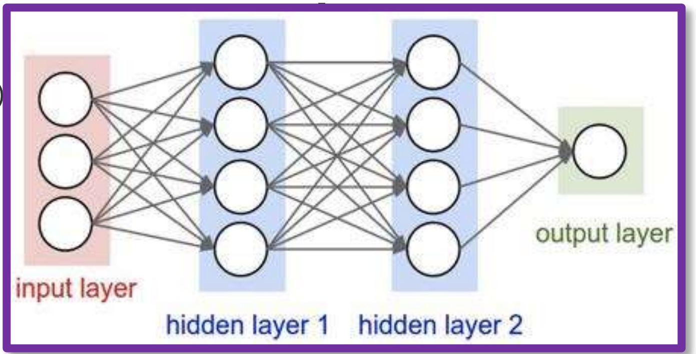


Image Source: www.helloacm.com



No
Yes
Yes
No
Yes
…

Artificial Neural Network Intuition

A Clip From The Today Show, 1994

What is Deep Learning?

What is Deep Learning?

What is Deep Learning?

1956

1980

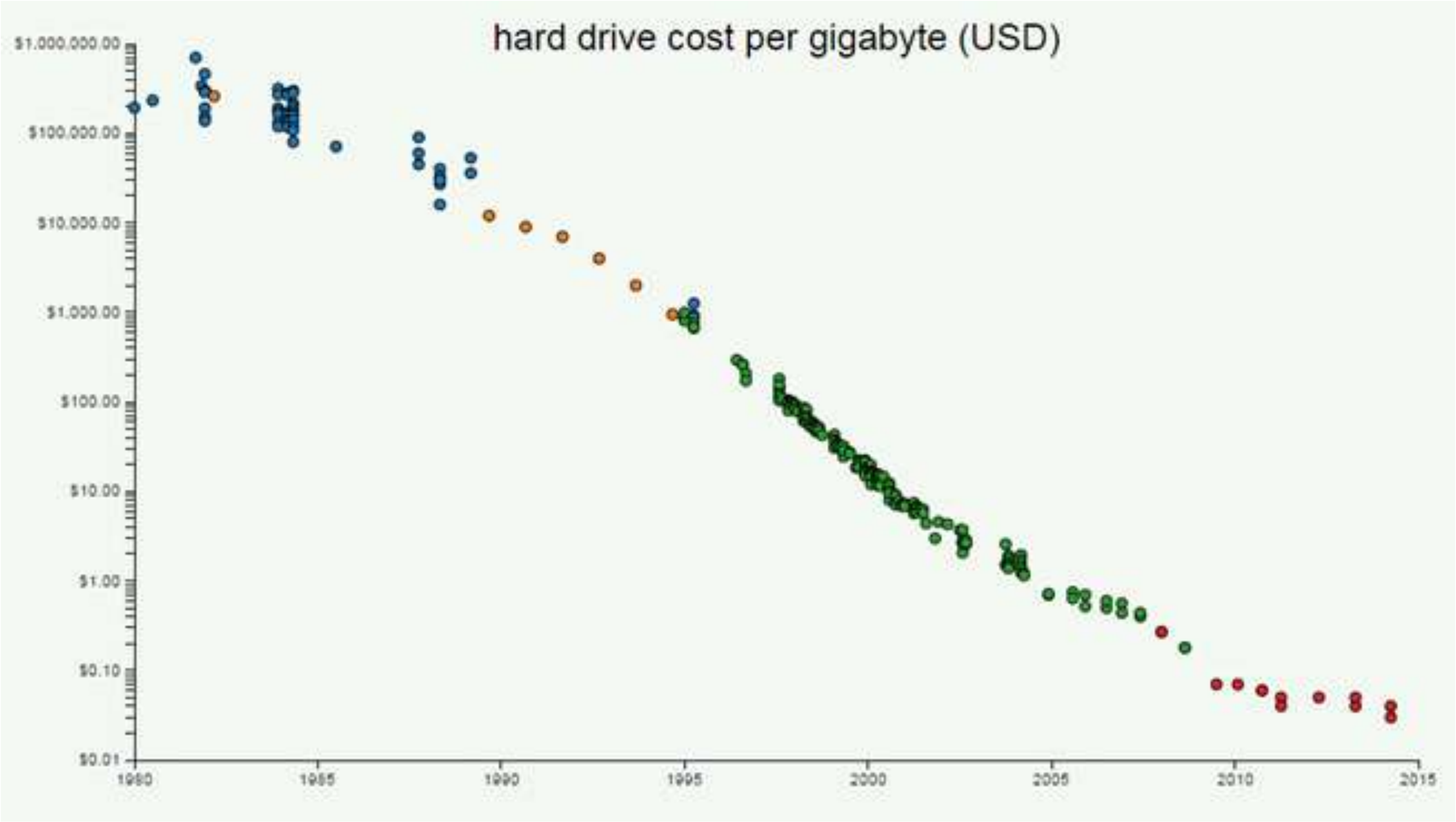
2017

2x

25,600x

What is Deep Learning?

Log-scale



Source: mkomo.com

What is Deep Learning?

What is Deep Learning?

STORAGE LIMITS

Estimates based on bacterial genetics suggest that digital DNA could one day rival or exceed today's storage technology.

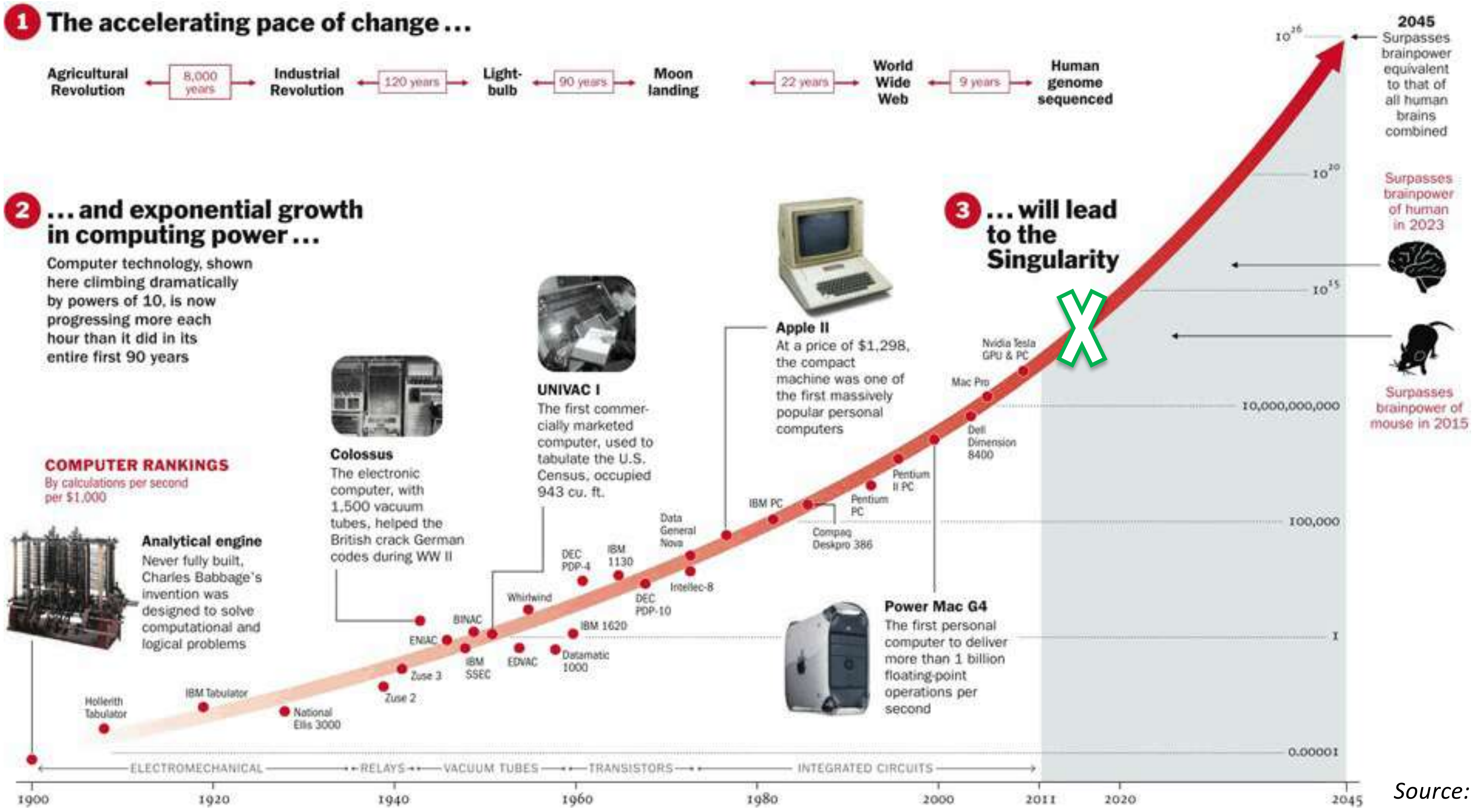
	Hard disk	Flash memory	Bacterial DNA
Read–write speed (μs per bit)	~3,000–5,000	~100	<100
Data retention (years)	>10	>10	>100
Power usage (watts per gigabyte)	~0.04	~0.01–0.04	$<10^{-10}$
Data density (bits per cm³)	$\sim10^{13}$	$\sim10^{16}$	$\sim10^{19}$

WEIGHT OF DNA NEEDED TO STORE WORLD'S DATA

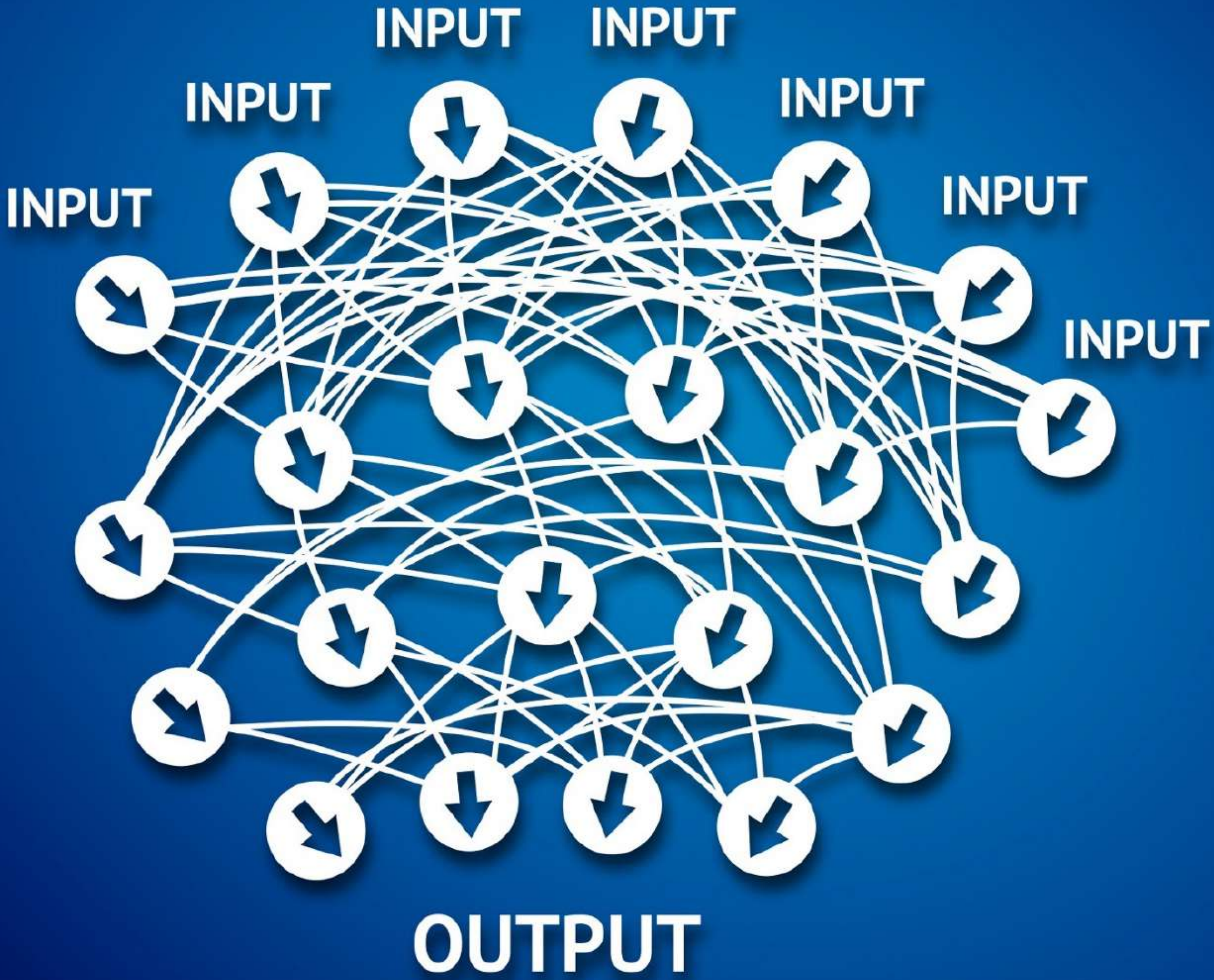
©nature

Source: nature.com

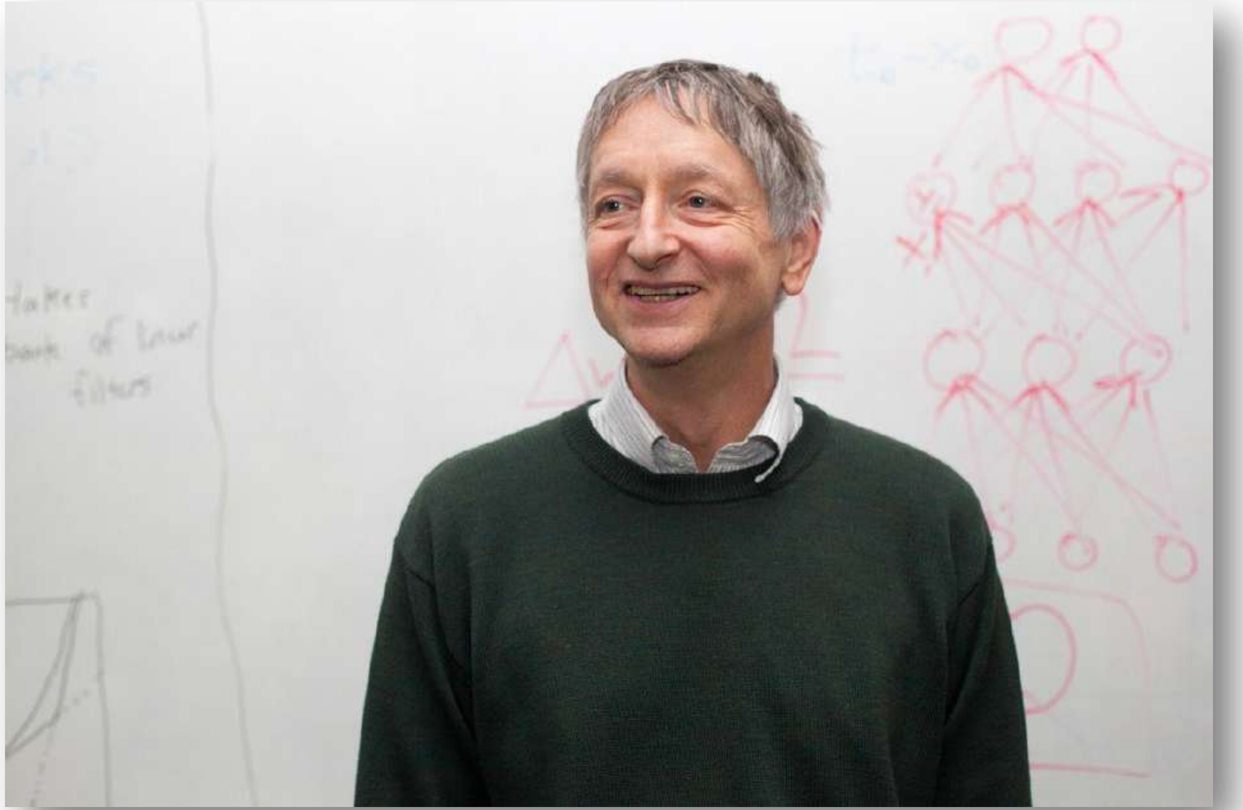
What is Deep Learning?



Source: Time Magazine

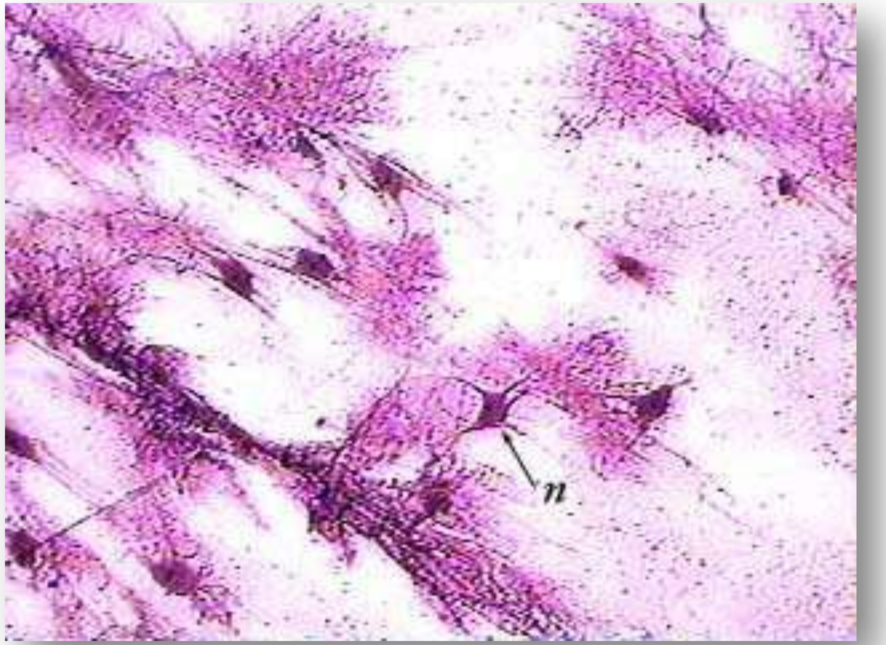

INPUT
INPUT
INPUT
INPUT
INPUT
INPUT
INPUT
OUTPUT

What is Deep Learning?



Geoffrey Hinton

What is Deep Learning?



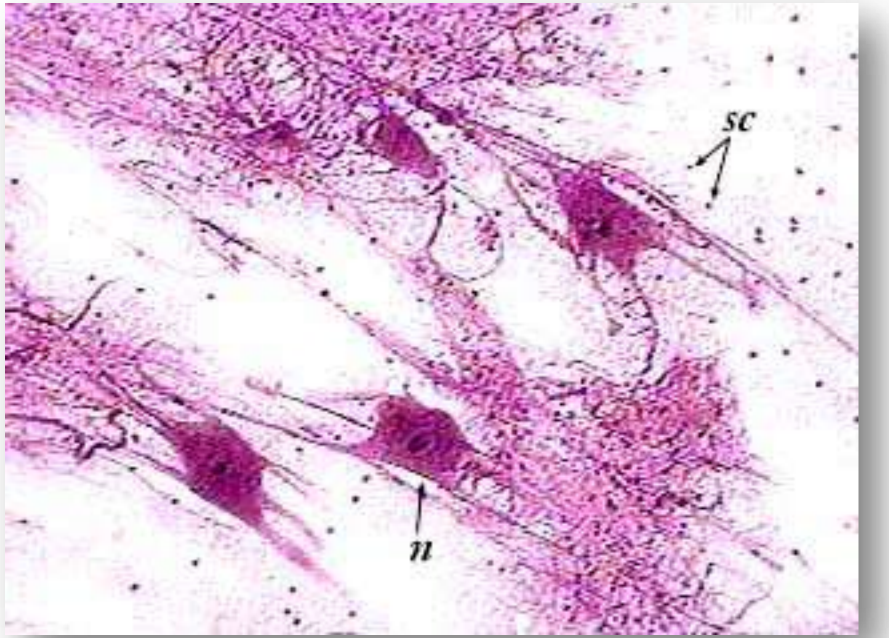
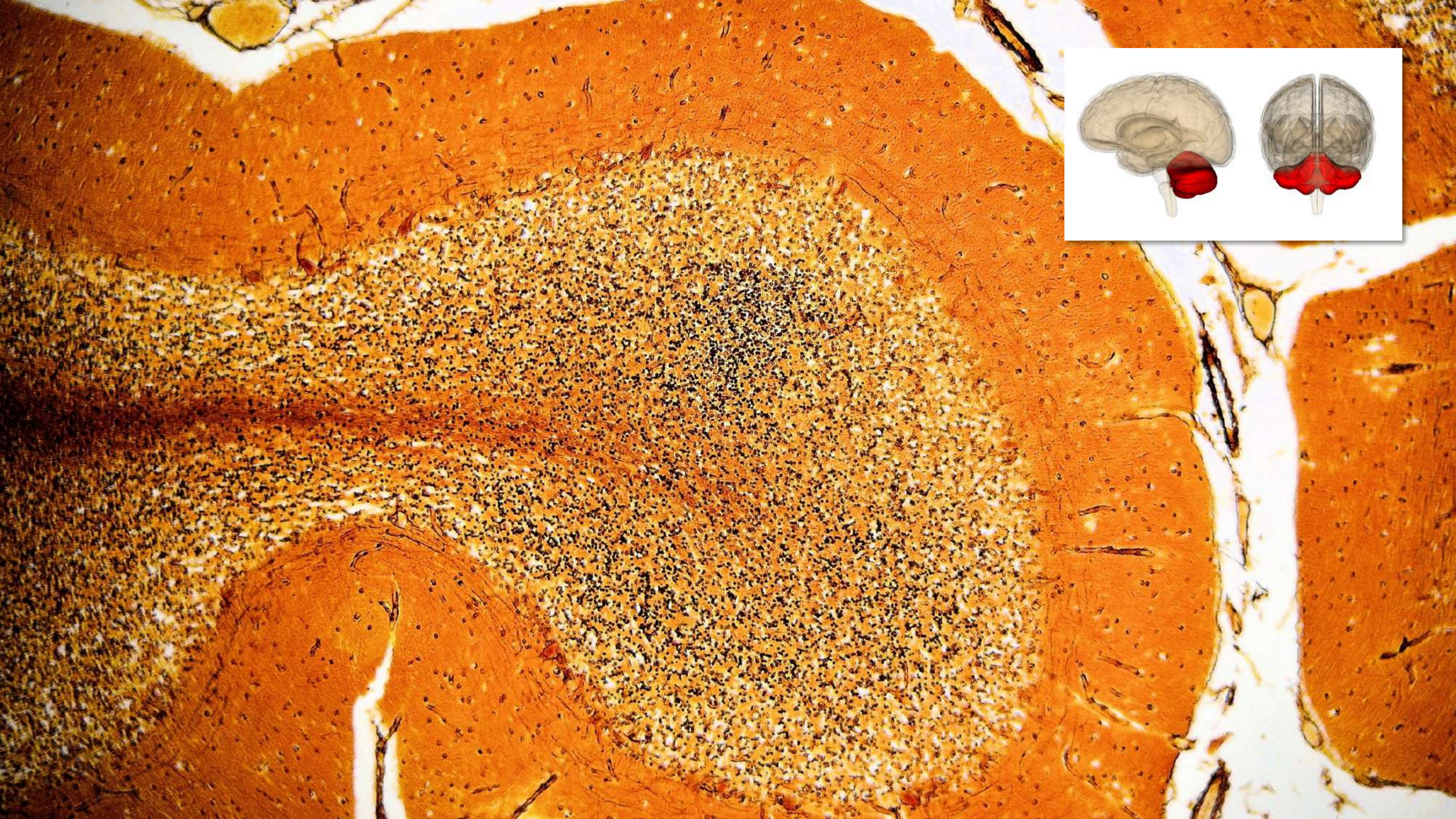
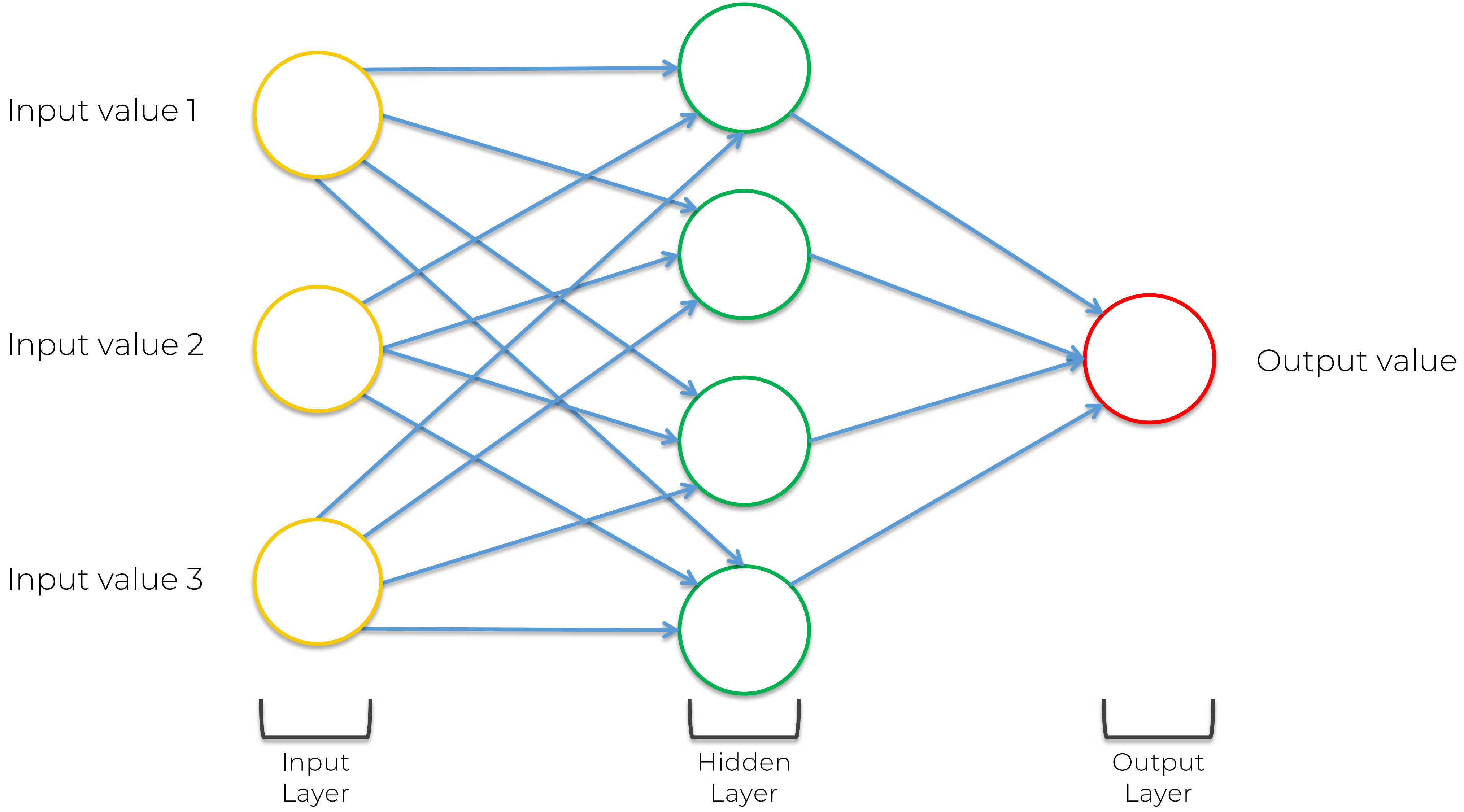


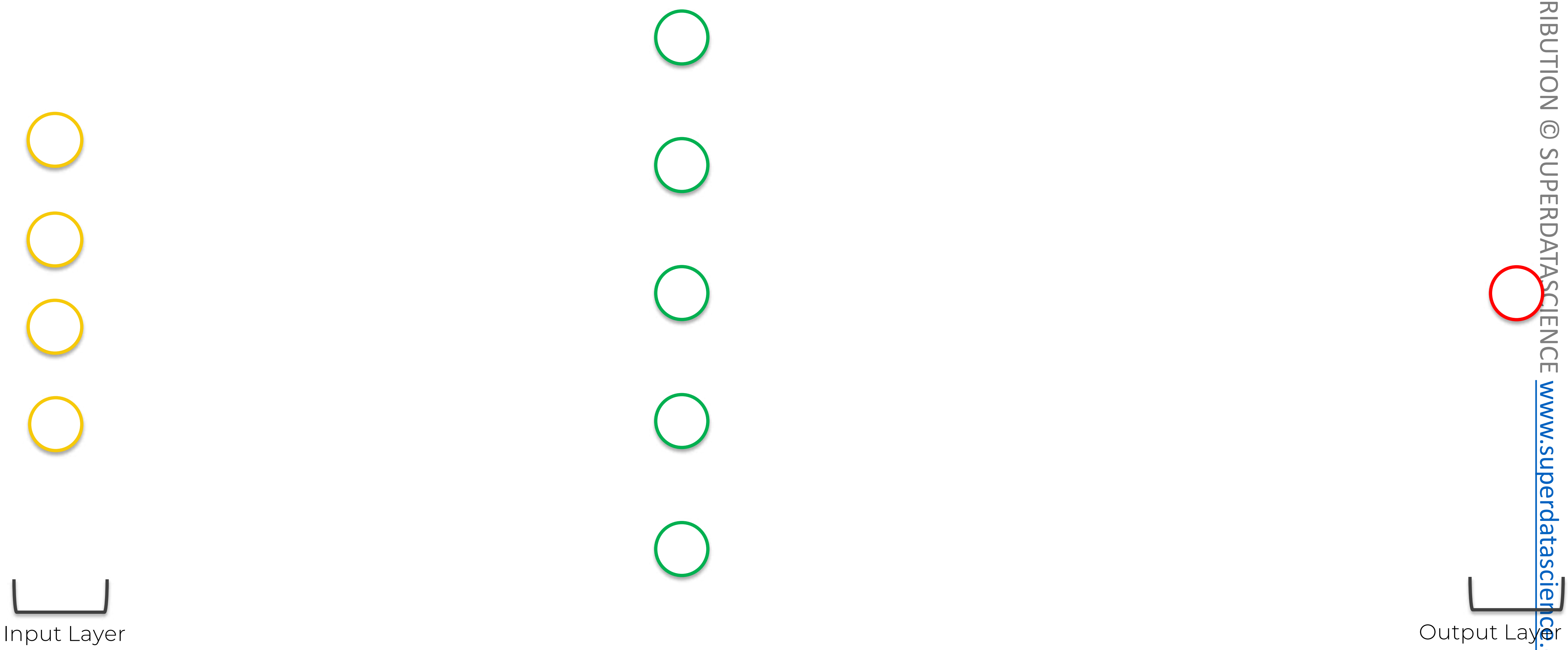
Image Source: www.austincc.edu



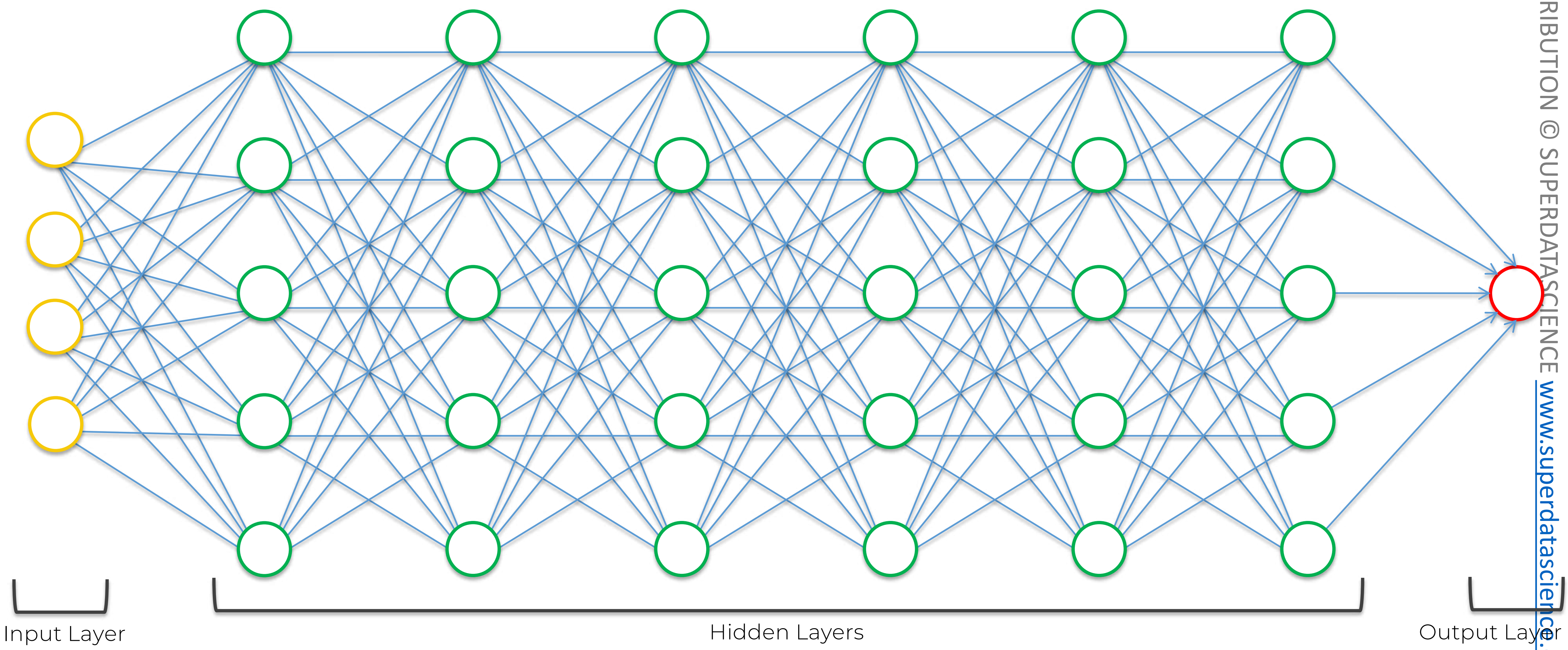
What is Deep Learning?



What is Deep Learning?



What is Deep Learning?



Supervised vs Unsupervised

Supervised vs Unsupervised

Supervised	Artificial Neural Networks	Used for Regression & Classification
	Convolutional Neural Networks	Used for Computer Vision
	Recurrent Neural Networks	Used for Time Series Analysis
Unsupervised	Self-Organizing Maps	Used for Feature Detection
	Deep Boltzmann Machines	Used for Recommendation Systems
	AutoEncoders	Used for Recommendation Systems

Plan of Attack

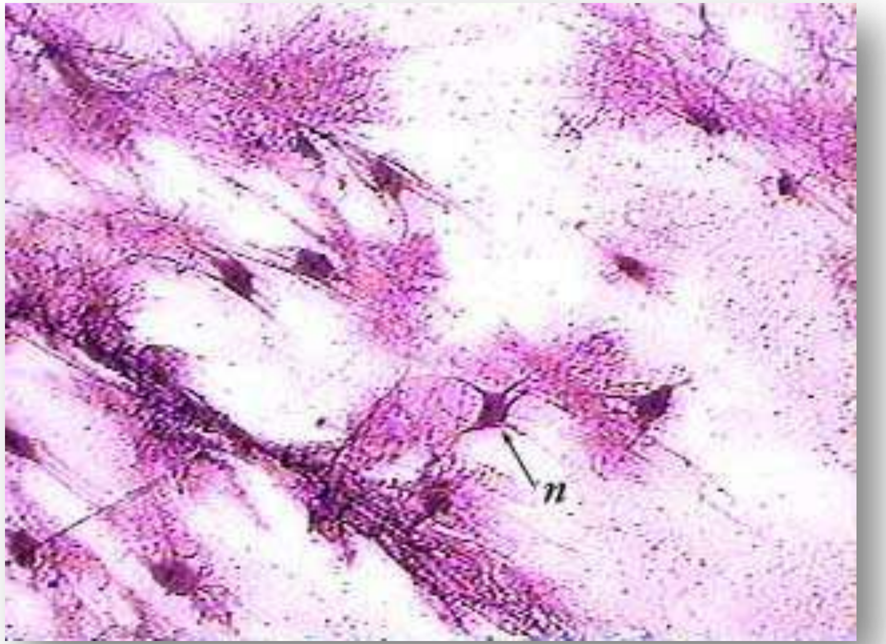
Plan of Attack

What we will learn in this section:

- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Backpropagation

The Neuron

The Neuron



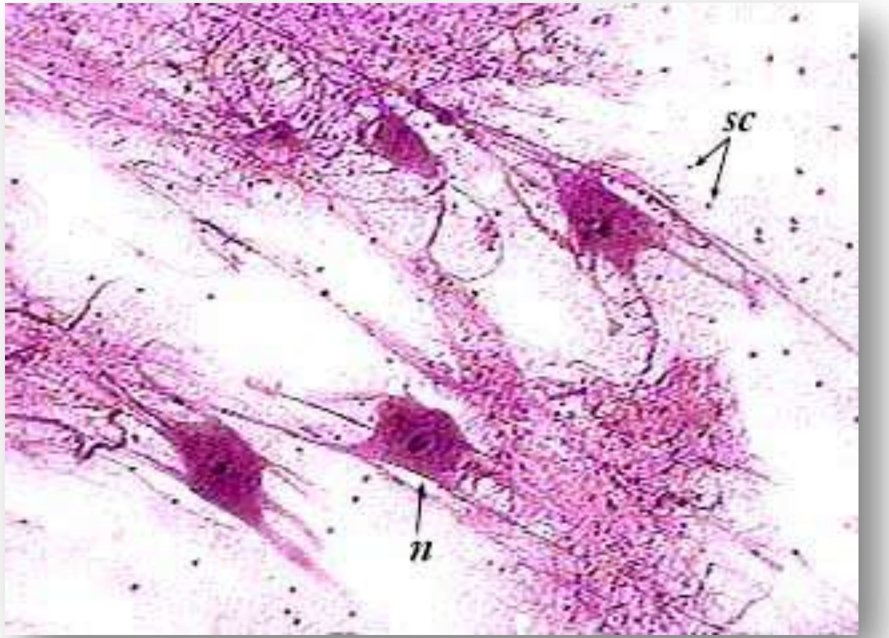


Image Source: www.austincc.edu

The Neuron

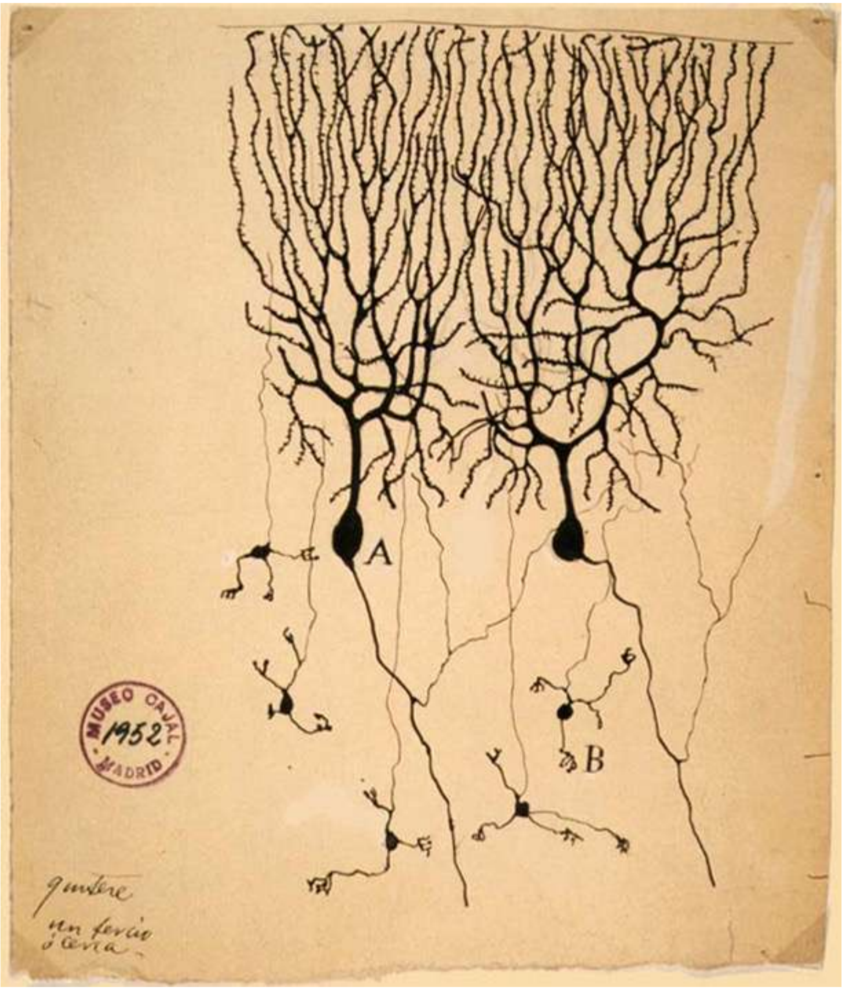


Image Source: Wikipedia

The Neuron

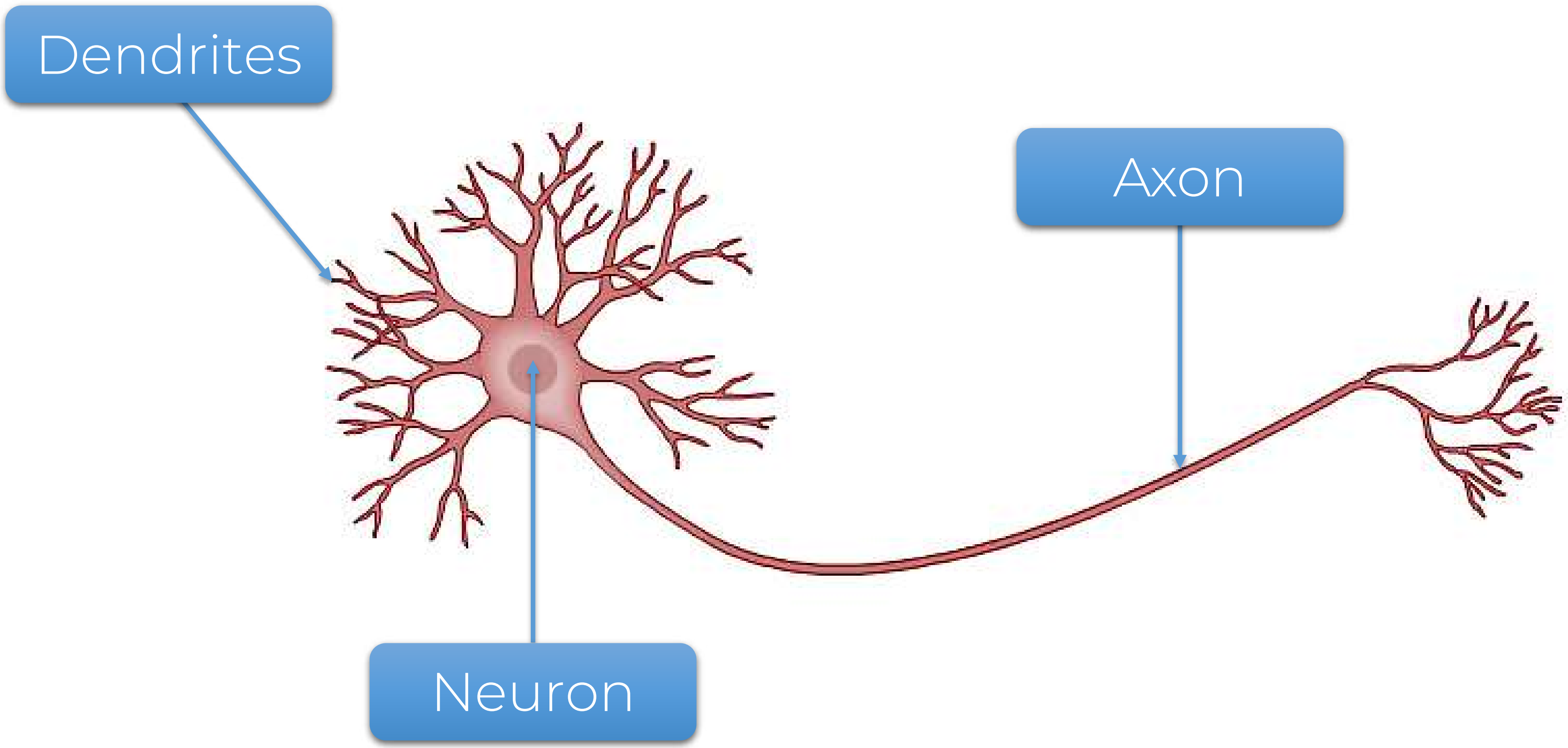


Image Source: Wikipedia

The Neuron

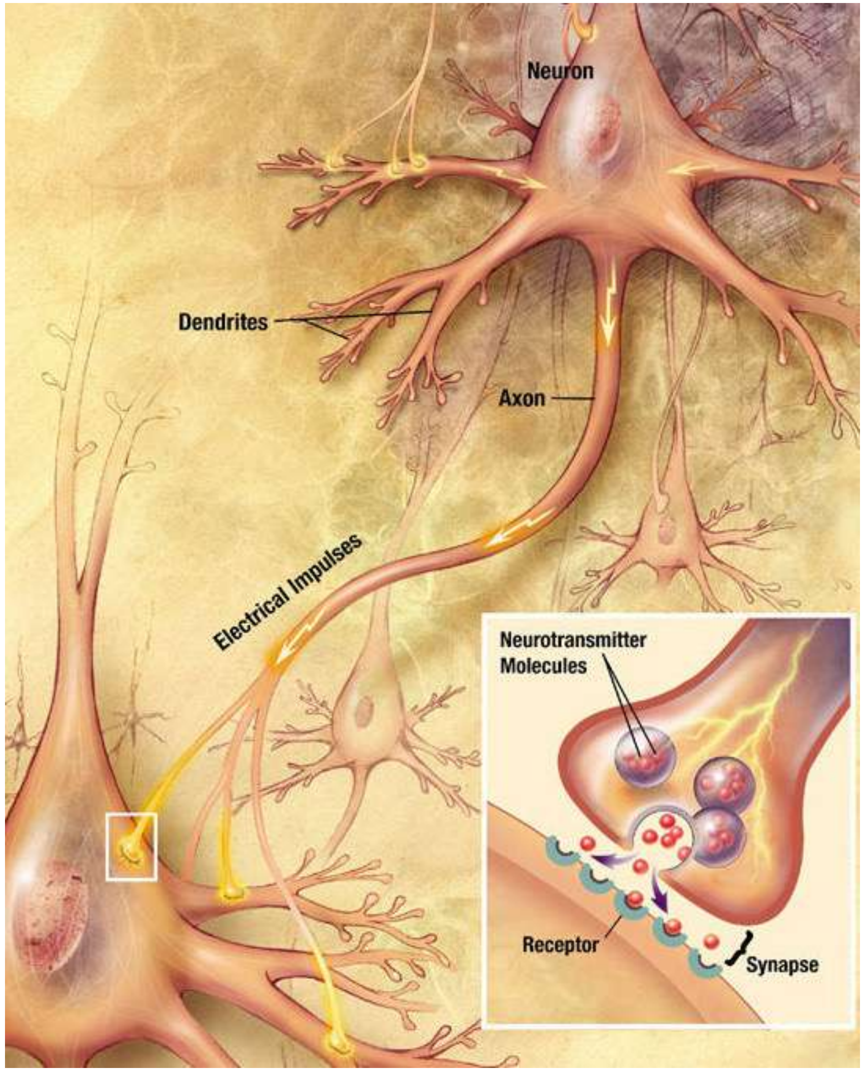
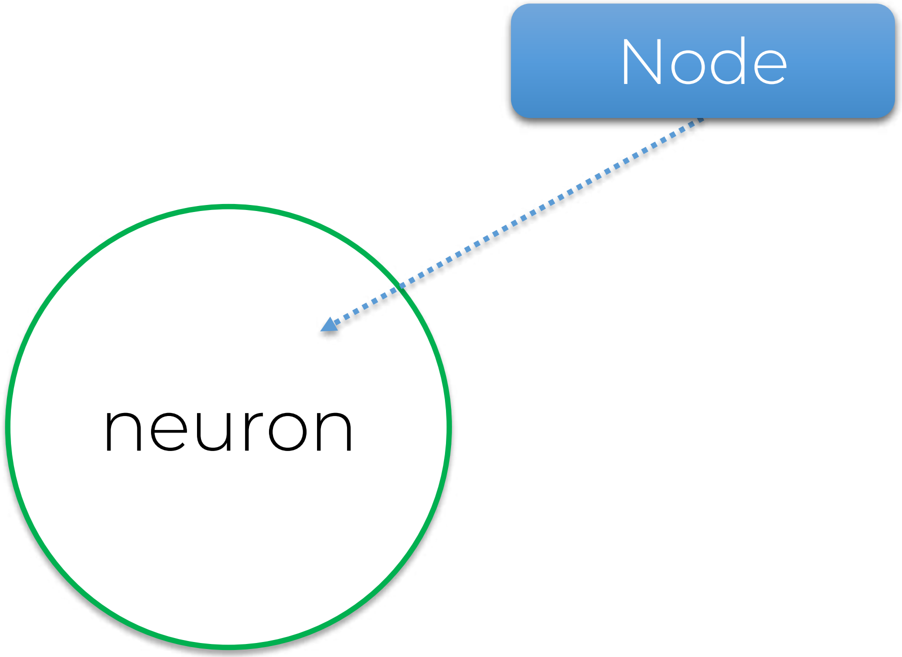
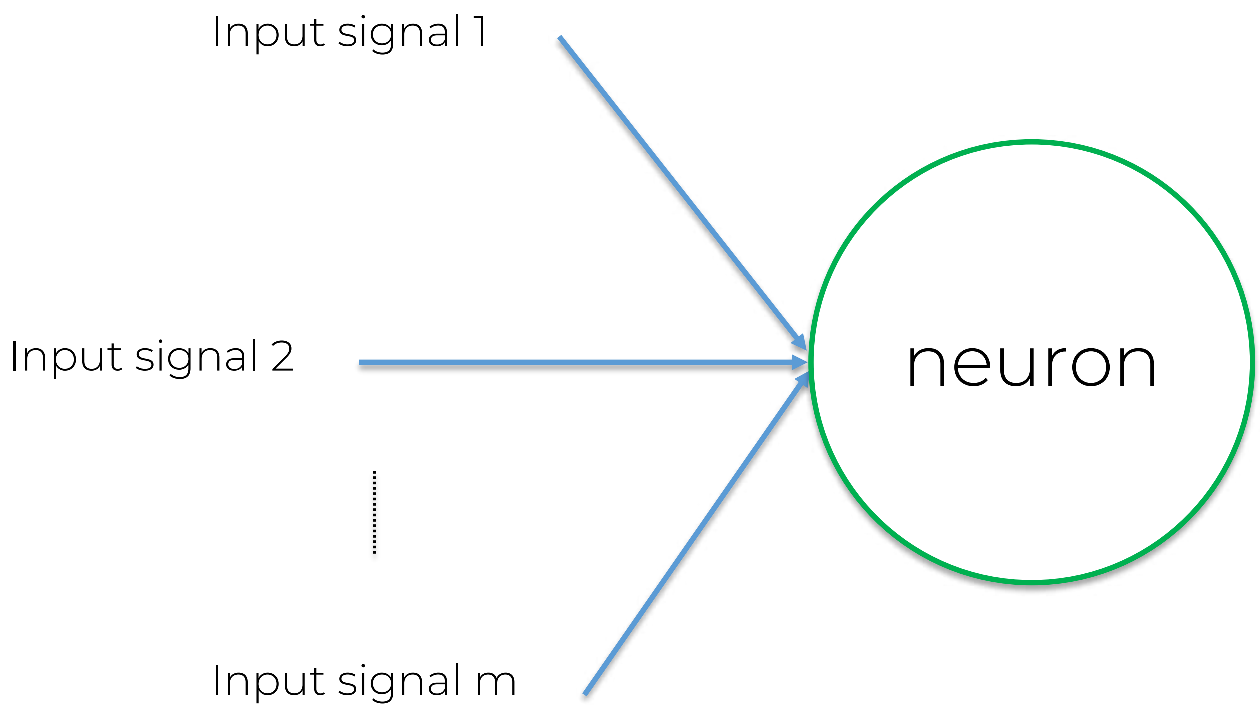


