

Machine Learning A-Z Course Slides

Welcome to the course!



Dear student,

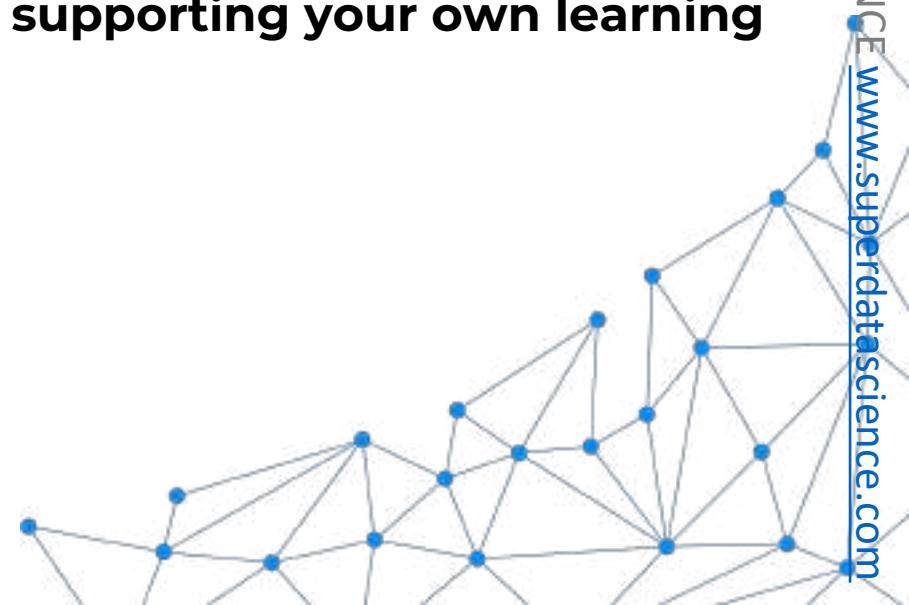
Welcome to the “Machine Learning A-Z” course brought to you by [SuperDataScience](#). We are super-excited to have you on board! In this class you will learn many interesting and useful concepts while having lots of fun.

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



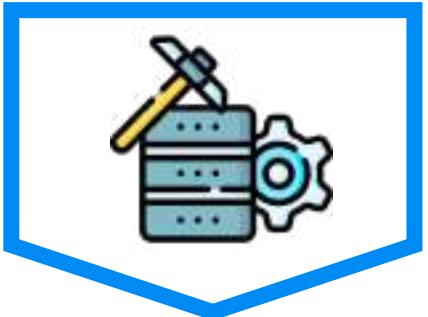
The Machine Learning Process

The Machine Learning Process



DISTRIBUTION © SUPERDATASCIENCE

www.superdatascience.com



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

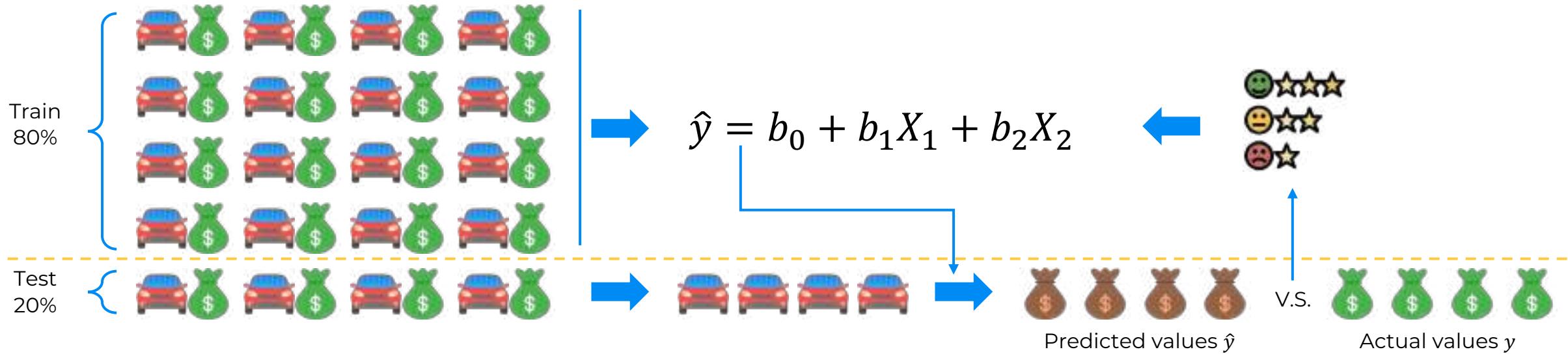




Training Set & Test Set



~





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]

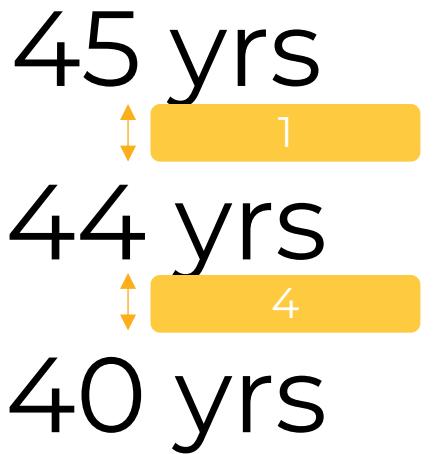
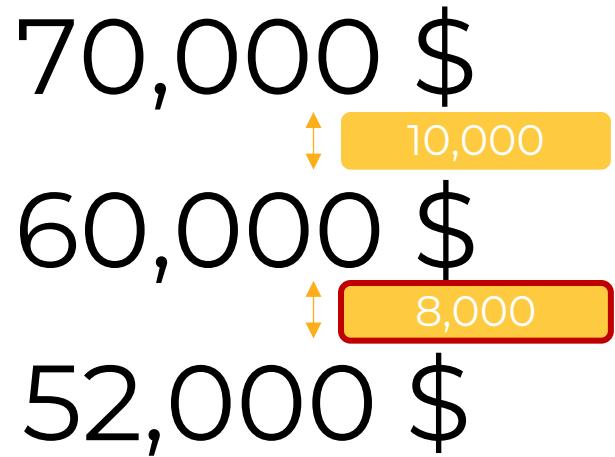
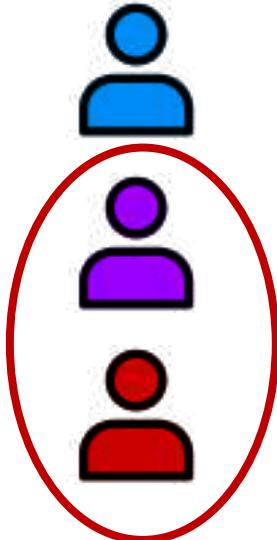


Feature Scaling



NOT FOR DISTRIBUTION

© SUPERDATASCIENCE

www.superdatascience.com

Feature Scaling

Normalization

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

[0 ; 1]