Image Source: Wikipedia

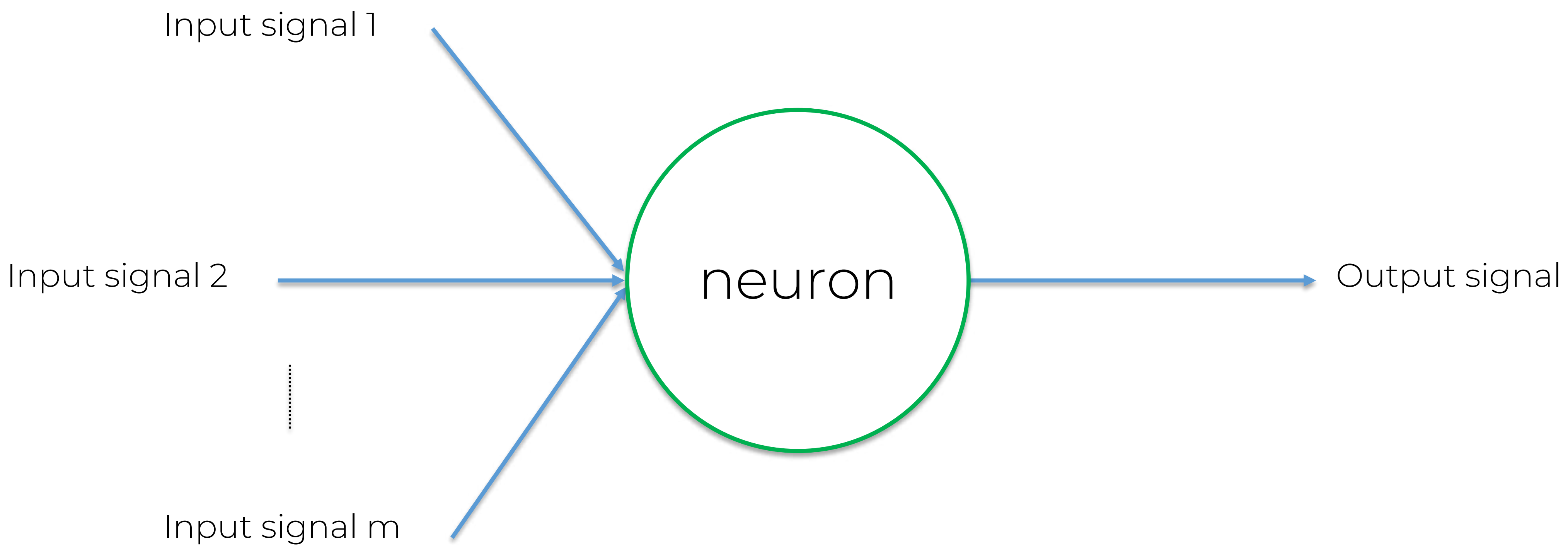
The Neuron



The Neuron

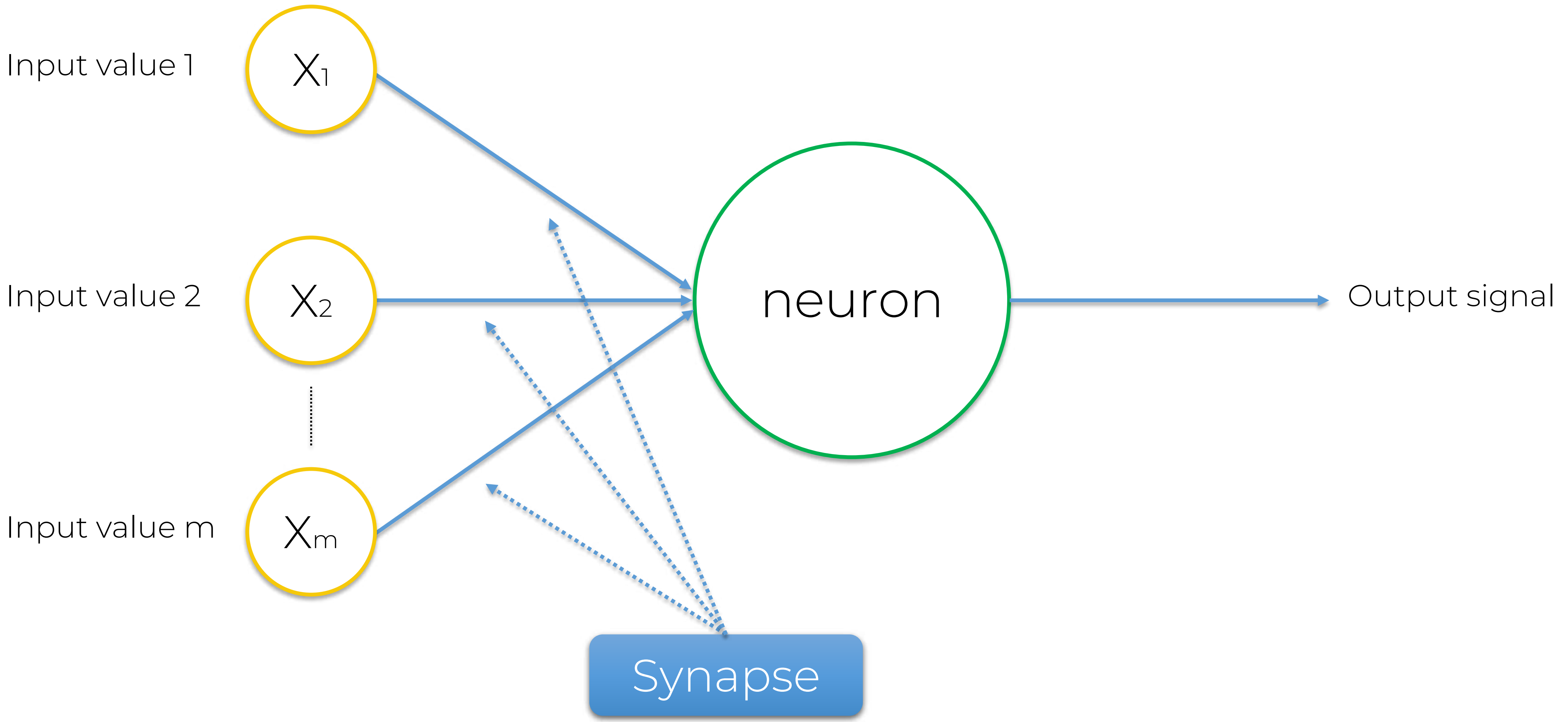


The Neuron

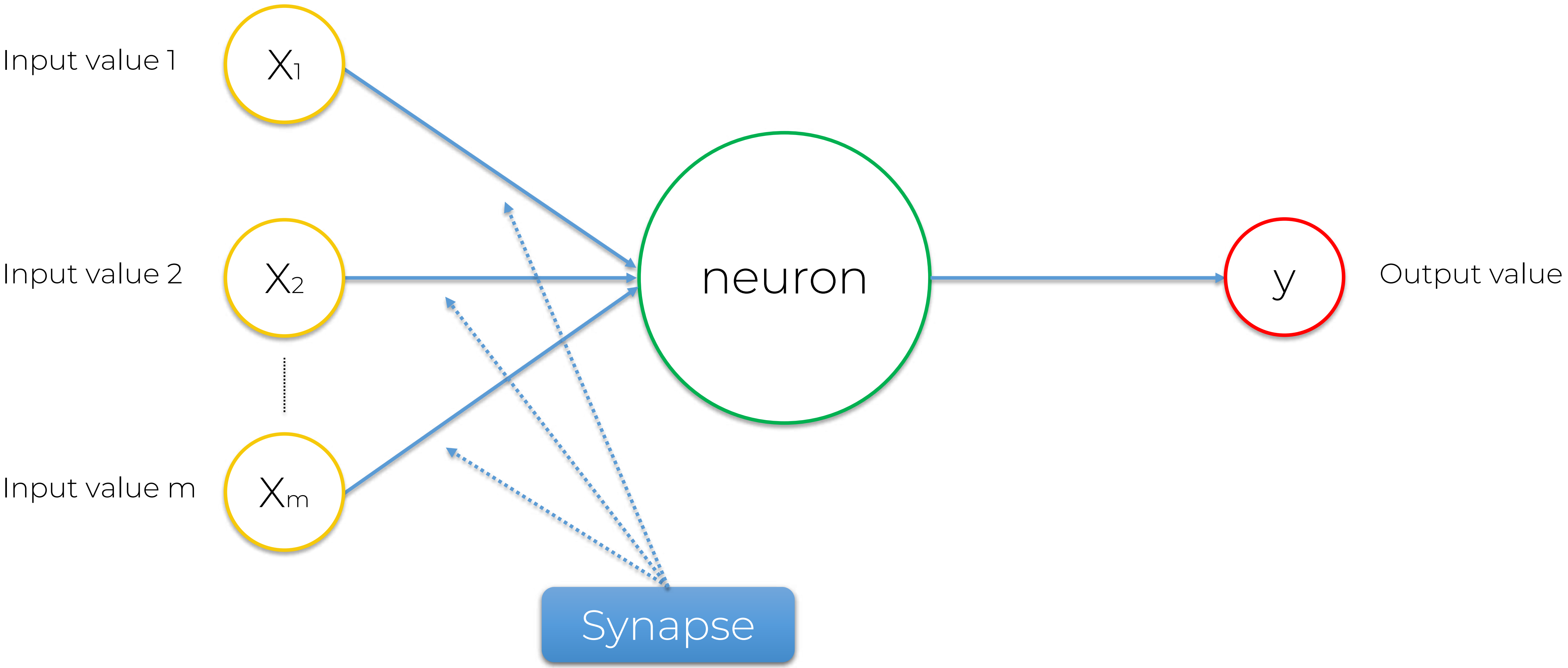


Input signal 1
Input signal 2
Input signal m
neuron
Output signal

The Neuron

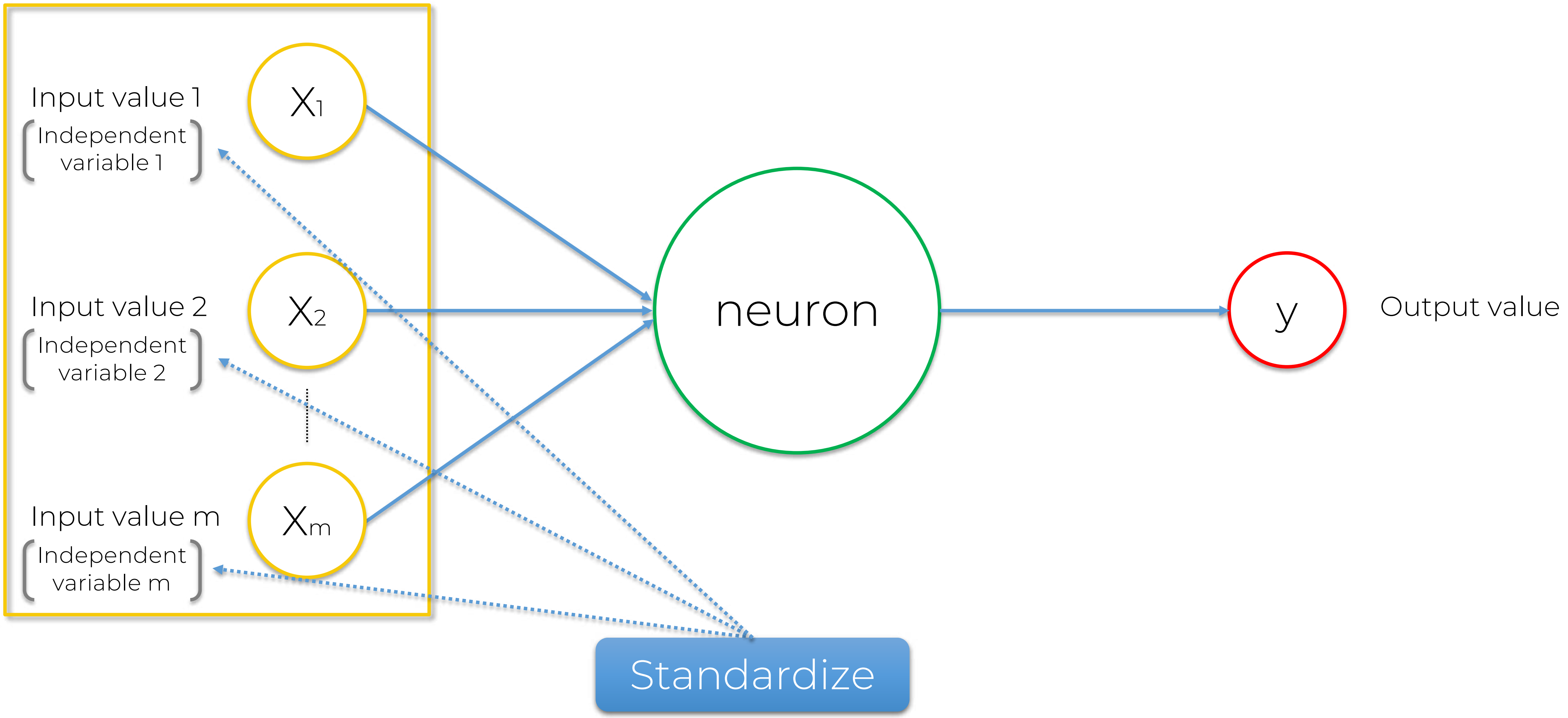

Input value 1
Input value 2
Input value m
X1
X2
Xm
neuron
Output signal
Synapse

The Neuron



Input value 1
Input value 2
Input value m
X1
X2
Xm
neuron
y
Output value
Synapse

The Neuron



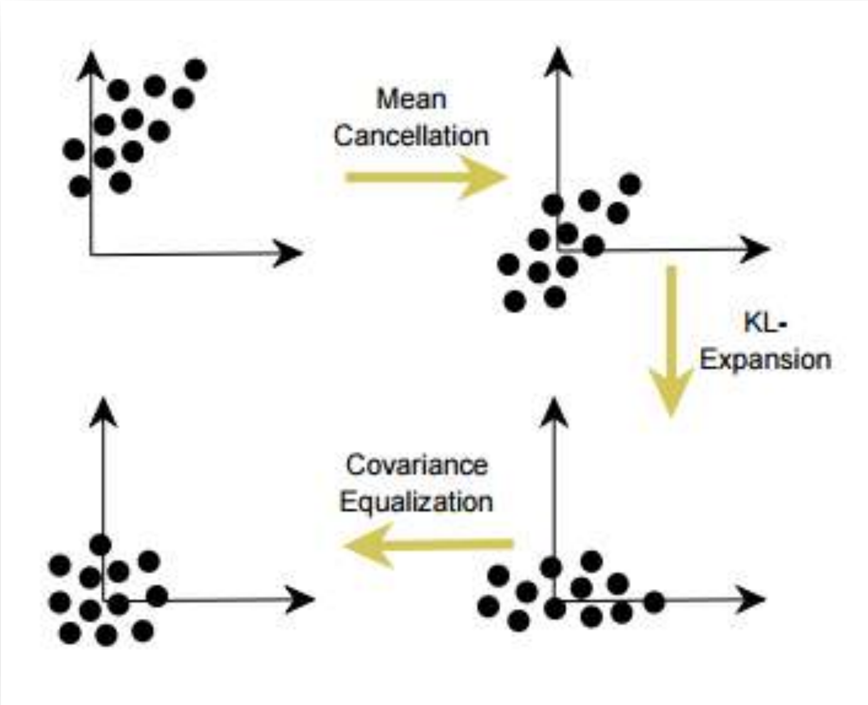
Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value
Standardize

The Neuron

Additional Reading:

Efficient BackProp

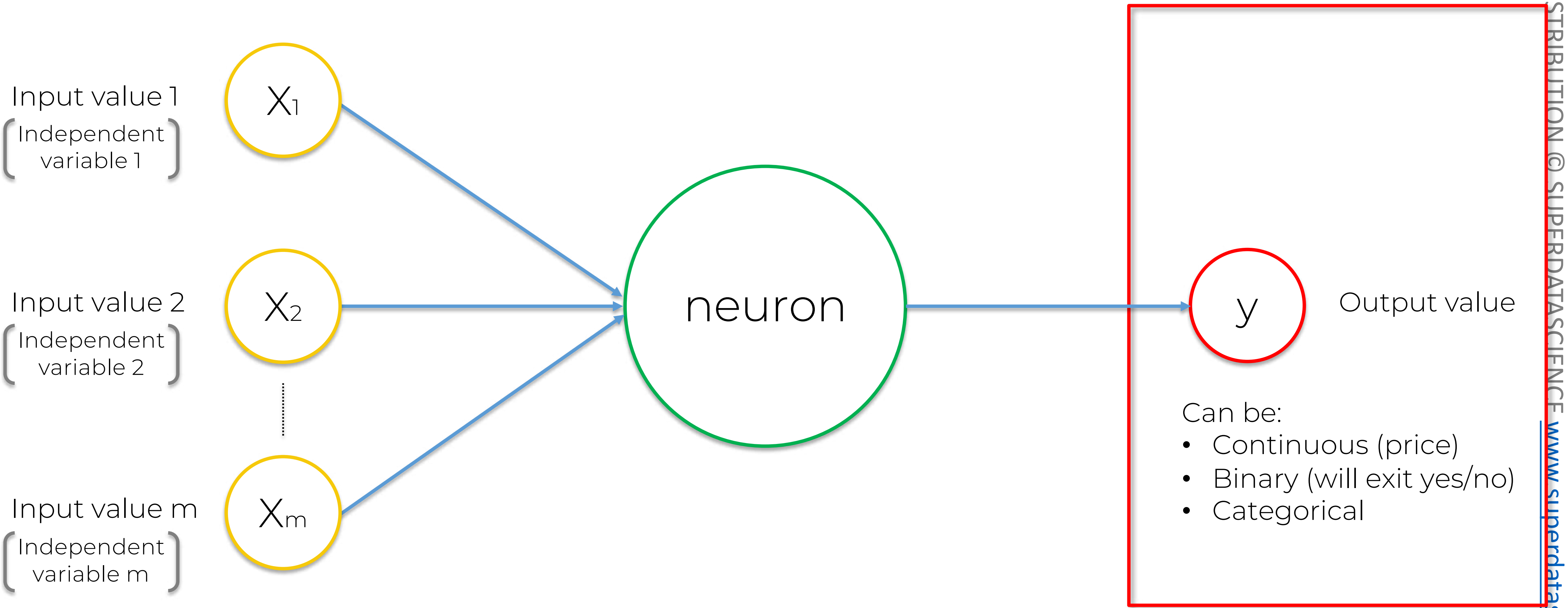
By Yann LeCun et al. (1998)



Link:

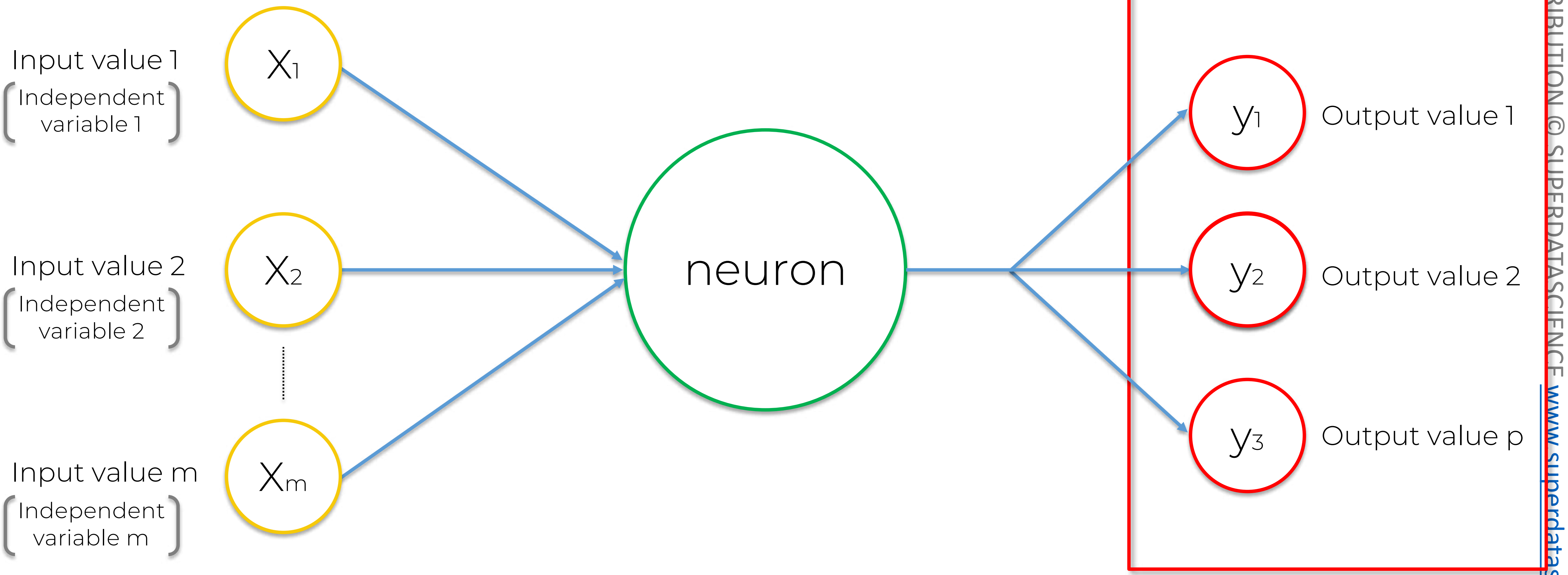
http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

The Neuron

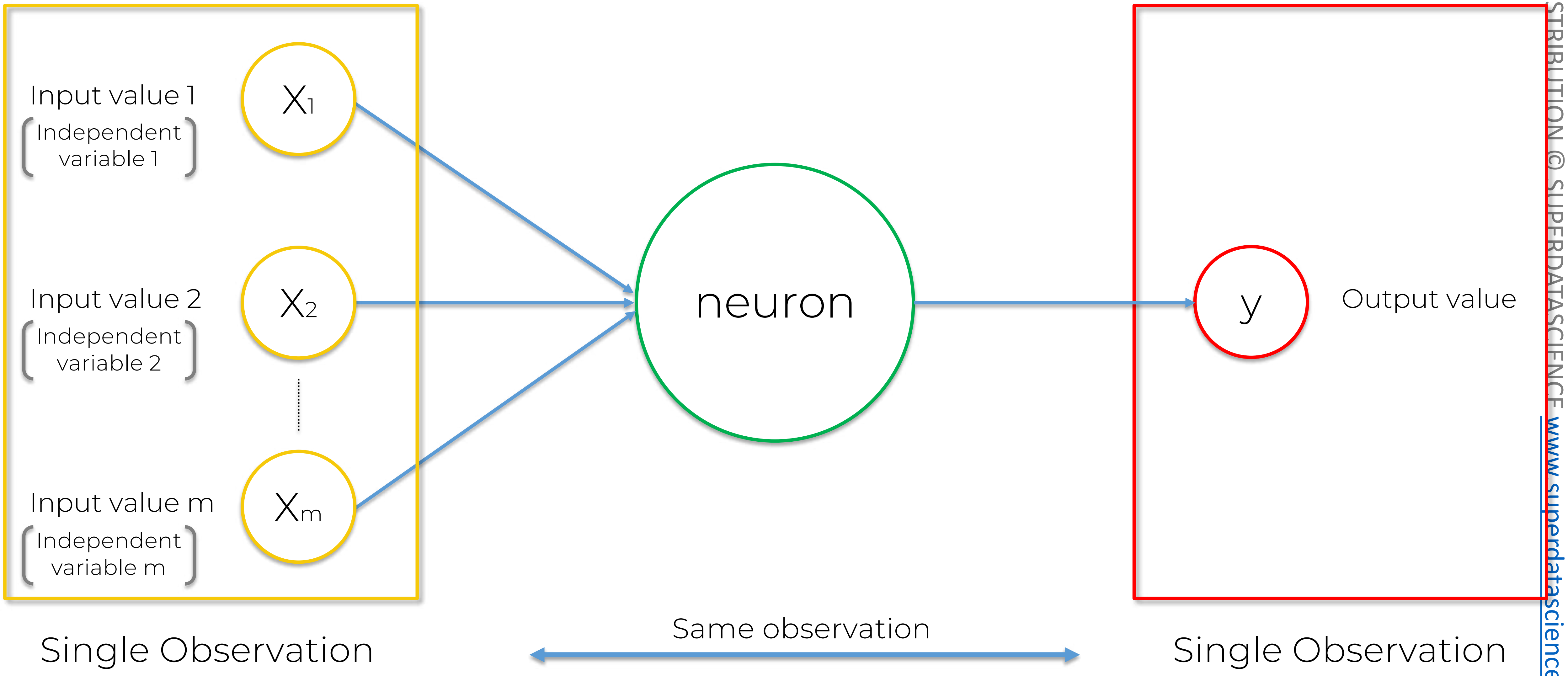


Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value
Can be:
• Continuous (price)
• Binary (will exit yes/no)
• Categorical

The Neuron

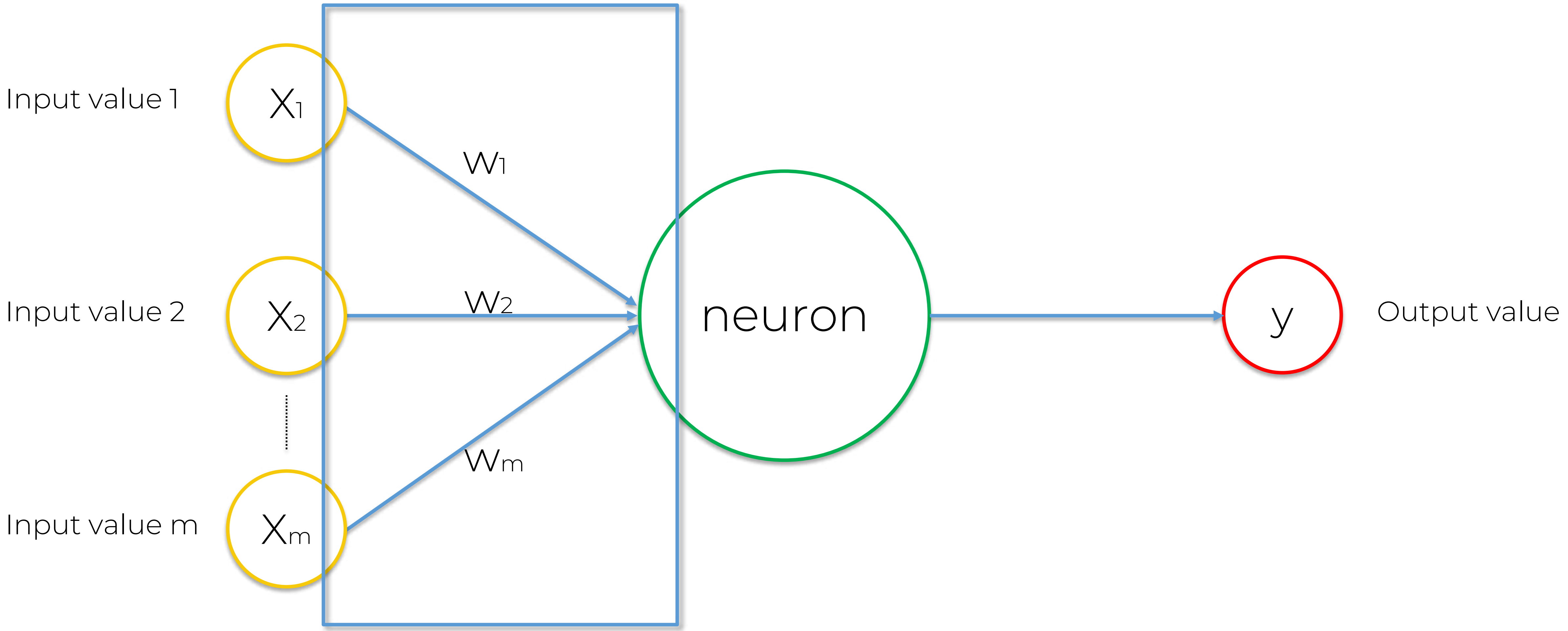

Input value 1
Independent variable 1
Input value 2
Independent variable 2
Input value m
Independent variable m
X1
X2
Xm
neuron
y1
y2
y3
Output value 1
Output value 2
Output value p

The Neuron

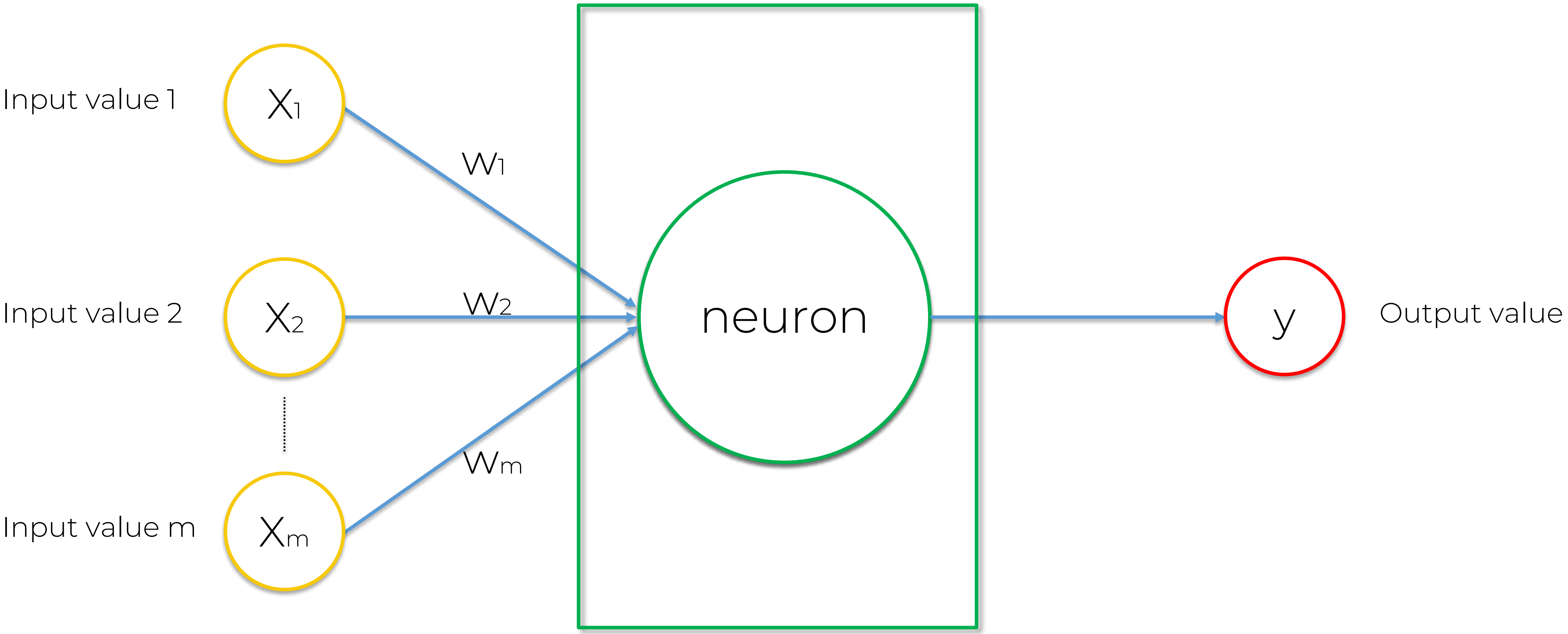


Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value
Single Observation
Same observation
Single Observation

The Neuron

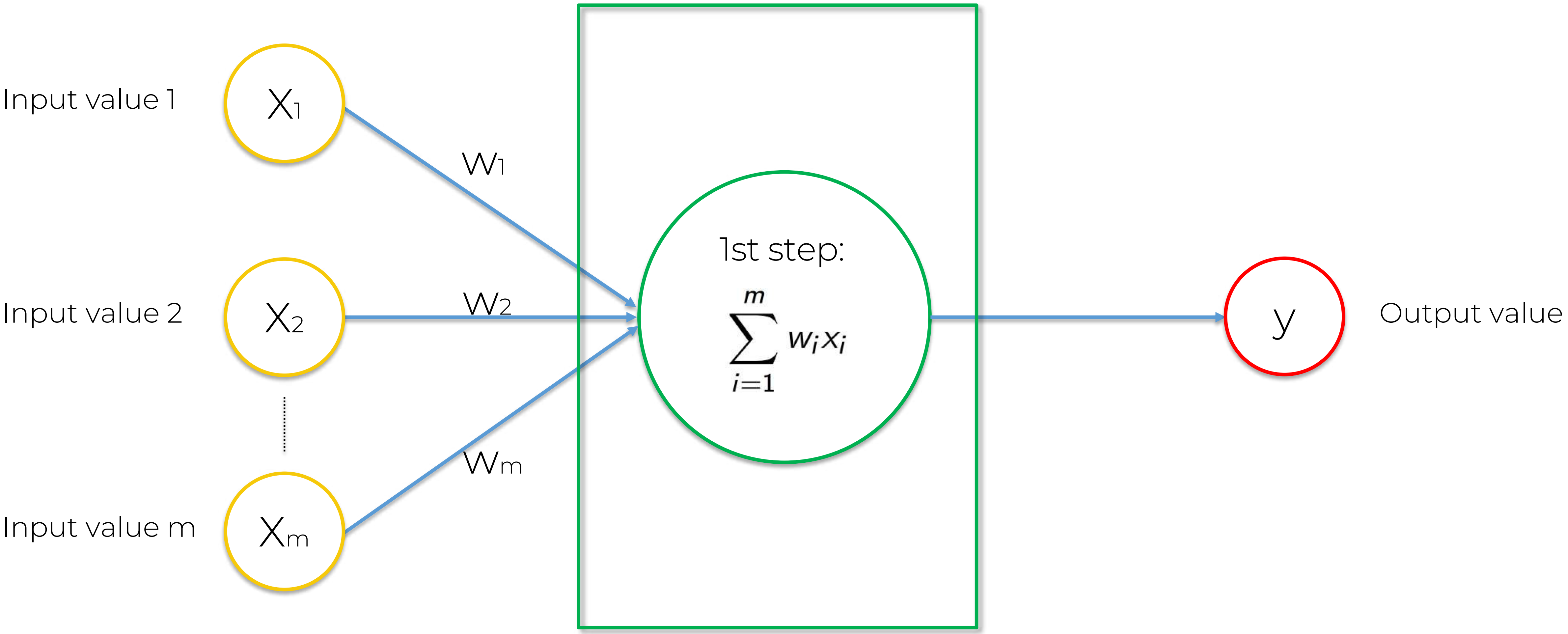

Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
W2
Wm
neuron
y
Output value

The Neuron



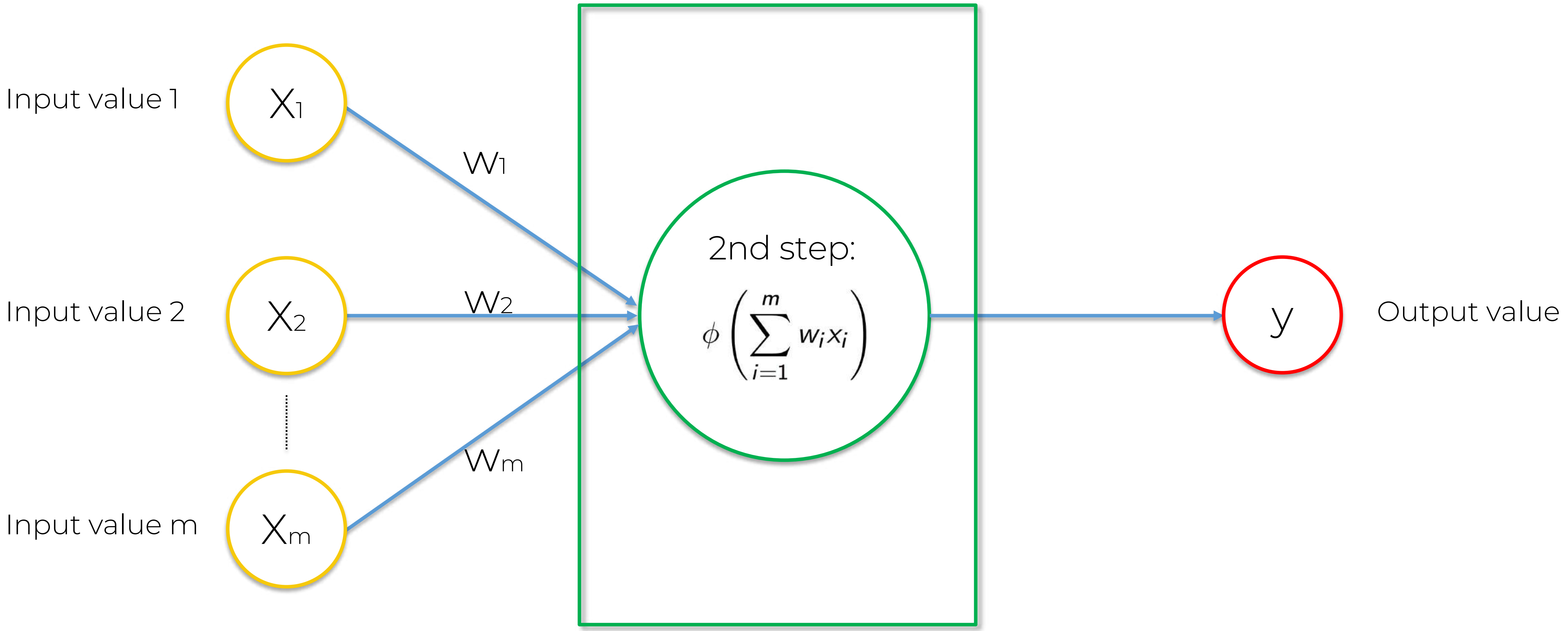
Input value 1
Input value 2
Input value m
X_1
X_2
X_m
W_1
W_2
W_m
neuron
y
Output value

The Neuron



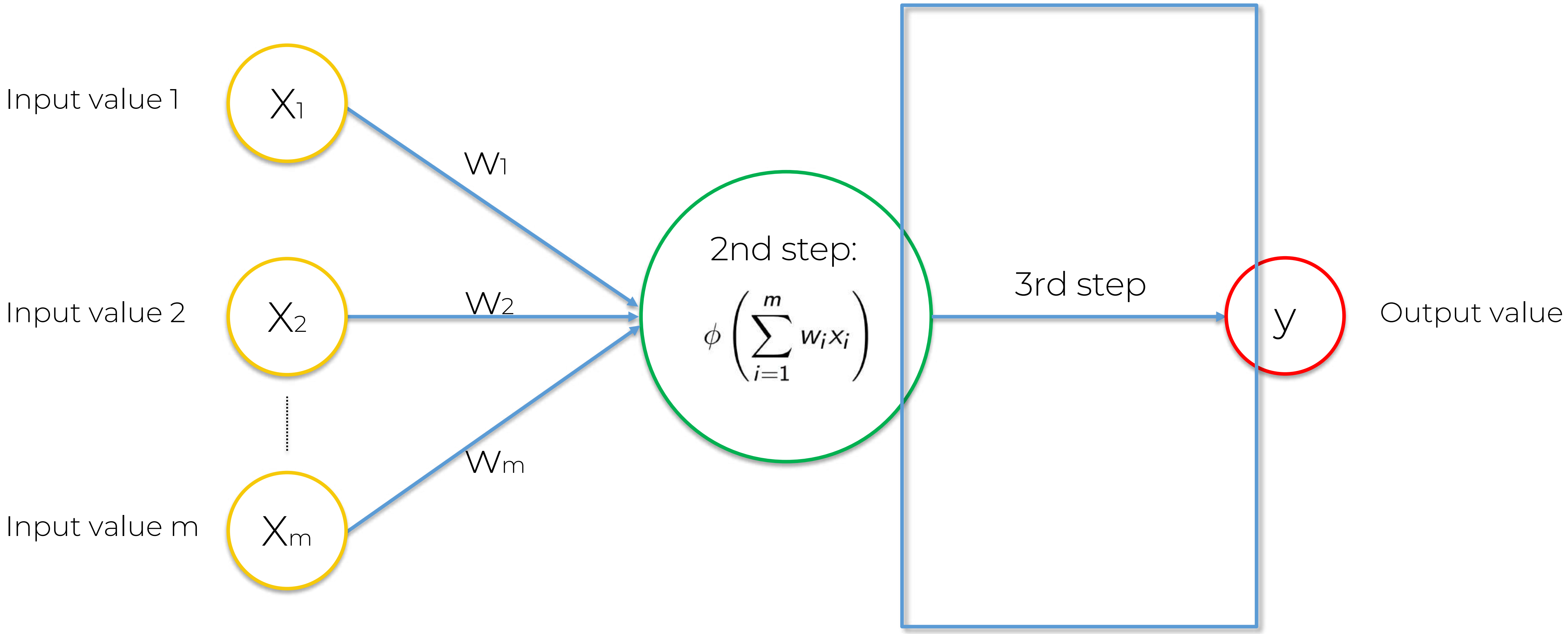
Input value 1
X_1
W_1
Input value 2
X_2
W_2
Input value m
X_m
W_m
1st step:
$\sum_{i=1}^{m} w_i x_i$
y
Output value

The Neuron



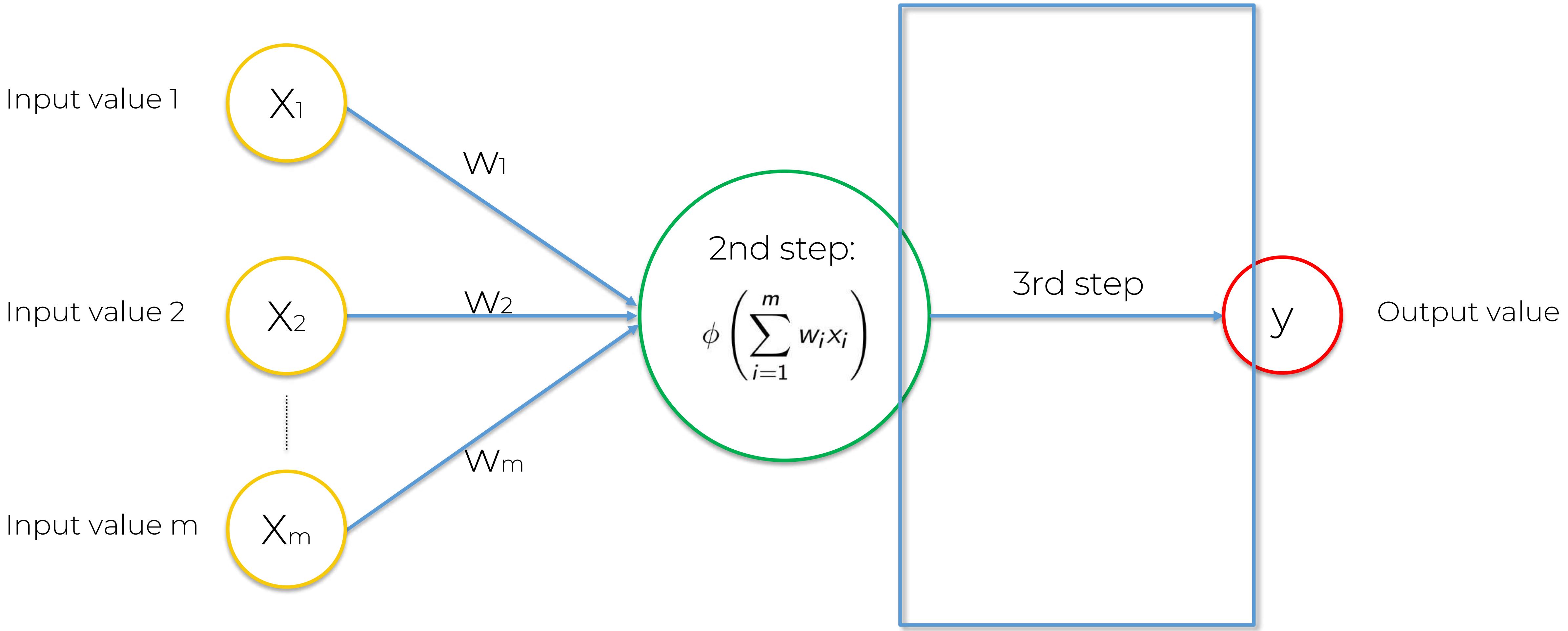
Input value 1
Input value 2
Input value m
X_1
X_2
X_m
W_1
W_2
W_m
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
y
Output value

The Neuron

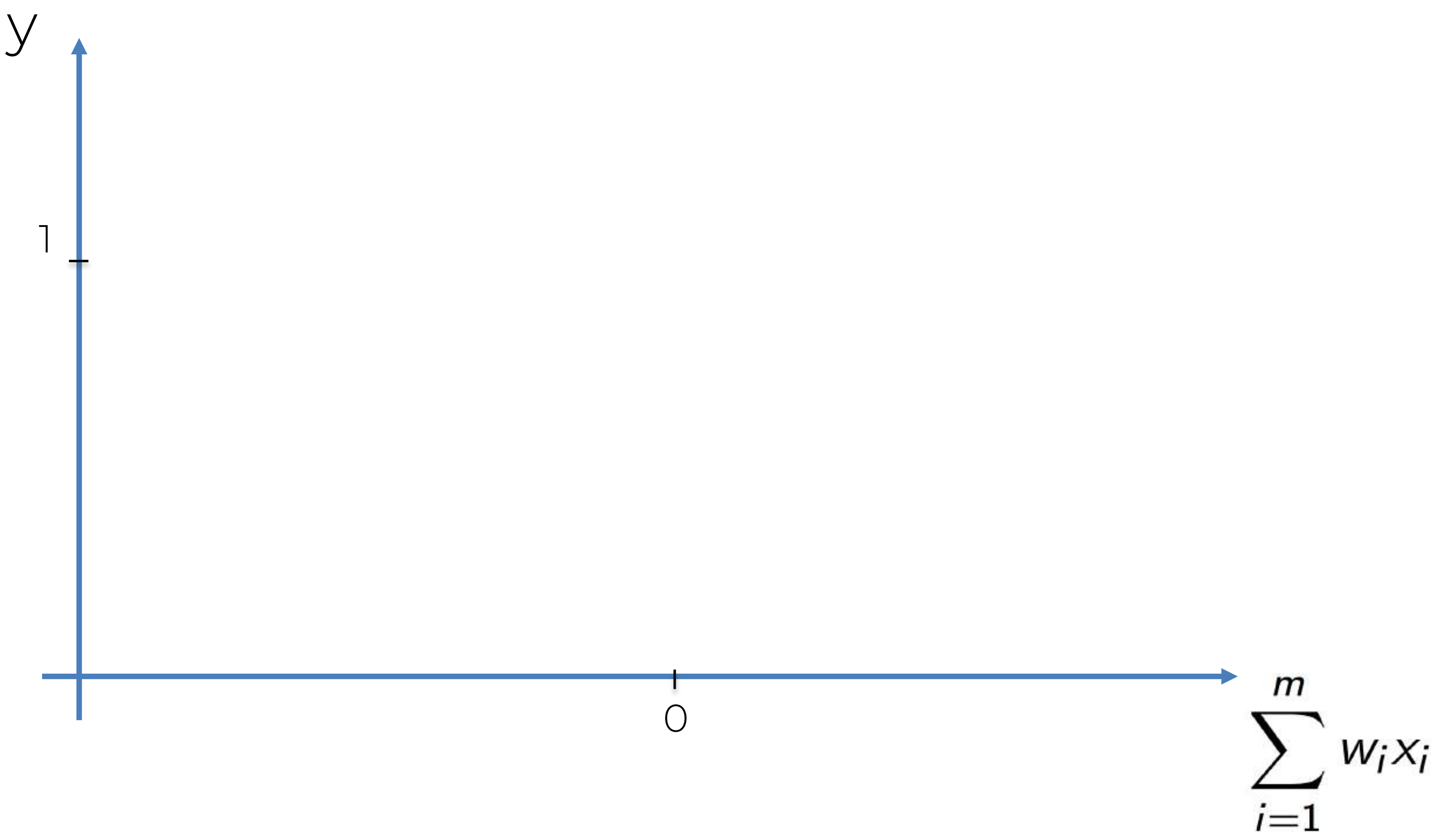

Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
W2
Wm
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value

The Activation Function

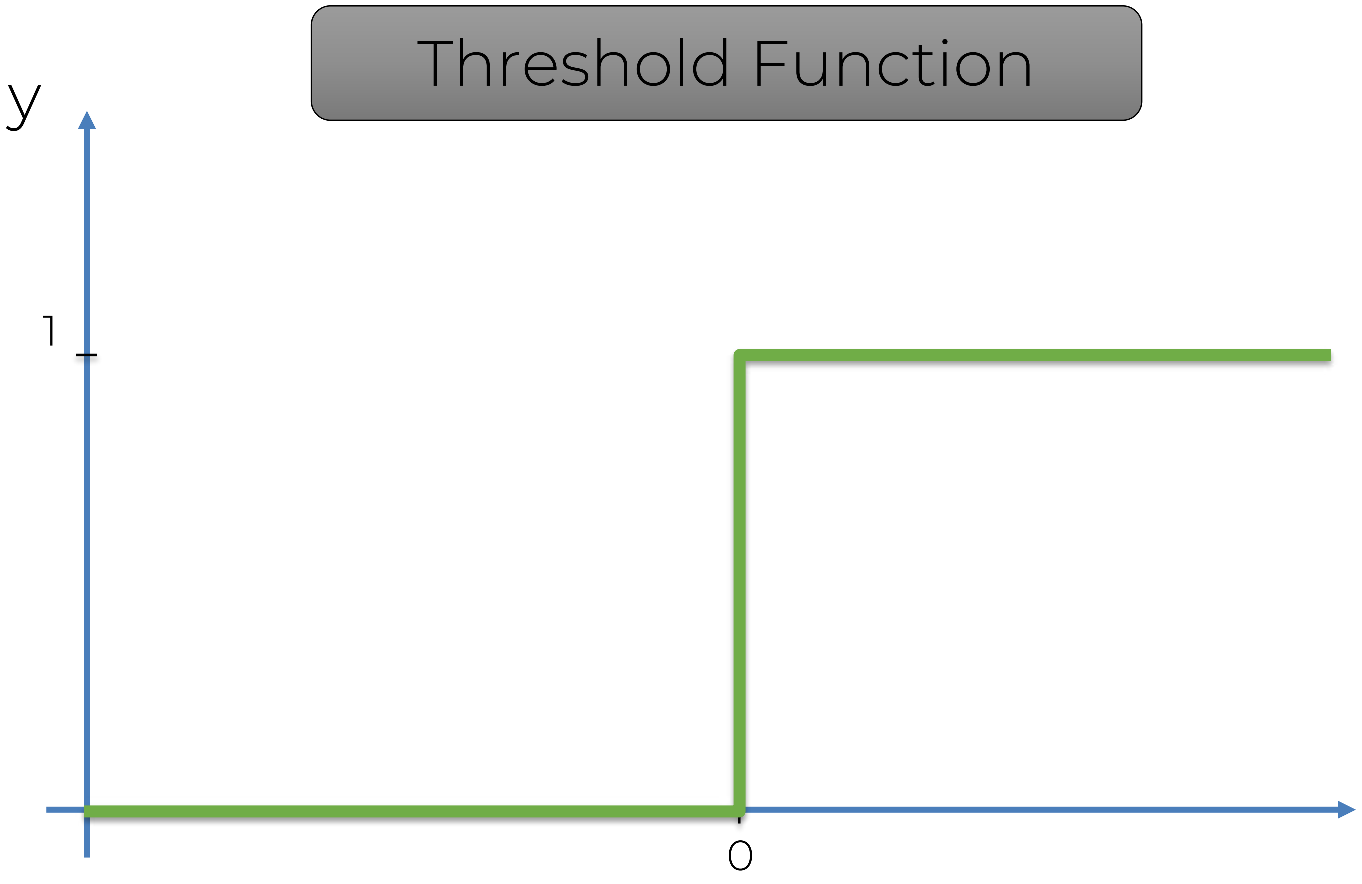
The Activation Function



The Activation Function



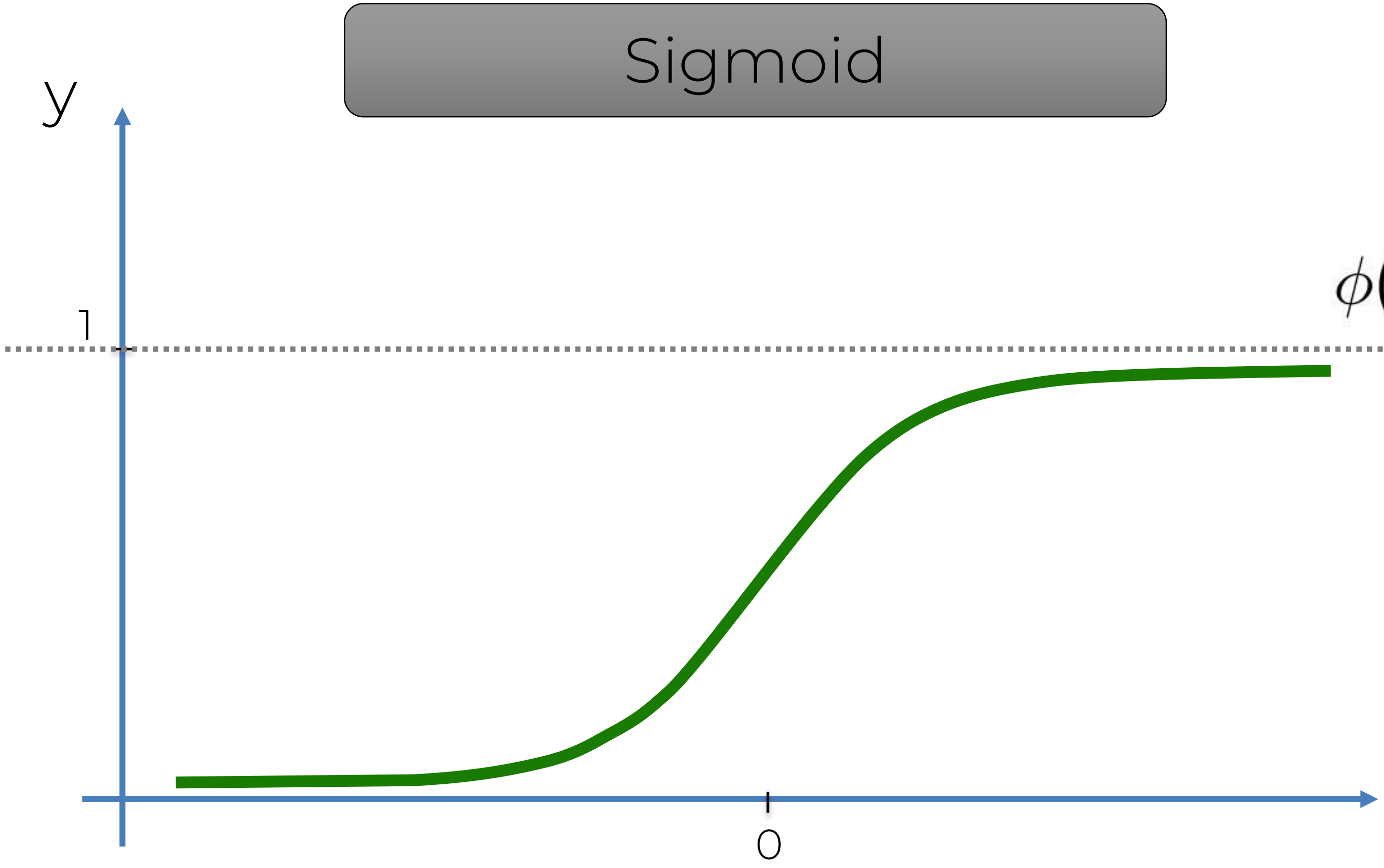
The Activation Function



$$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

$$\sum_{i=1}^{m} w_i x_i$$

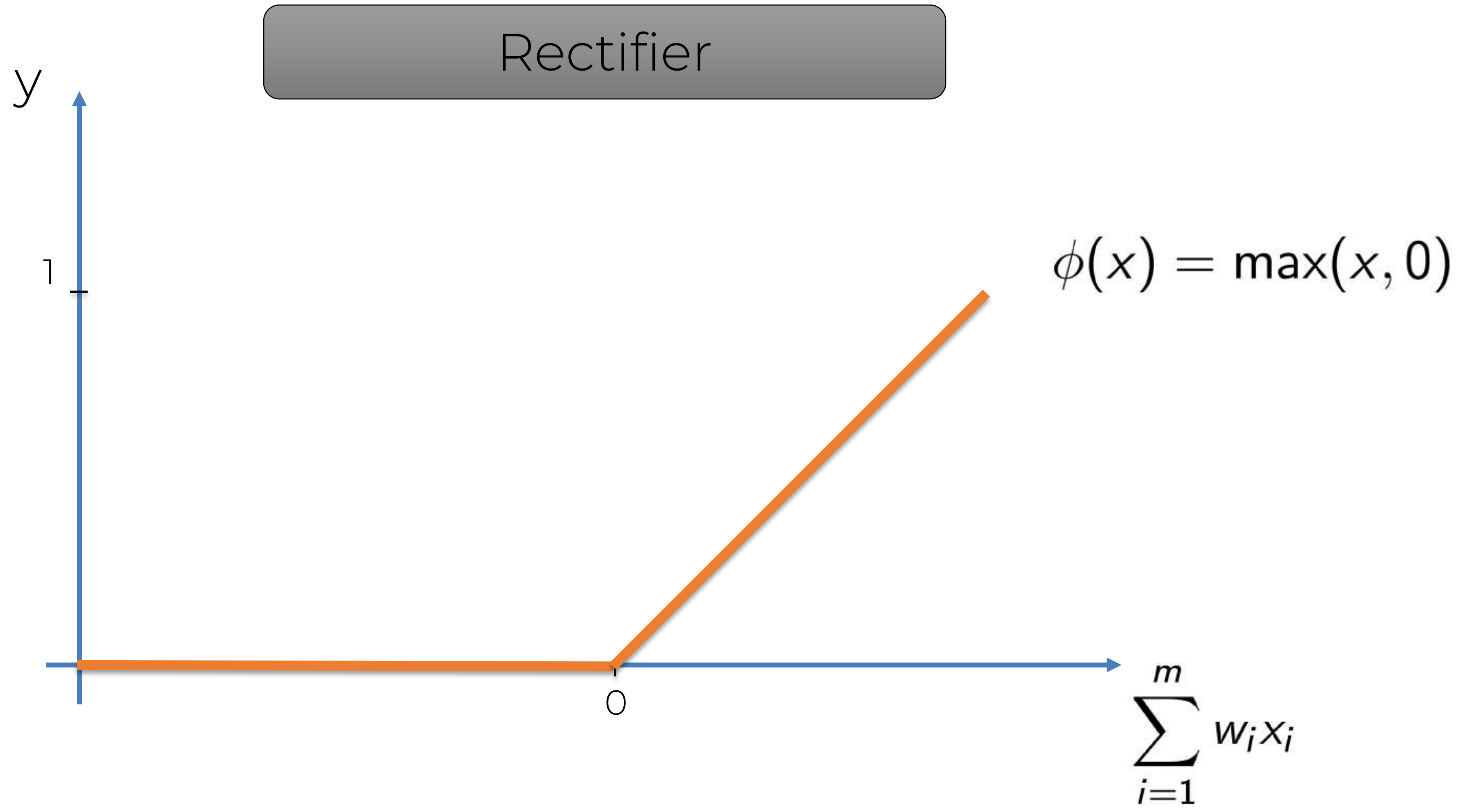
The Activation Function



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

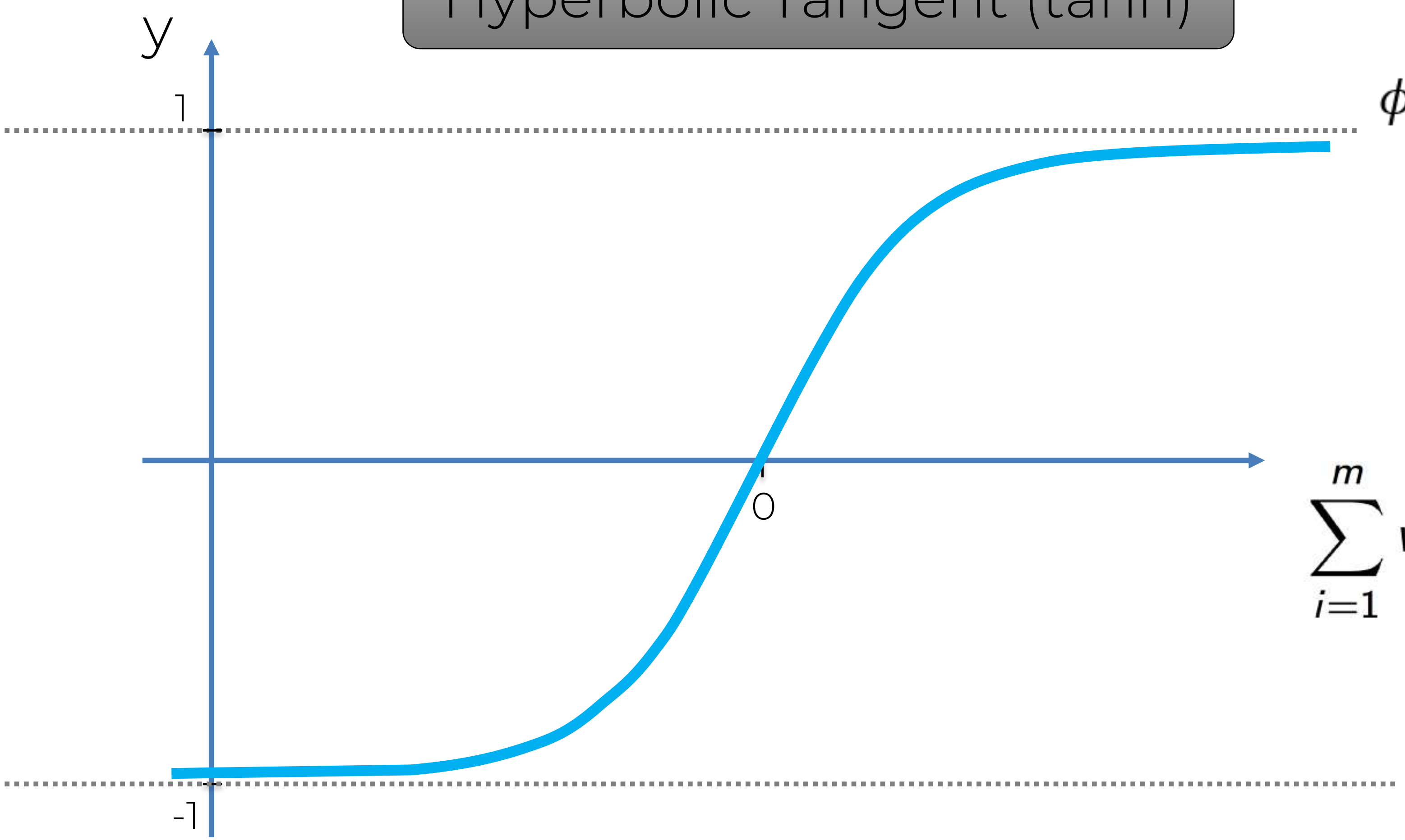
$$\sum_{i=1}^{m} w_i x_i$$

The Activation Function



The Activation Function

Hyperbolic Tangent (tanh)



$$\phi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

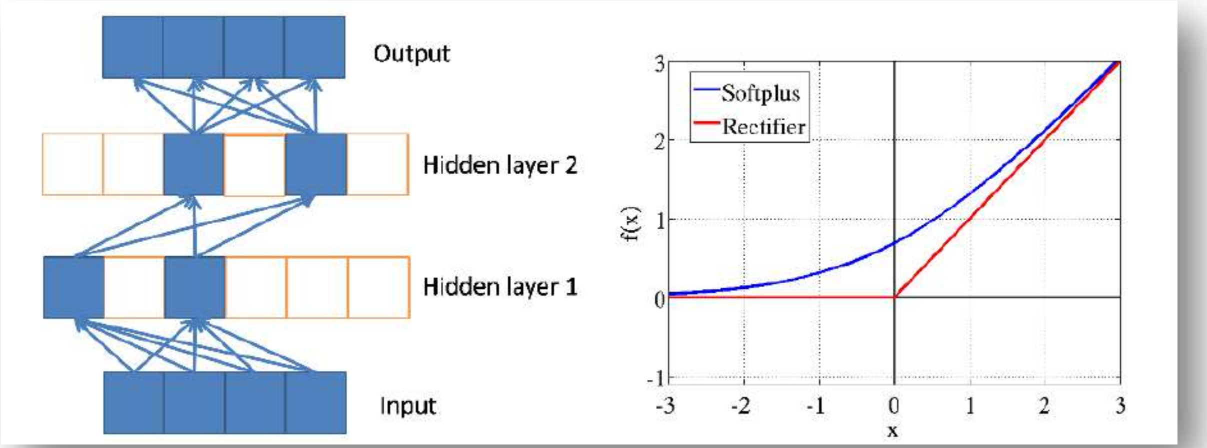
$$\sum_{i=1}^{m} w_i x_i$$

The Activation Function

Additional Reading:

Deep sparse rectifier neural networks

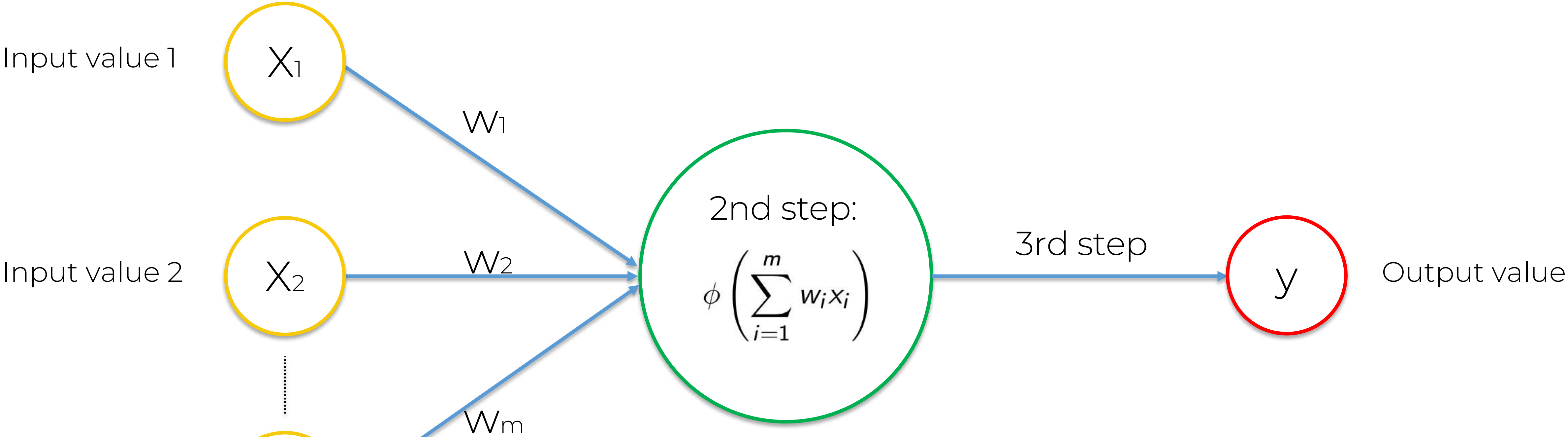
By Xavier Glorot et al. (2011)



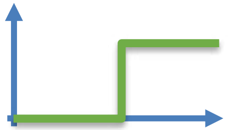
Link:

http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

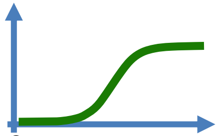
The Activation Function



Assuming the DV is binary (y = 0 or 1)

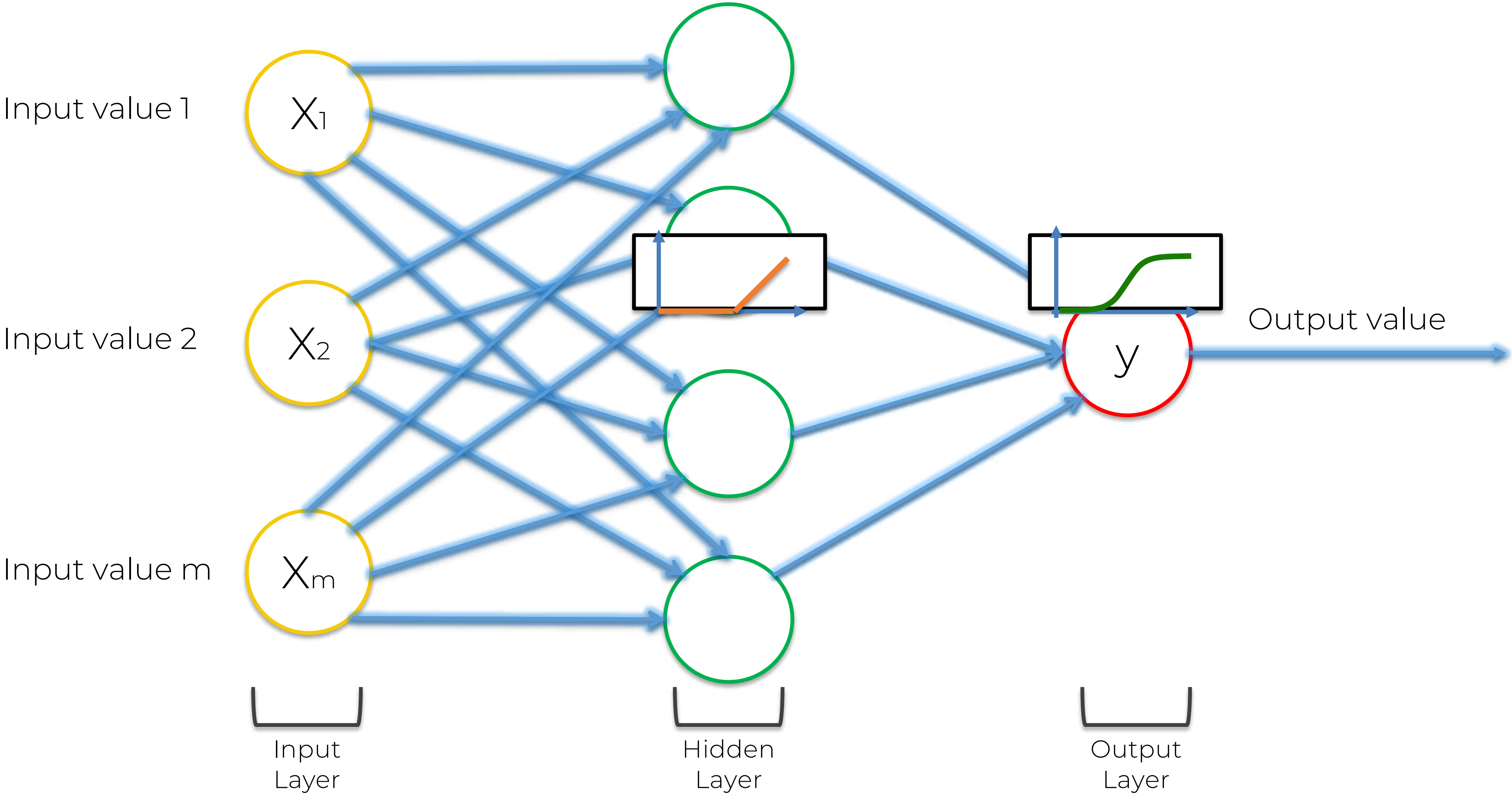


If threshold activation function: $y = \phi\left(\sum_{i=1}^{m} w_i x_i\right)$



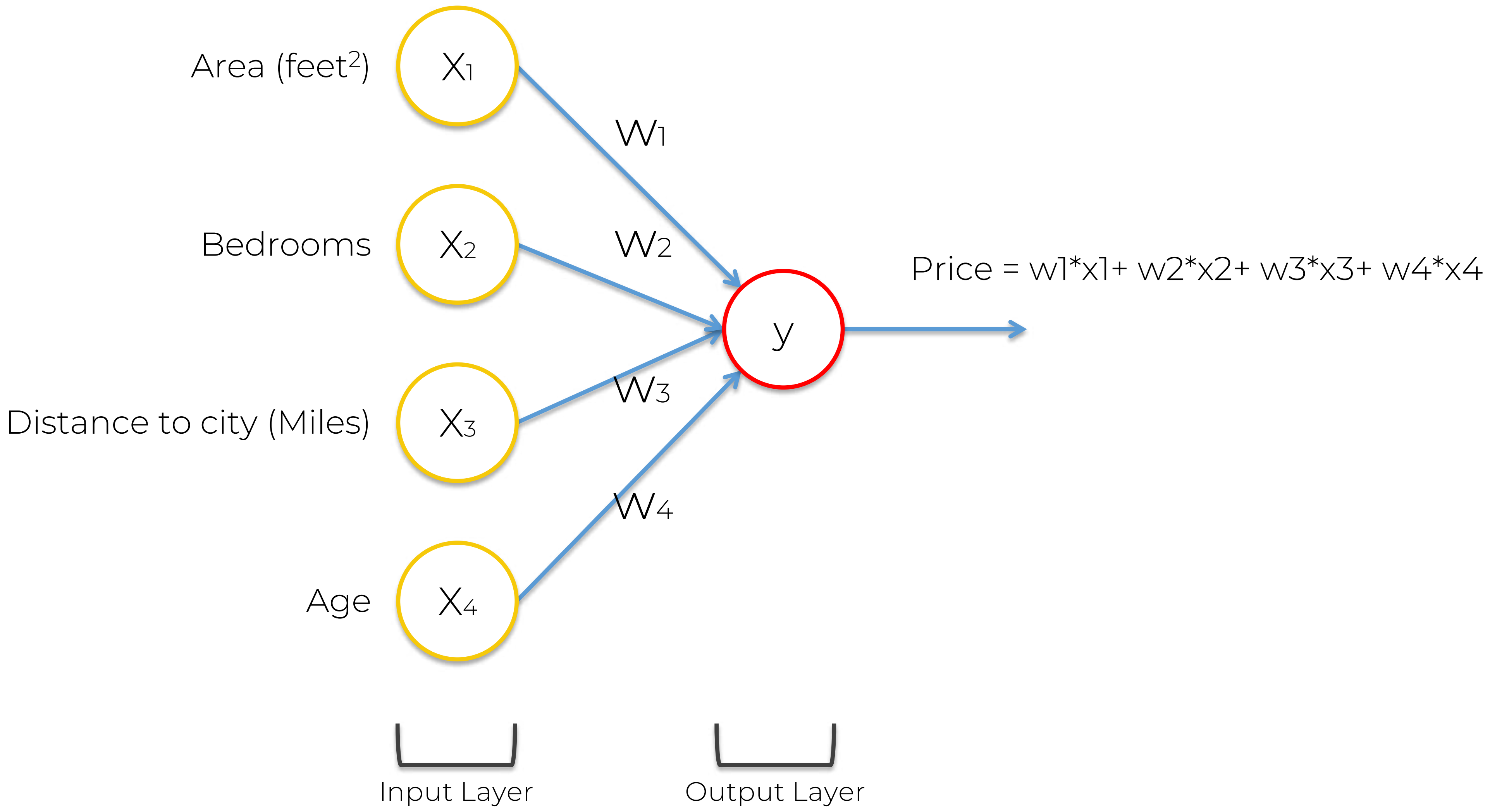
If sigmoid activation function: $\mathbb{P}(y = 1) = \phi\left(\sum_{i=1}^{m} w_i x_i\right)$

The Activation Function



How do NNs Work?

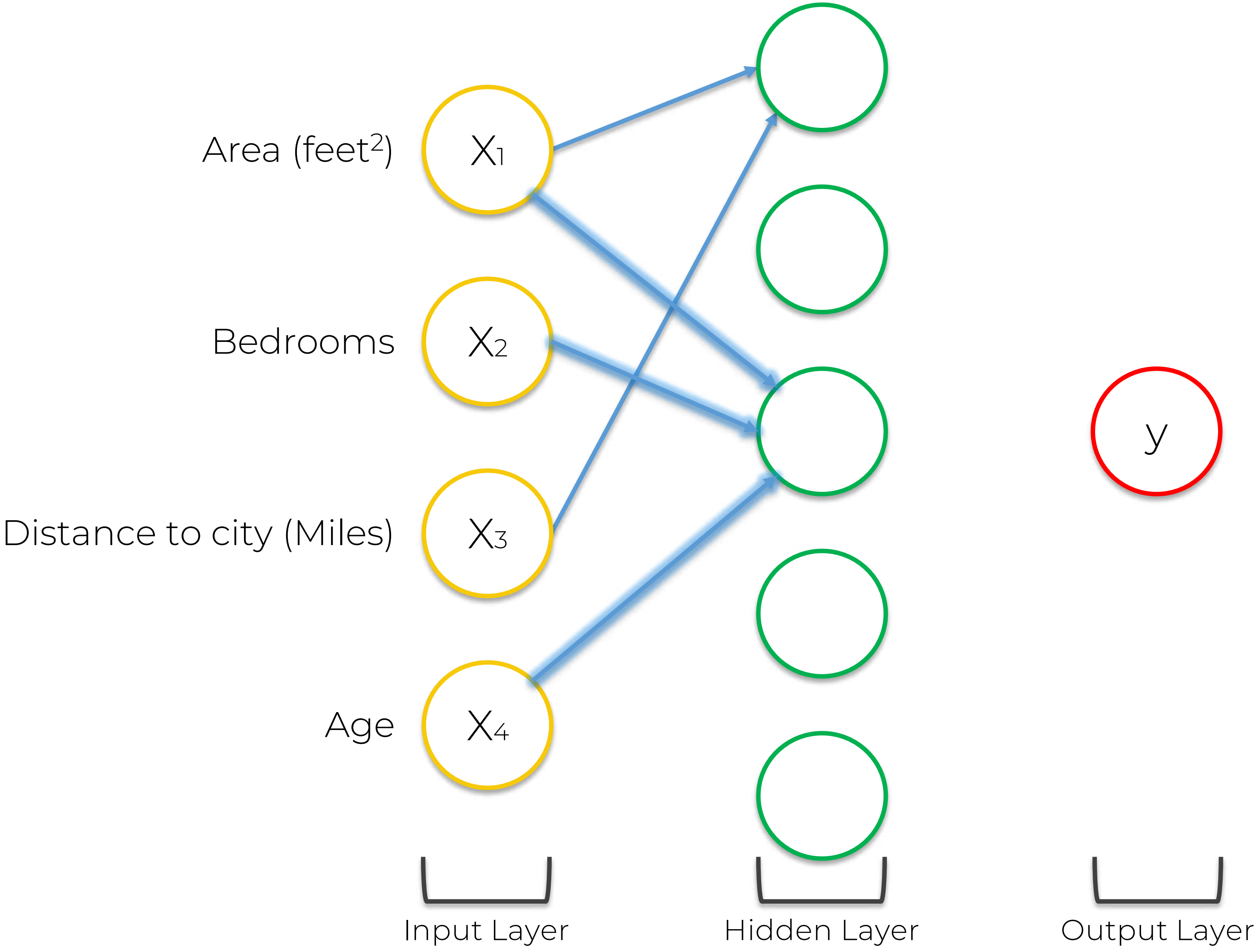
How Do Neural Networks Work?



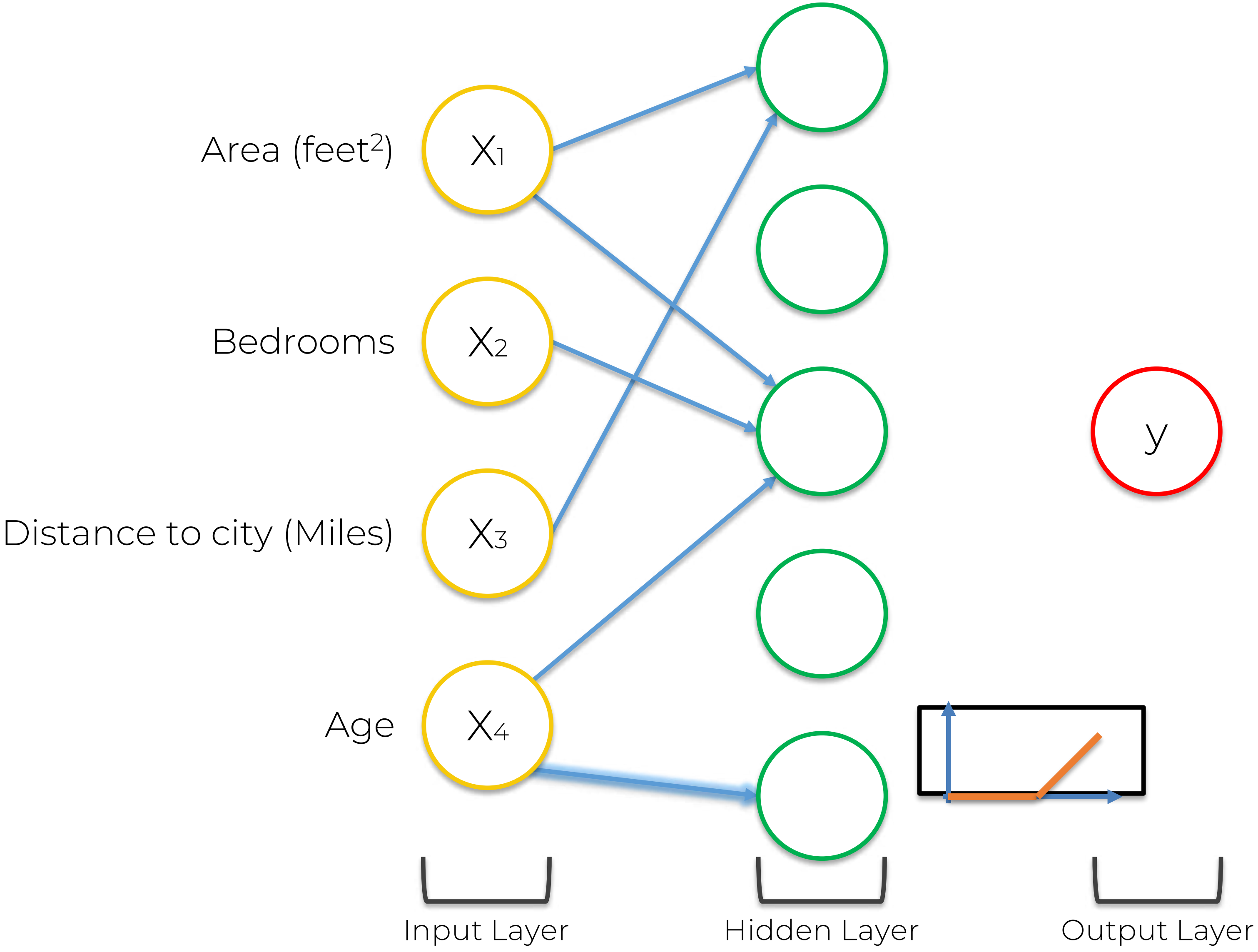
How Do Neural Networks Work?

How Do Neural Networks Work?

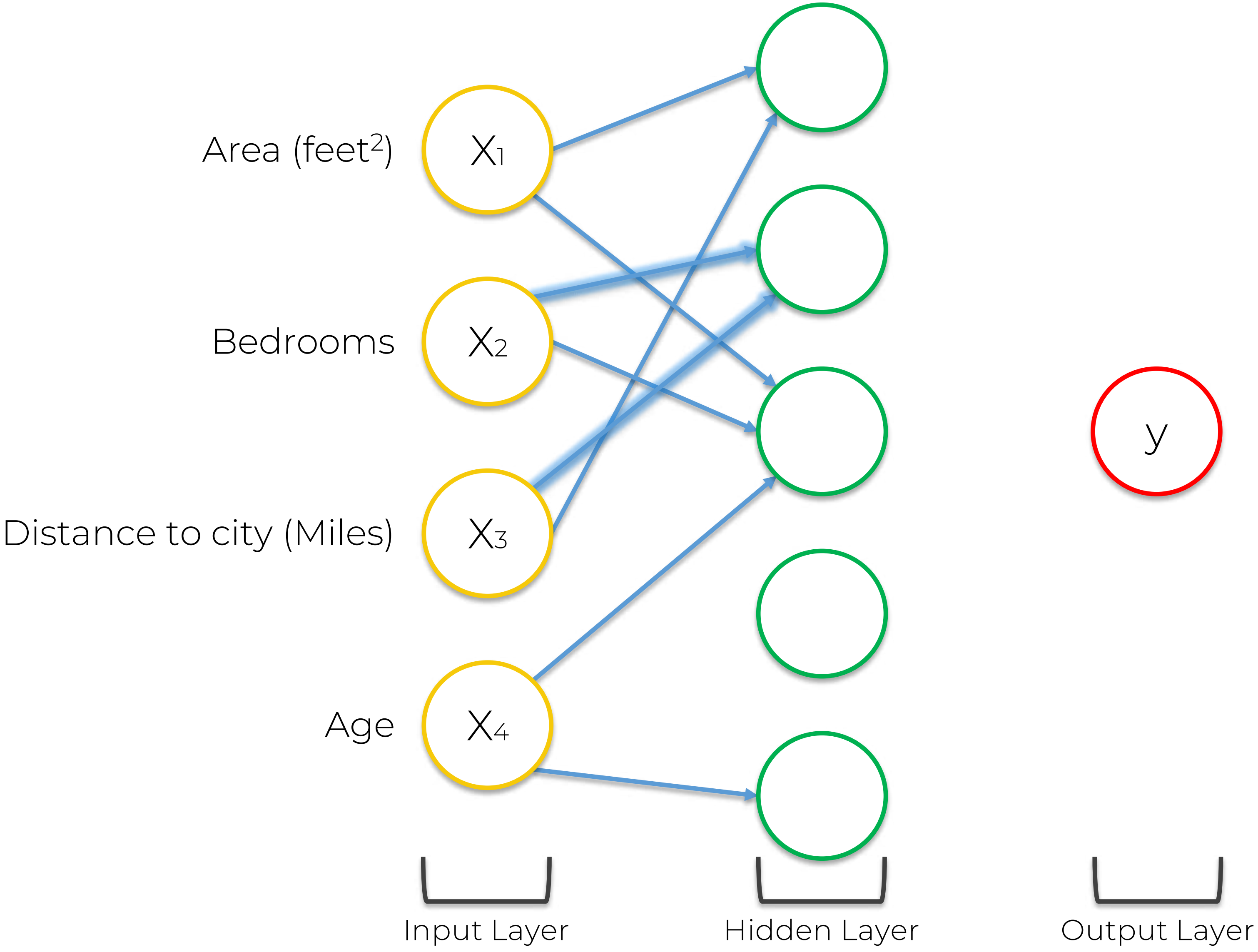
How Do Neural Networks Work?



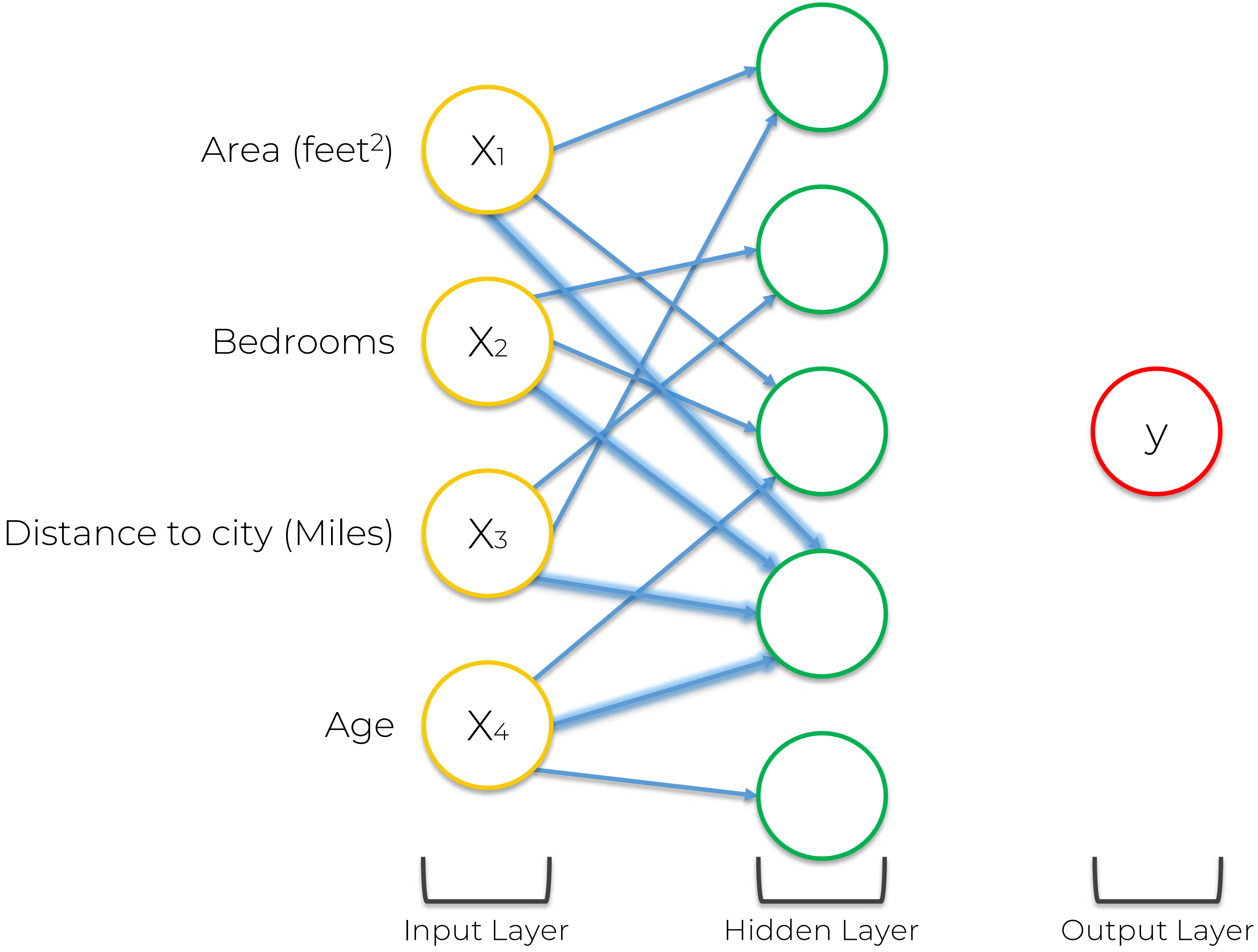
How Do Neural Networks Work?