Feature Scaling



NOT FOR DISTRIBUTION

© SUPERDATASCIENCE

www.superdatascience.com



70,000 \$
60,000 \$
52,000 \$



45 yrs
44 yrs
40 yrs



Feature Scaling



1
0.444
0



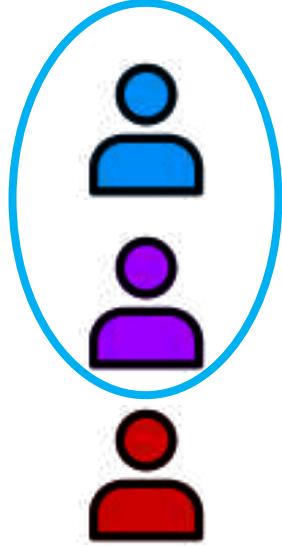
45 yrs
44 yrs
40 yrs



Feature Scaling

NOT FOR
DISTRIBUTION

© SUPERDATASCIENCE

www.superdatascience.com

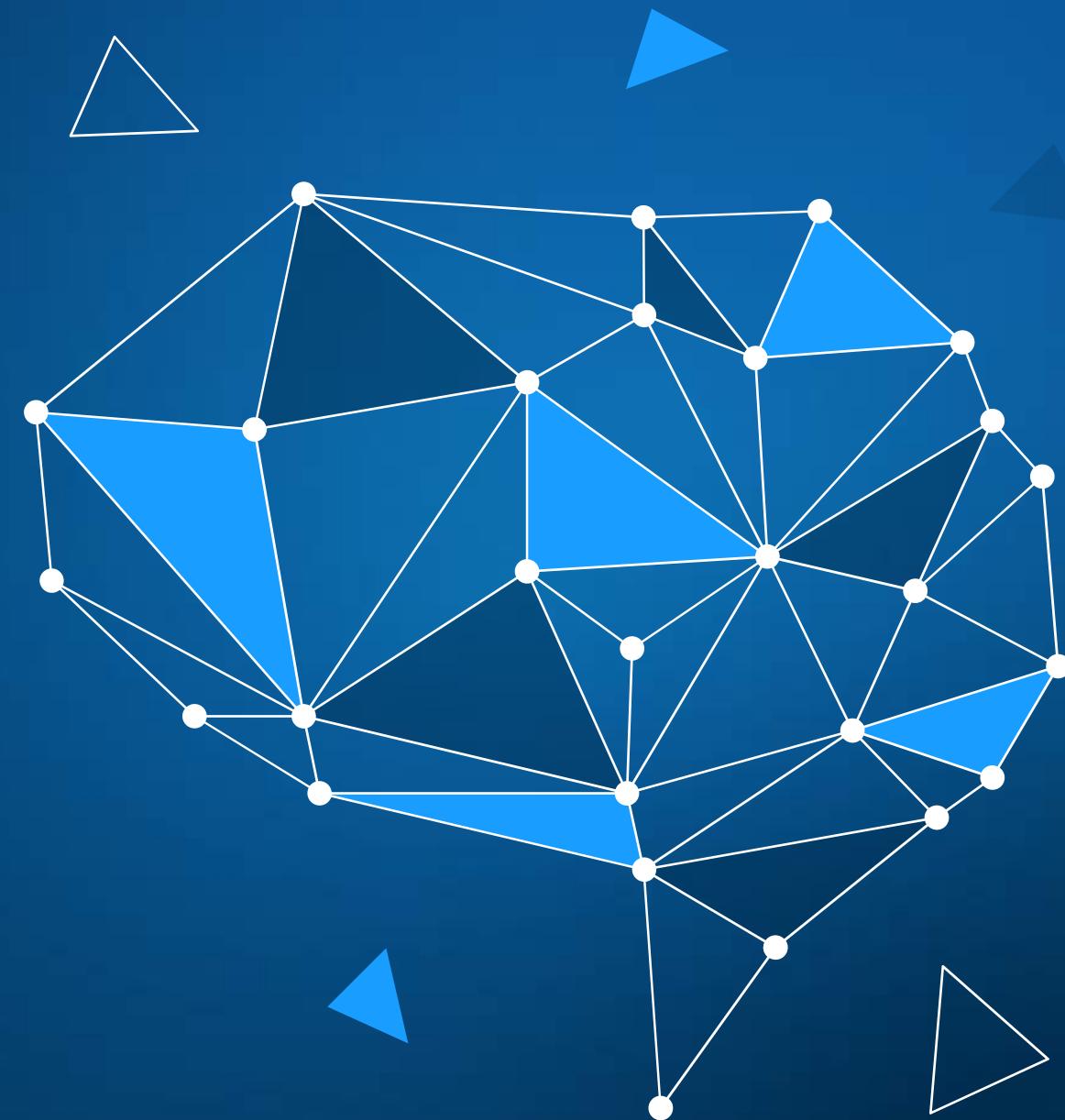
1
0.444
0



1
0.75
0



Regression



Simple Linear Regression





Simple Linear Regression

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

Dependent variable

y-intercept (constant)

Independent variable

Slope coefficient





Simple Linear Regression



~

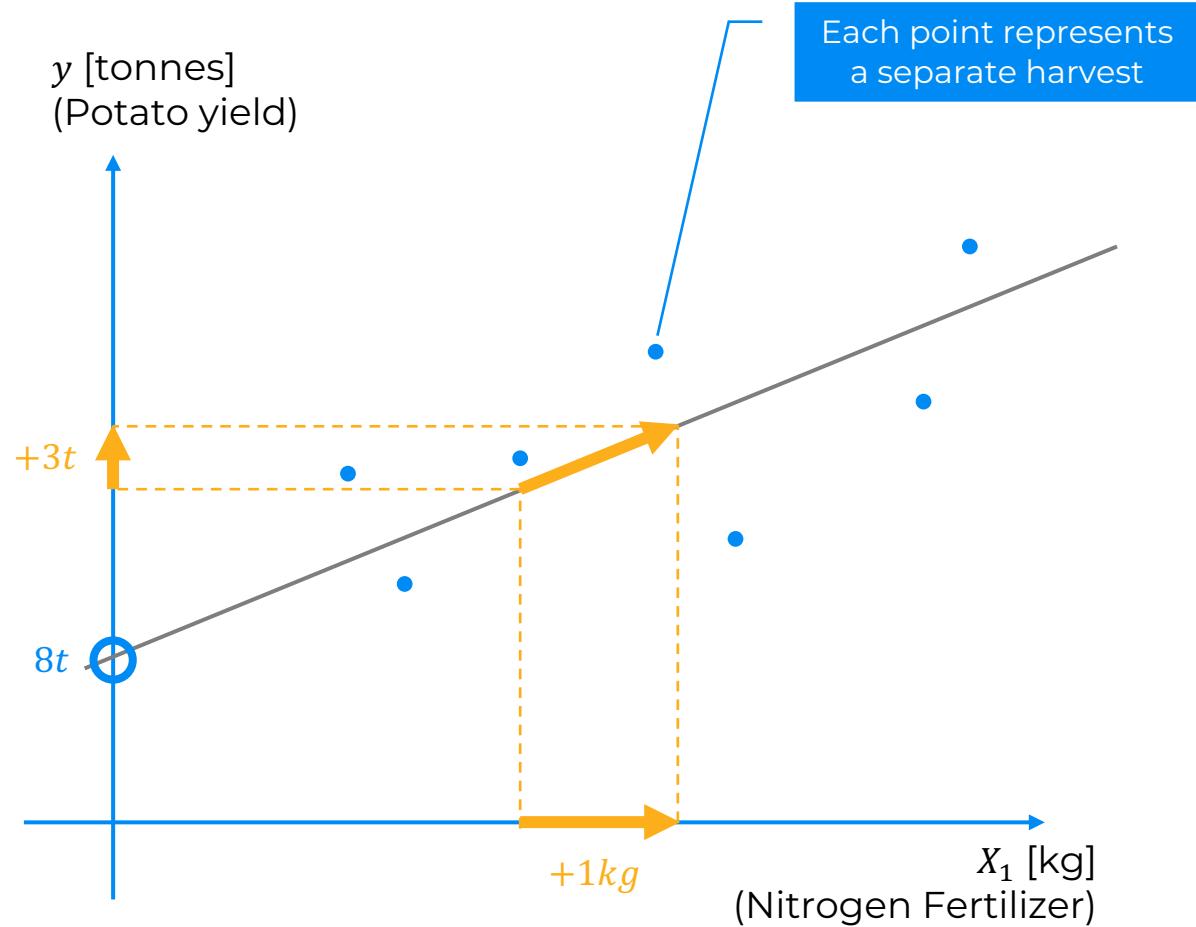


$$\hat{y} = b_0 + b_1 X_1$$

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

$$b_0 = 8[t]$$

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



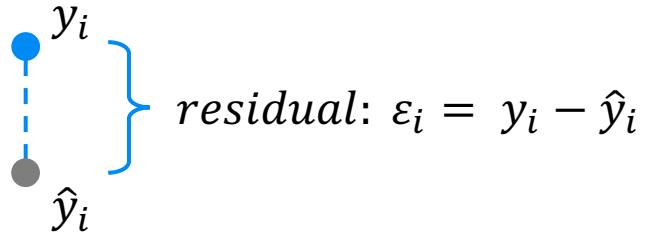
Ordinary Least Squares





Simple Linear Regression

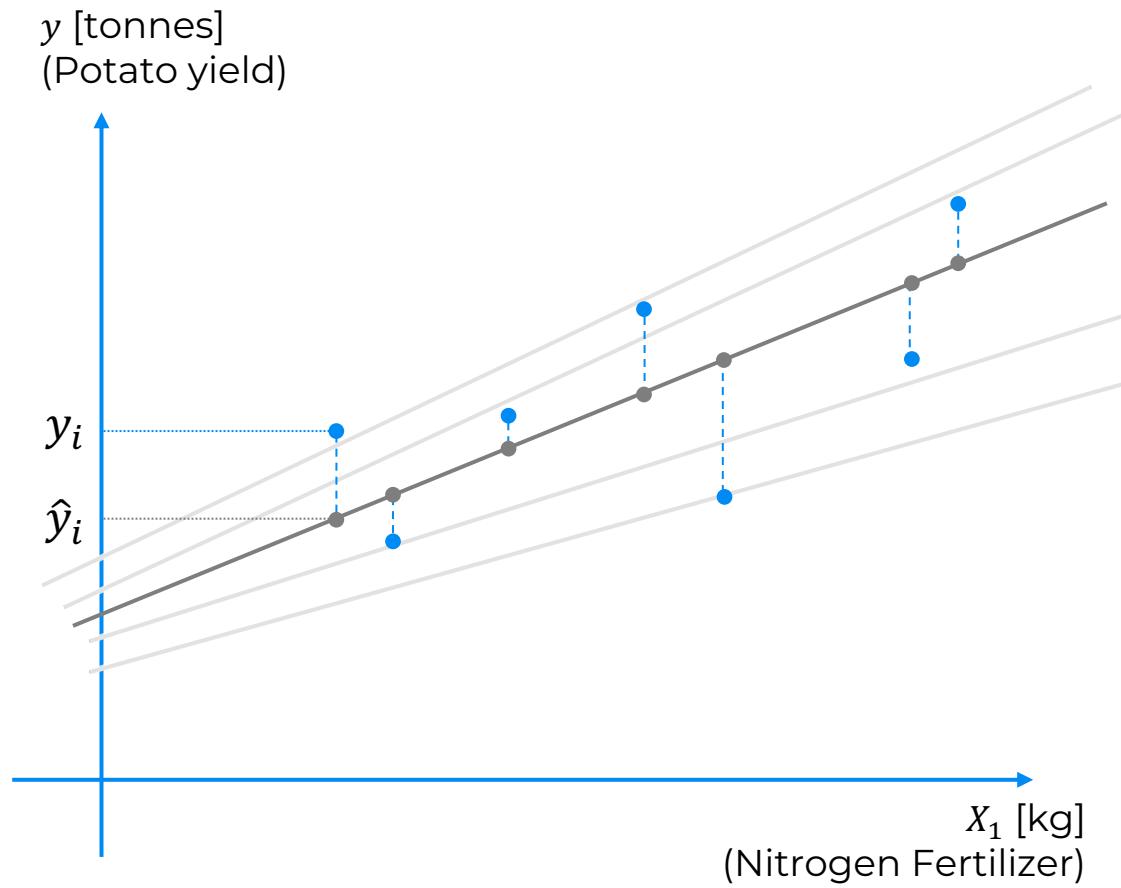
Ordinary Least Squares:



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

b_0, b_1 such that:

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



Multiple Linear Regression





Multiple Linear Regression

$$\hat{y} = b_0 + b_1X_1 + b_2X_2 + \cdots + b_nX_n$$

Dependent variable
y-intercept (constant)

Independent variable 1
Slope coefficient 1

Independent variable 2
Slope coefficient 2

Independent variable n
Slope coefficient n





Multiple Linear Regression



~



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

Additional Reading

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

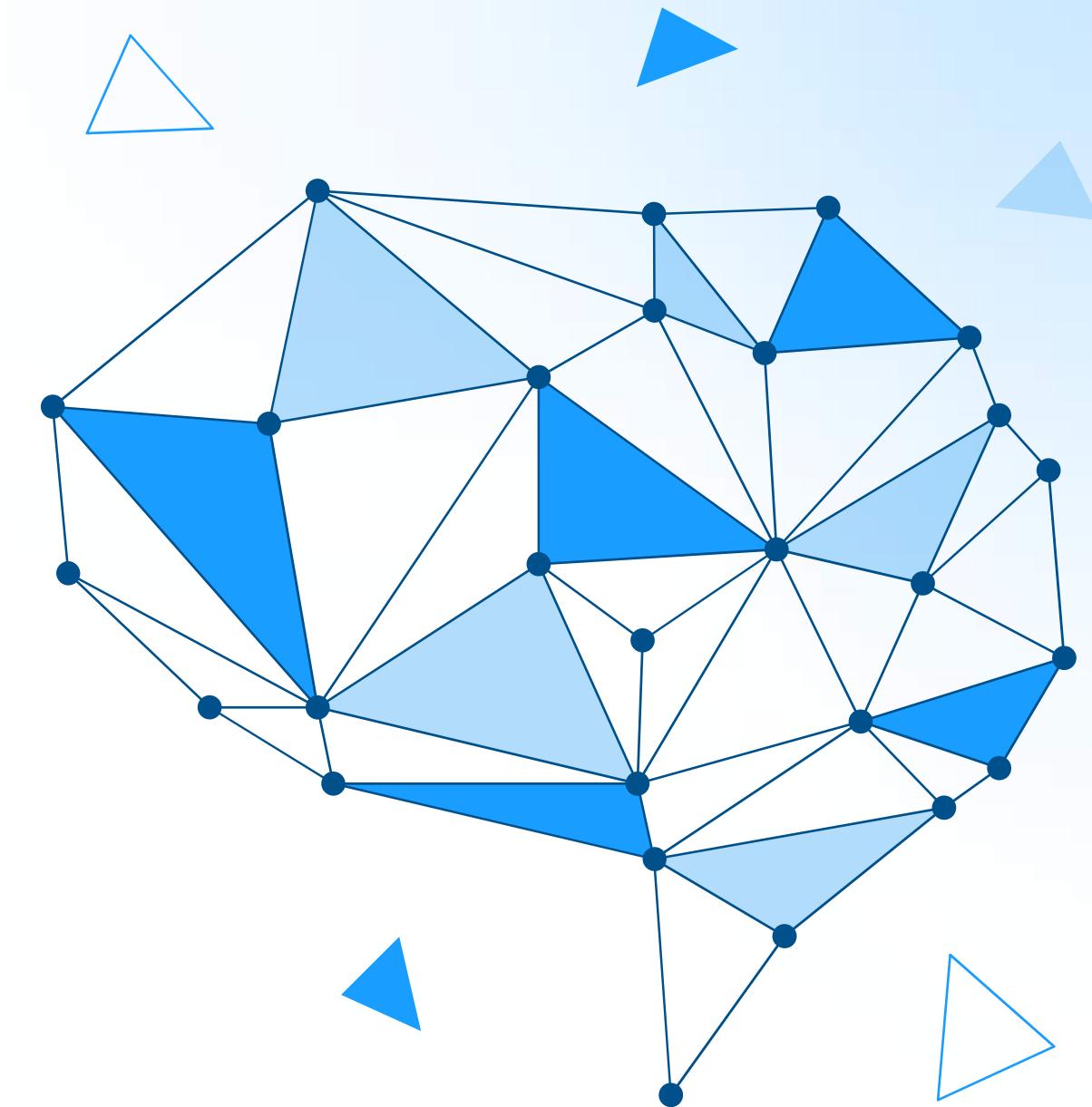
Magdalena Piekutowska et. al. (2021)

Link:

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

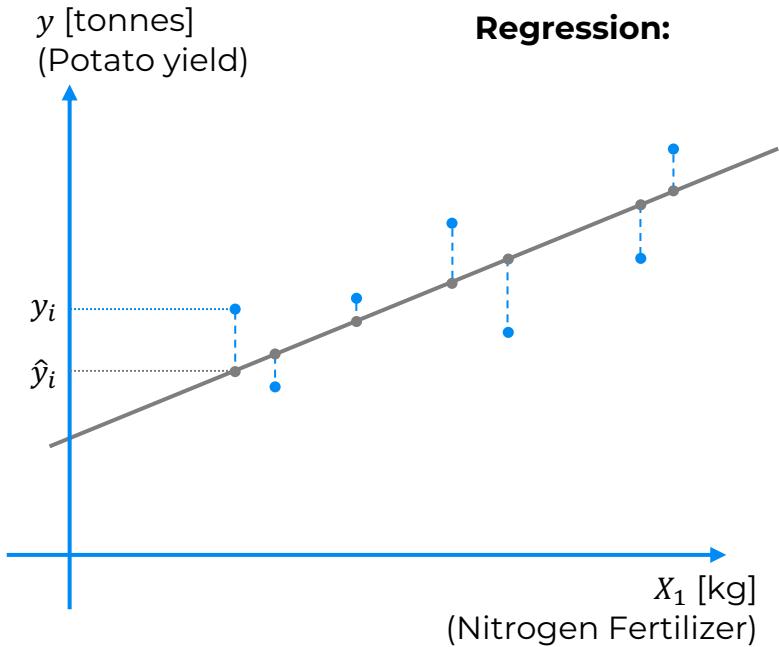
Quantitative Yield Forecast		
Models RY1 and NY1	Yield Forecast before Harvest (40 Days from Full Emergence)	Data Range
INSO	insolation sum [h] in the periods: planting—June 20	275.3–711.7
TEMP	average daily air temperature [$^{\circ}\text{C}$] in the periods: planting—20 June	10.8–15.7
PREC	precipitation [mm] in the periods: planting—20 June	38.7–258.2
NETRO	sum of nitrogen fertilization [$\text{kg}\cdot\text{ha}^{-1}$] in the periods: planting—20 June	80–150
PHOSP	sum of phosphorus fertilization [$\text{kg}\cdot\text{ha}^{-1}$]	28.2–150
POTAS	sums of potassium fertilization [$\text{kg}\cdot\text{ha}^{-1}$]	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 $\text{P}_2\text{O}_5\cdot 100 \text{ g}^{-1}$ soil]	14–26.2
SFERTK	soil fertility in potassium [mg $\text{K}_2\text{O}\cdot 100 \text{ g}^{-1}$ soil]	11.7–19.2
SFERTM	soil fertility in magnesium [mg $\text{Mg}\cdot 100 \text{ g}^{-1}$ soil]	3–9.1
YIELDPI	tuber yield [$\text{t}\cdot\text{ha}^{-1}$], harvest 40 days from full emergence	11.6–41.3

R Squared

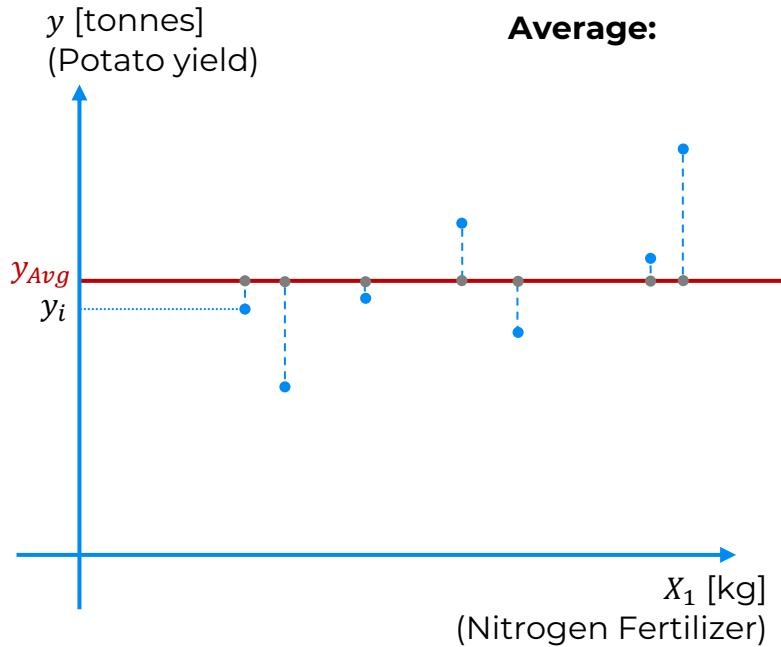




R Squared



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



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

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

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

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

R² – Goodness of fit
(greater is better)