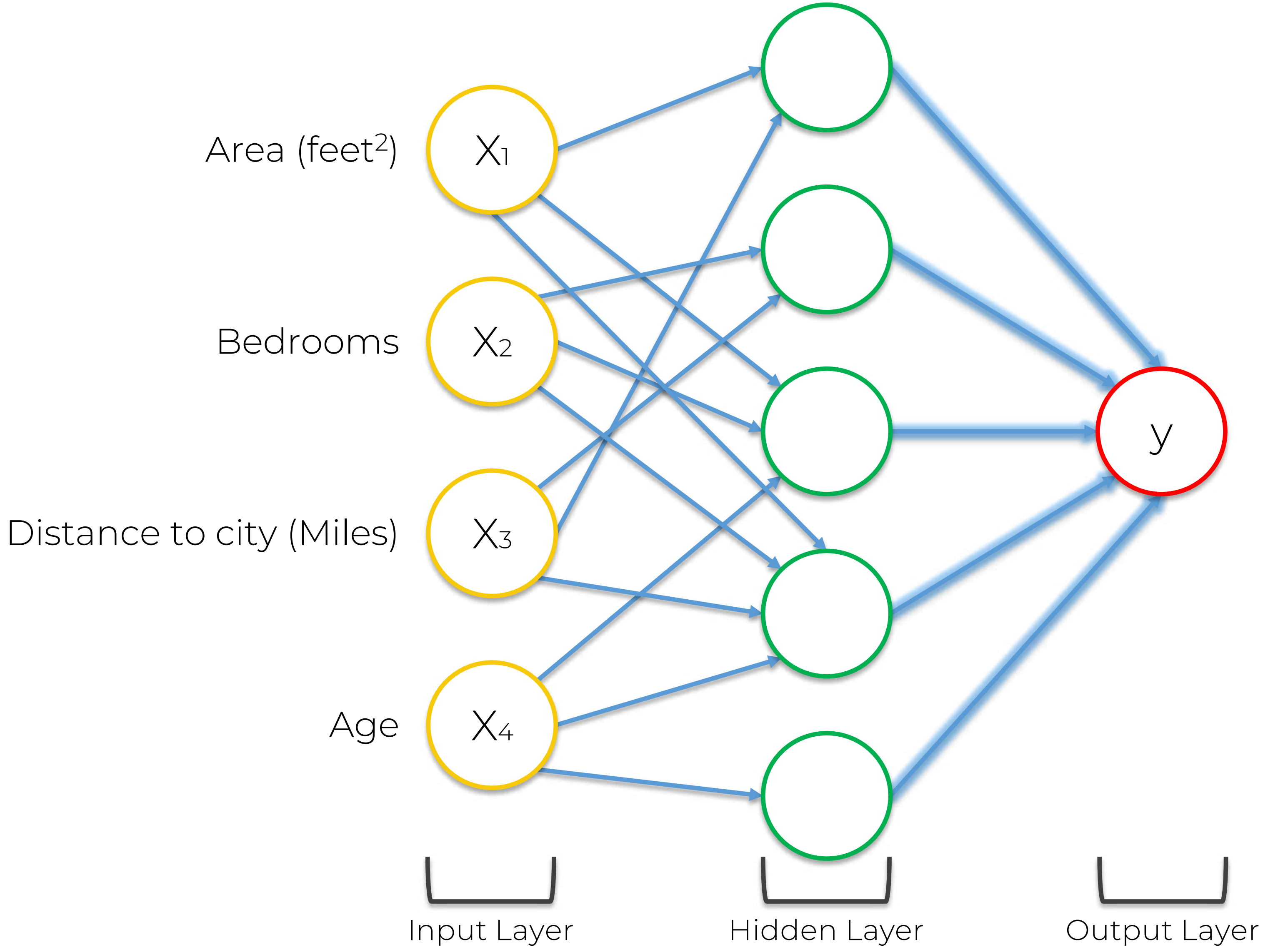
How Do Neural Networks Work?



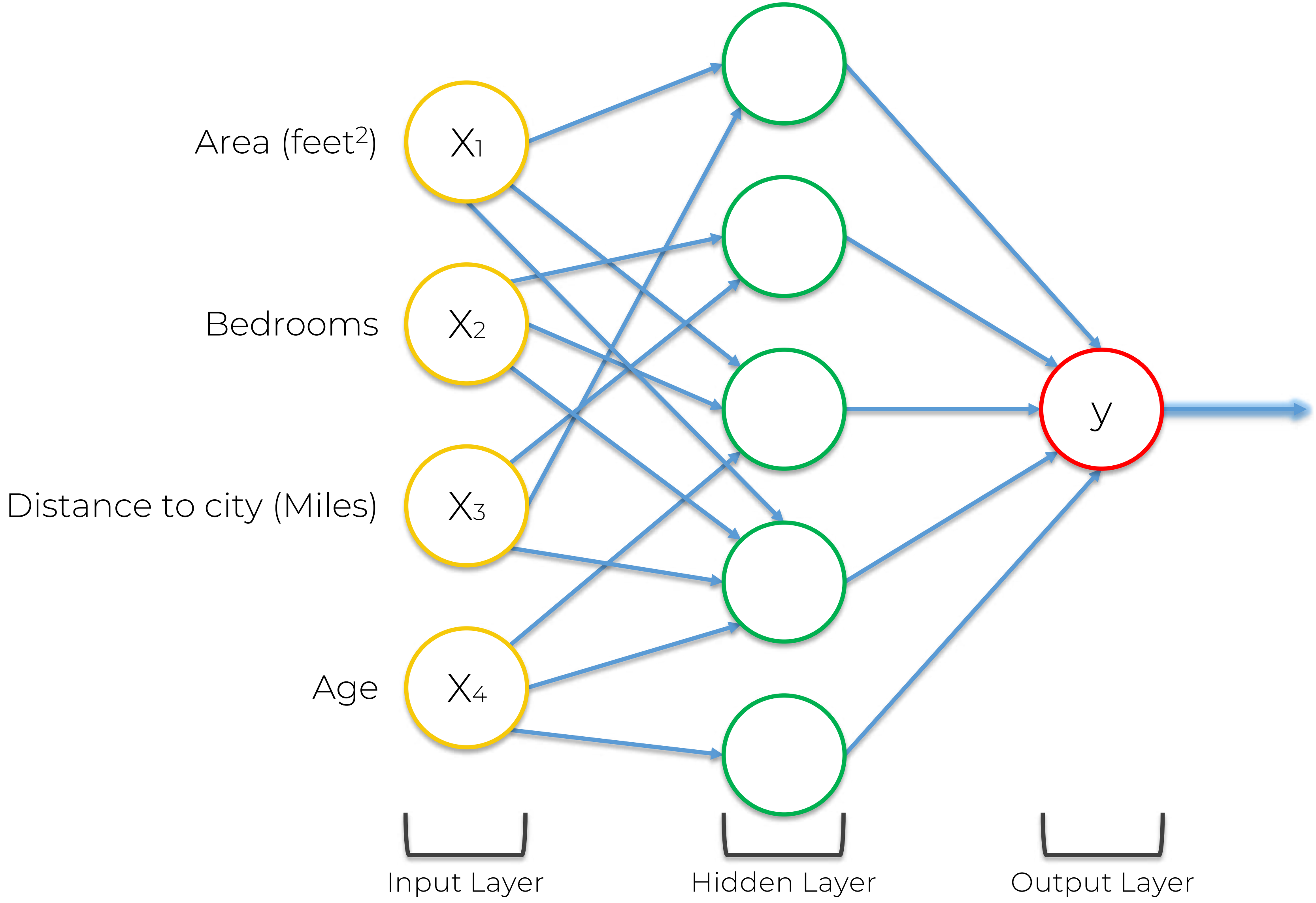
How Do Neural Networks Work?



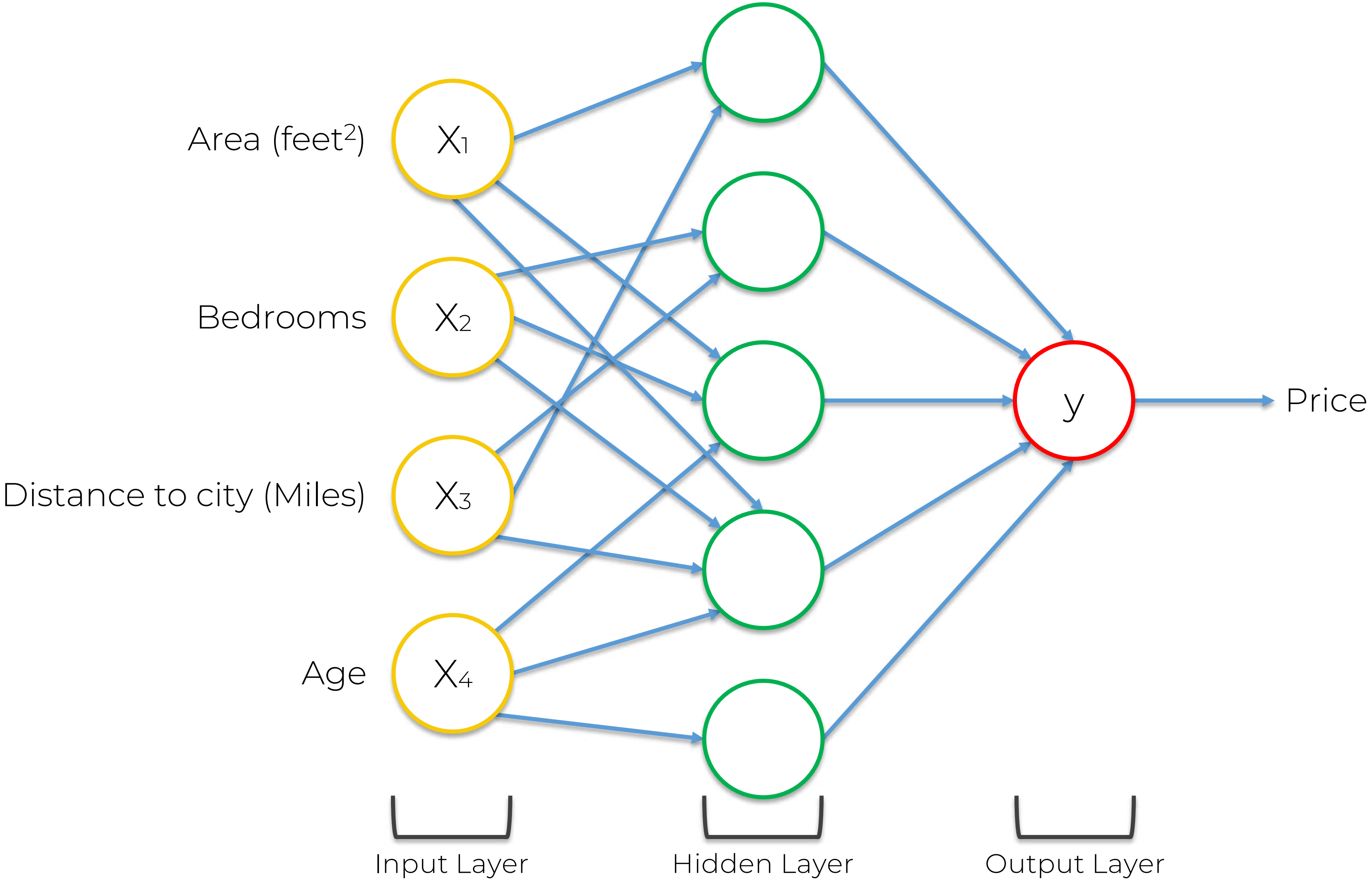
How Do Neural Networks Work?



How Do Neural Networks Work?



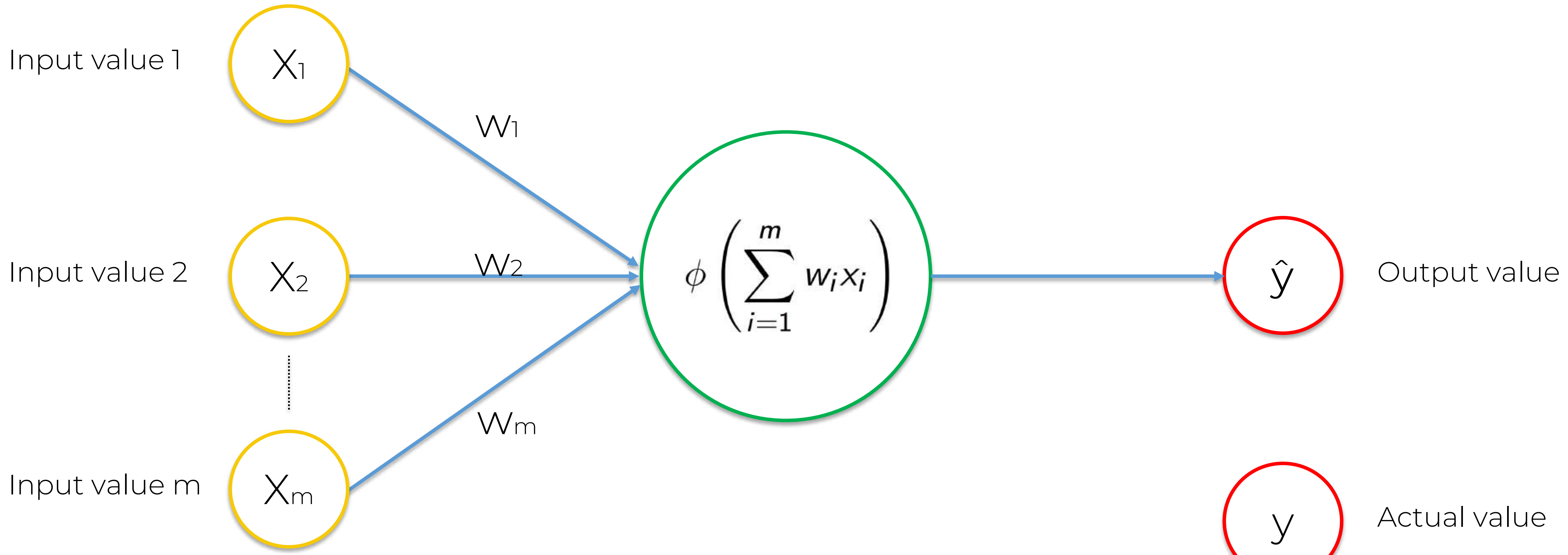
How Do Neural Networks Work?



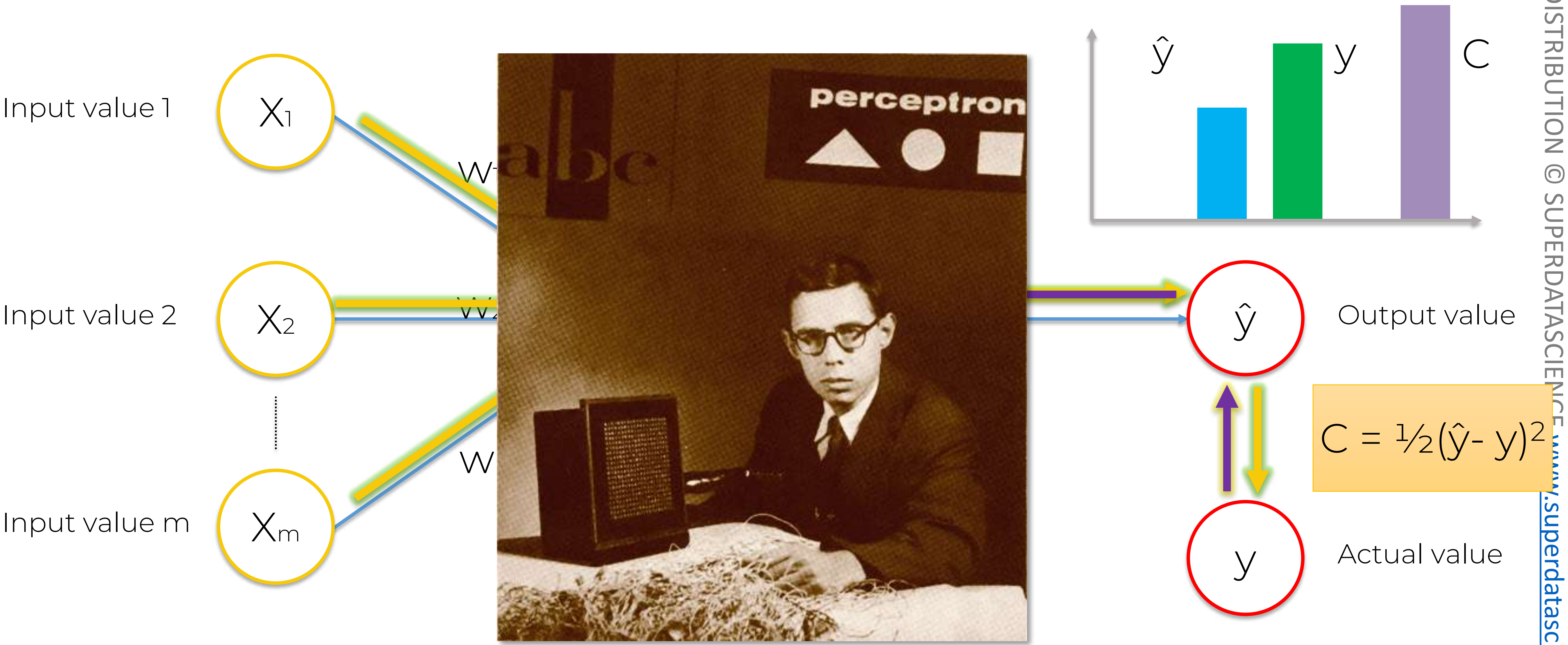
How do NNs Learn?

How do Neural Networks learn?

How do Neural Networks learn?

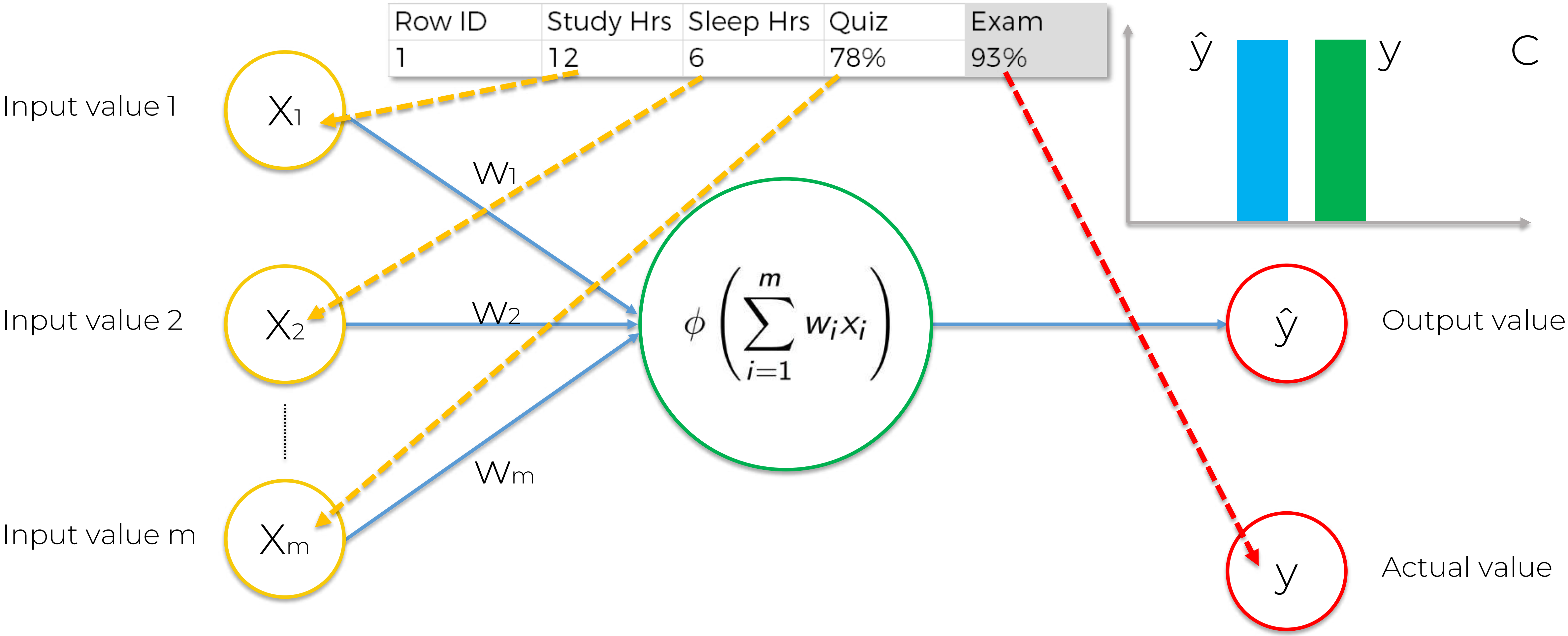


How do Neural Networks learn?



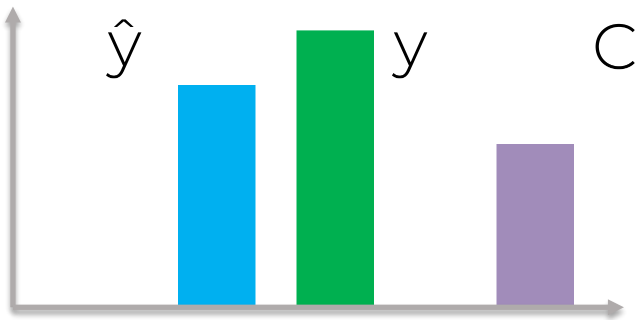
How do Neural Networks learn?

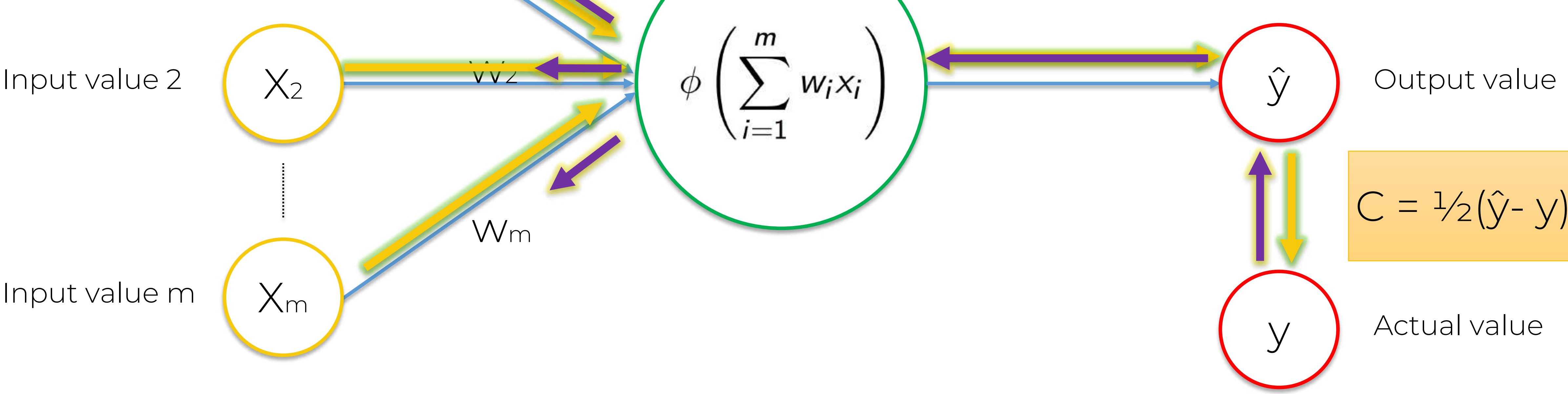
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

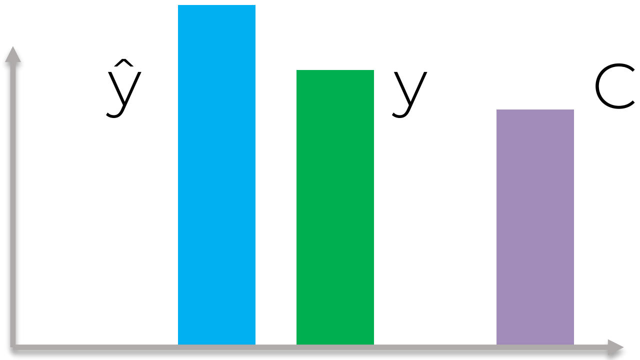


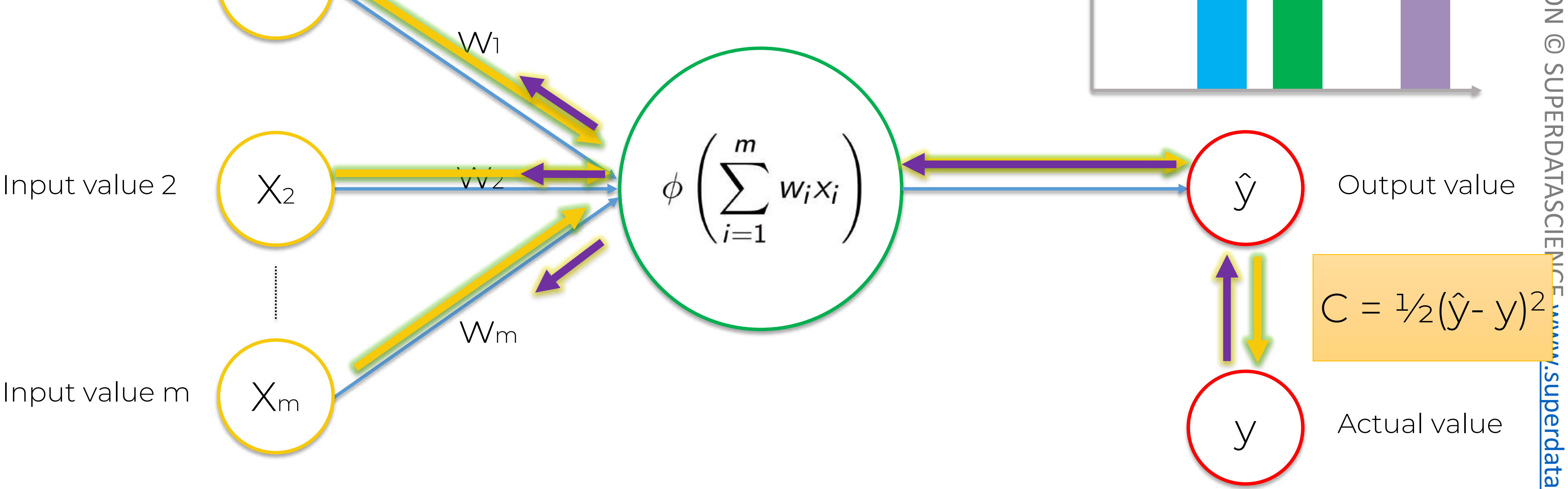


$C = \frac{1}{2}(\hat{y} - y)^2$

How do Neural Networks learn?

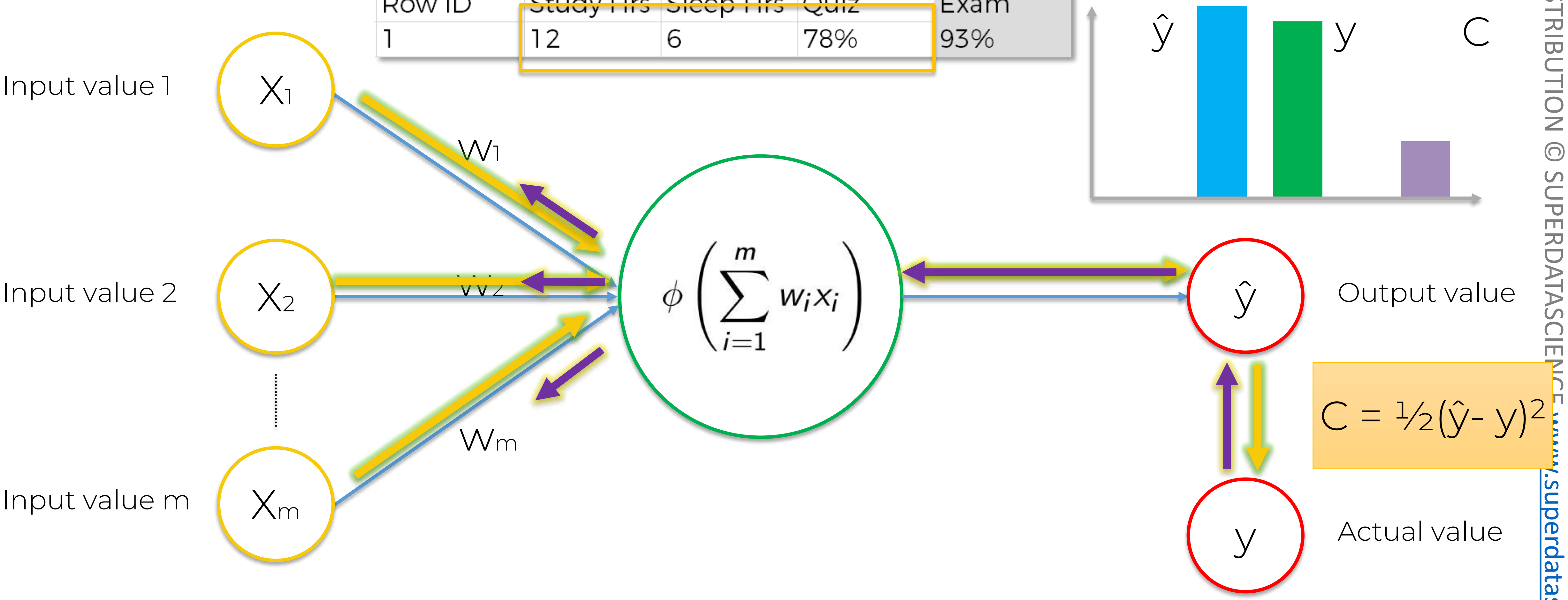
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%





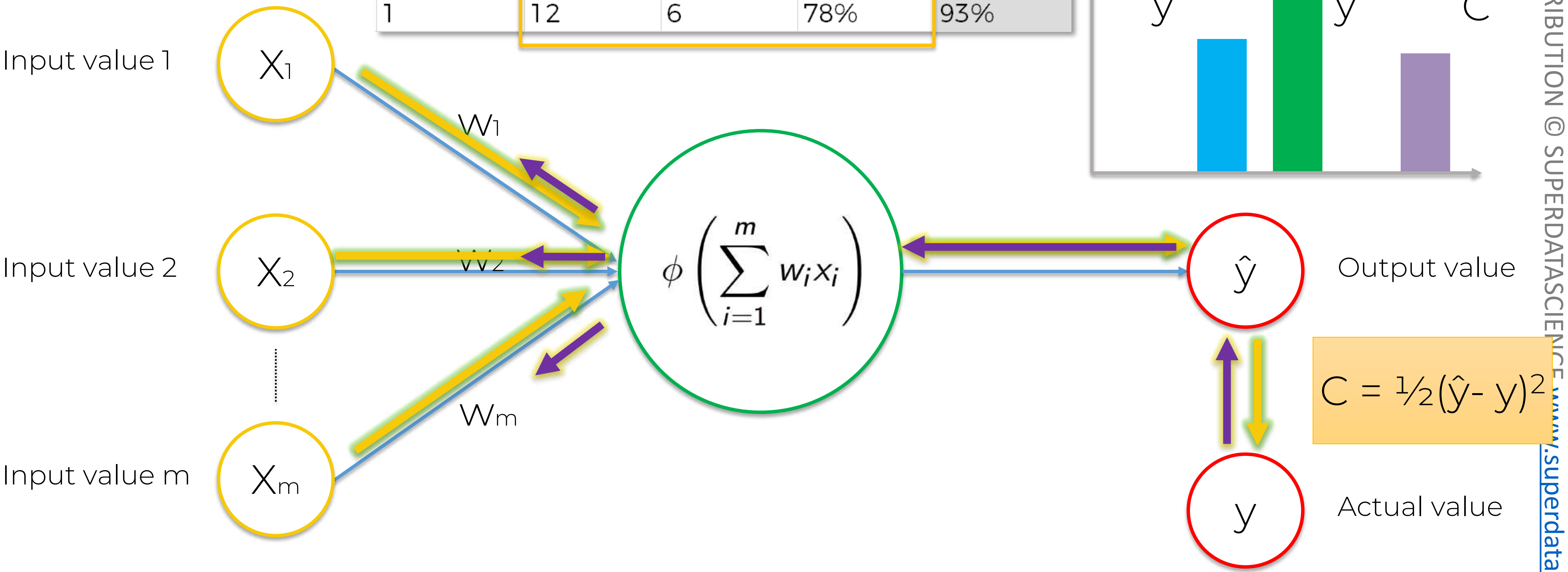
How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



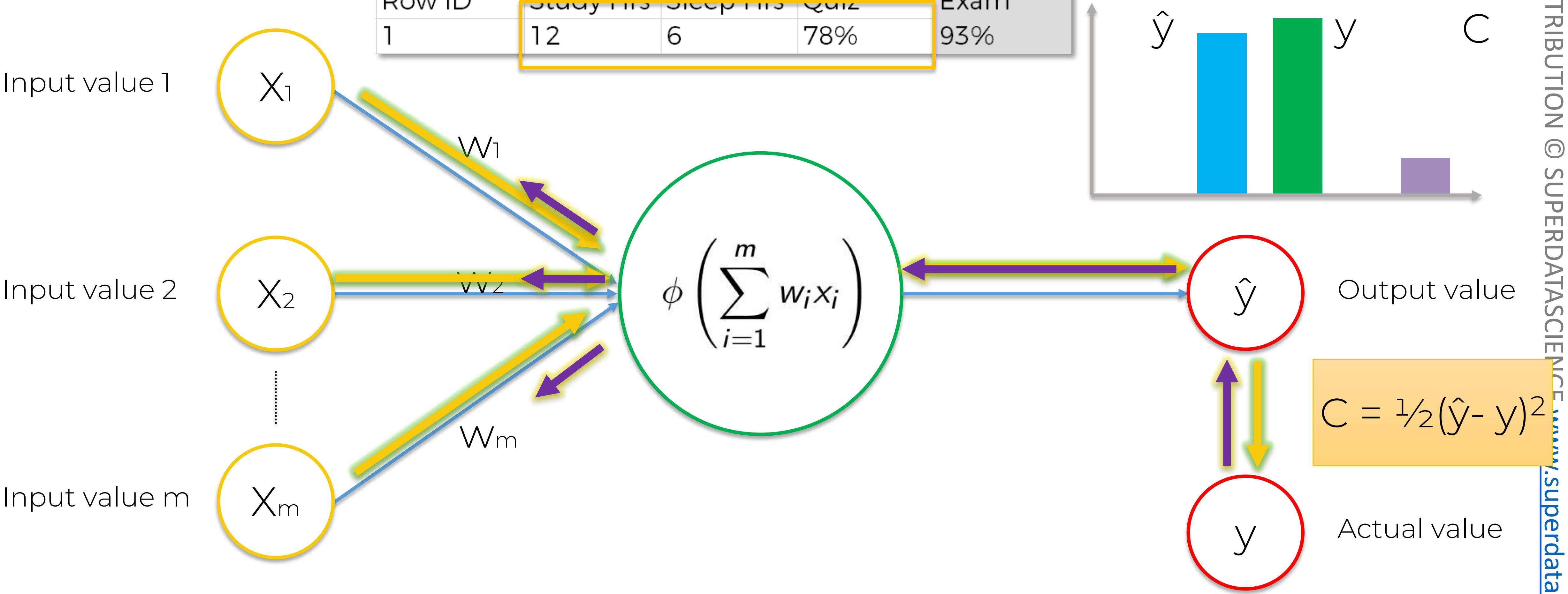
How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



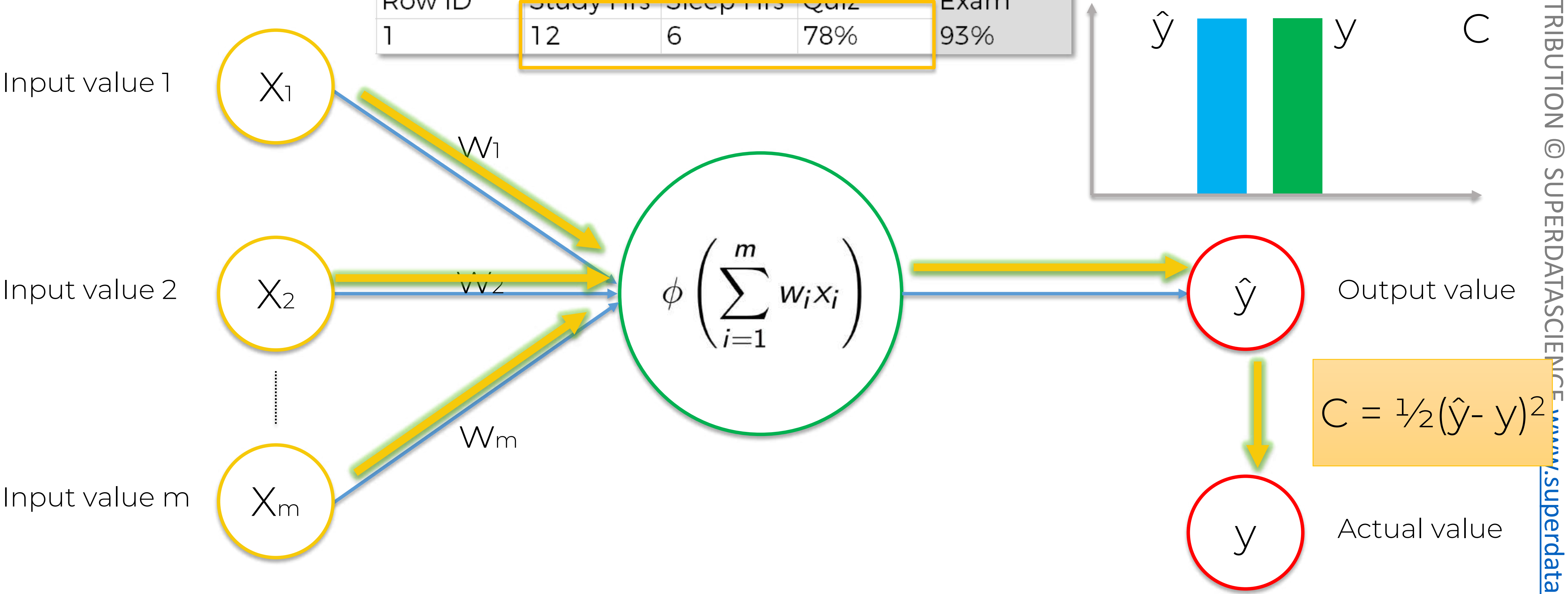
How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

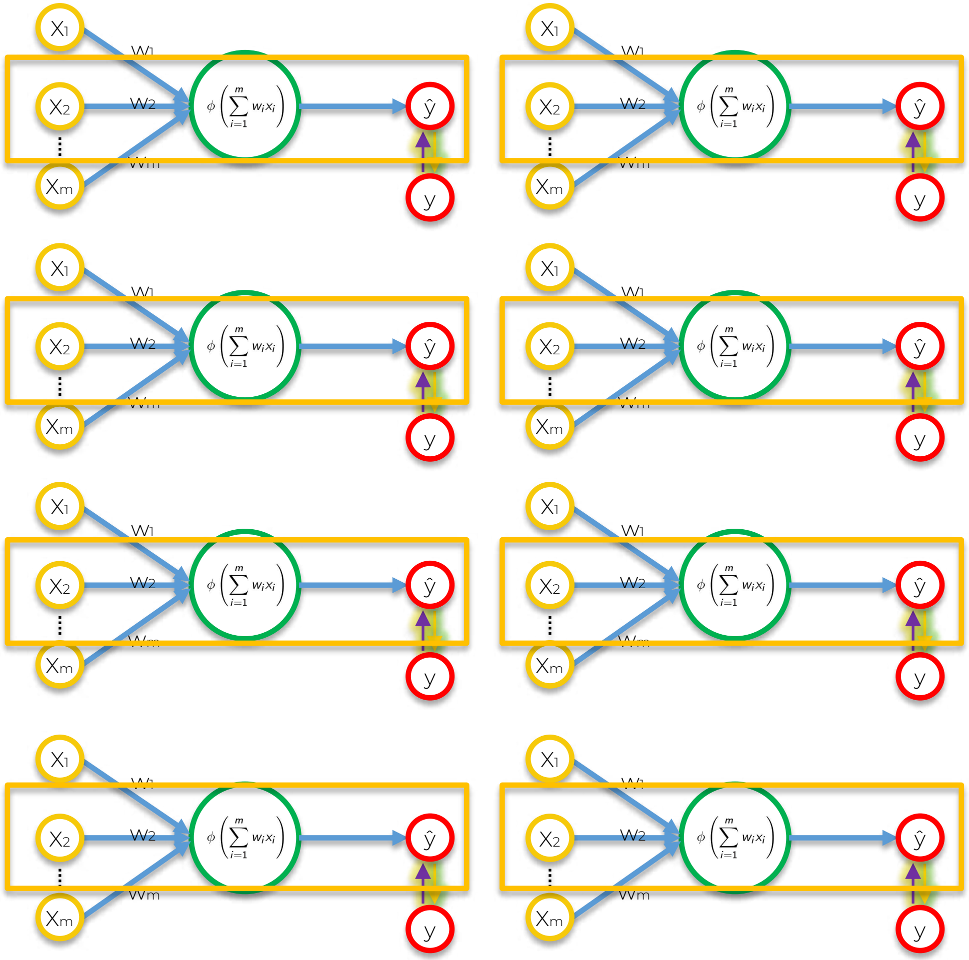


How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



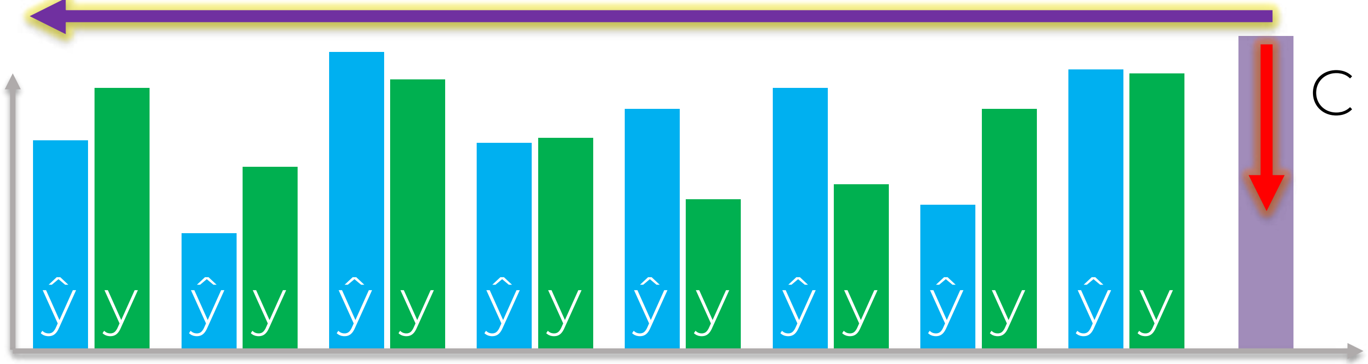
How do Neural Networks learn?



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

Adjust w_1, w_2, w_3



How do Neural Networks learn?