Problem:

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

SS_{tot} doesn't change

SS_{res} will decrease or stay the same

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

(This is because of Ordinary Least Squares: $SS_{res} \rightarrow \text{Min}$)

Solution:

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

k – number of independent variables

n – sample size



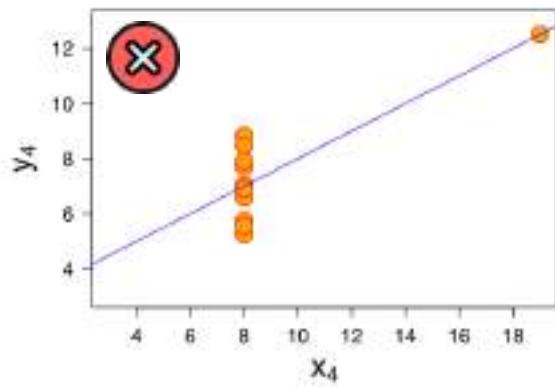
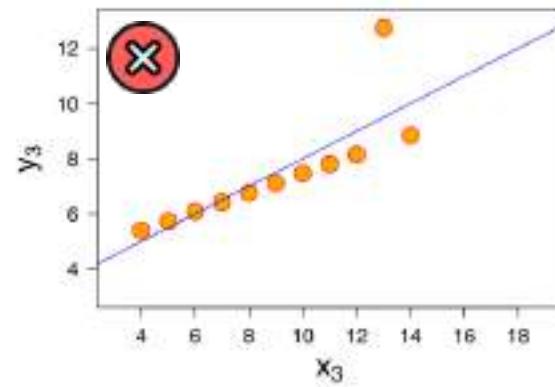
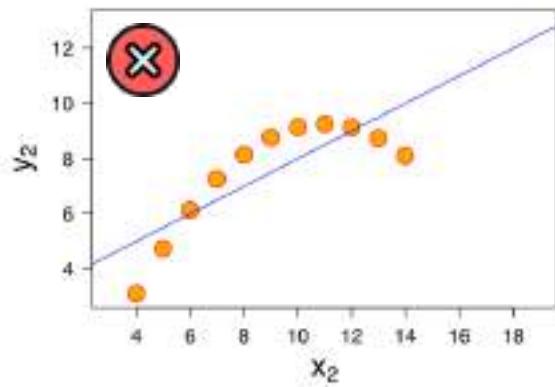
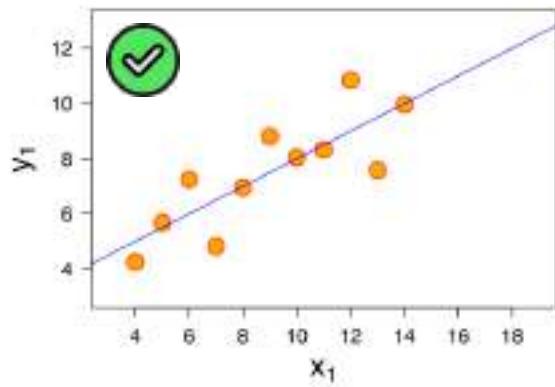
Assumptions Of Linear Regression





Assumptions of Linear Regression

Anscombe's quartet (1973):

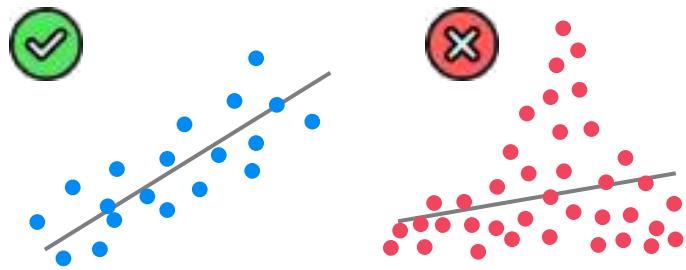




Assumptions of Linear Regression

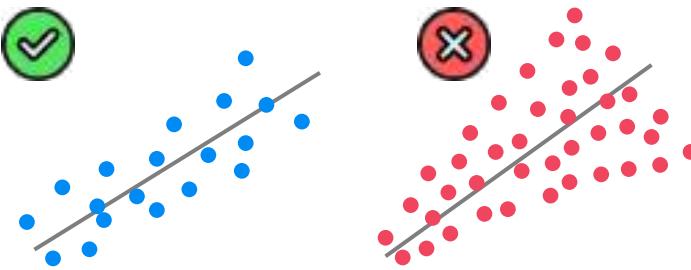
1. Linearity

(Linear relationship between Y and each X)



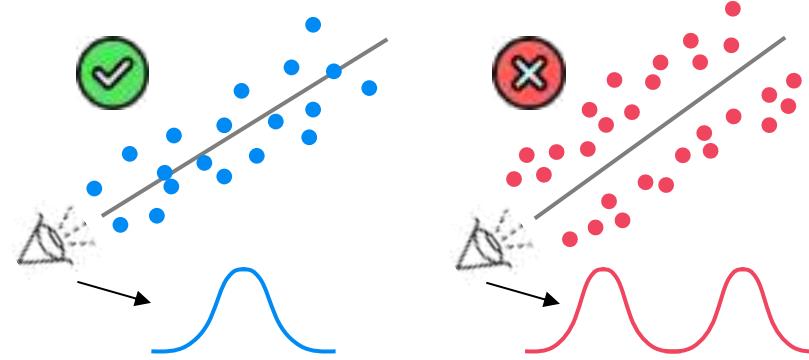
2. Homoscedasticity

(Equal variance)



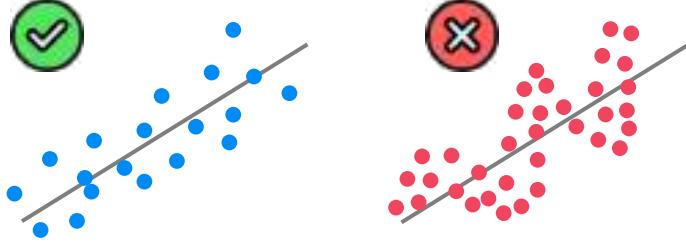
3. Multivariate Normality

(Normality of error distribution)



4. Independence

(of observations. Includes “no autocorrelation”)



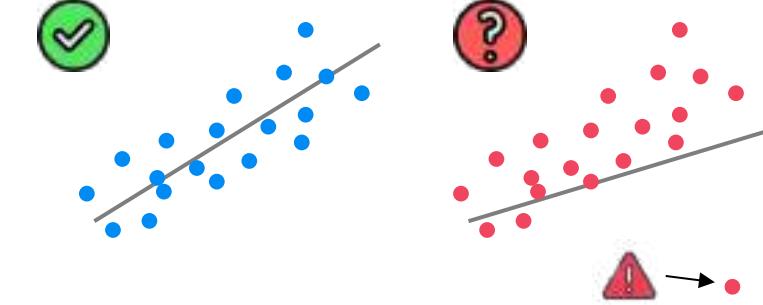
5. Lack of Multicollinearity

(Predictors are not correlated with each other)

$$\checkmark \quad X_1 \not\sim X_2 \quad \times \quad X_1 \sim X_2$$

6. The Outlier Check

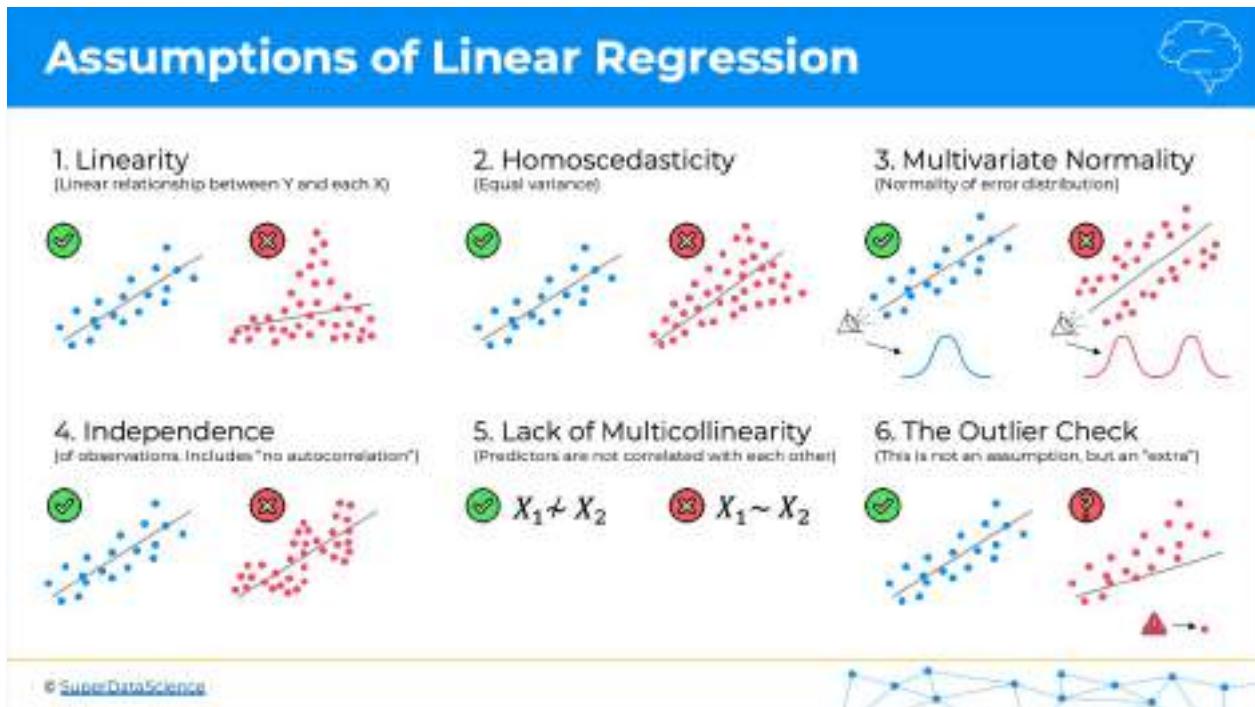
(This is not an assumption, but an “extra”)



Bonus



Download the Assumptions poster at:
superdatascience.com/assumptions



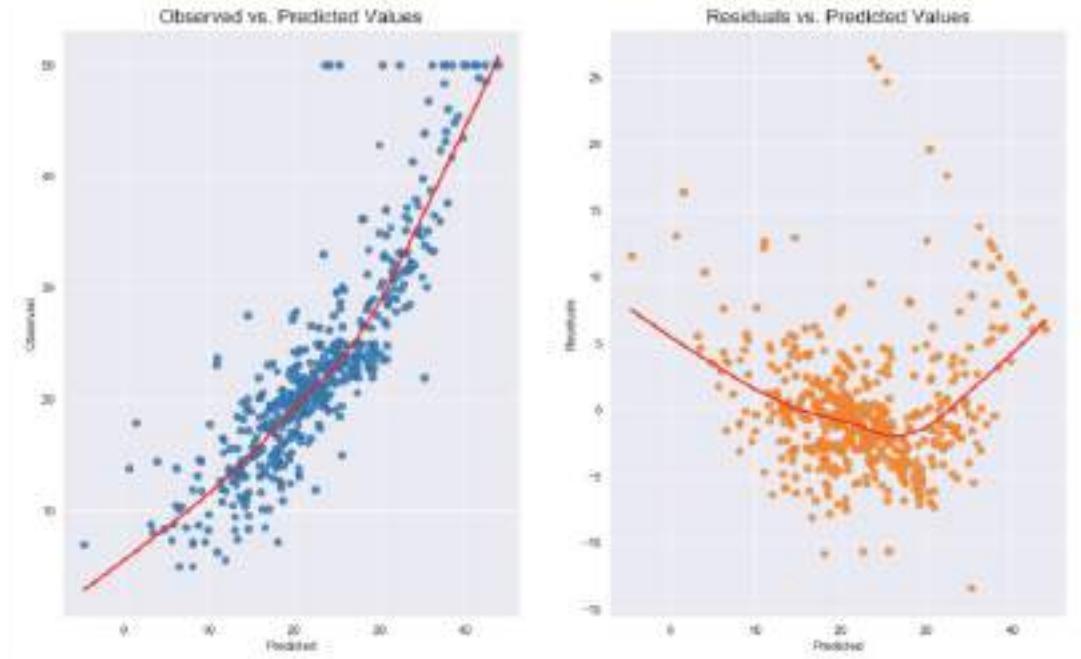
Additional Reading

*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](https://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

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D_1$$



Dummy Var. Trap

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	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

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D_1$$

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			California
191,050.39	153,441.51			California
182,901.99	144,372.41			New York
166,187.94	142,107.34			California

$$D_2 = 1 - D_1$$

Dummy Variables

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

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



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	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

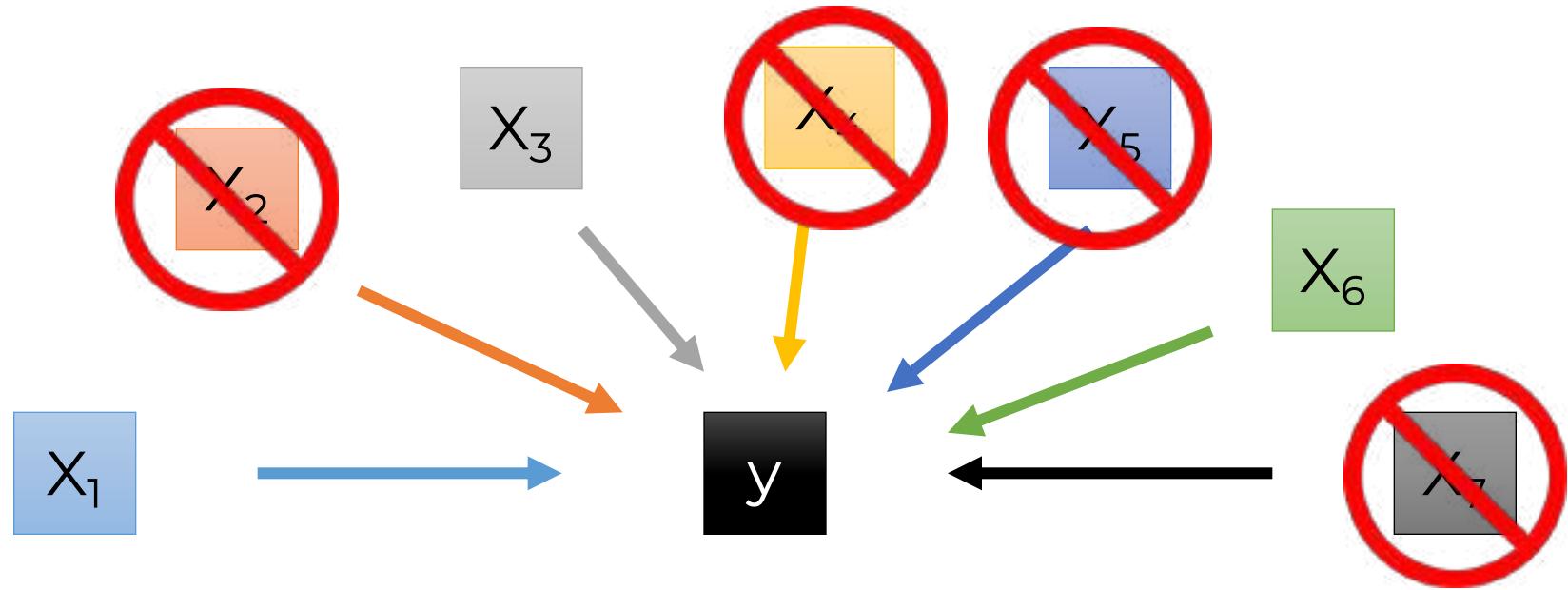
$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D_1 + b_5 * D_2$$



Always omit one dummy variable

Building A Model (Step-By-Step)

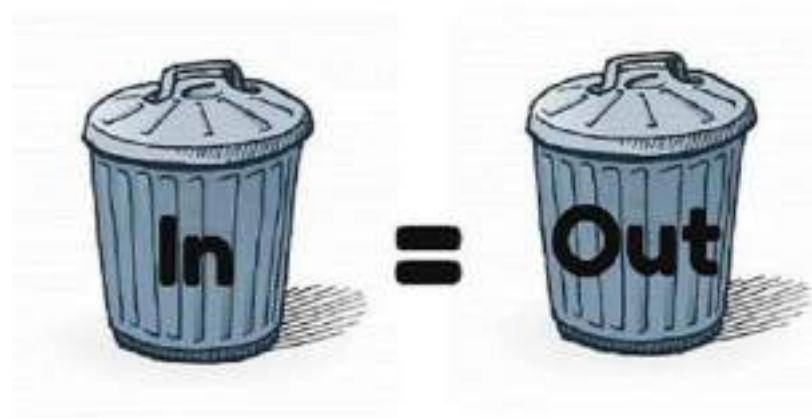
Building A Model



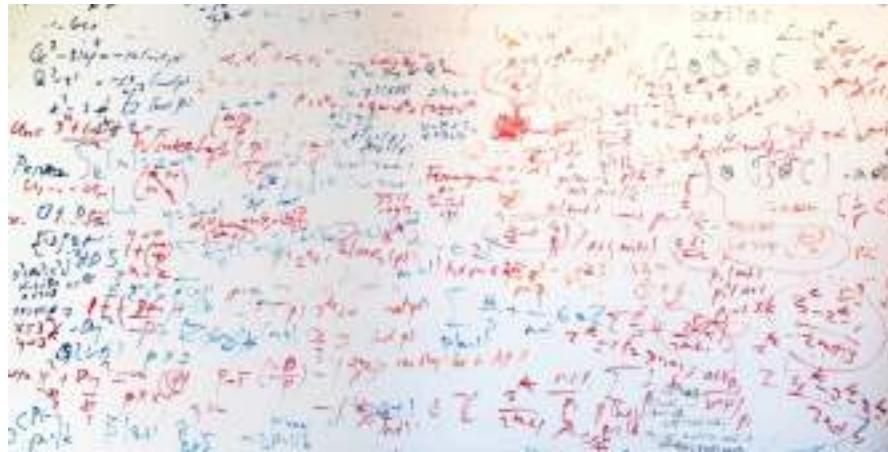
Why?