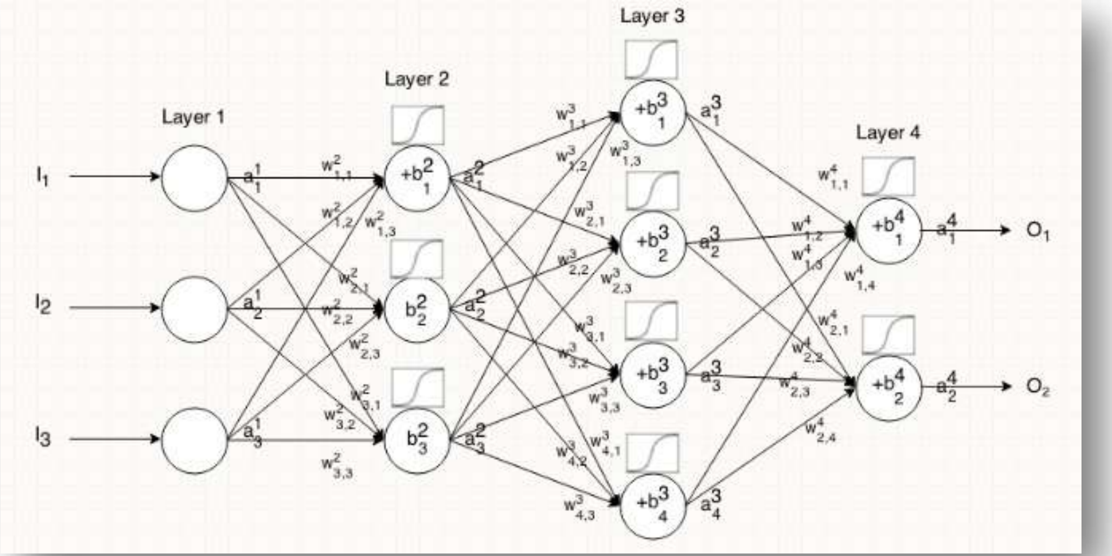
Additional Reading:

A list of cost functions used in neural networks, alongside applications

CrossValidated (2015)

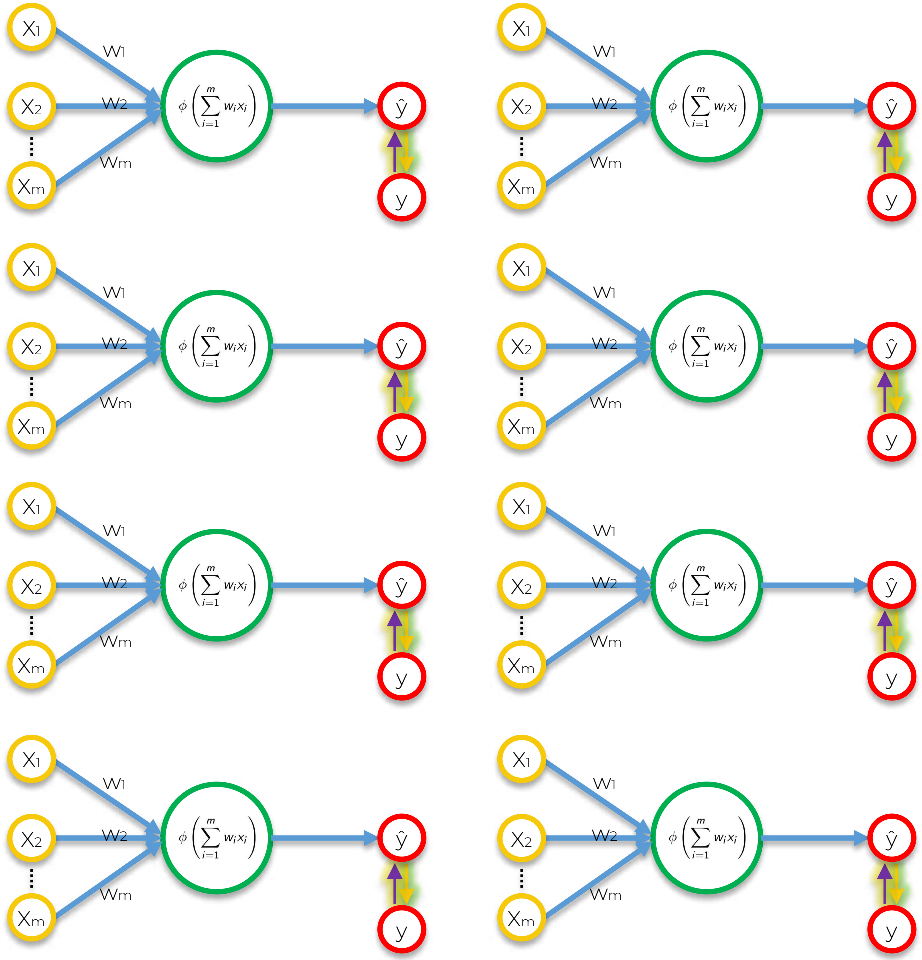
Link:

http://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications



Gradient Descent

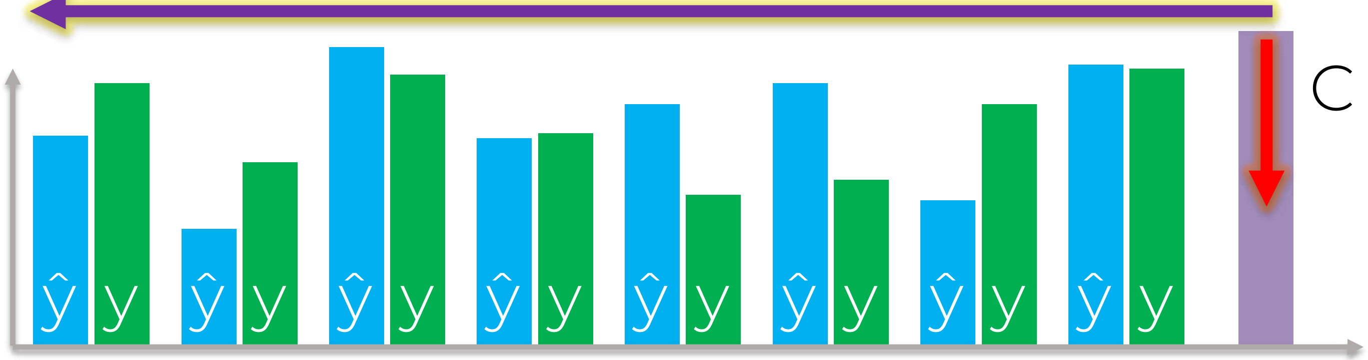
Gradient Descent



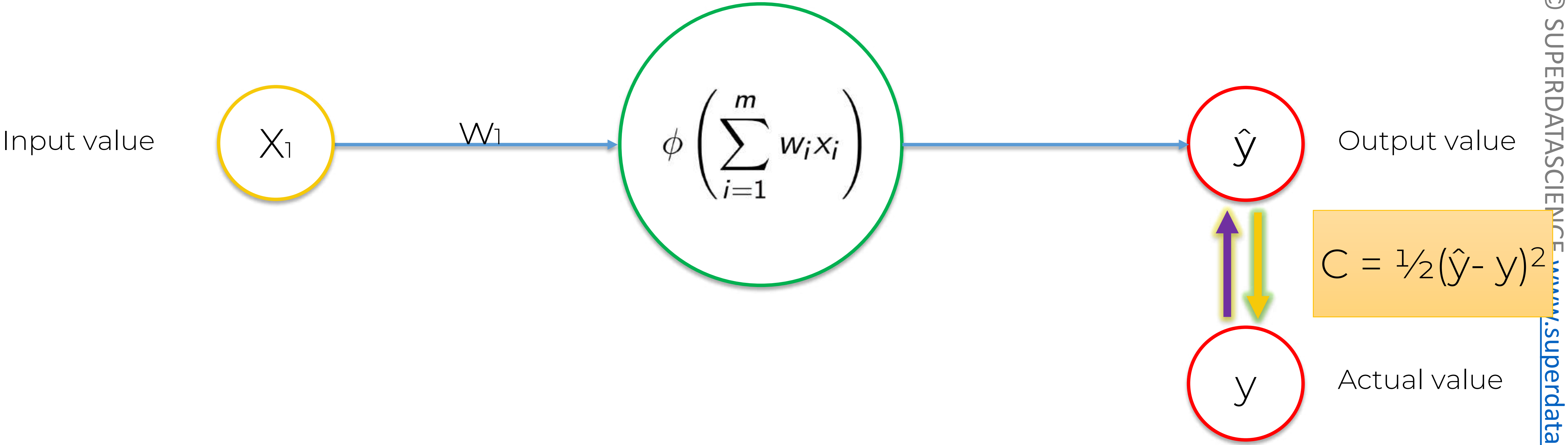
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

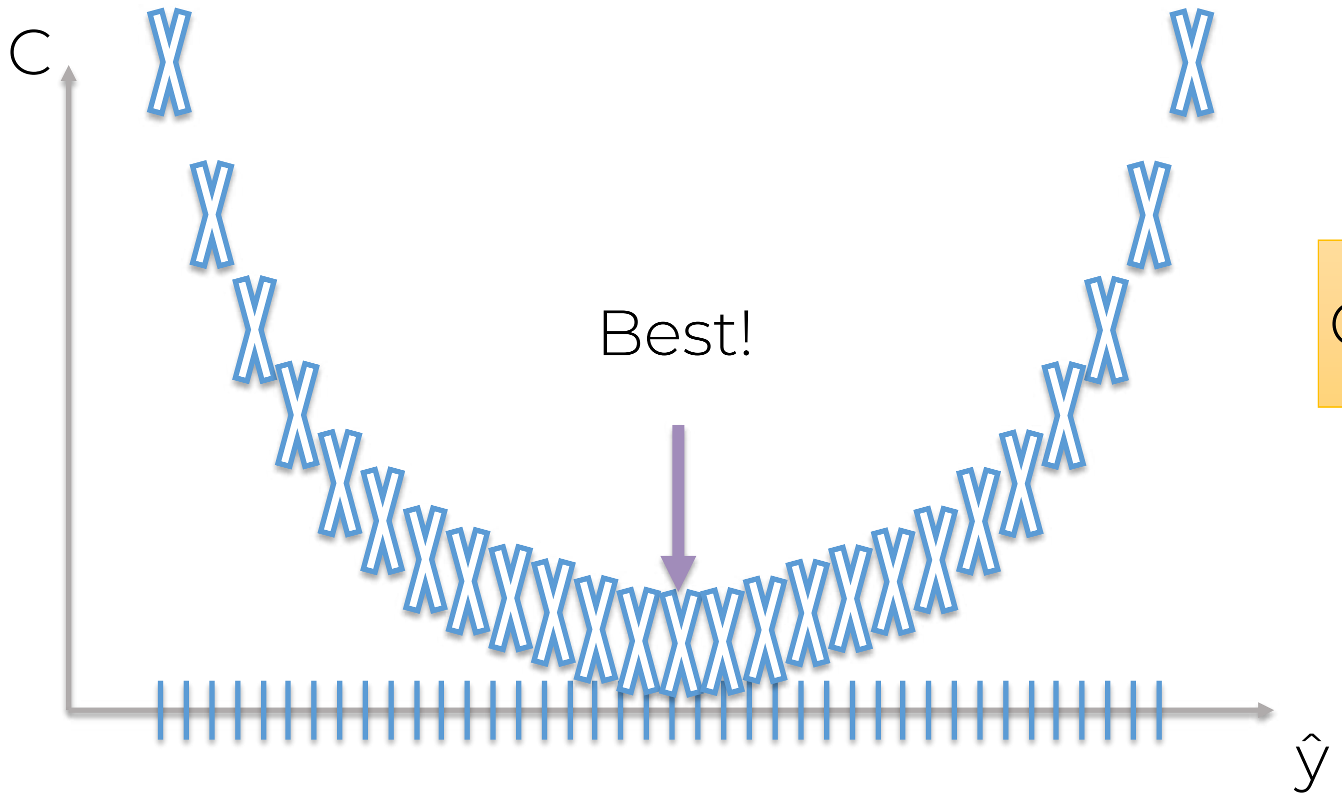
Adjust w_1, w_2, w_3



Gradient Descent



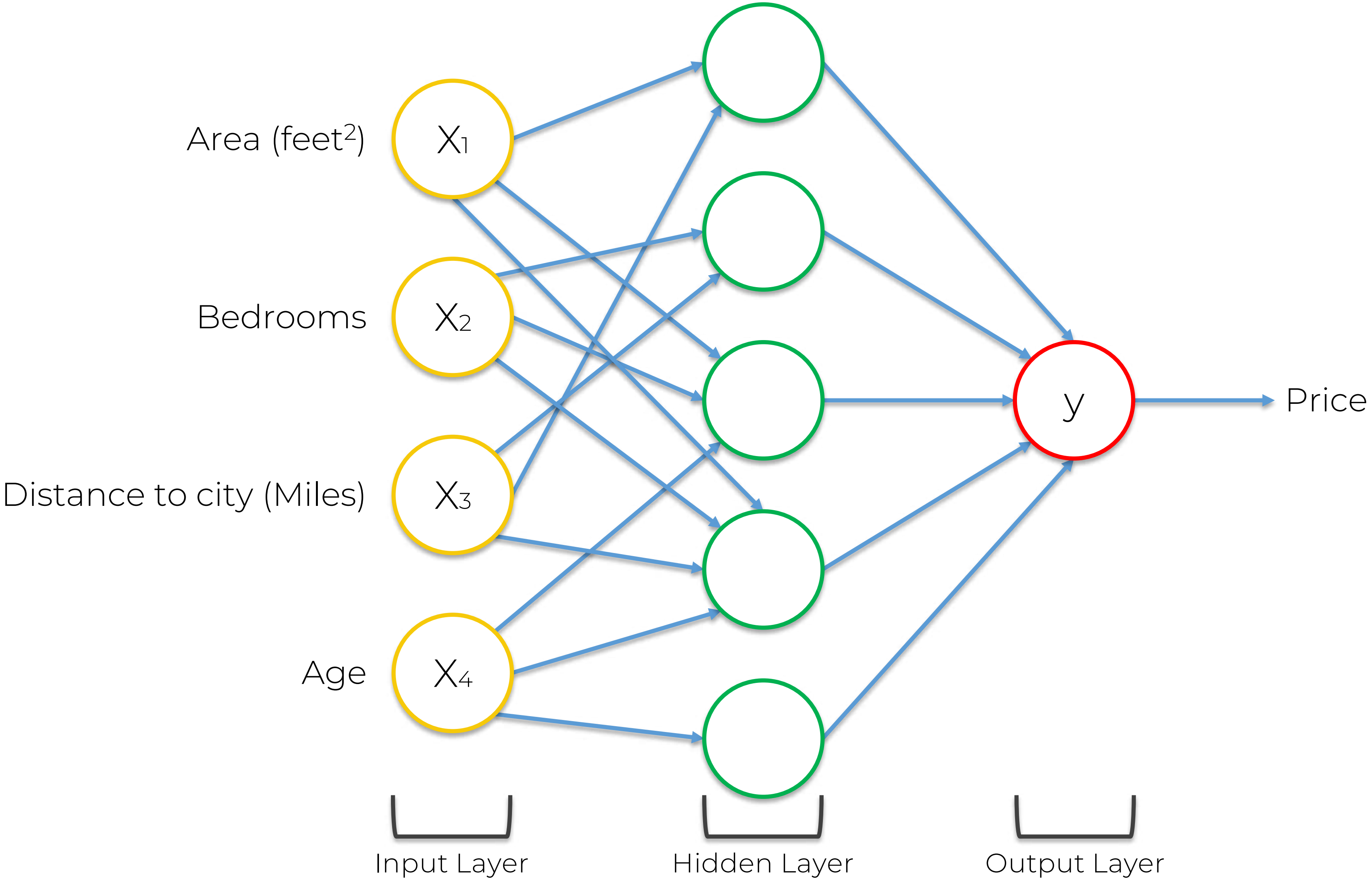
Gradient Descent



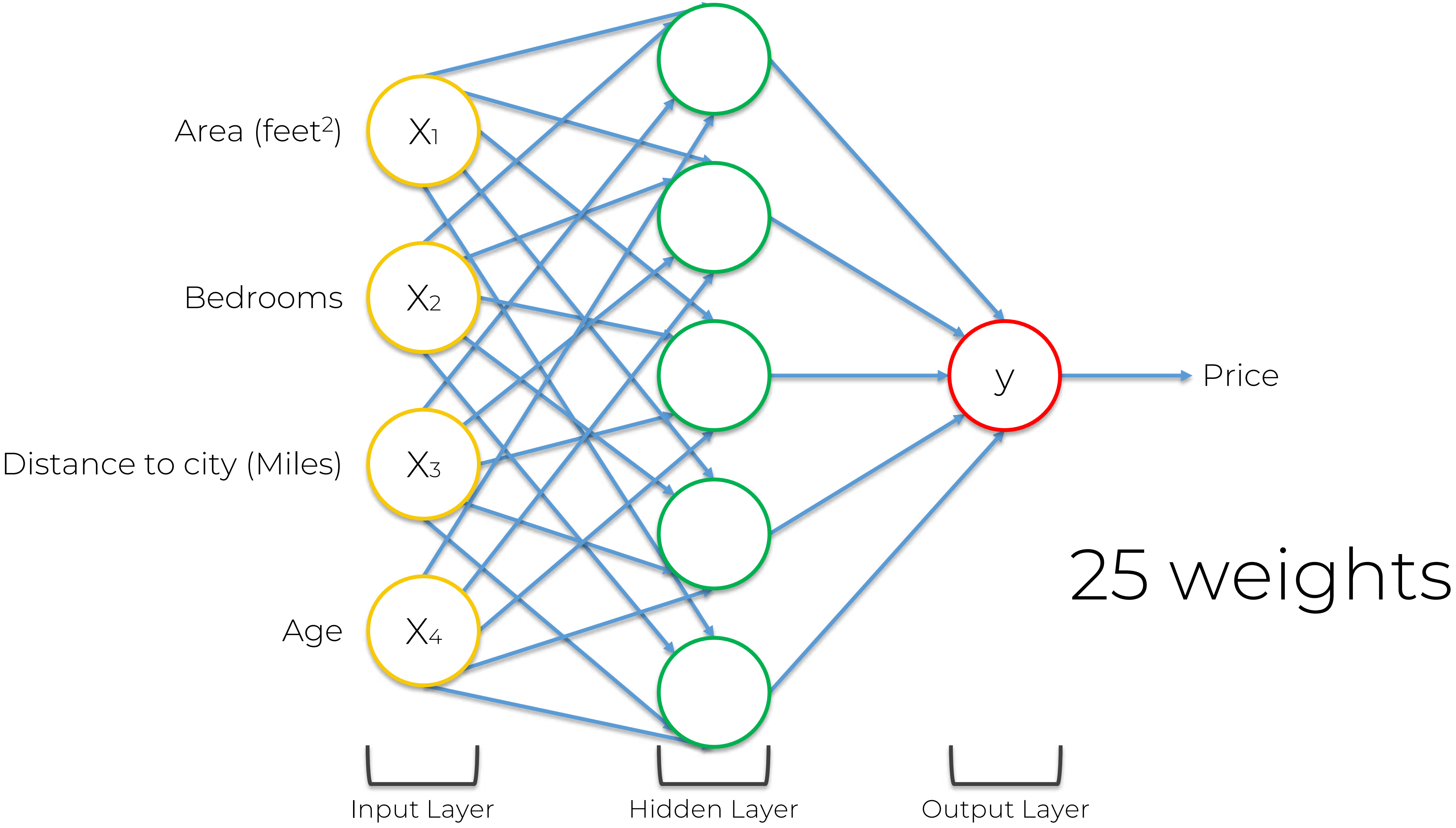
$C = \frac{1}{2}(\hat{y} - y)^2$

Curse of Dimensionality

Gradient Descent



Gradient Descent



Gradient Descent

$1{,}000 \times 1{,}000 \times \ldots \times 1{,}000 = 1{,}000^{25} = 10^{75}$ combinations

Sunway TaihuLight: World's fastest Super Computer

93 PFLOPS

93×10^{15}

$10^{75} / (93 \times 10^{15})$

$= 1.08 \times 10^{58}$ seconds

$= 3.42 \times 10^{50}$ years

Gradient Descent

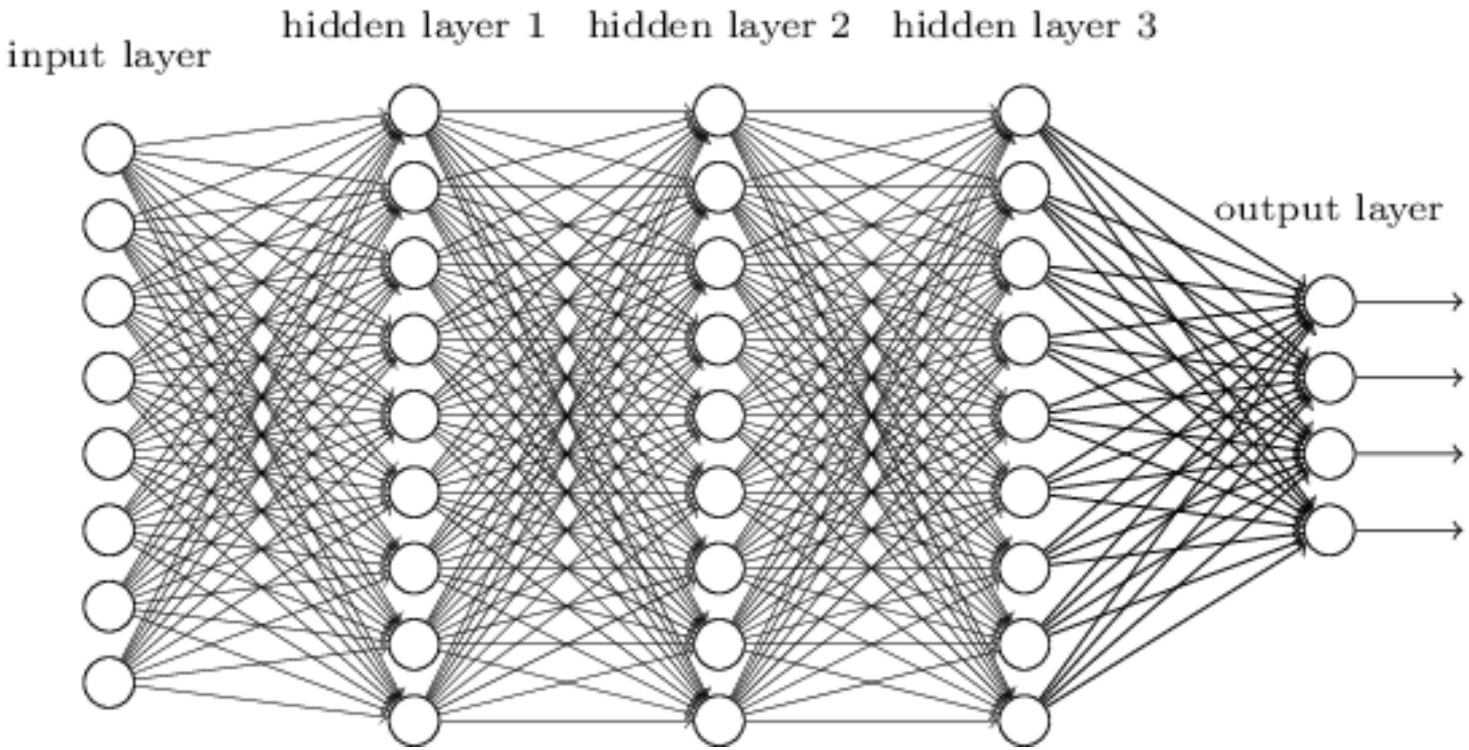
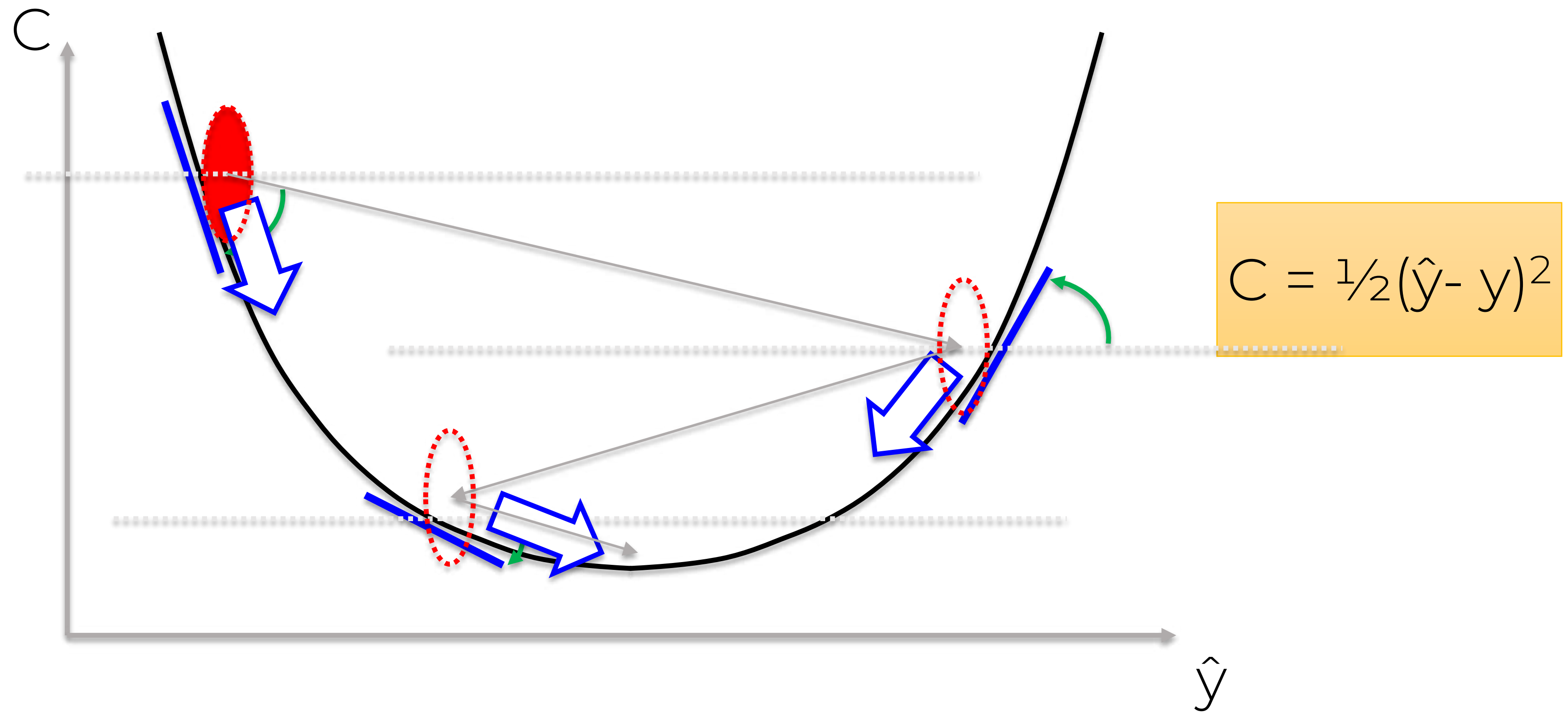


Image Source: neuralnetworksanddeeplearning.com

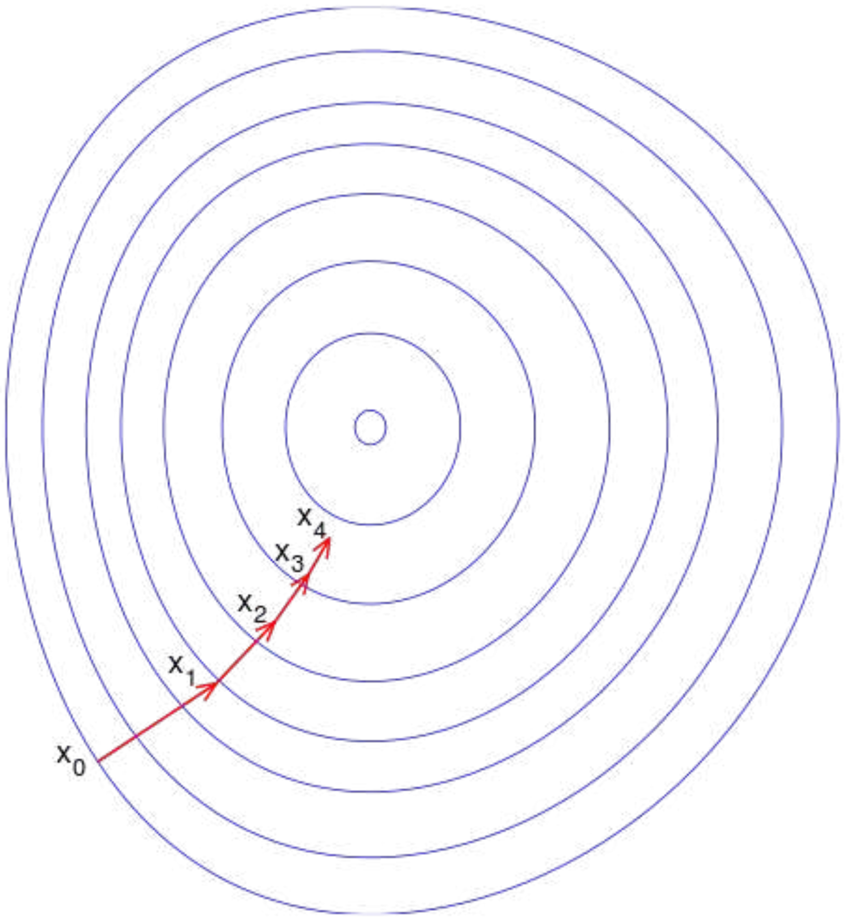
Gradient Descent

Gradient Descent

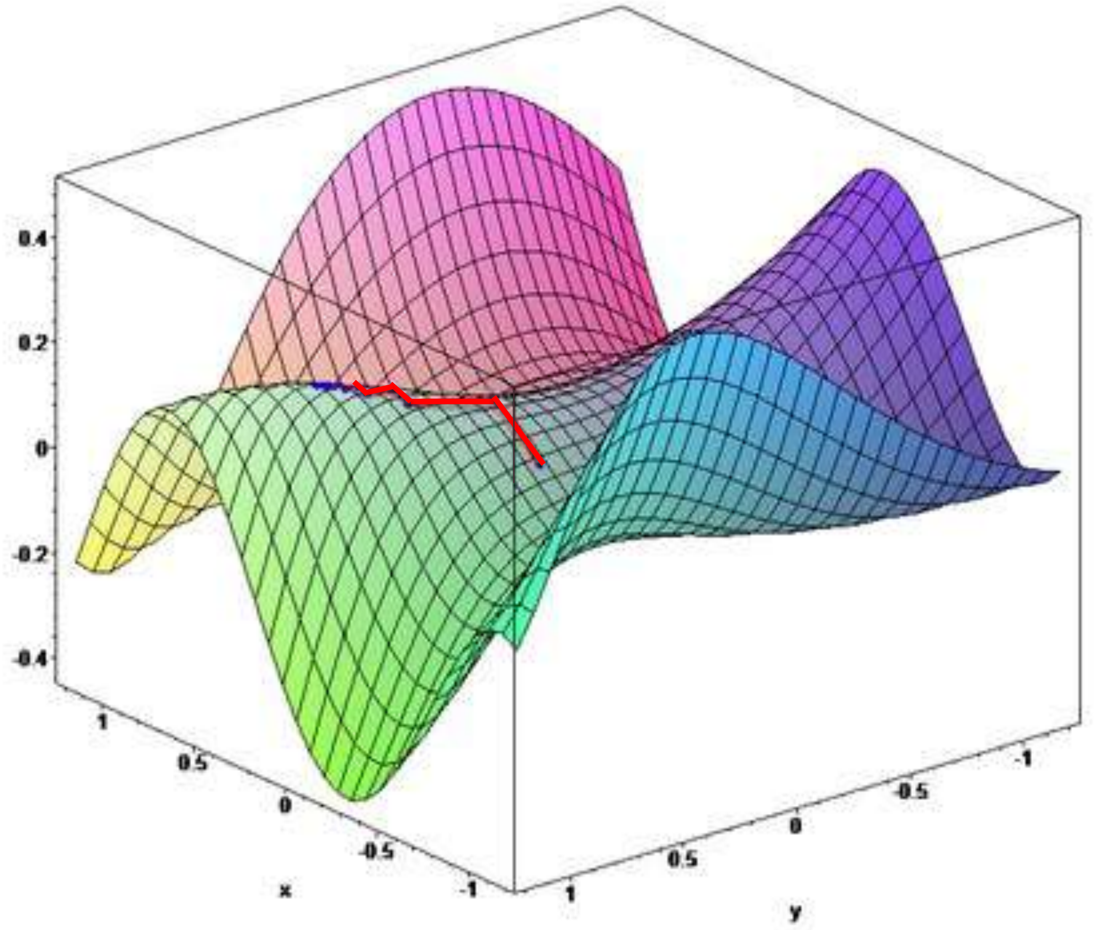
Gradient Descent

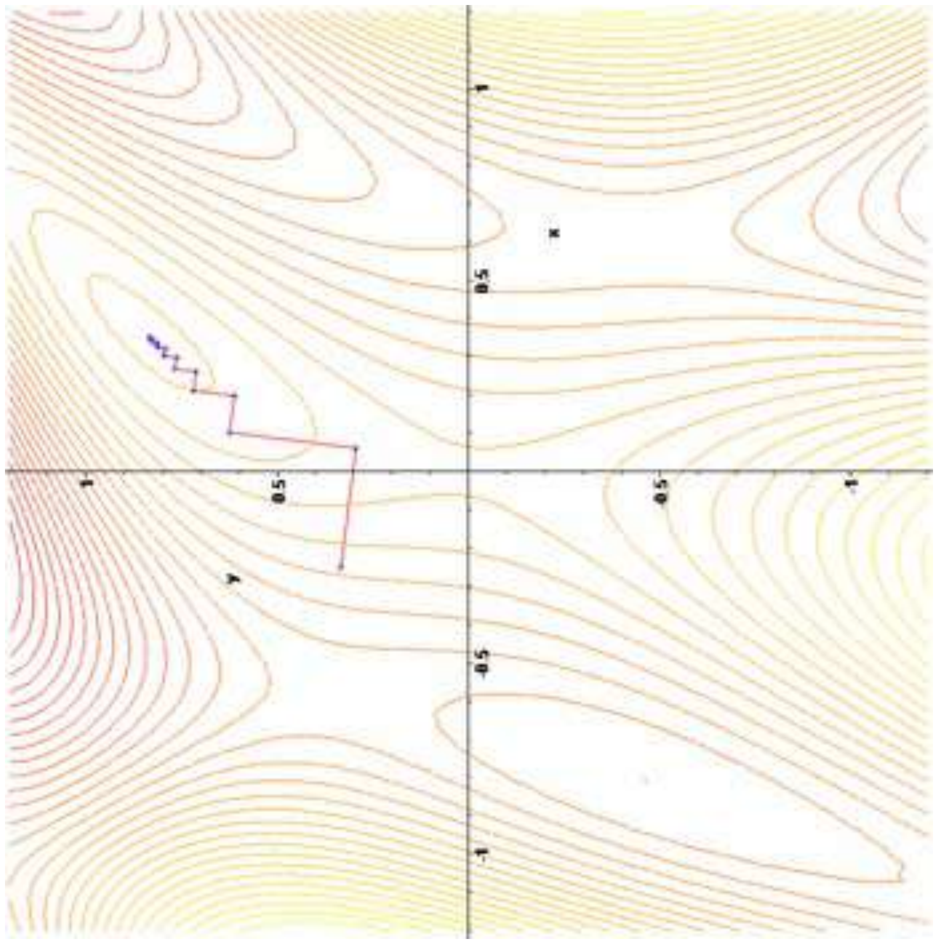


Gradient Descent



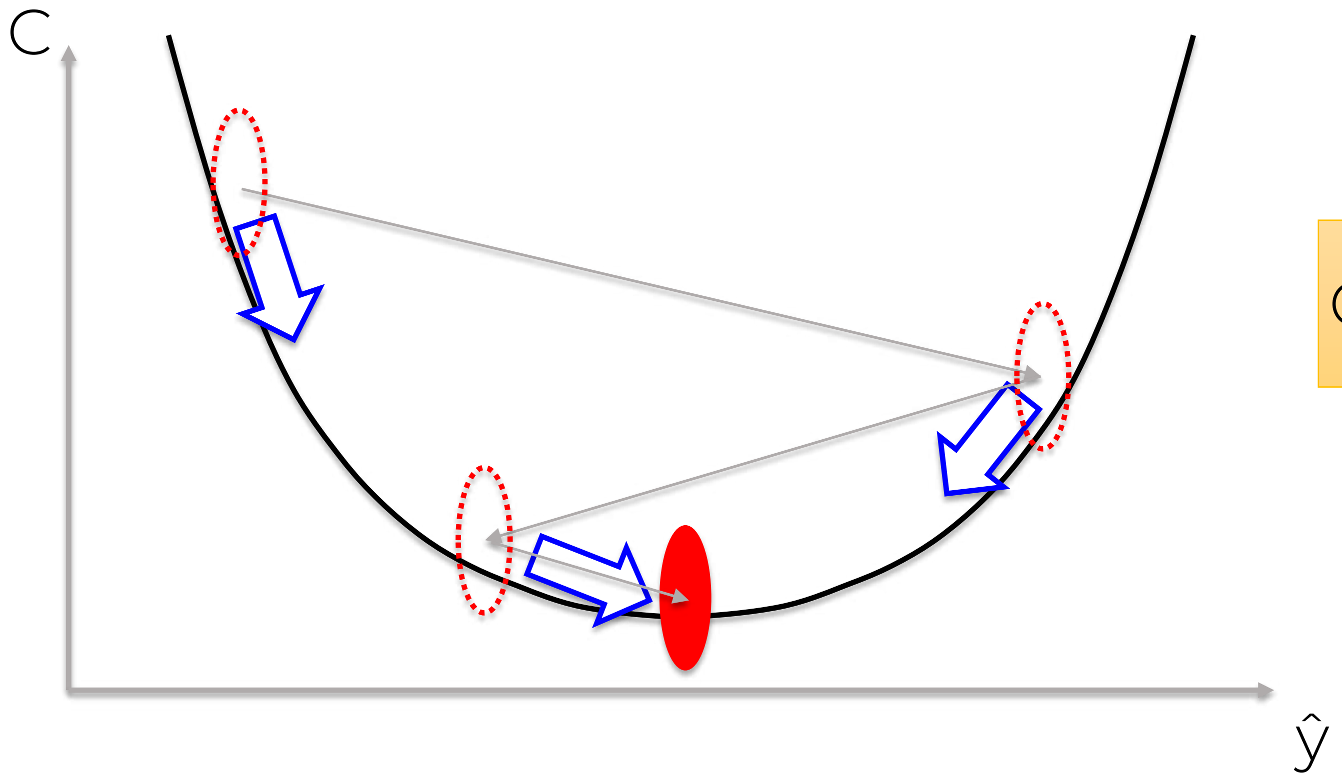
Gradient Descent





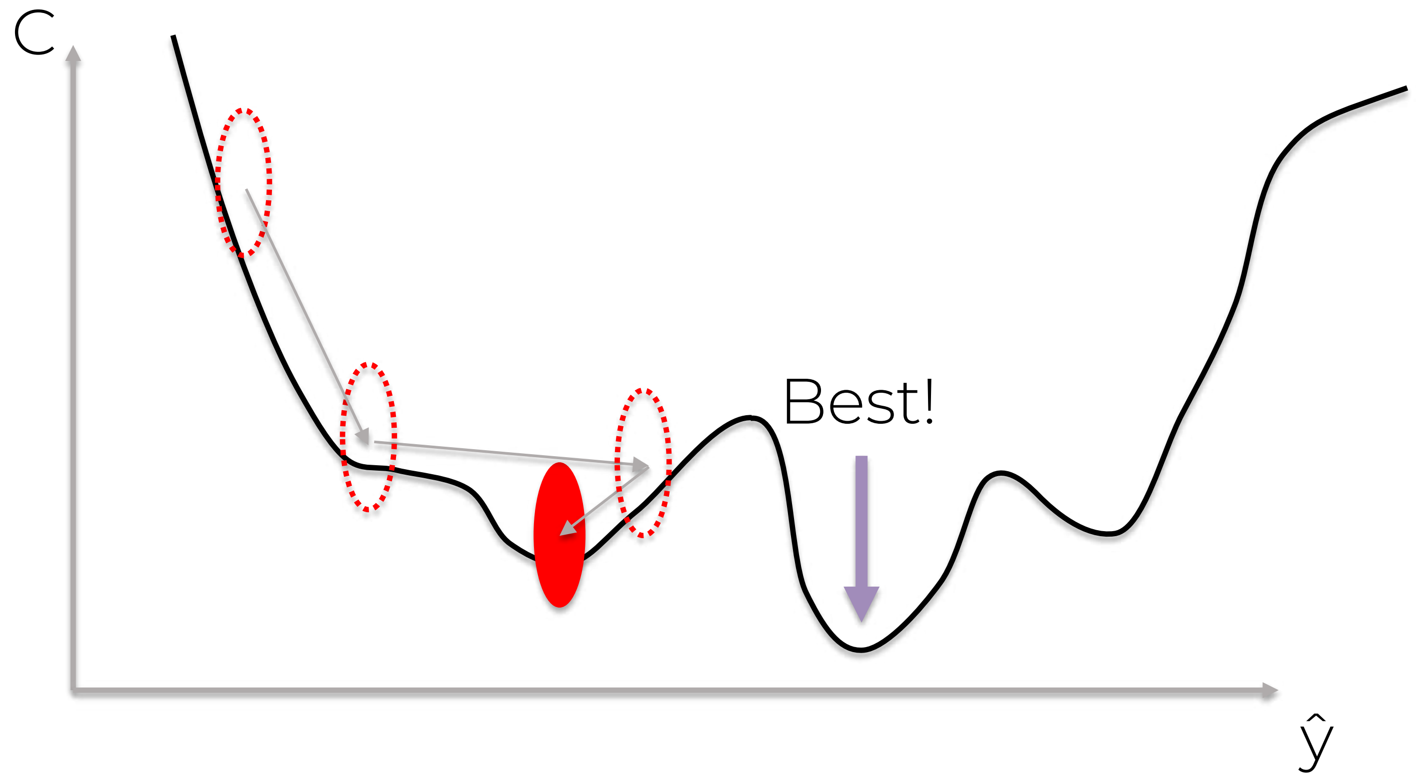
Stochastic Gradient Descent

Stochastic Gradient Descent

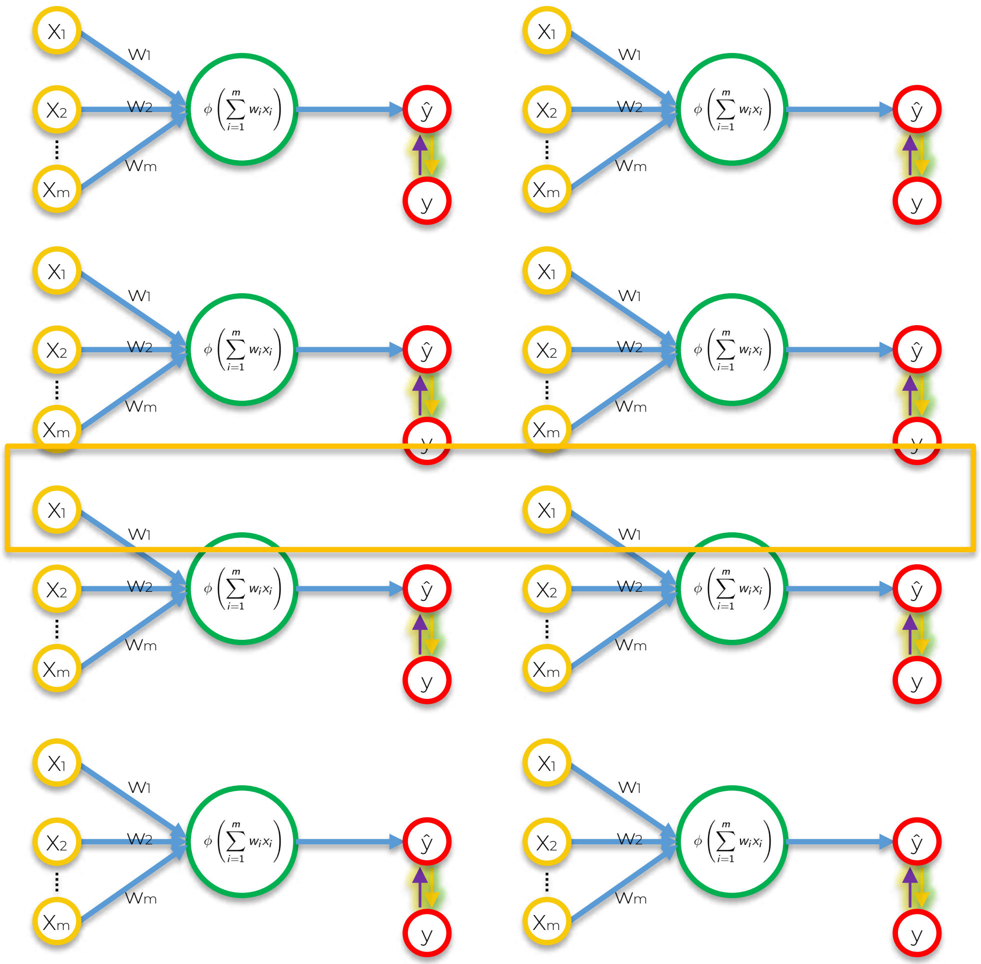


$$C = \frac{1}{2}(\hat{y} - y)^2$$

Stochastic Gradient Descent



Stochastic Gradient Descent



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

Stochastic Gradient Descent

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

Stochastic Gradient Descent

Upd w's

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

Batch Gradient Descent

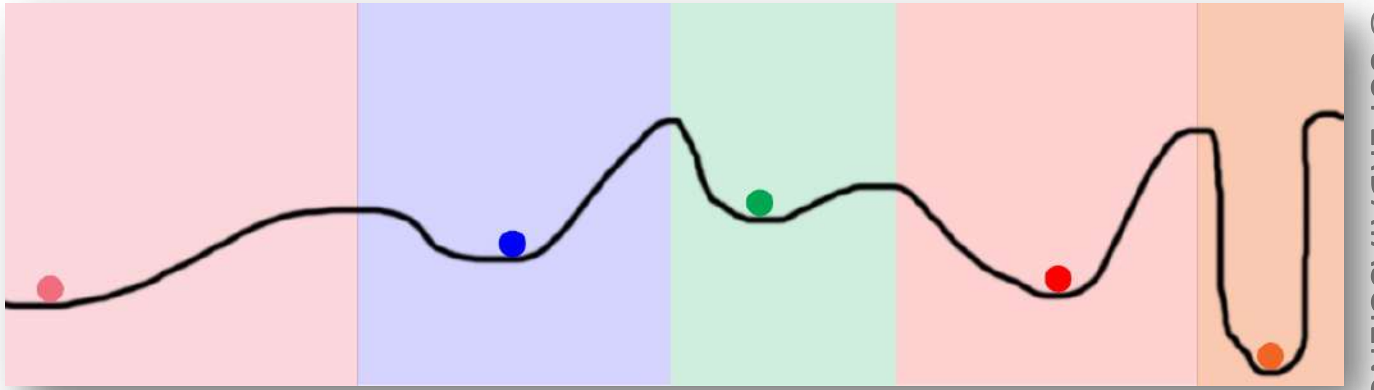
	Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
Upd w's	1	12	6	78%	93%
Upd w's	2	22	6.5	24%	68%
Upd w's	3	115	4	100%	95%
Upd w's	4	31	9	67%	75%
Upd w's	5	0	10	58%	51%
Upd w's	6	5	8	78%	60%
Upd w's	7	92	6	82%	89%
Upd w's	8	57	8	91%	97%

Stochastic Gradient Descent

Stochastic Gradient Descent

Additional Reading:

A Neural Network in 13 lines of Python (Part 2 - Gradient Descent)



Andrew Trask (2015)

Link:

https://iamtrask.github.io/2015/07/27/python-network-part2/

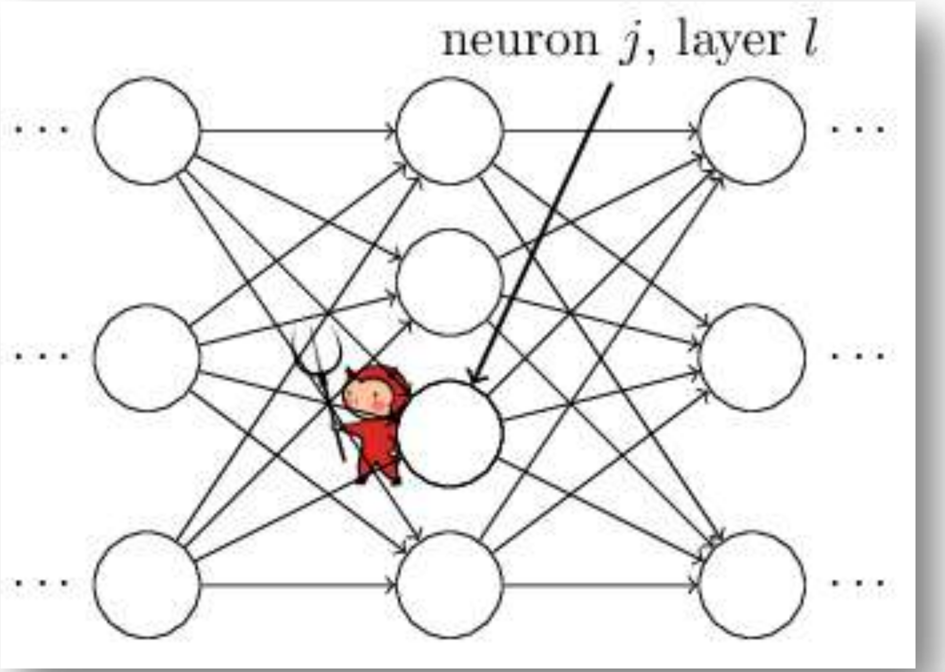
Stochastic Gradient Descent

Additional Reading:

Neural Networks and Deep Learning

Michael Nielsen (2015)

Link:



http://neuralnetworksanddeeplearning.com/chap2.html

Backpropagation

Gradient Descent

Forward Propagation

Backpropagation

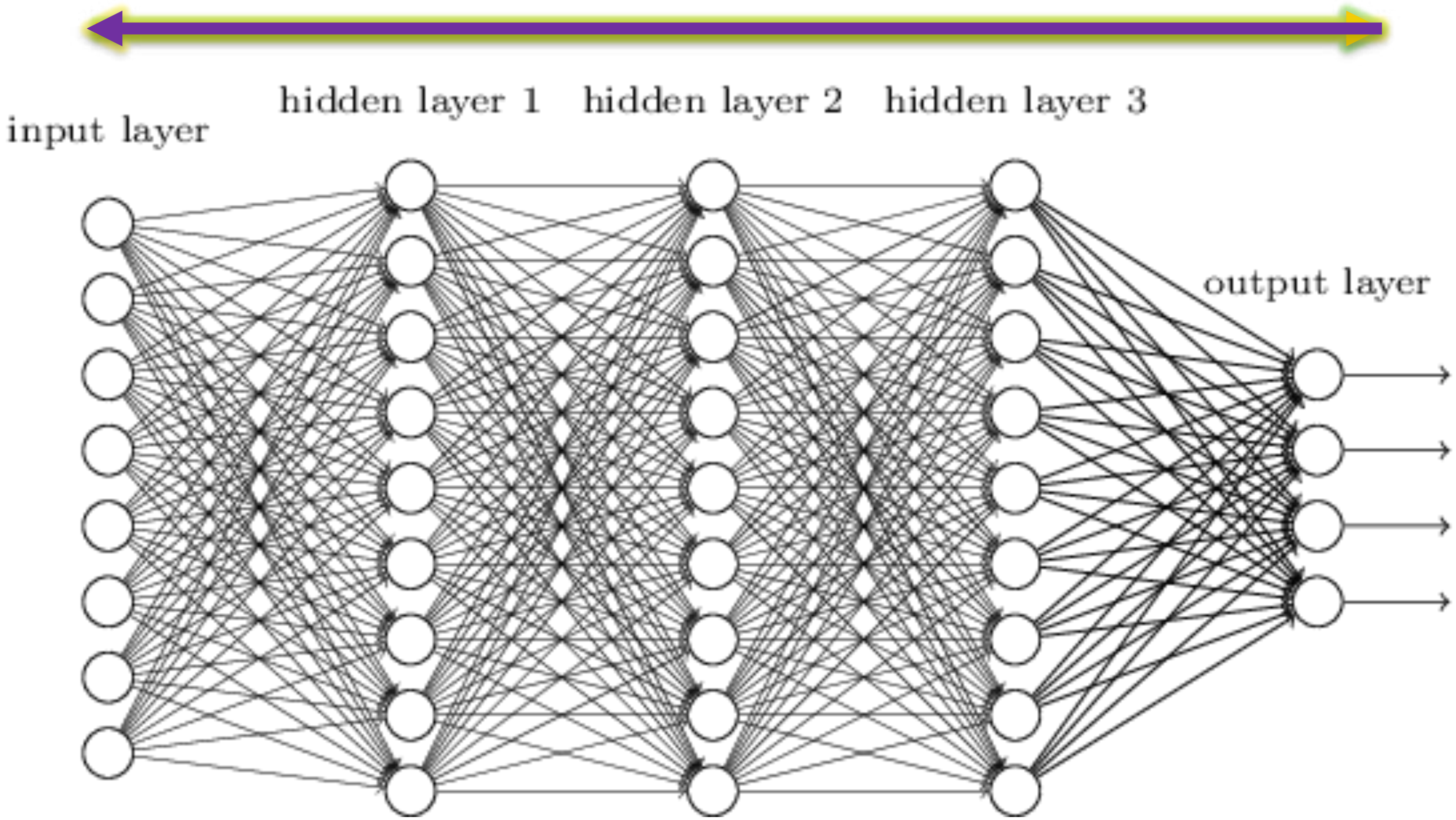


Image Source: neuralnetworksanddeeplearning.com

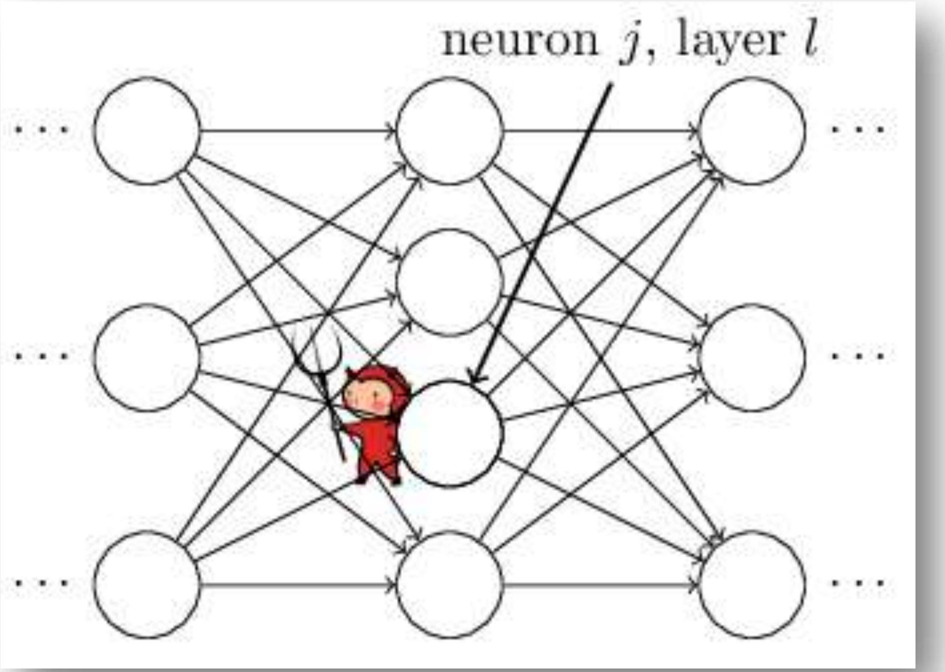
Stochastic Gradient Descent

Additional Reading:

Neural Networks and Deep Learning

Michael Nielsen (2015)

Link:



http://neuralnetworksanddeeplearning.com/chap2.html

Training the ANN with Stochastic Gradient Descent

STEP 1: Randomly initialise the weights to small numbers close to 0 (but not 0).