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



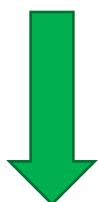
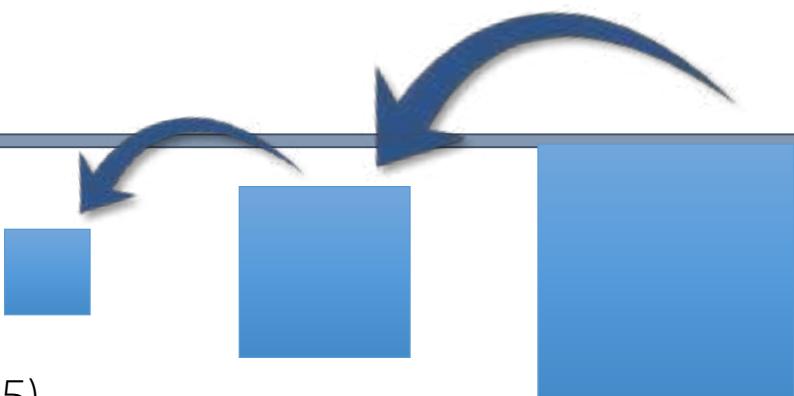
STEP 3: Consider the predictor with the highest P-value. If $P > SL$, go to STEP 4, otherwise go to FIN



STEP 4: Remove the predictor



STEP 5: Fit model without this variable*



FIN: Your Model Is Ready

Building A Model

Forward Selection

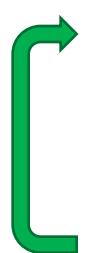
STEP 1: Select a significance level to enter the model (e.g. SL = 0.05)



STEP 2: Fit all simple regression models $y \sim x_n$ 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 lowest P-value. If $P < SL$, go to STEP 3, otherwise go to FIN

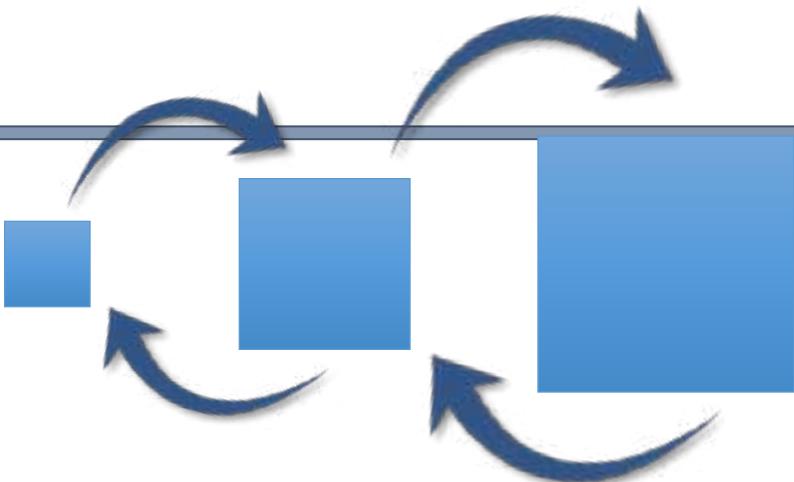


FIN: Keep the previous model

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: $P < \text{SLENTER}$ to enter)

STEP 3: Perform ALL steps of Backward Elimination (old variables must have $P < \text{SLSTAY}$ 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

$$y = b_0 + b_1 x_1$$

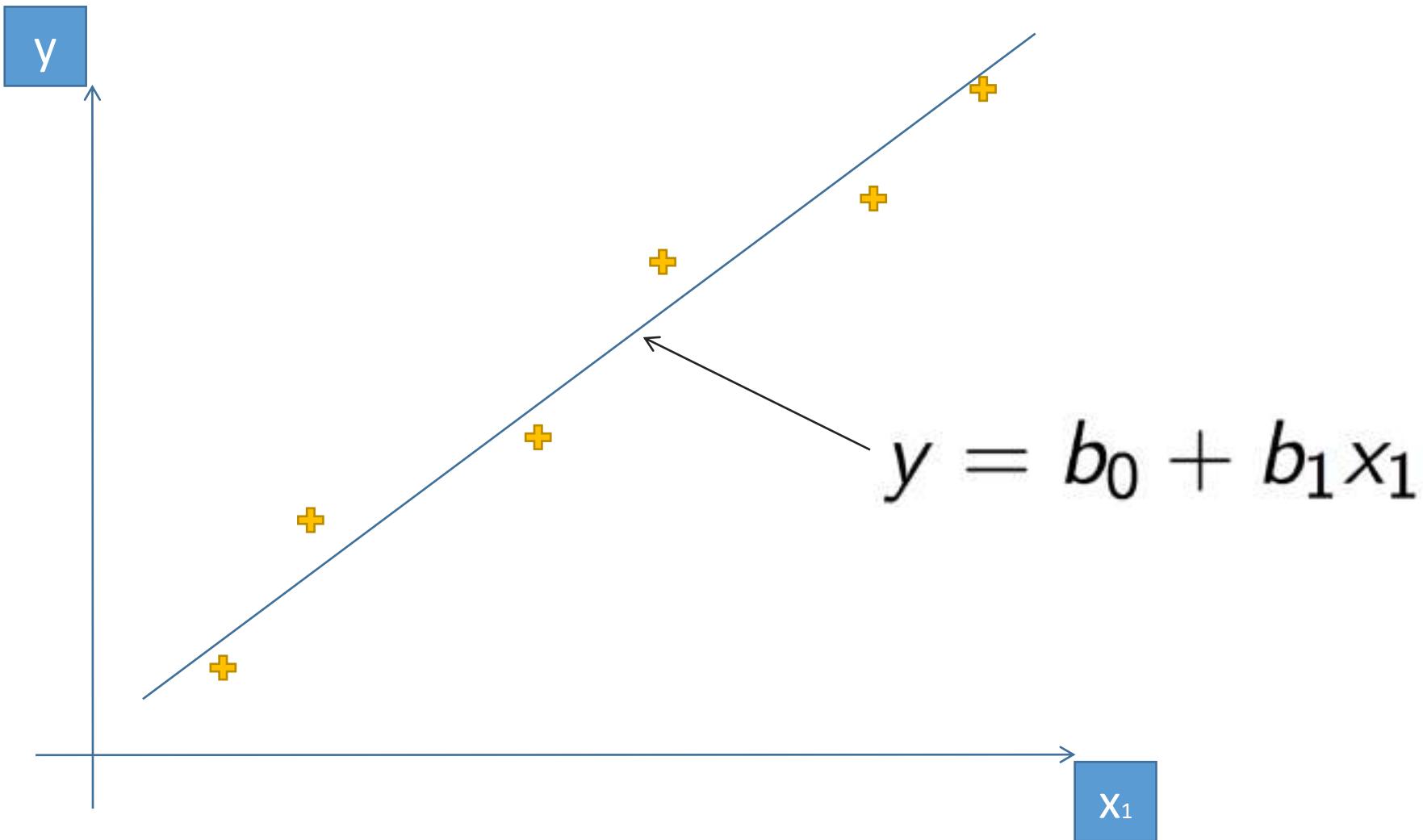
Multiple
Linear
Regression

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

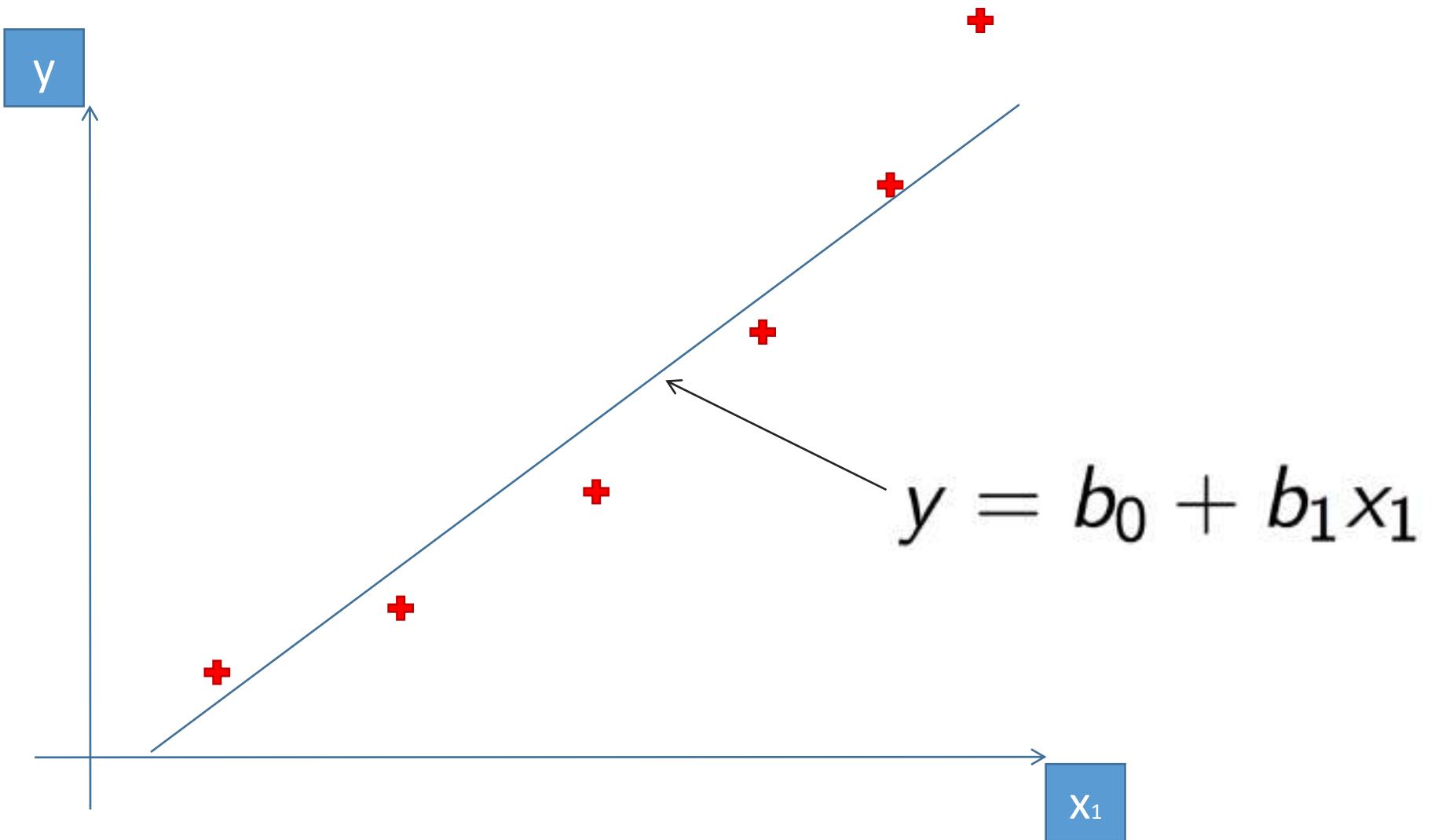
Polynomial
Linear
Regression

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

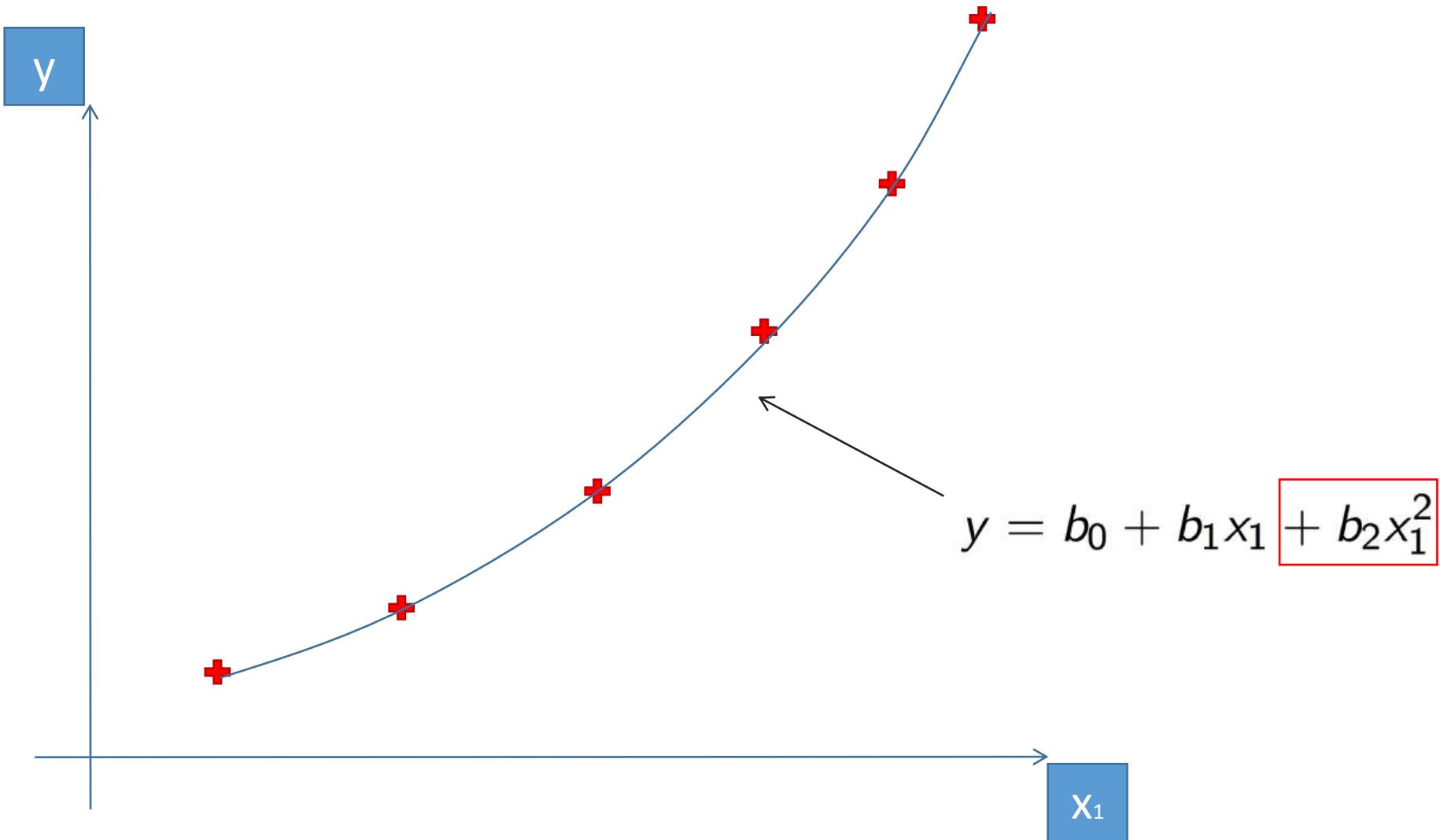
Simple Linear Regression



Simple Linear Regression



Polynomial Regression



Polynomial Regression

One Question: Why “Linear”?

Polynomial Regression

Polynomial
Linear
Regression

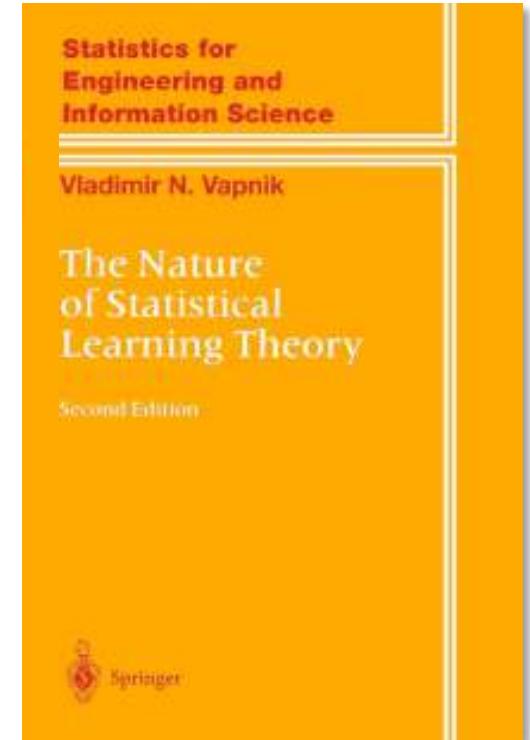
$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \dots + b_n x_1^n$$