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning). Or: Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs.

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

Plan of Attack

Plan of Attack

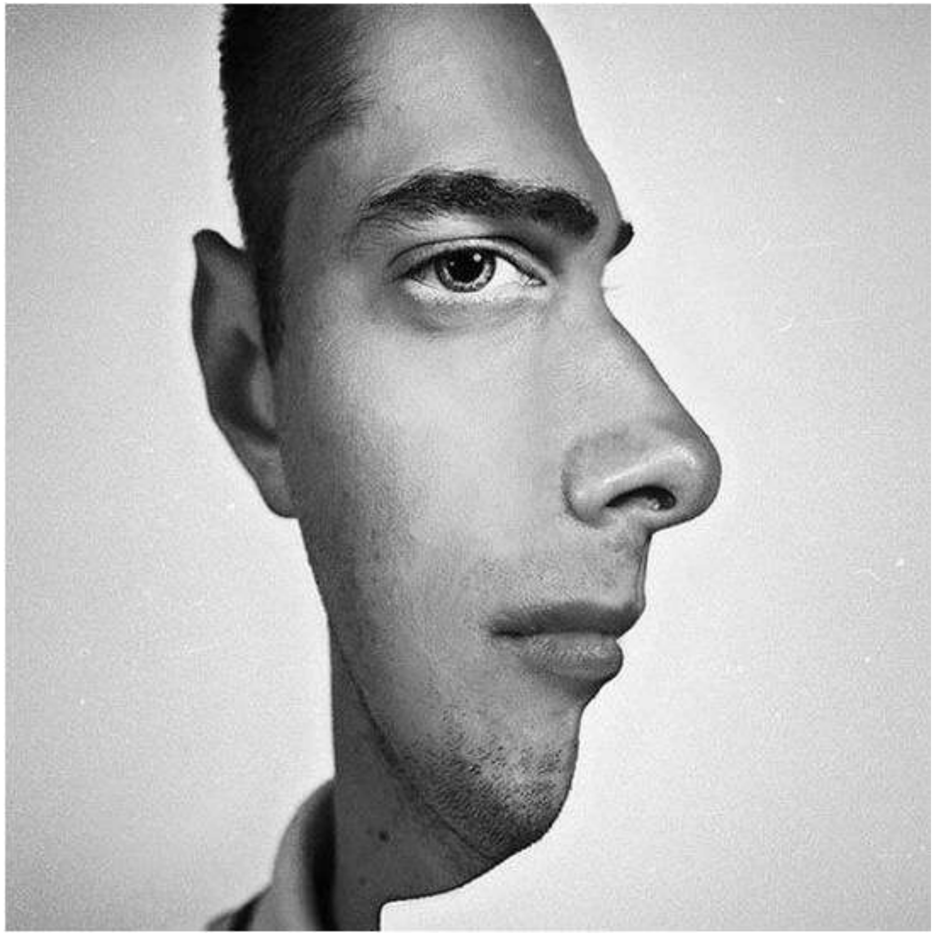
What we will learn in this section:

- What are Convolutional Neural Networks?
- Step 1 - Convolution Operation
- Step 1(b) - ReLU Layer
- Step 2 - Pooling
- Step 3 - Flattening
- Step 4 - Full Connection
- Summary

- EXTRA: Softmax & Cross-Entropy

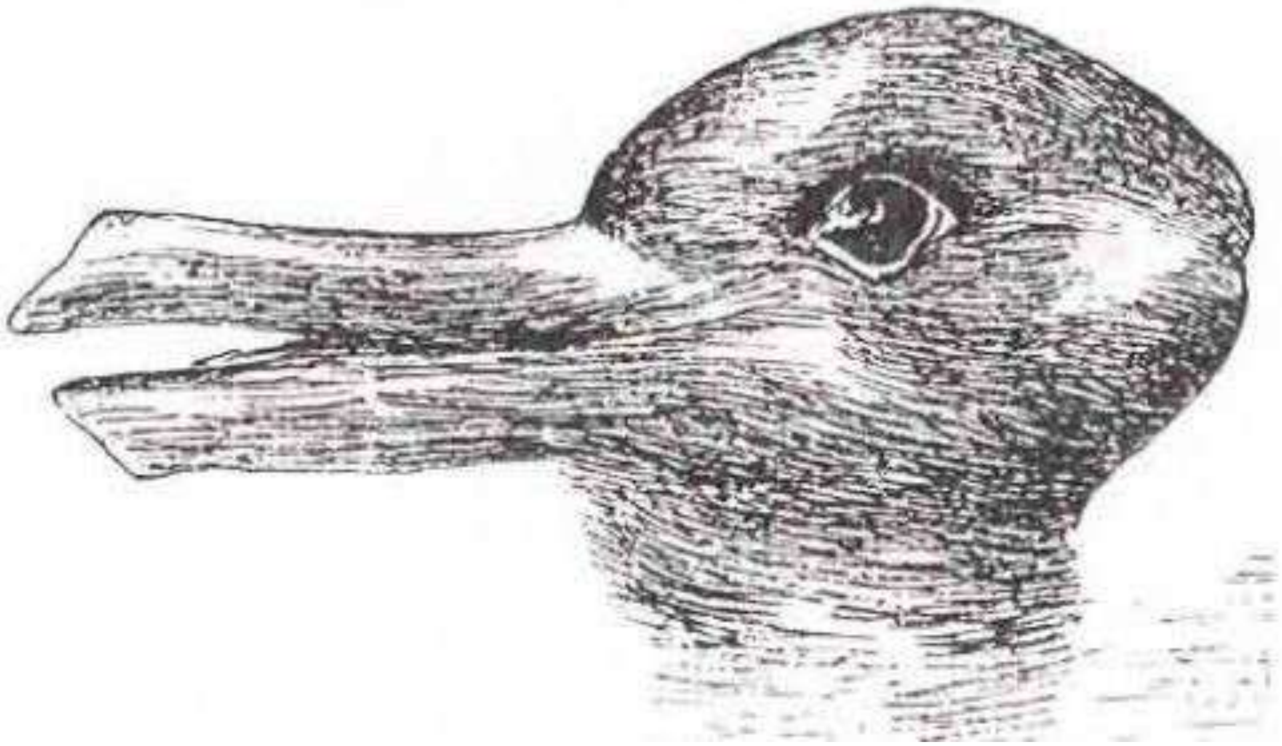
Convolutional Neural Networks

Convolutional Neural Networks

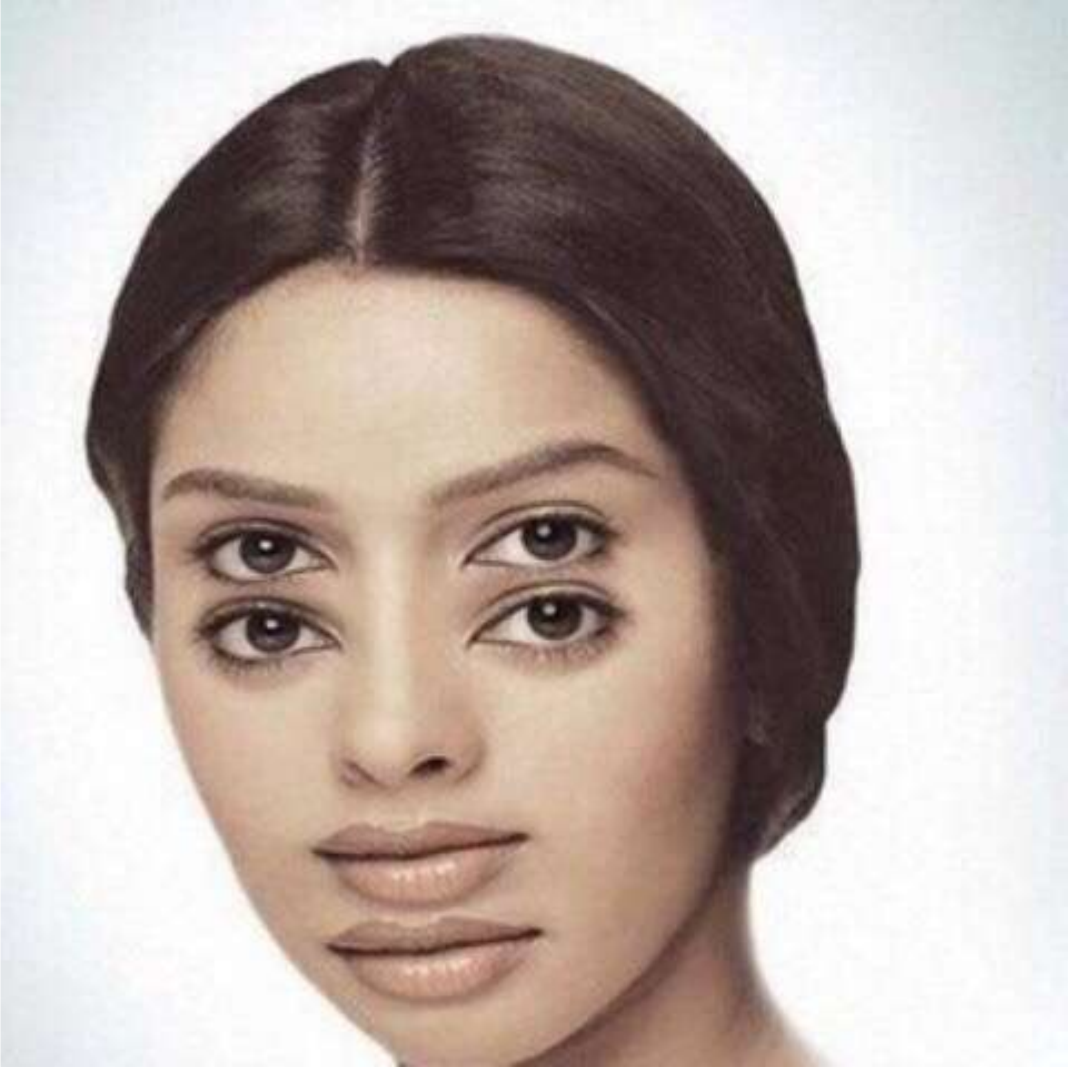


Convolutional Neural Networks

Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks

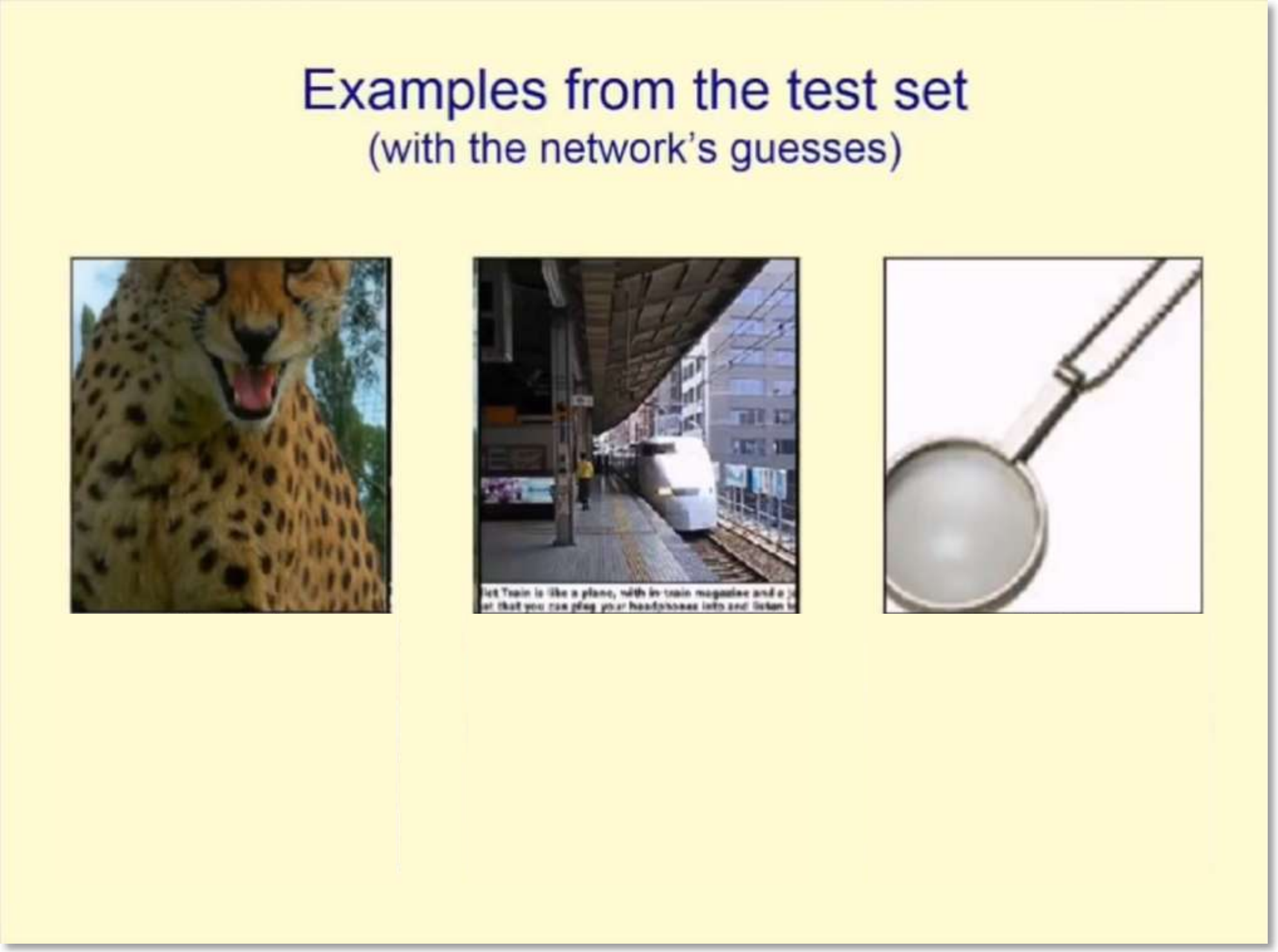
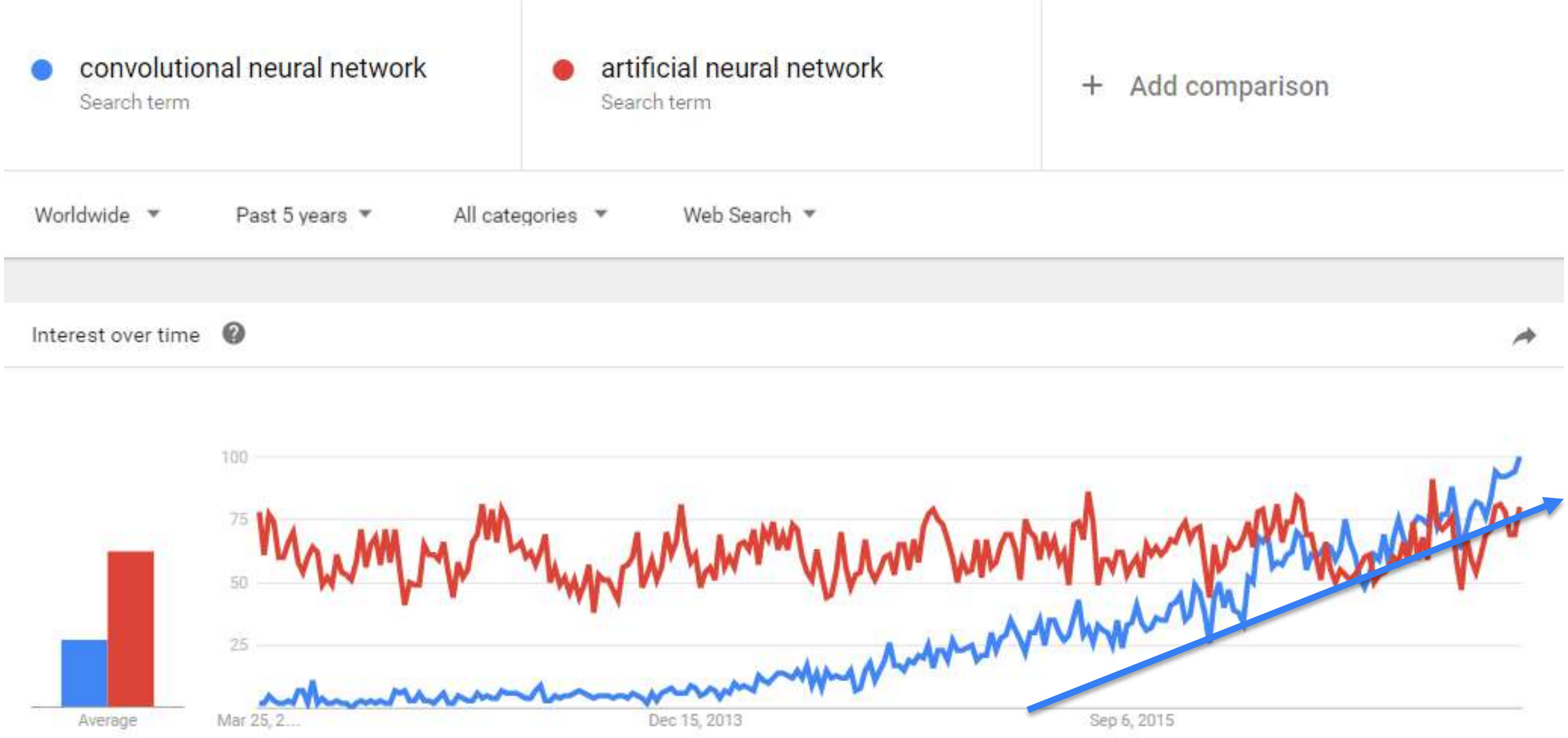


Image Source: a talk by Geoffrey Hinton

Convolutional Neural Networks



Source: google trends

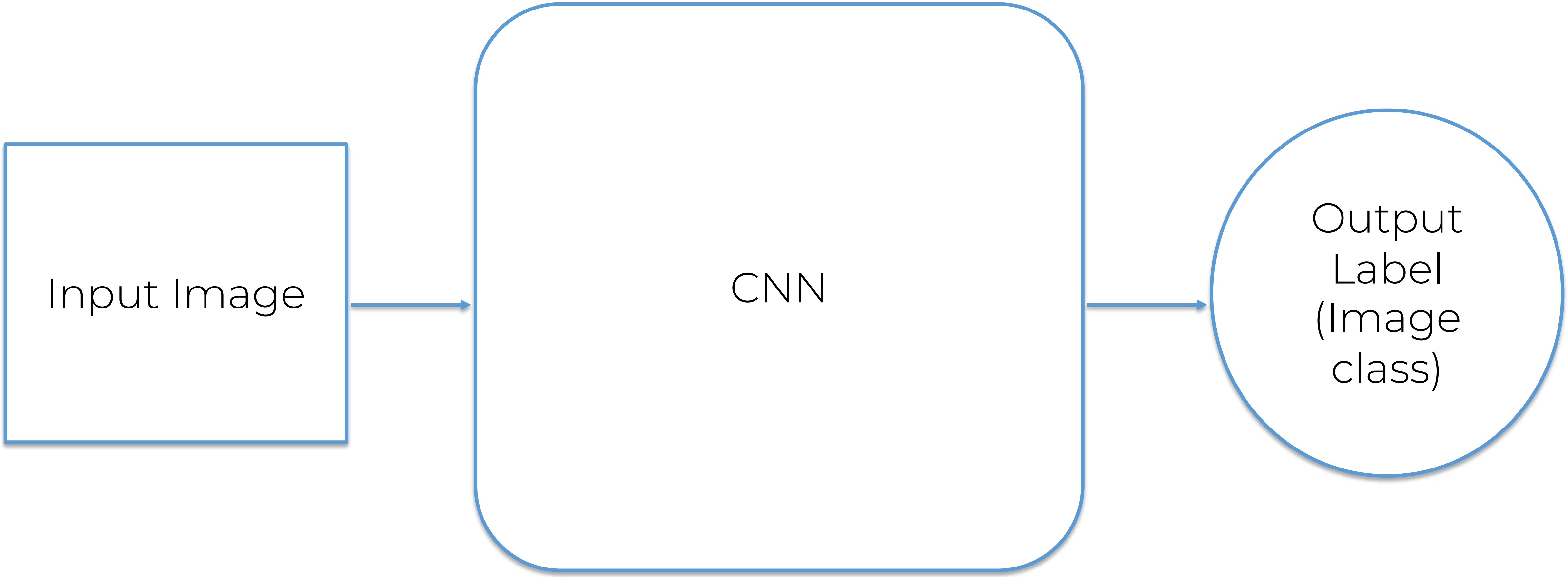
Convolutional Neural Networks

Yann Lecun

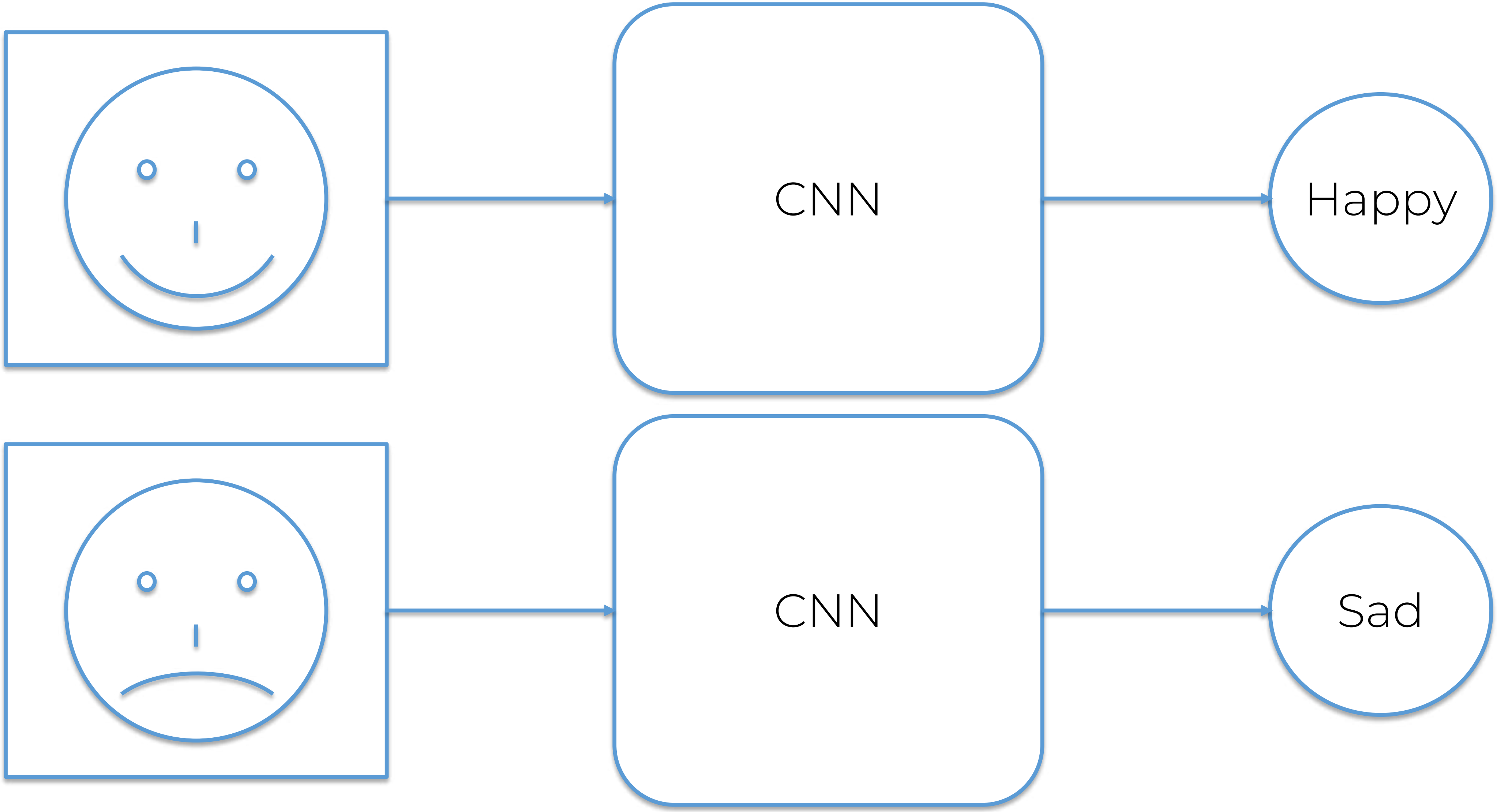
Convolutional Neural Networks

Google Facebook

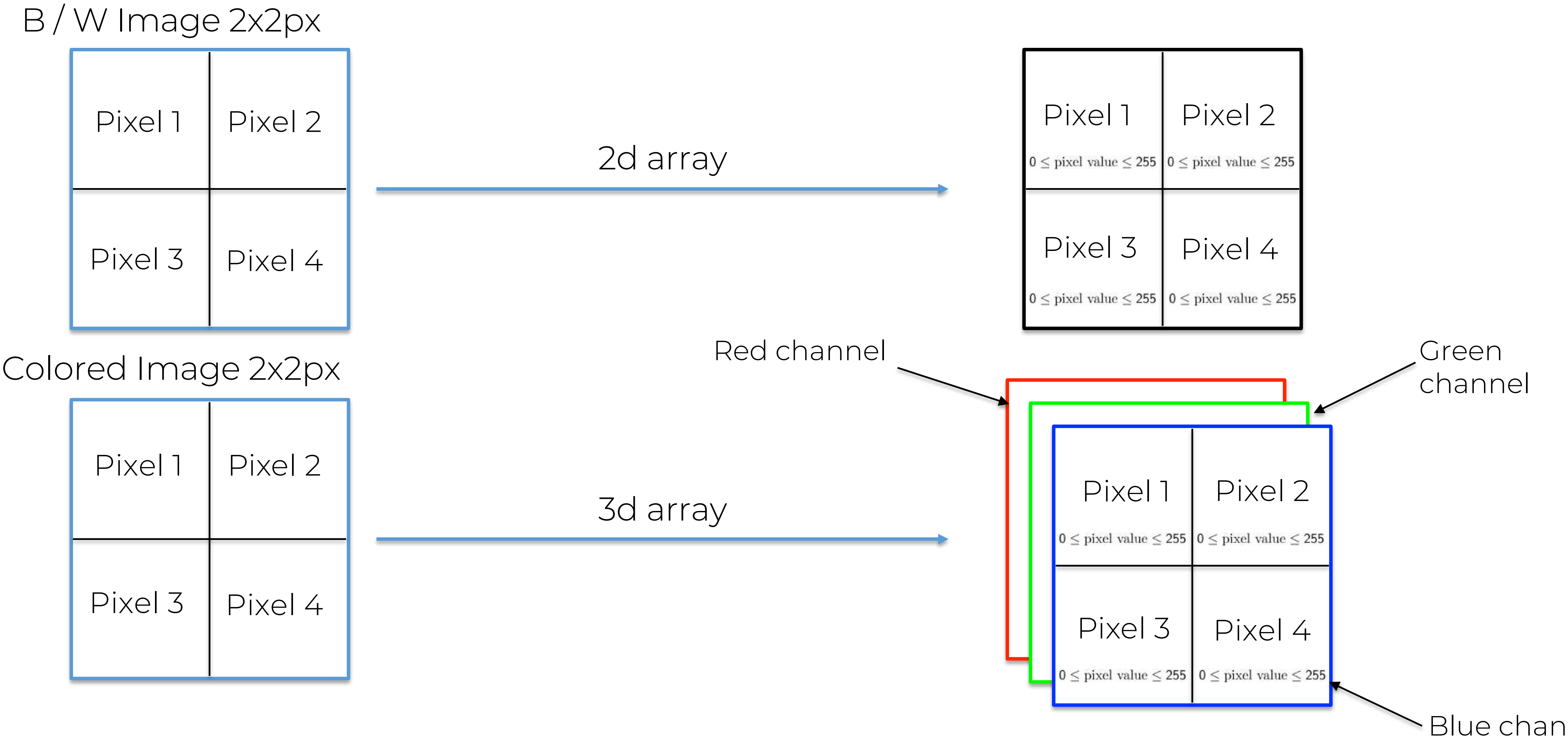
Convolutional Neural Networks



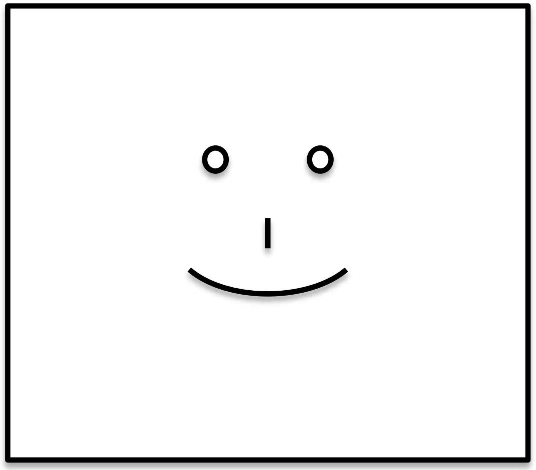
Convolutional Neural Networks

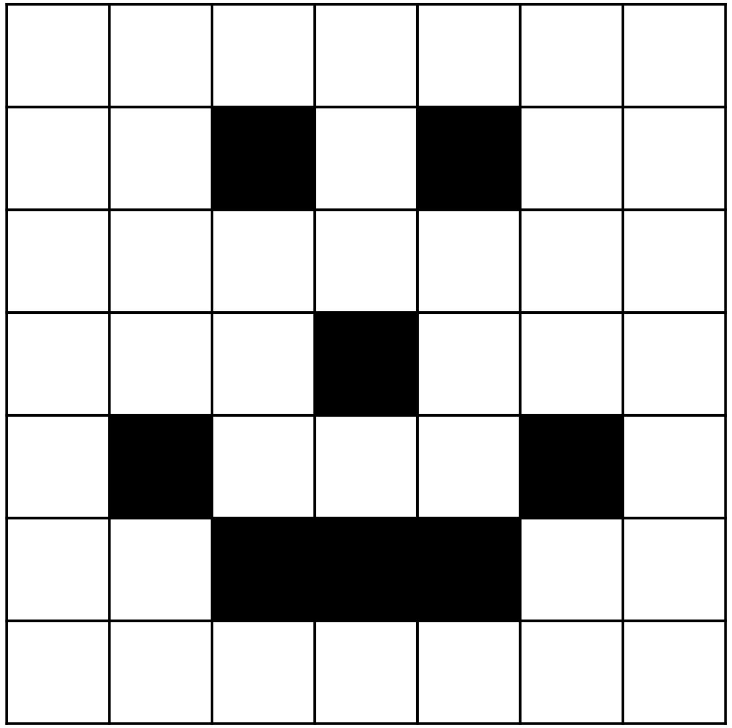


Convolutional Neural Networks



Convolutional Neural Networks





0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Convolutional Neural Networks

STEP 1: Convolution

⇩

STEP 2: Max Pooling

⇩

STEP 3: Flattening

⇩

STEP 4: Full Connection

Convolutional Neural Networks

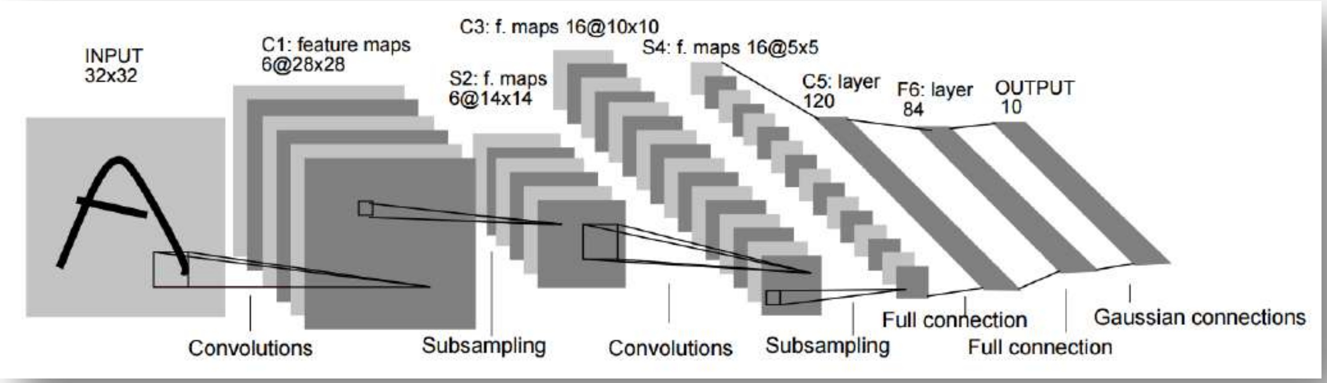
Additional Reading:

Gradient-Based Learning Applied to Document Recognition

By Yann LeCun et al. (1998)

Link:

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf



Step 1 – Convolution

Step 1 - Convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Step 1 - Convolution

Additional Reading:

Introduction to Convolutional Neural Networks

By Jianxin Wu (2017)

Link: http://cs.nju.edu.cn/wujx/paper/CNN.pdf

$$
\begin{aligned}
\frac{\partial z}{\partial (\operatorname{vec}(\boldsymbol{y})^T)}(F^T \otimes I) &= \left((F \otimes I) \frac{\partial z}{\partial \operatorname{vec}(\boldsymbol{y})} \right)^T \\
&= \left((F \otimes I) \operatorname{vec}\left(\frac{\partial z}{\partial Y} \right) \right)^T \\
&= \operatorname{vec}\left(I \frac{\partial z}{\partial Y} F^T \right)^T \\
&= \operatorname{vec}\left(\frac{\partial z}{\partial Y} F^T \right)^T ,
\end{aligned}
$$

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

0	0	1
1	0	0
0	1	1

Feature Detector

Step 1- Convolution

Input Image

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

$\otimes$

Feature Detector

0	0	1
1	0	0
0	1	1

$=$

Feature Map

0				

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1			

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0		

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	

Feature Map

Step 1 - Convolution

Input Image

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

$\otimes$

Feature Detector

0	0	1
1	0	0
0	1	1

$=$

Feature Map

0	1	0	0	0

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0				

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1			

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1		

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	0
1				

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	0
1	0			

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1		

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1				

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4			

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2		

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0				

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0			

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1		

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input
Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature
Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	

Feature Map

Step 1 - Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

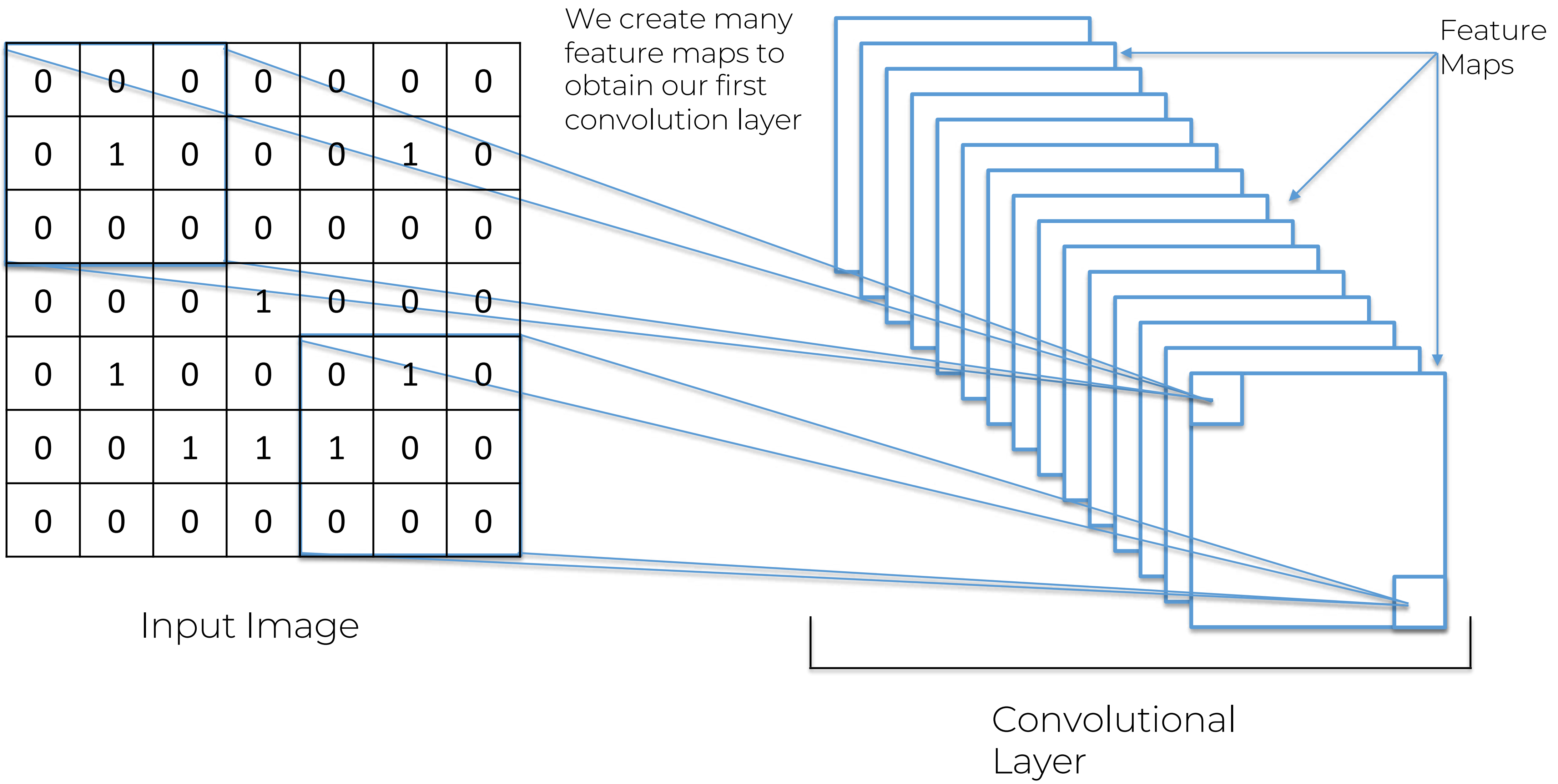
Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Step 1 - Convolution



Step 1 - Convolution

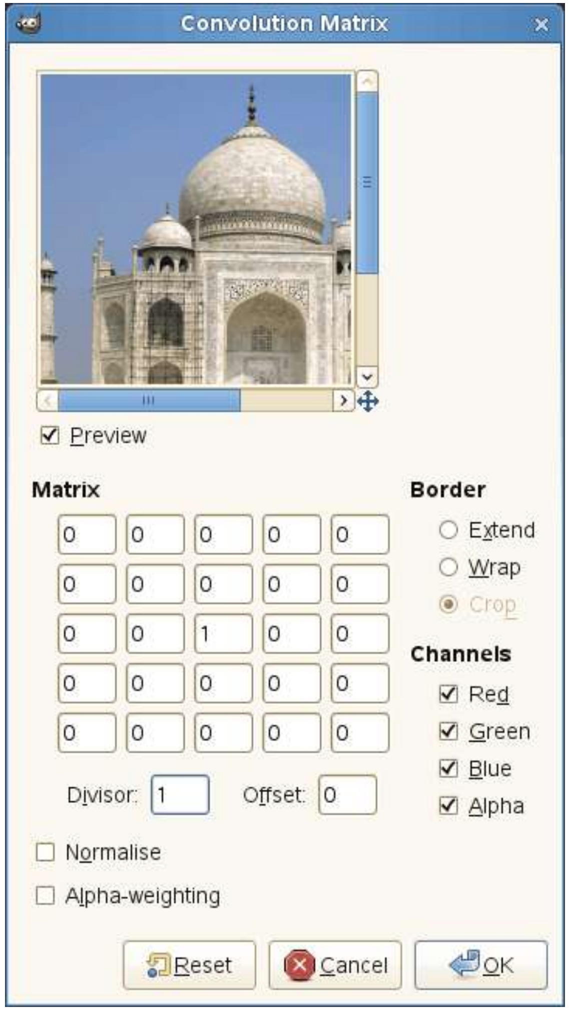


Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Sharpen:

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Blur:

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Edge Enhance:

	0	0	0	
	-1	1	0	
	0	0	0	

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Edge Detect:

	0	1	0	
	1	-4	1	
	0	1	0	

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

Step 1 - Convolution

Emboss:

	-2	-1	0	
	-1	1	1	
	0	1	2	

Image Source: docs.gimp.org/en/plug-in-convmatrix.html

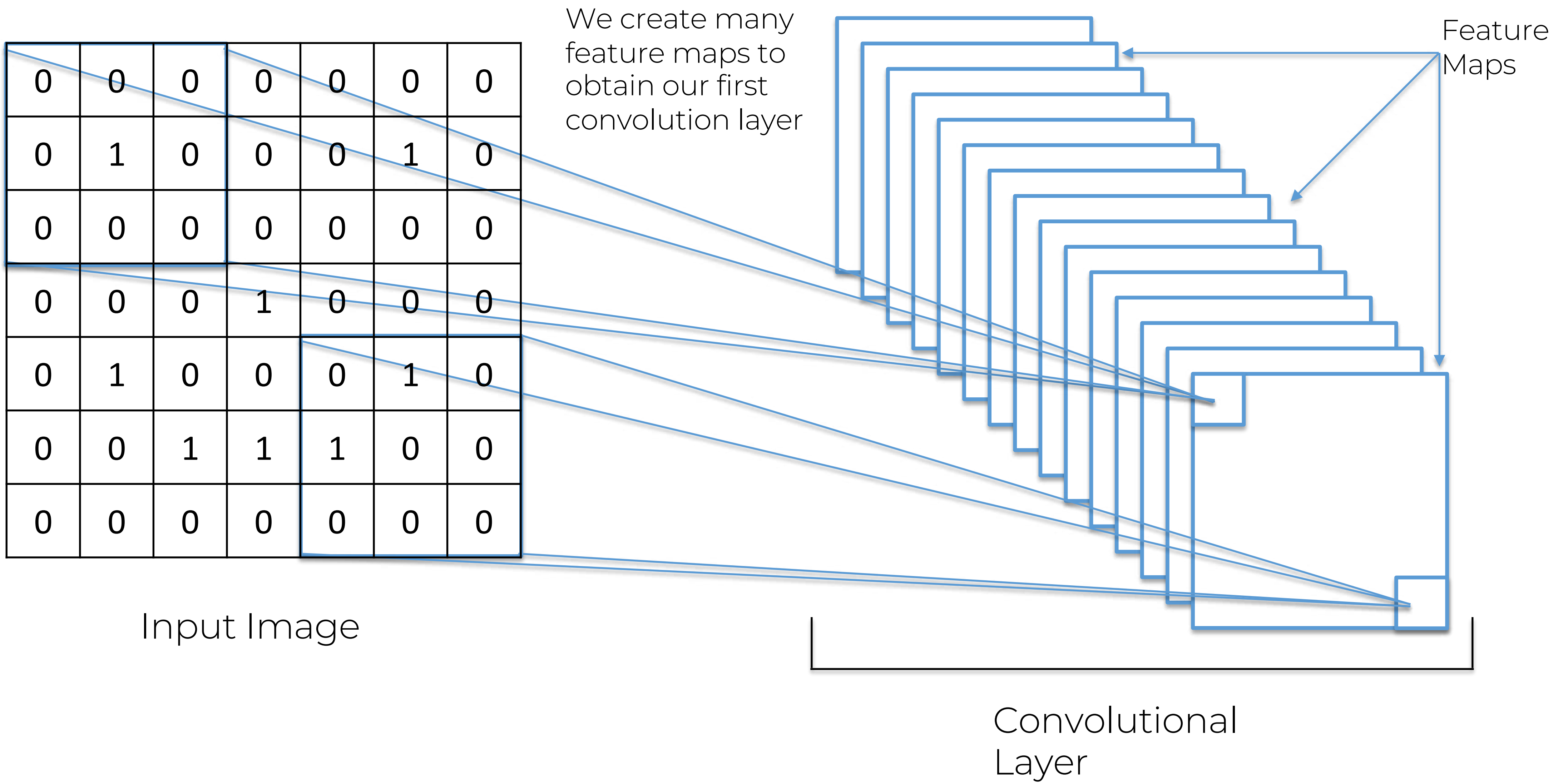
Step 1 - Convolution

$$* \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

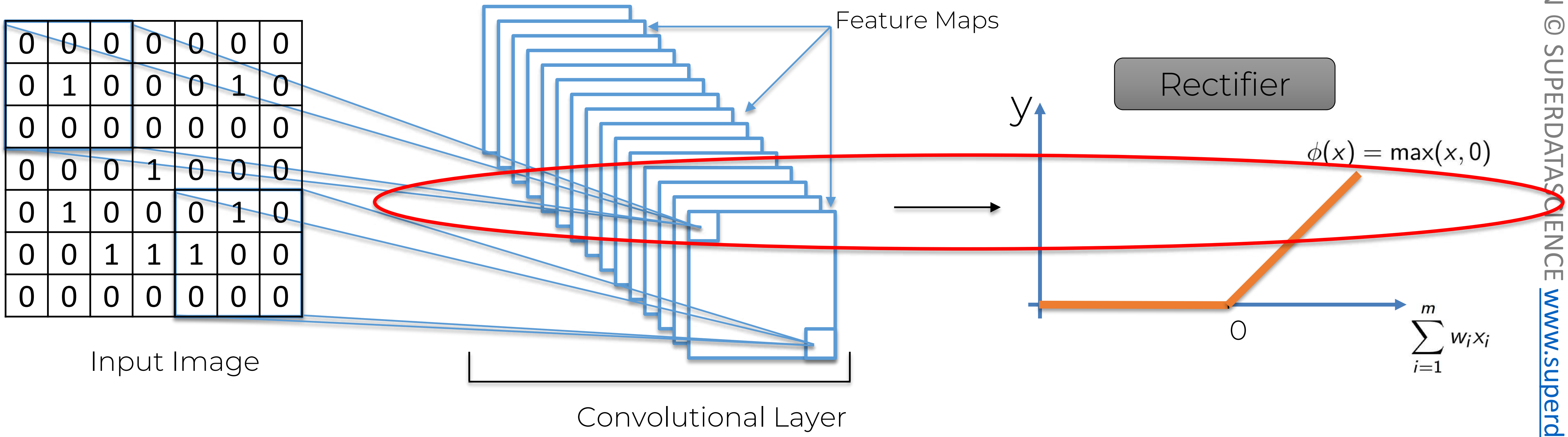
Image Source: eonardoaraujosantos.gitbooks.io

Step 1(B) – ReLU Layer

Step 1(B) – ReLU Layer



Step 1(B) – ReLU Layer



Step 1(B) – ReLU Layer

Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Step 1(B) – ReLU Layer

Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Step 1(B) – ReLU Layer

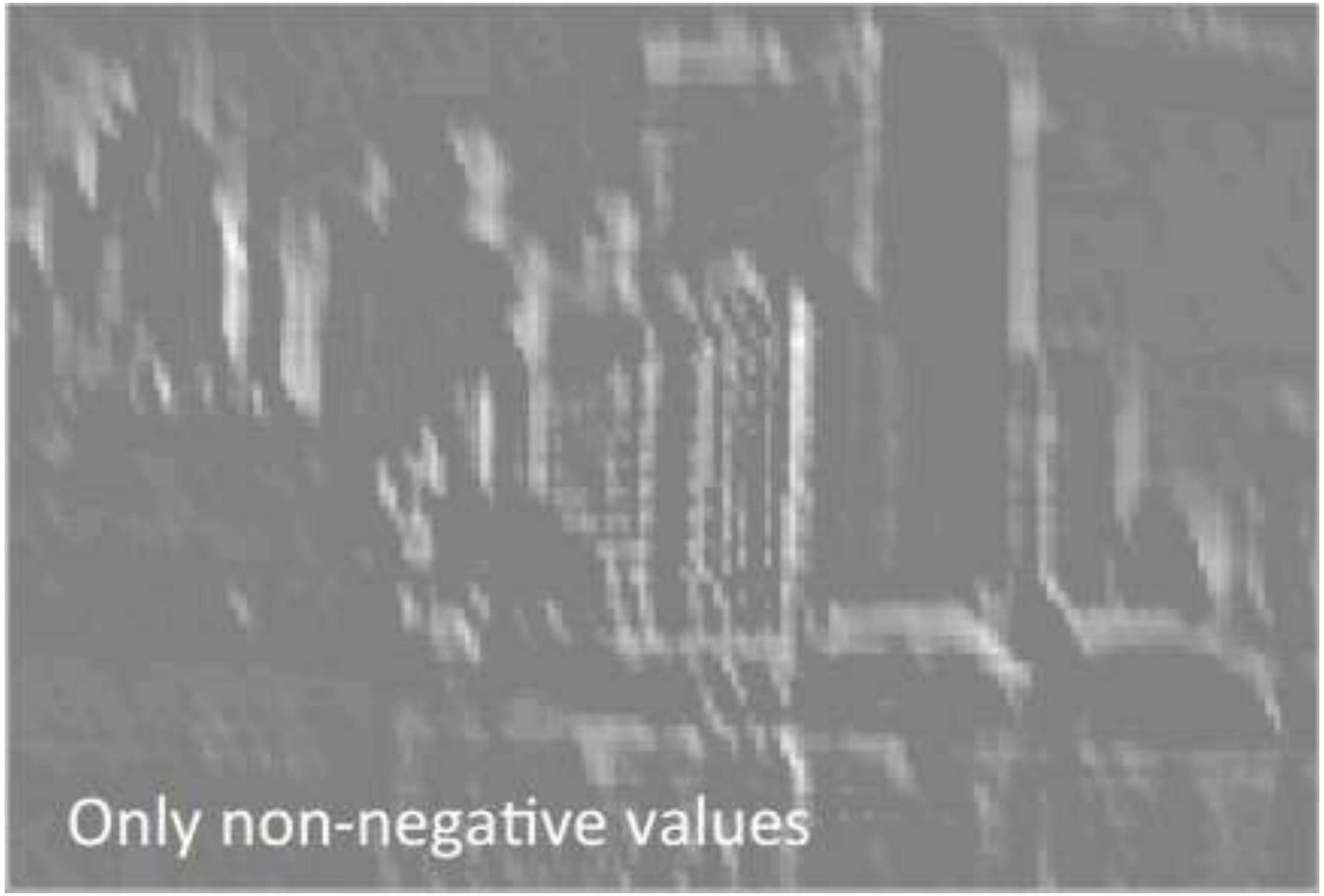


Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Step 1(B) – ReLU Layer

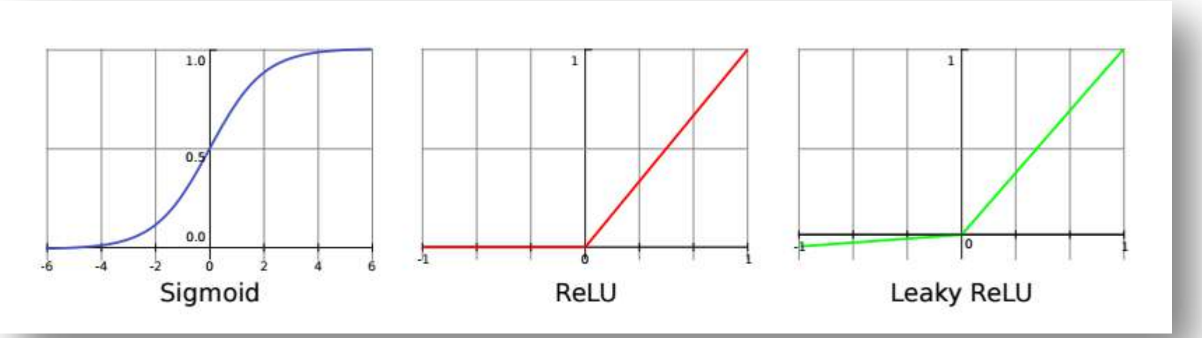
Additional Reading:

Understanding Convolutional Neural Networks with A Mathematical Model

By C.-C. Jay Kuo (2016)

Link:

https://arxiv.org/pdf/1609.04112.pdf



Step 1(B) – ReLU Layer

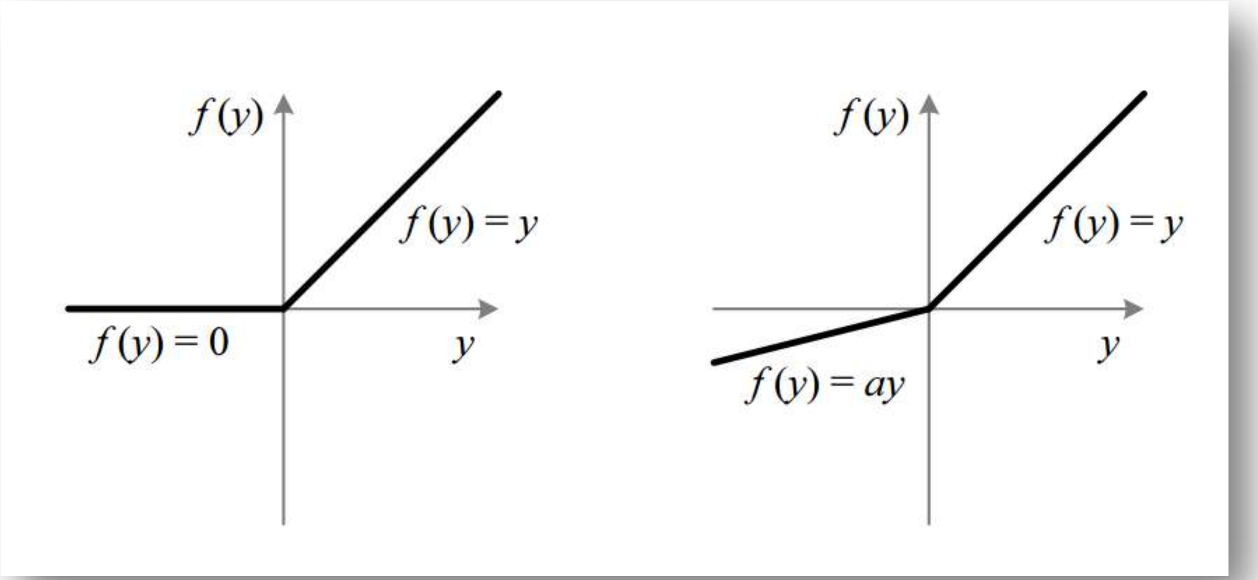
Additional Reading:

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

By Kaiming He et al. (2015)

Link:

https://arxiv.org/pdf/1502.01852.pdf



Step 2 – Max Pooling

Step 2 - Max Pooling

Image Source: Wikipedia

Step 2 - Max Pooling

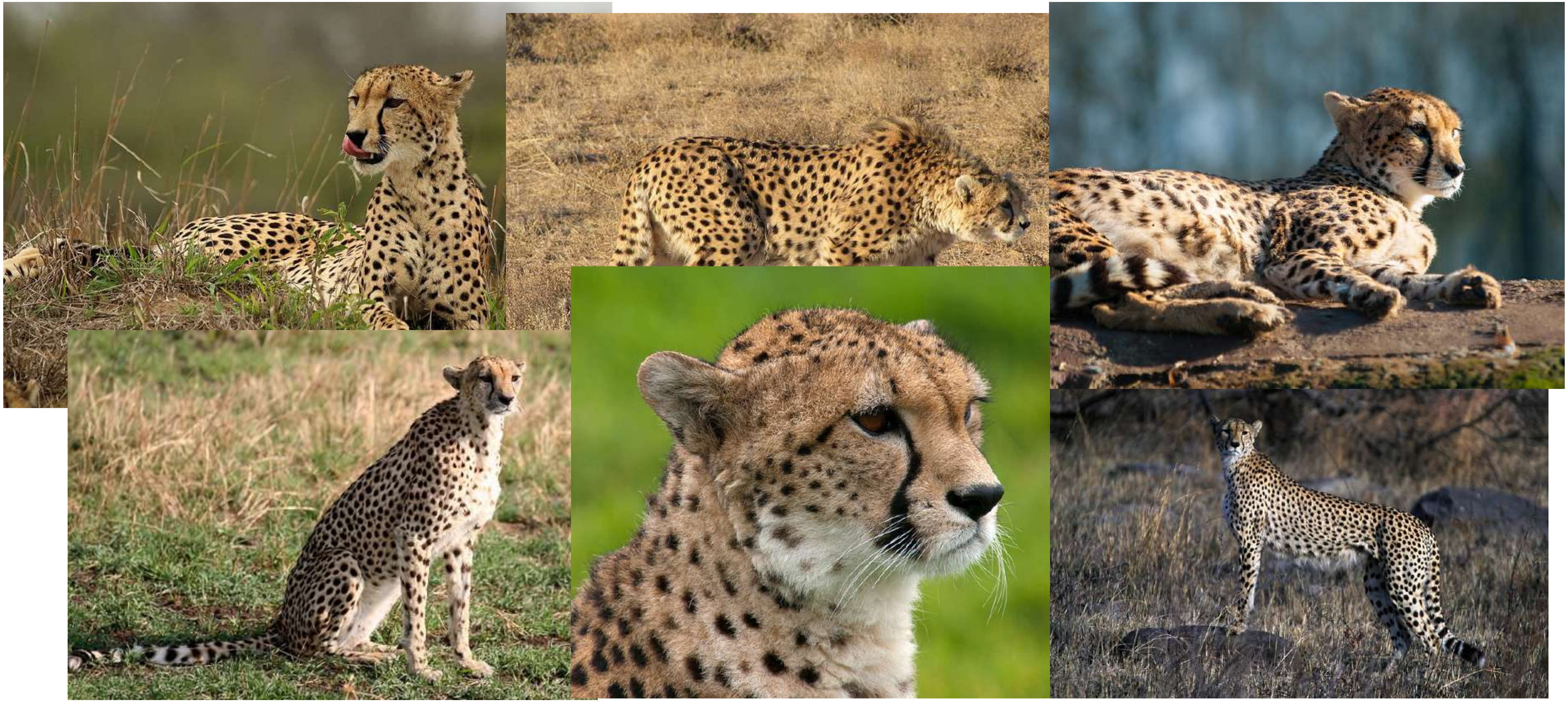


Image Source: Wikipedia

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

1		

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

1	1	

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

1	1	0

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling

1	1	0
4		

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling

1	1	0
4	2	

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

1	1	0
4	2	1

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

1	1	0
4	2	1
0		

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling →

1	1	0
4	2	1
0	2	

Pooled Feature Map

Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling $\longrightarrow$

1	1	0
4	2	1
0	2	1

Pooled Feature Map

Step 2 - Max Pooling

Additional Reading:

Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition

By Dominik Scherer et al. (2010)

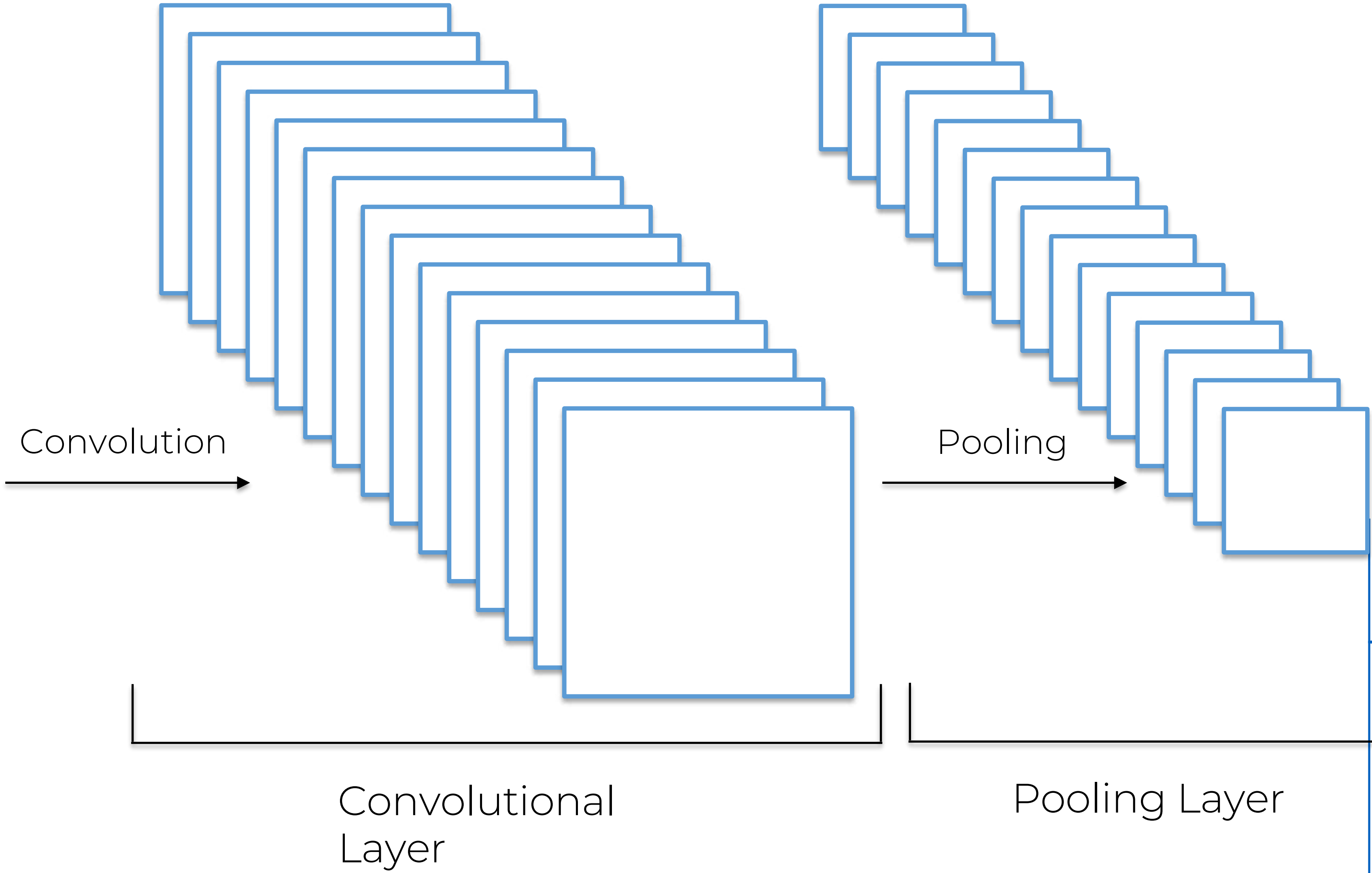
Link:

http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf

Step 2 - Max Pooling

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image



Example

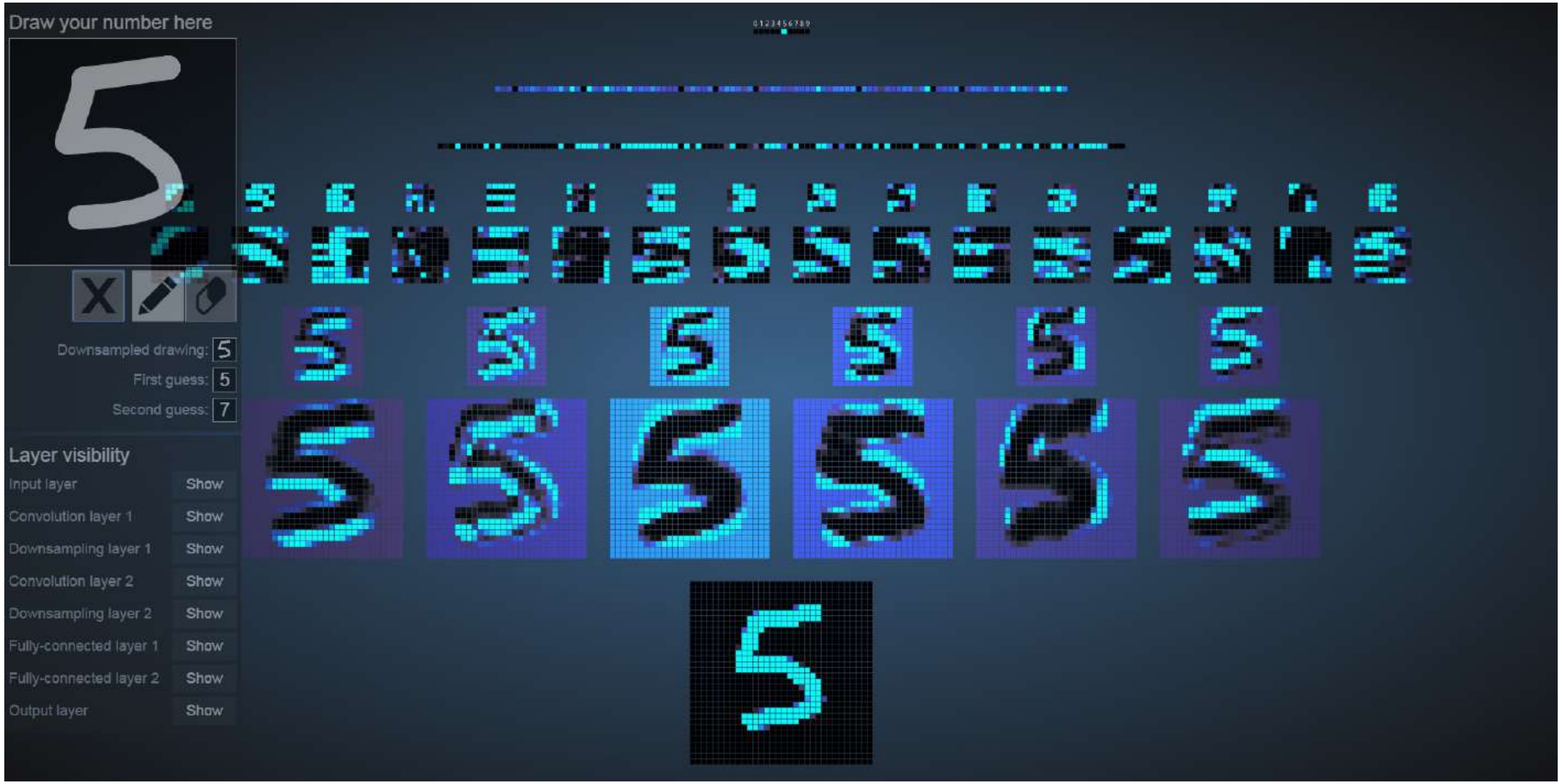


Image Source: scs.ryerson.ca/~aharley/vis/conv/flat.html

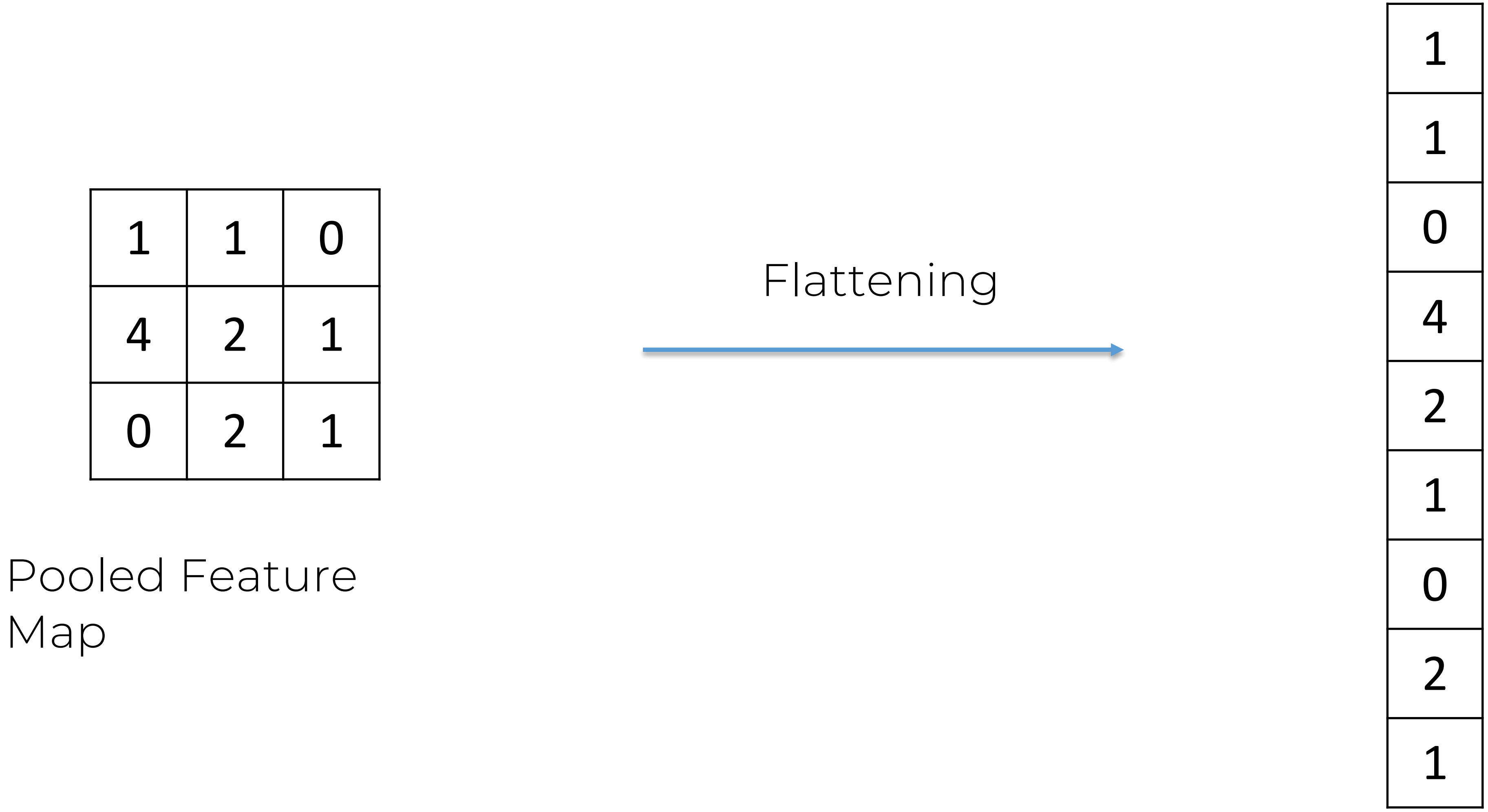
Step 3 – Flattening

Step 3 - Flattening

1	1	0
4	2	1
0	2	1

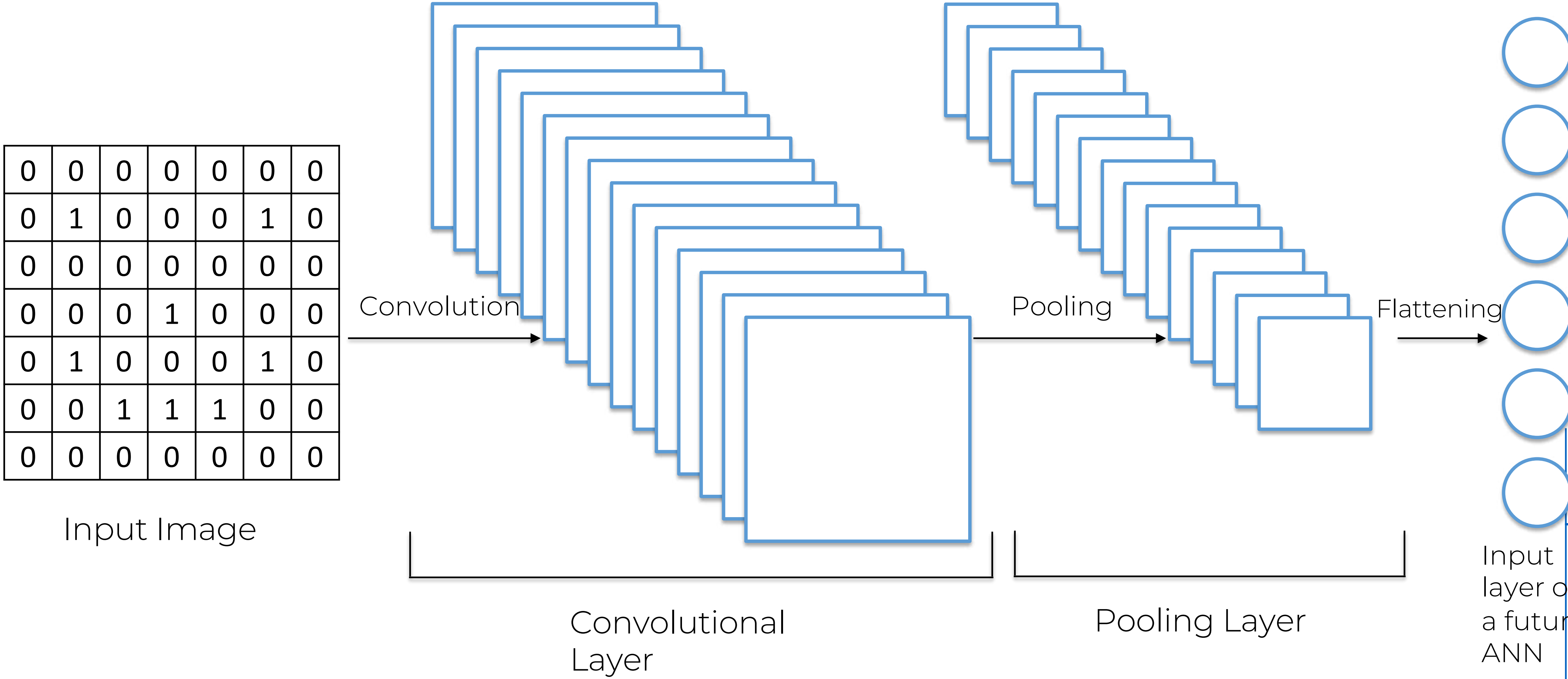
Pooled Feature Map

Step 3 - Flattening



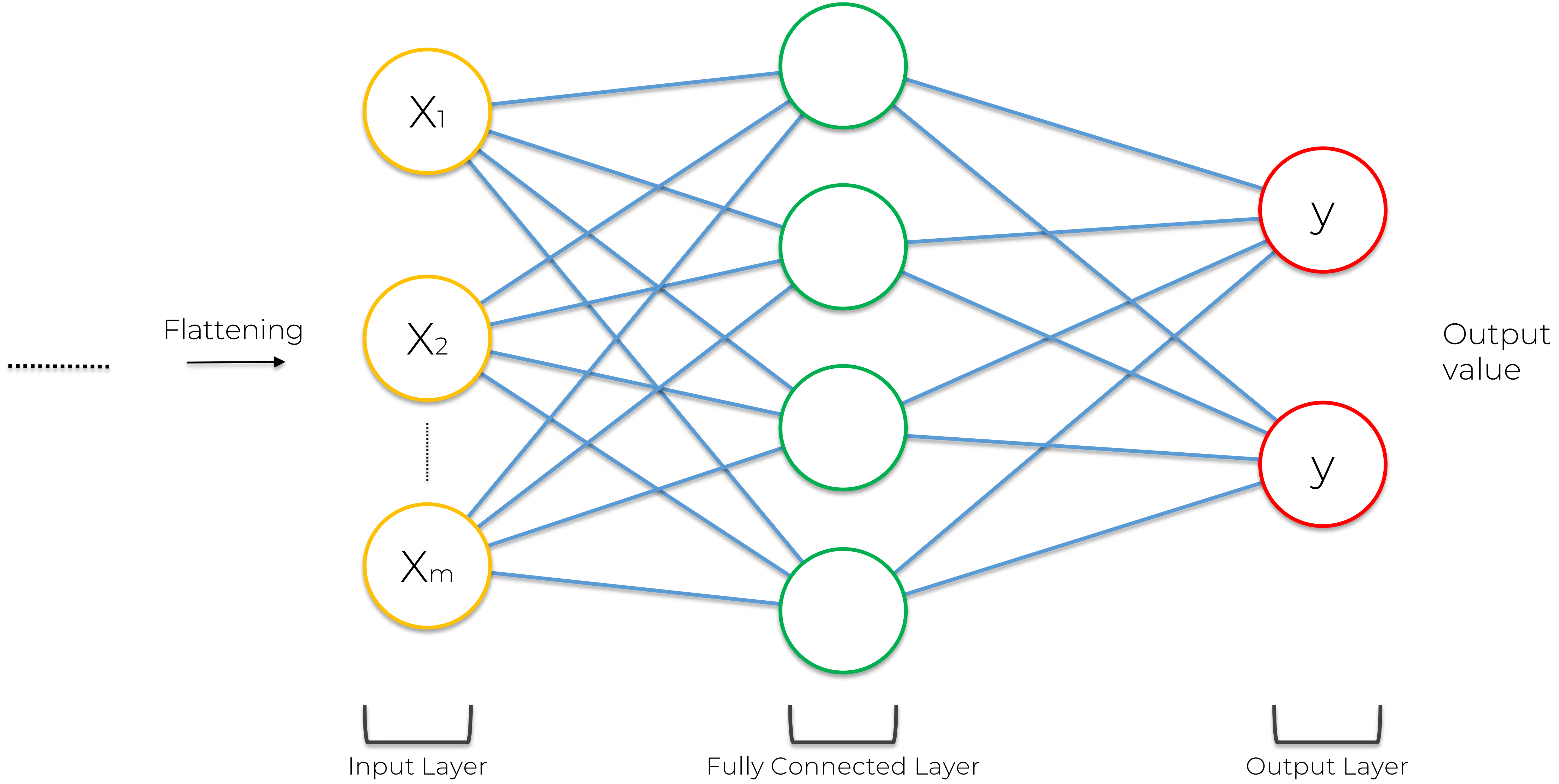
Step 3 - Flattening

Step 3 - Flattening

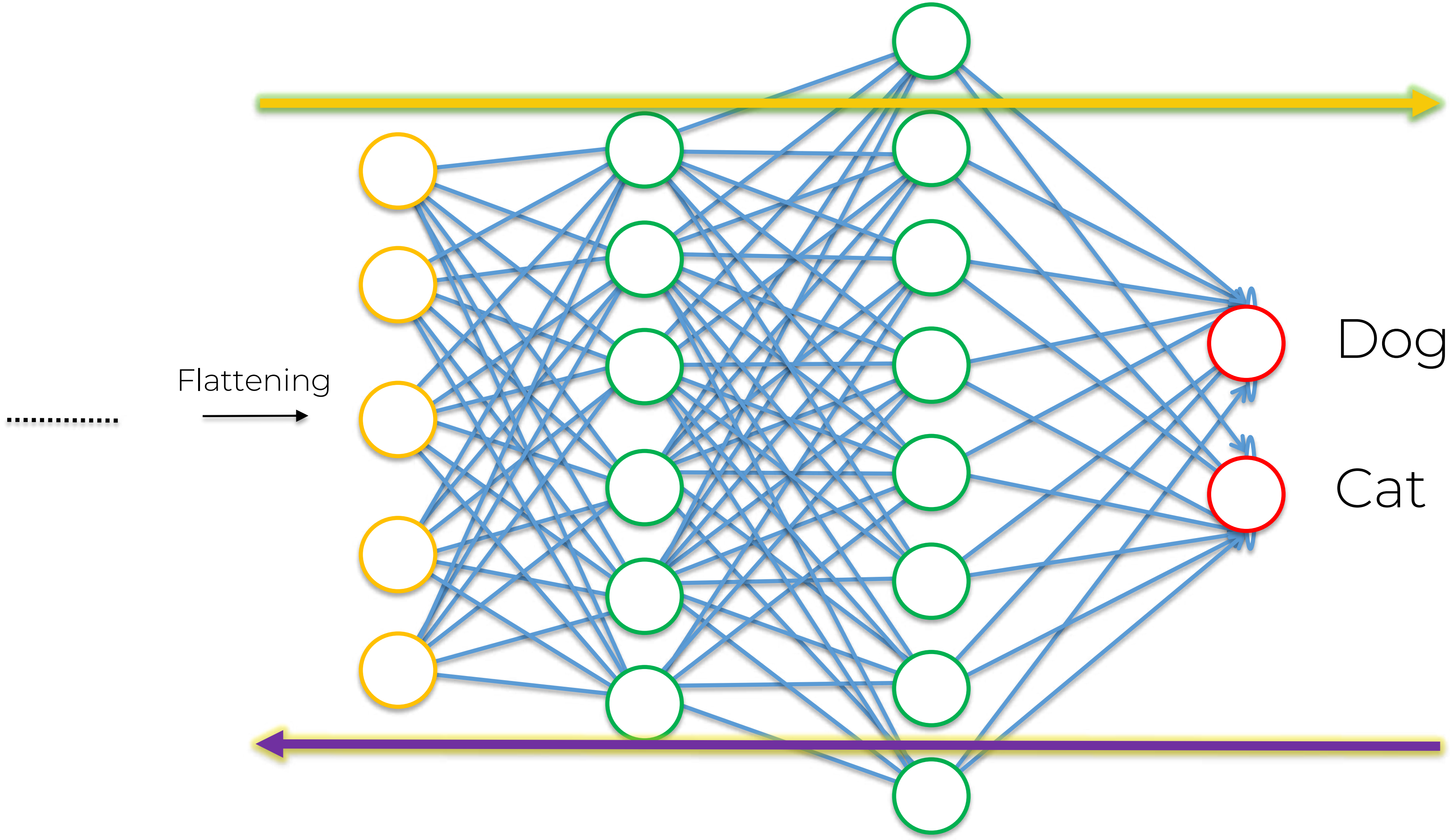


Step 4 – Full Connection

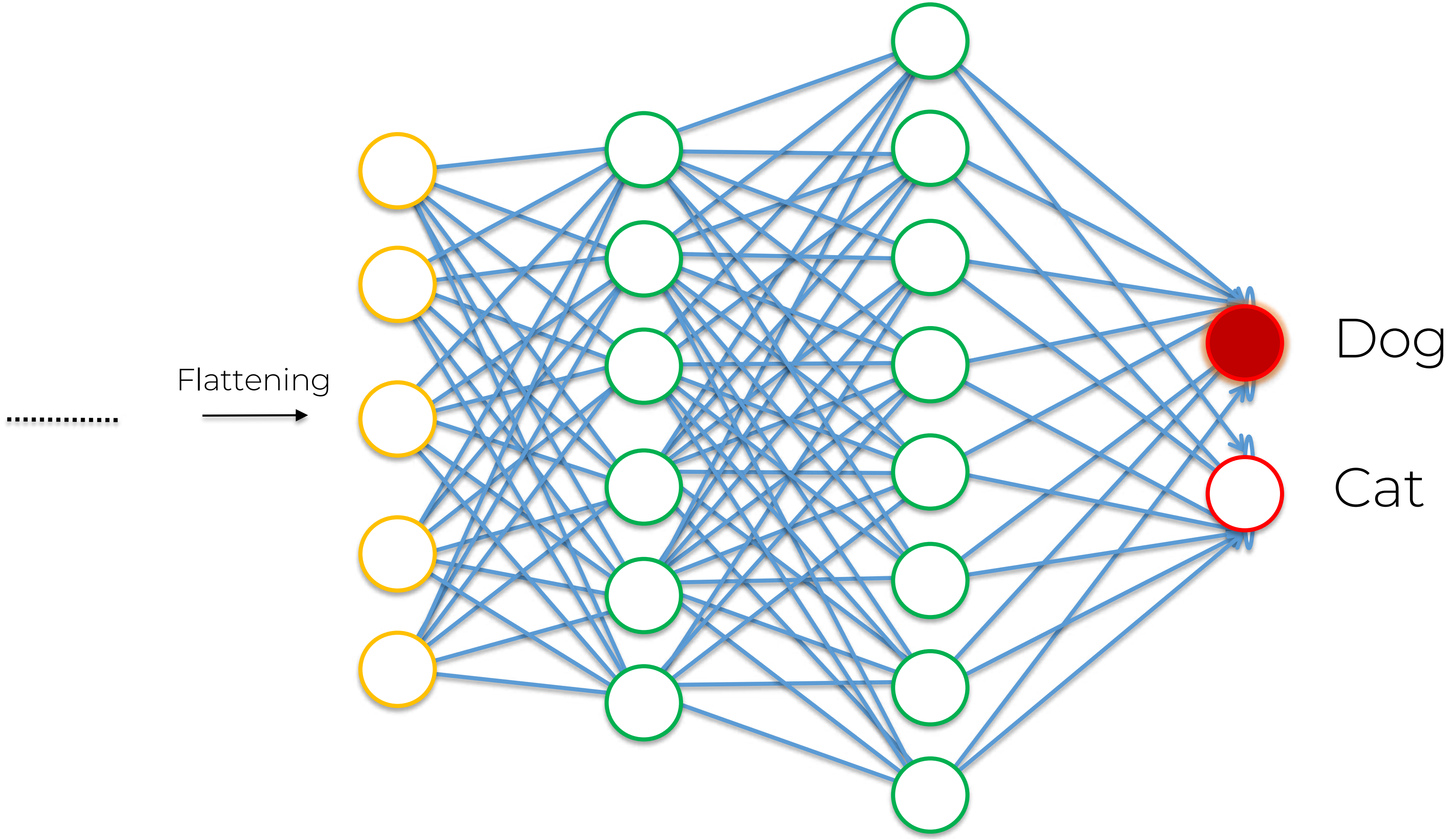
Step 4 - Full Connection



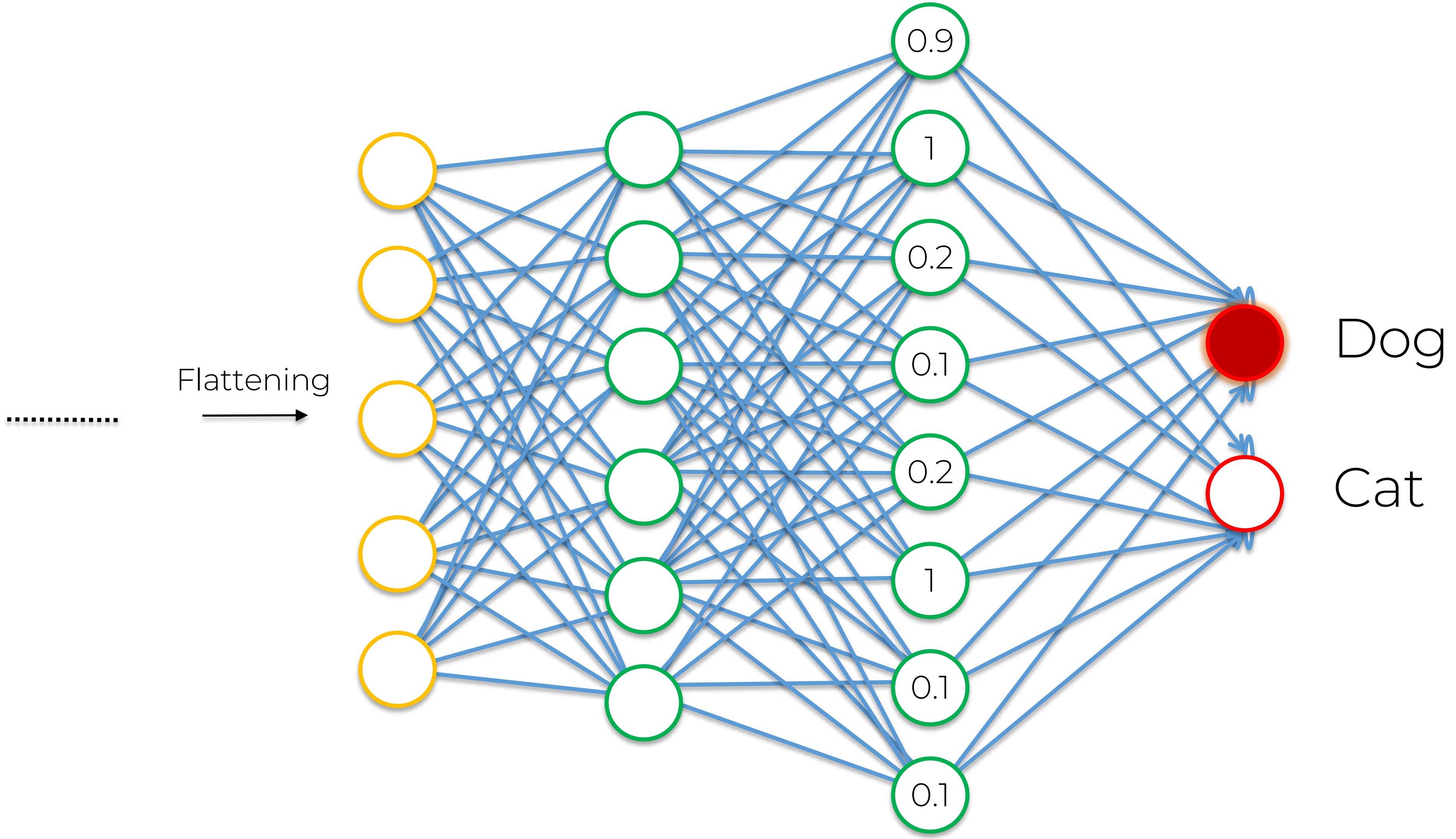
Step 4 - Full Connection



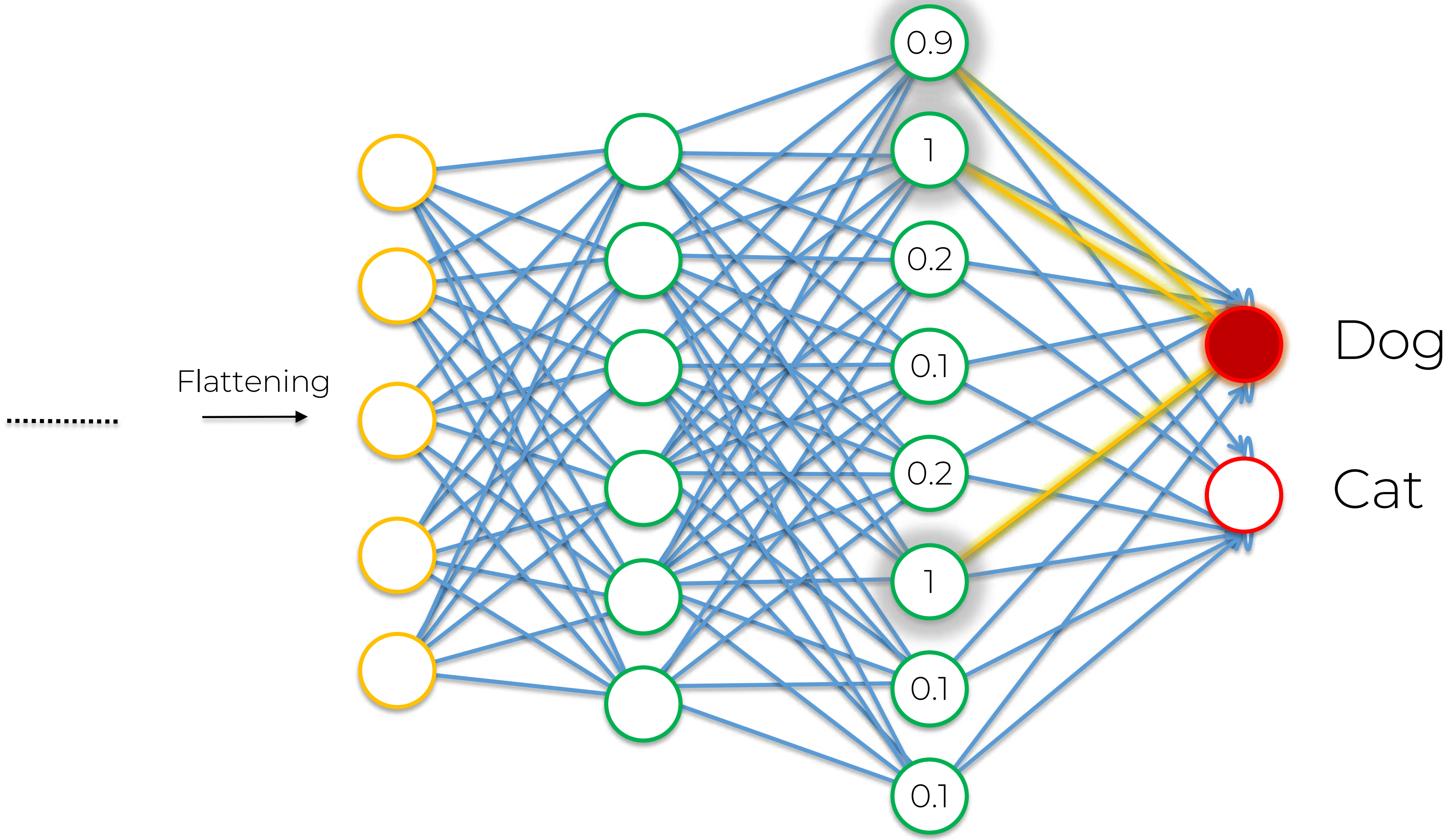
Step 4 - Full Connection



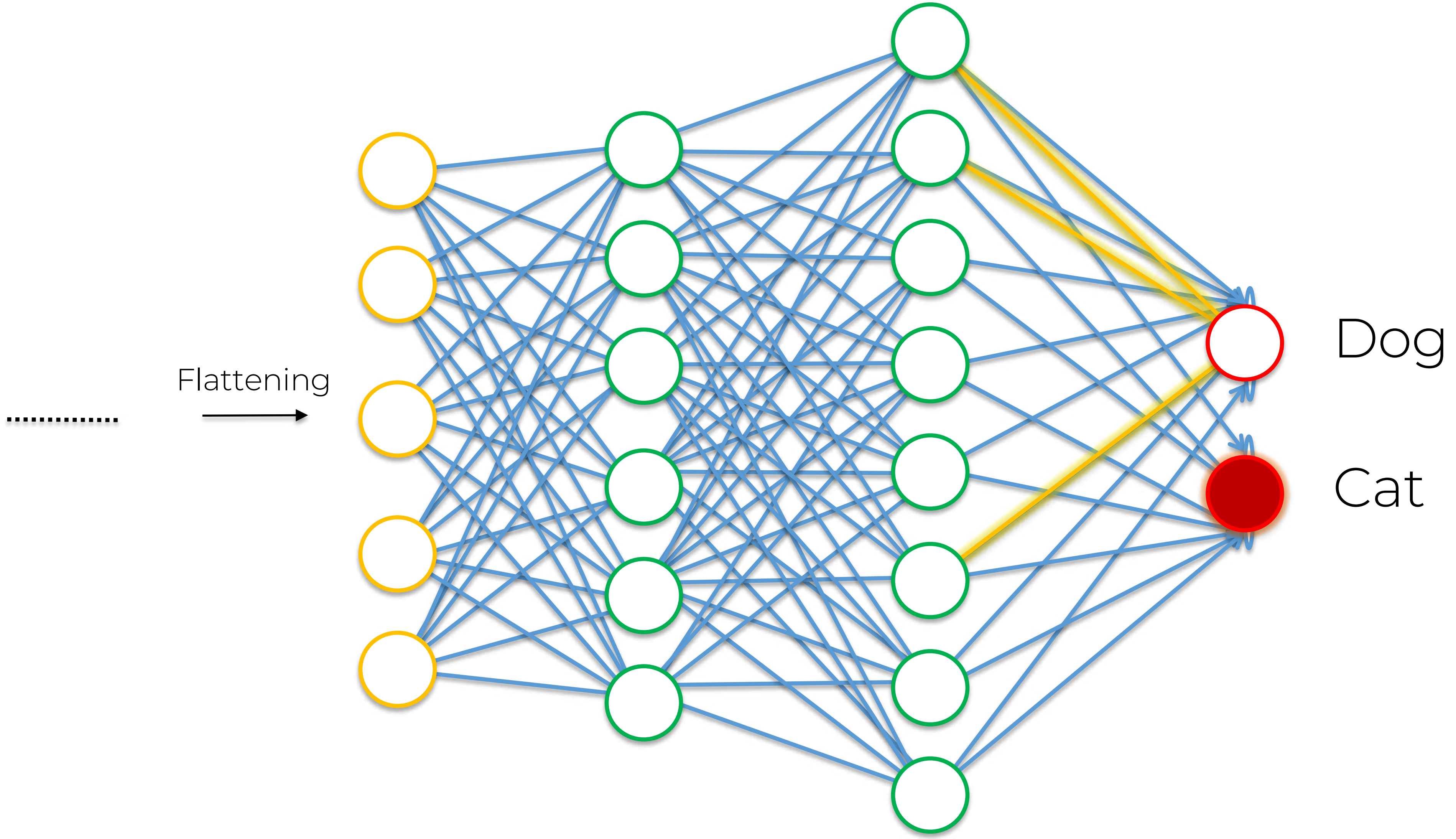
Step 4 - Full Connection



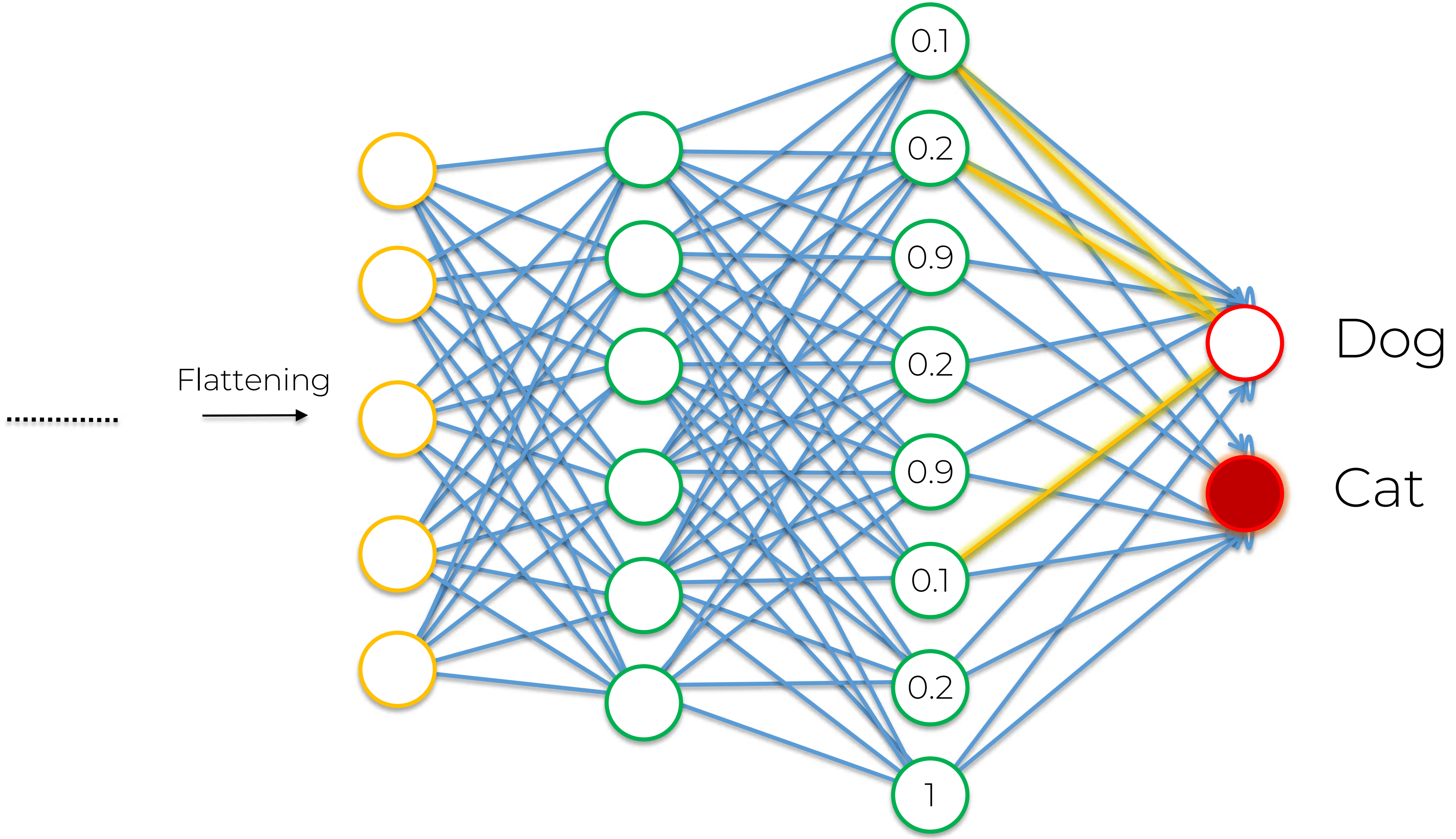
Step 4 - Full Connection



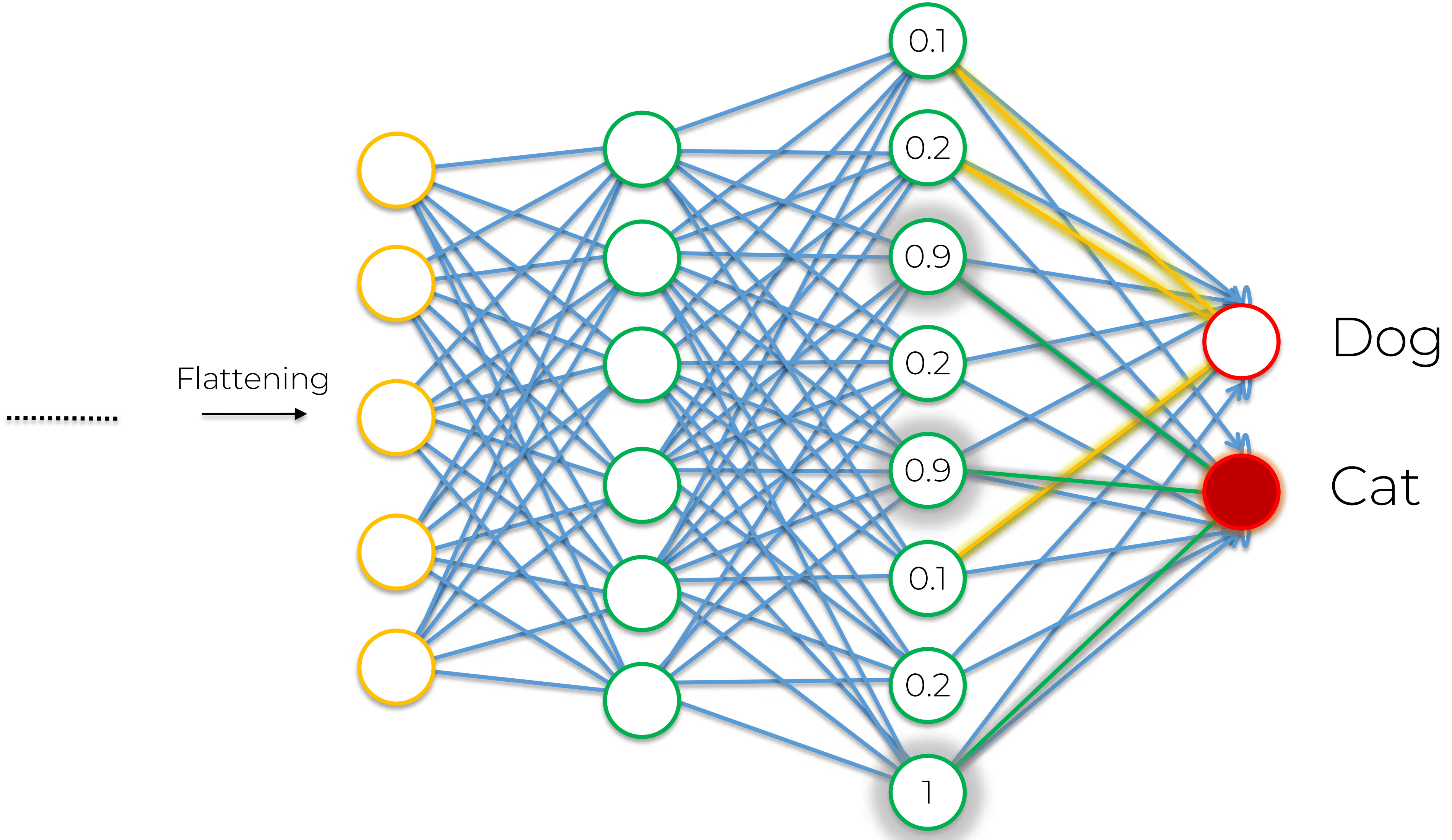
Step 4 - Full Connection



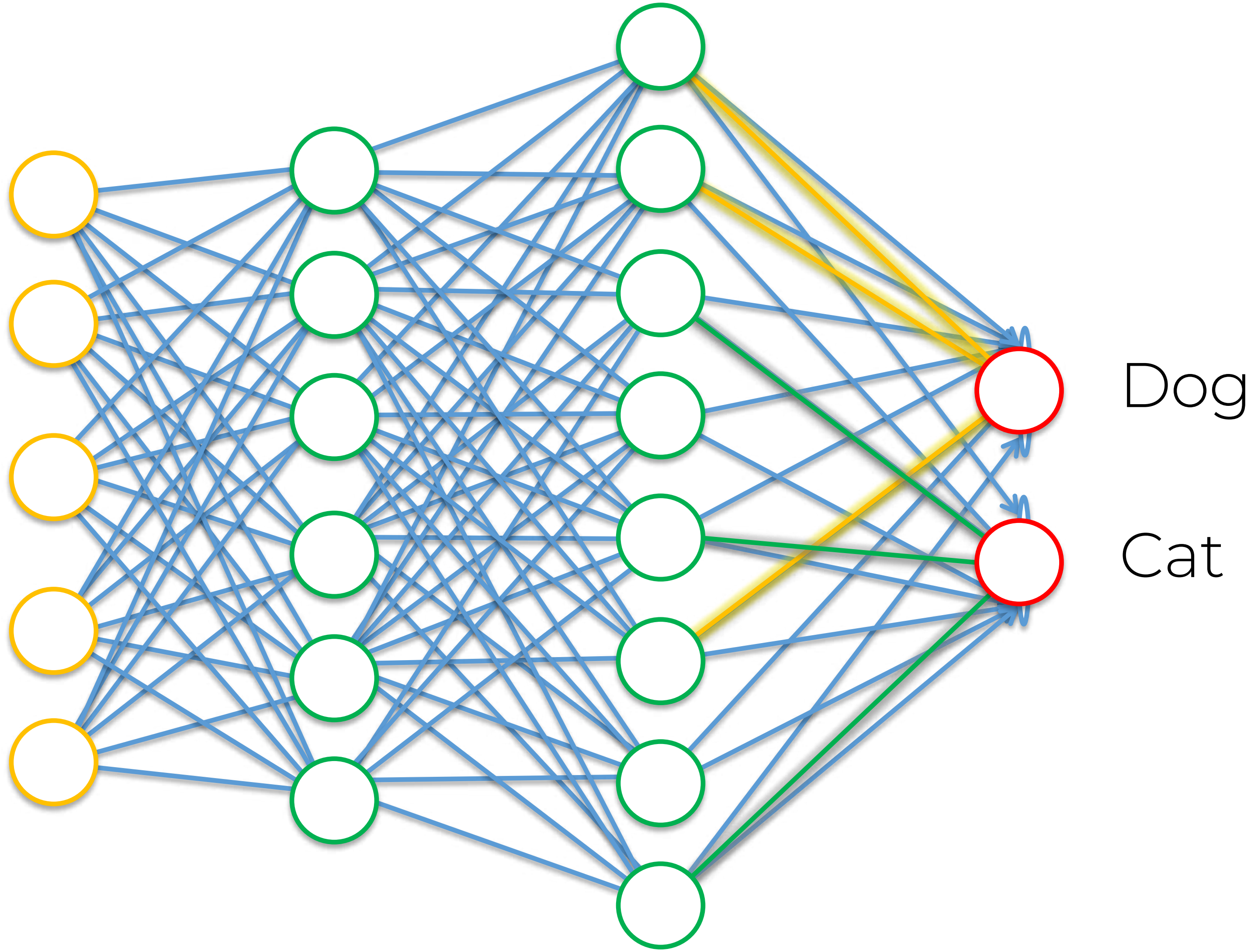
Step 4 - Full Connection



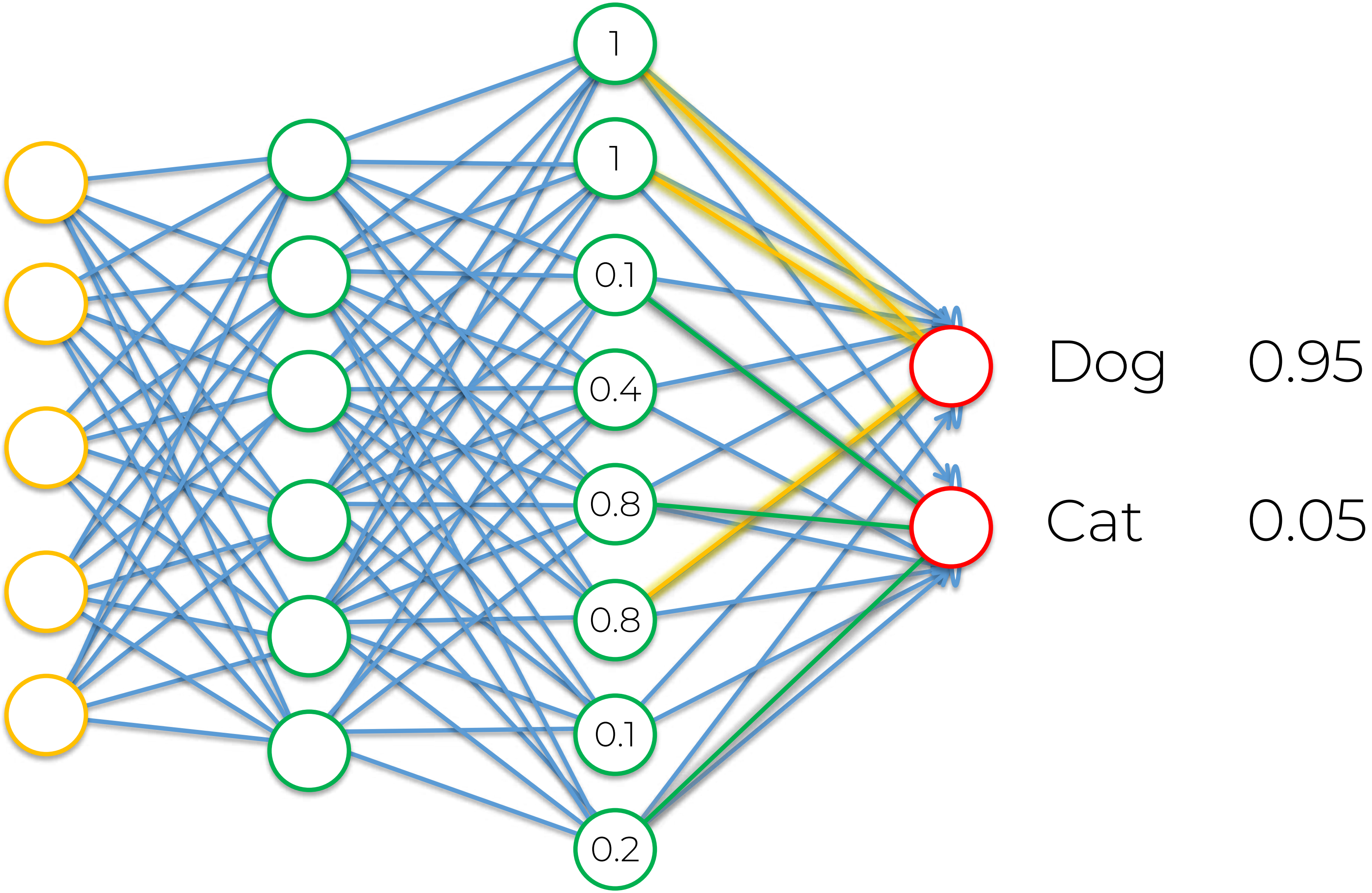
Step 4 - Full Connection



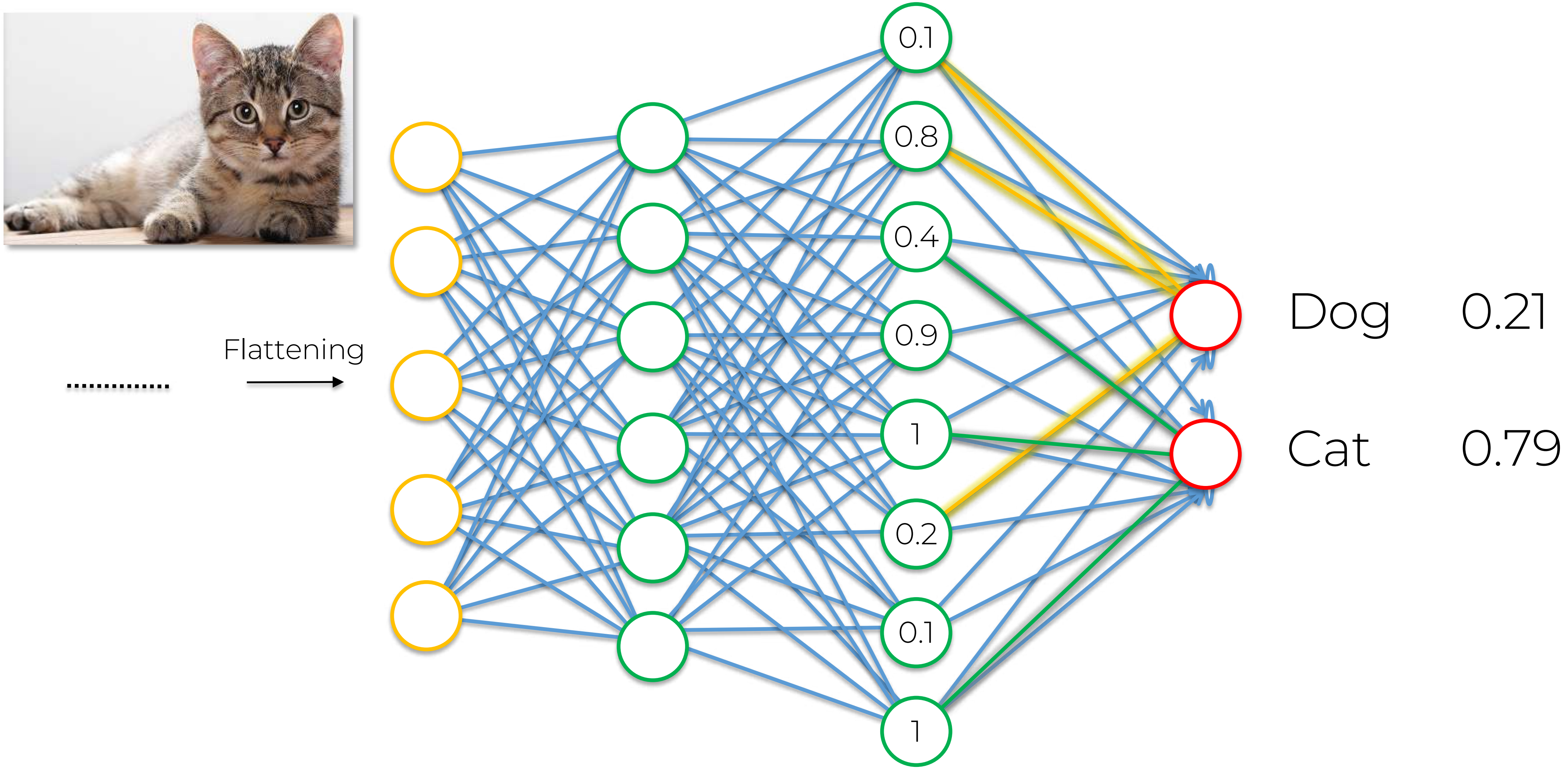
Step 4 - Full Connection



Step 4 - Full Connection



Step 4 - Full Connection



Step 4 - Full Connection

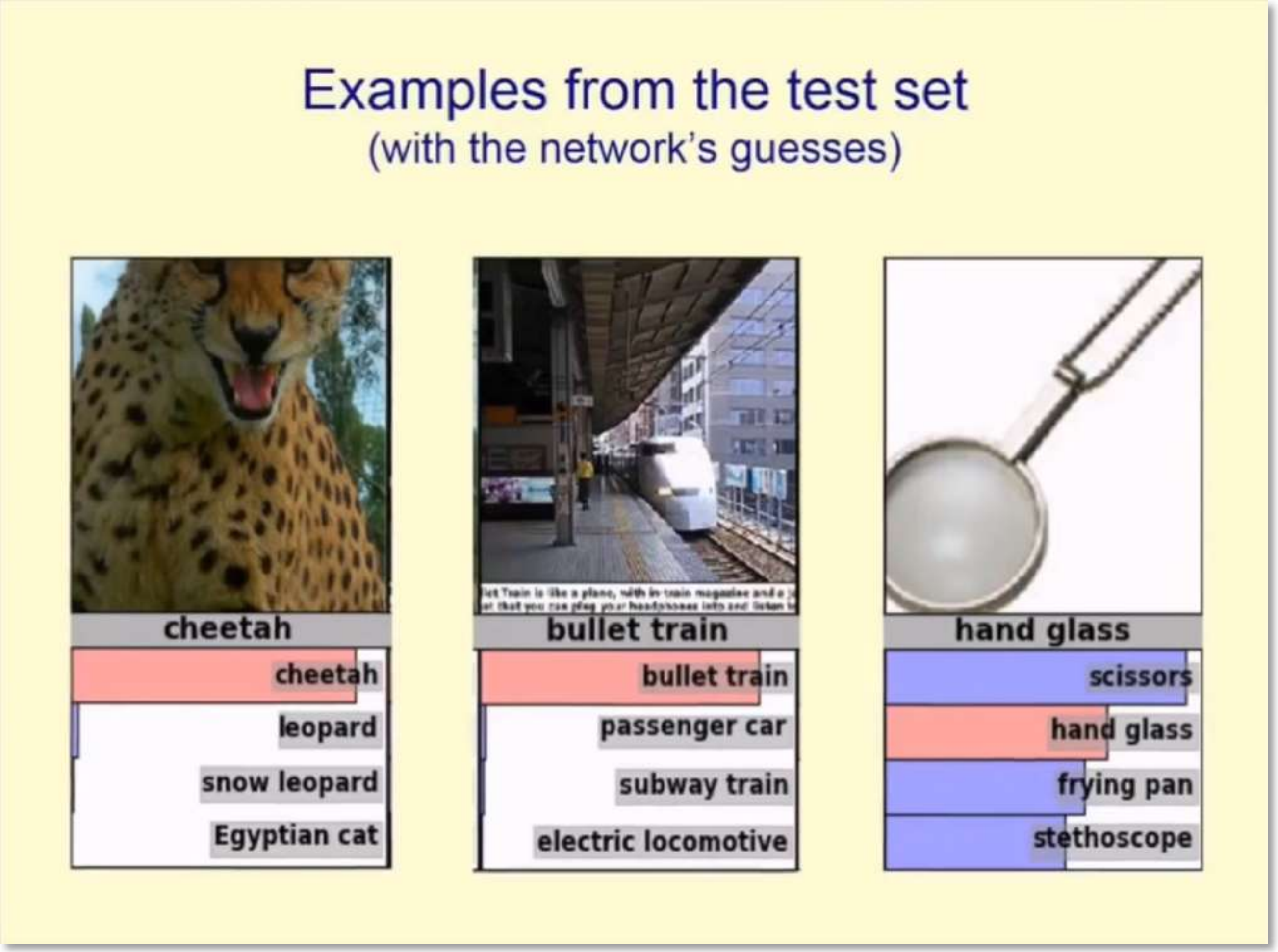
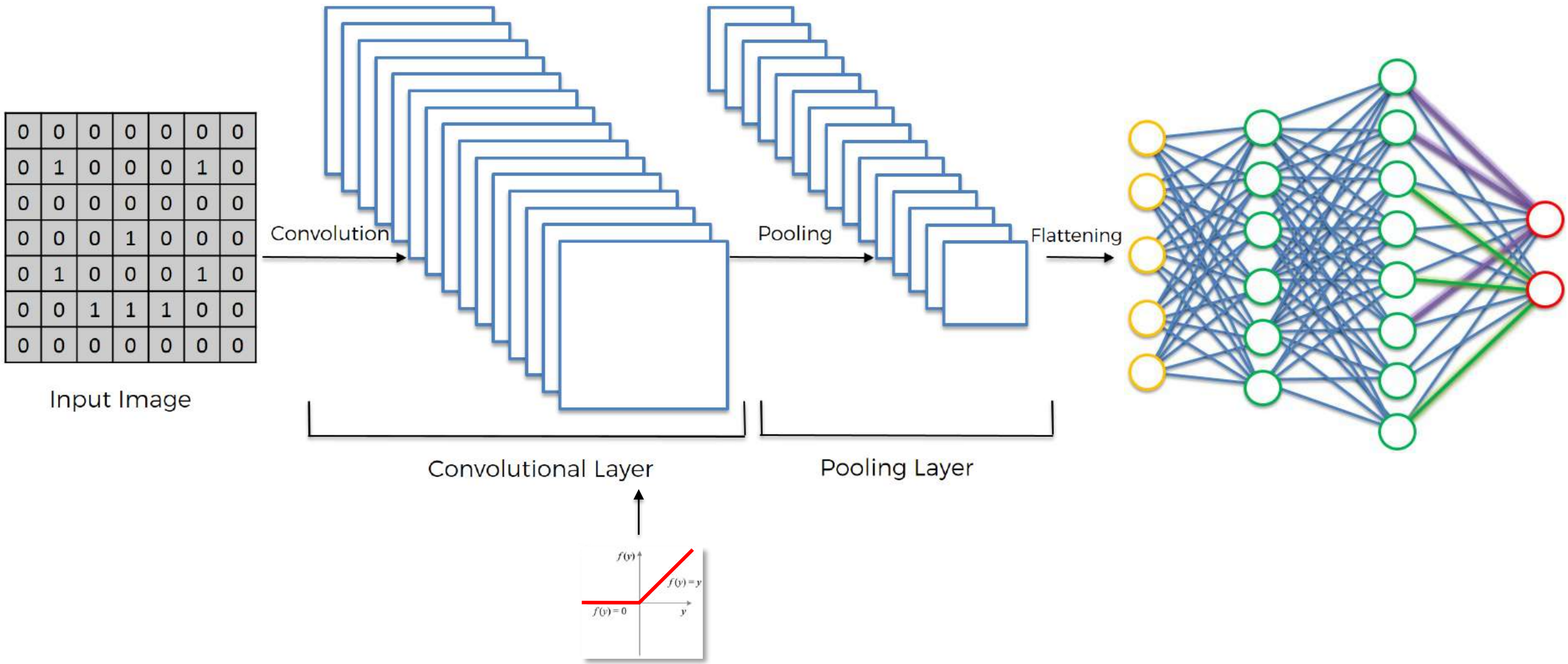


Image Source: a talk by Geoffrey Hinton

Summary

Summary



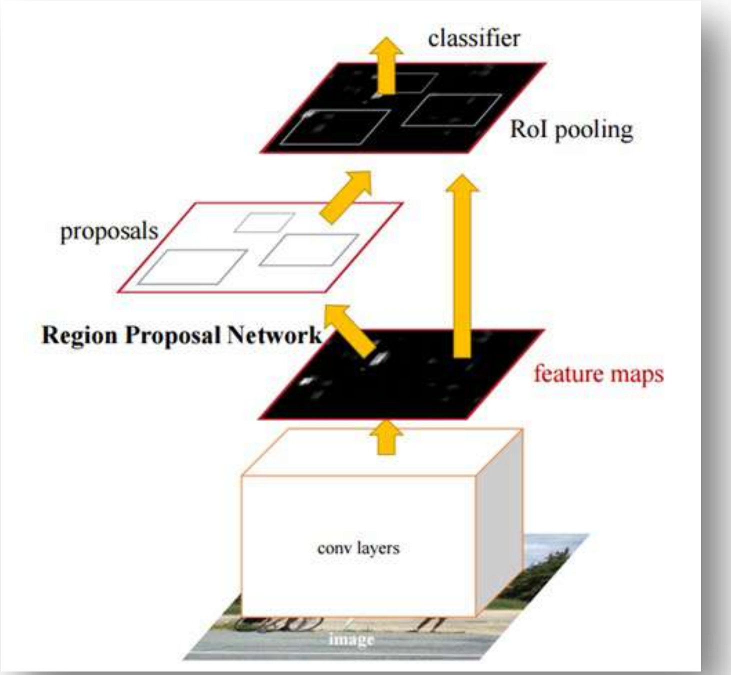
0 0 0 0 0 0 0
0 1 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 1 0
0 0 1 1 1 0 0
0 0 0 0 0 0 0
Input Image
Convolution
Pooling
Flattening
Convolutional Layer
Pooling Layer
f(y)
f(y) = y
f(y) = 0
y

Summary

Additional Reading:

The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)

Adit Deshpande (2016)

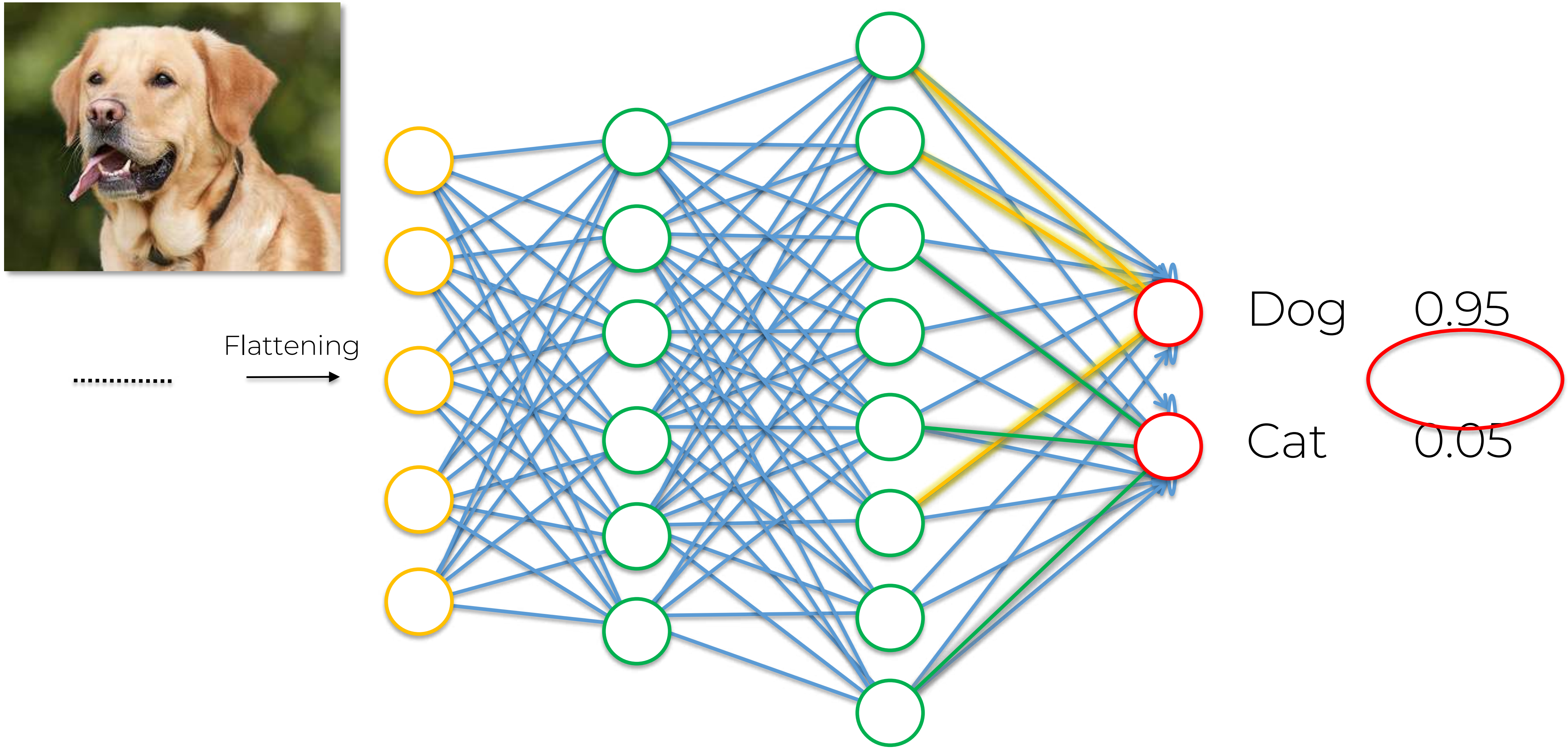


Link:

https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

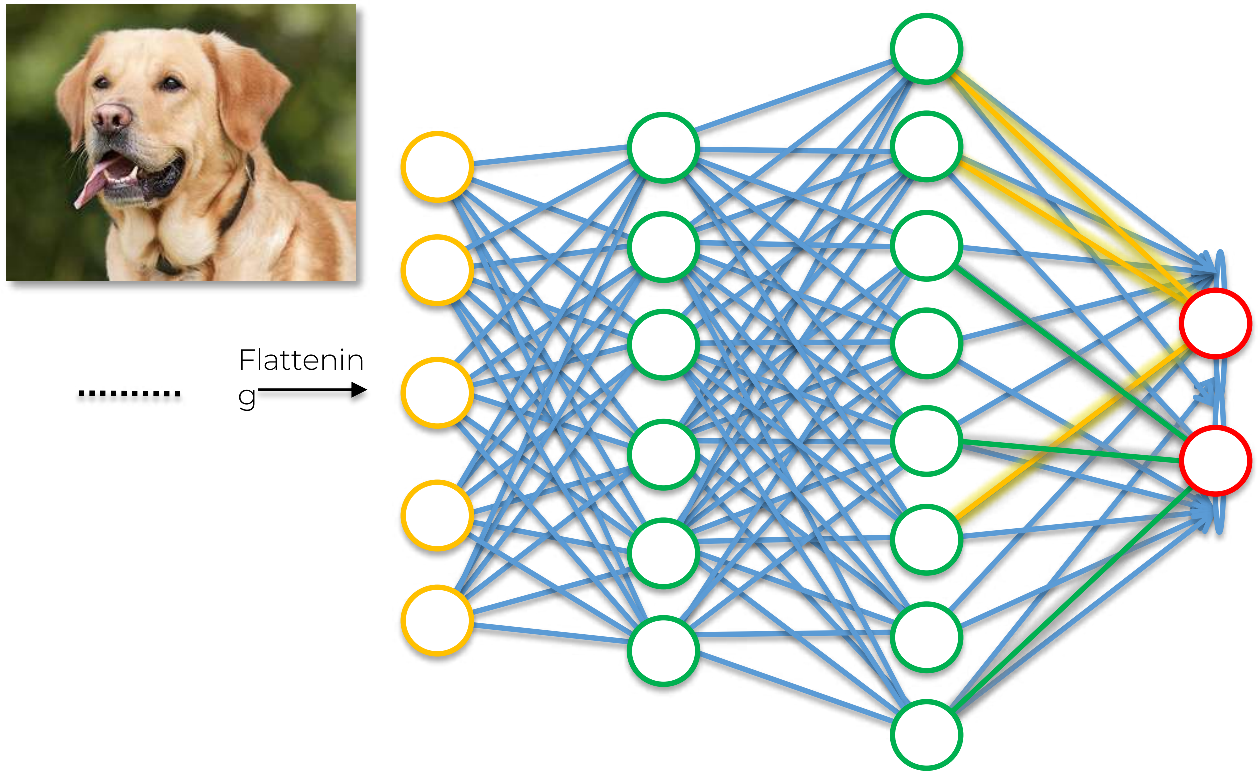
Softmax & Cross-Entropy

Softmax & Cross-Entropy



Softmax & Cross-Entropy

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



Dog ⟶ z_1 ⟶ 0.95

Cat ⟶ z_2 ⟶ 0.05

Softmax & Cross-Entropy

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

1
0

Softmax & Cross-Entropy

			NN1	NN2
	Dog	1	0.9	0.6
	Cat	0	0.1	0.4
	Dog	0	0.1	0.3
	Cat	1	0.9	0.7
	Dog	1	0.4	0.1
	Cat	0	0.6	0.9

Softmax & Cross-Entropy

NN1

Row	Dog ^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog ^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

NN1: 1/3 = 0.33 NN2: 1/3 = 0.33

Mean Squared Error

NN1: 0.25 NN2: 0.71

Cross-Entropy

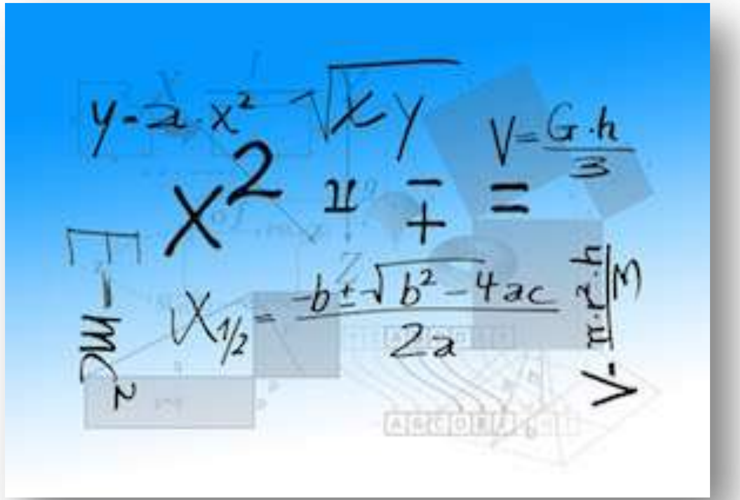
NN1: 0.38 NN2: 1.06

Softmax & Cross-Entropy

Additional Reading:

A Friendly Introduction to Cross-Entropy Loss

By Rob DiPietro (2016)



Link:

https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

Softmax & Cross-Entropy

Additional Reading:

How to implement a neural network Intermezzo 2

By Peter Roelants (2016)

$$
\begin{aligned}
\frac{\partial \xi}{\partial z_i} &= -\sum_{j=1}^{C} \frac{\partial t_j log(y_j)}{\partial z_i} = -\sum_{j=1}^{C} t_j \frac{\partial log(y_j)}{\partial z_i} = -\sum_{j=1}^{C} t_j \frac{1}{y_j} \frac{\partial y_j}{\partial z_i} \\
&= -\frac{t_i}{y_i} \frac{\partial y_i}{\partial z_i} - \sum_{j \neq i}^{C} \frac{t_j}{y_j} \frac{\partial y_j}{\partial z_i} = -\frac{t_i}{y_i} y_i (1 - y_i) - \sum_{j \neq i}^{C} \frac{t_j}{y_j} (-y_j y_i) \\
&= -t_i + t_i y_i + \sum_{j \neq i}^{C} t_j y_i = -t_i + \sum_{j=1}^{C} t_j y_i = -t_i + y_i \sum_{j=1}^{C} t_j \\
&= y_i - t_i
\end{aligned}
$$

Link:

http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/