SVR Intuition

SVR Intuition



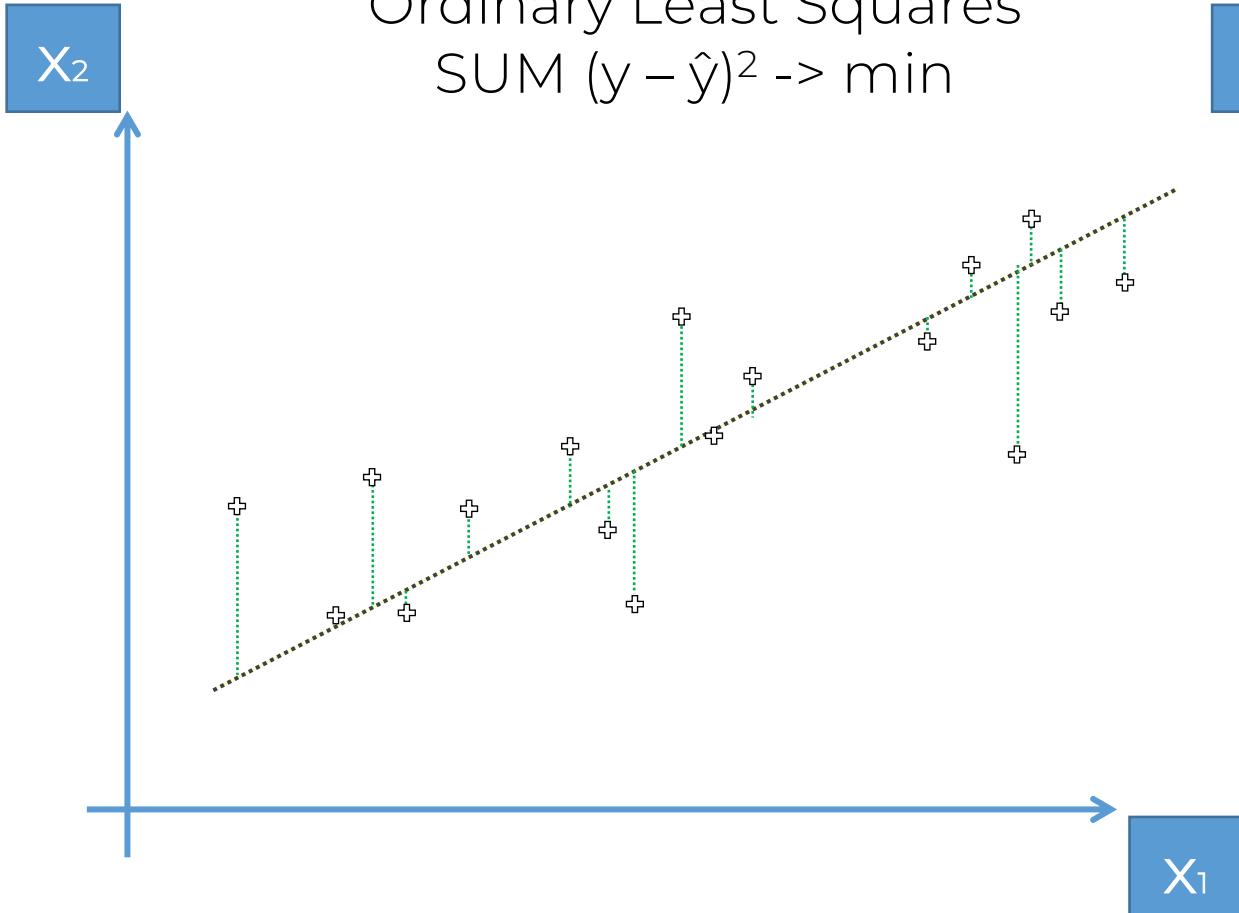
Vladimir Vapnik



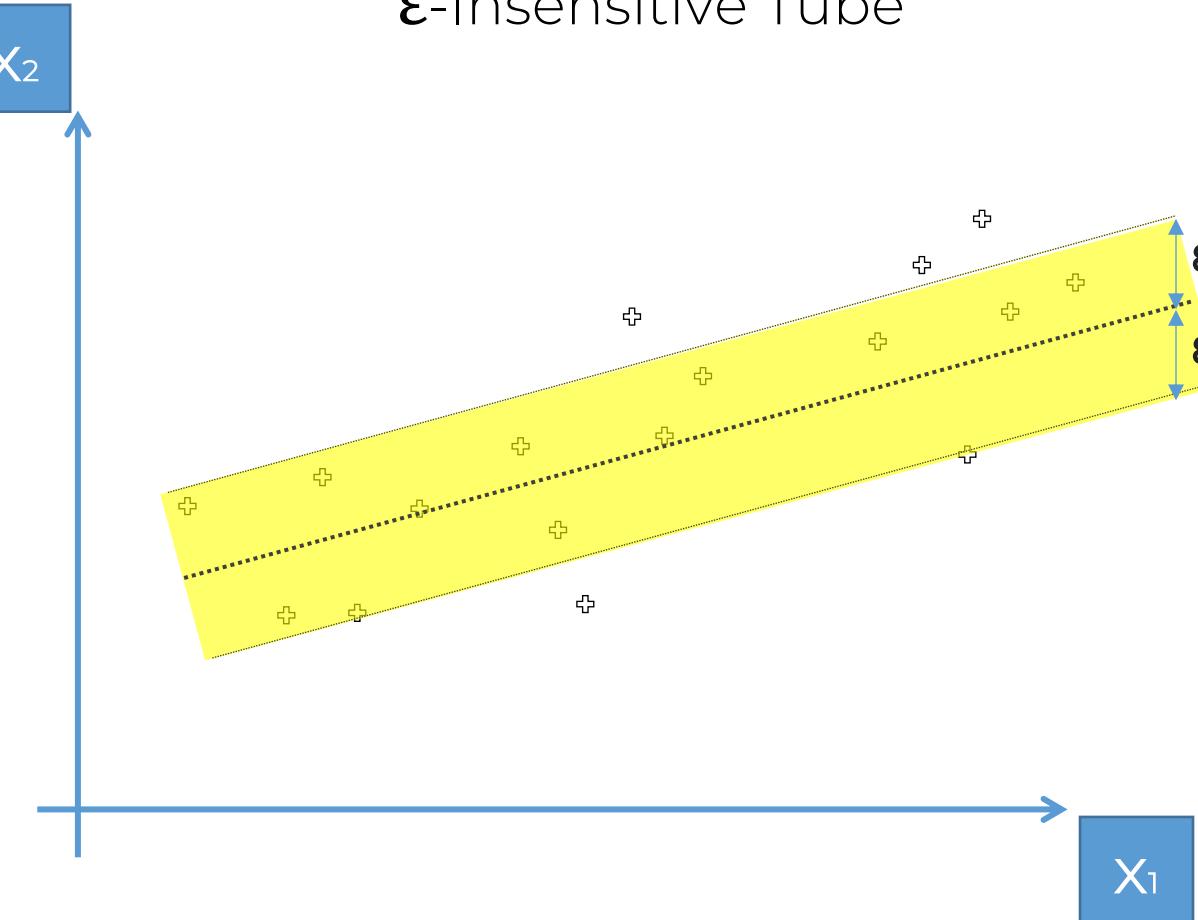
1992

SVR Intuition

Ordinary Least Squares
 $\text{SUM } (y - \hat{y})^2 \rightarrow \min$



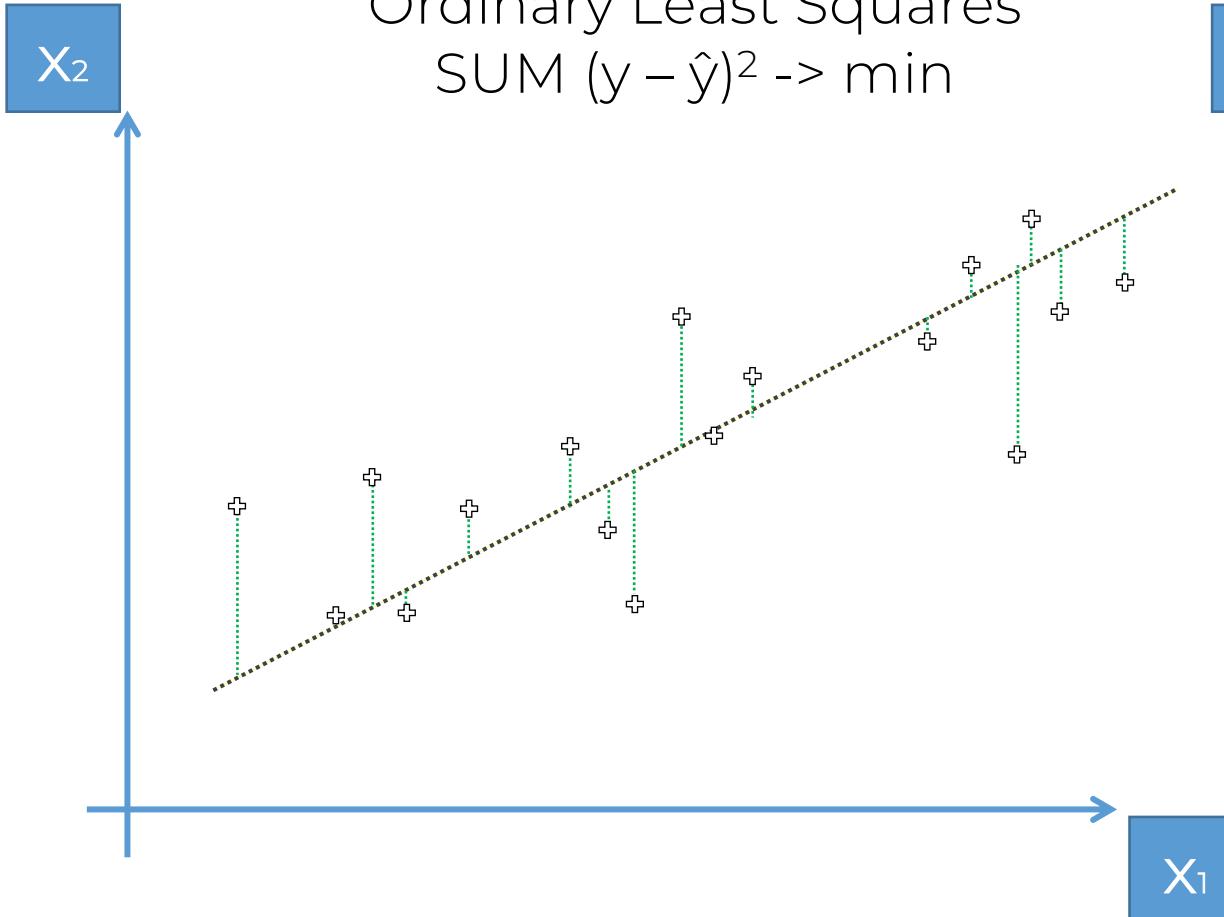
ϵ -Insensitive Tube



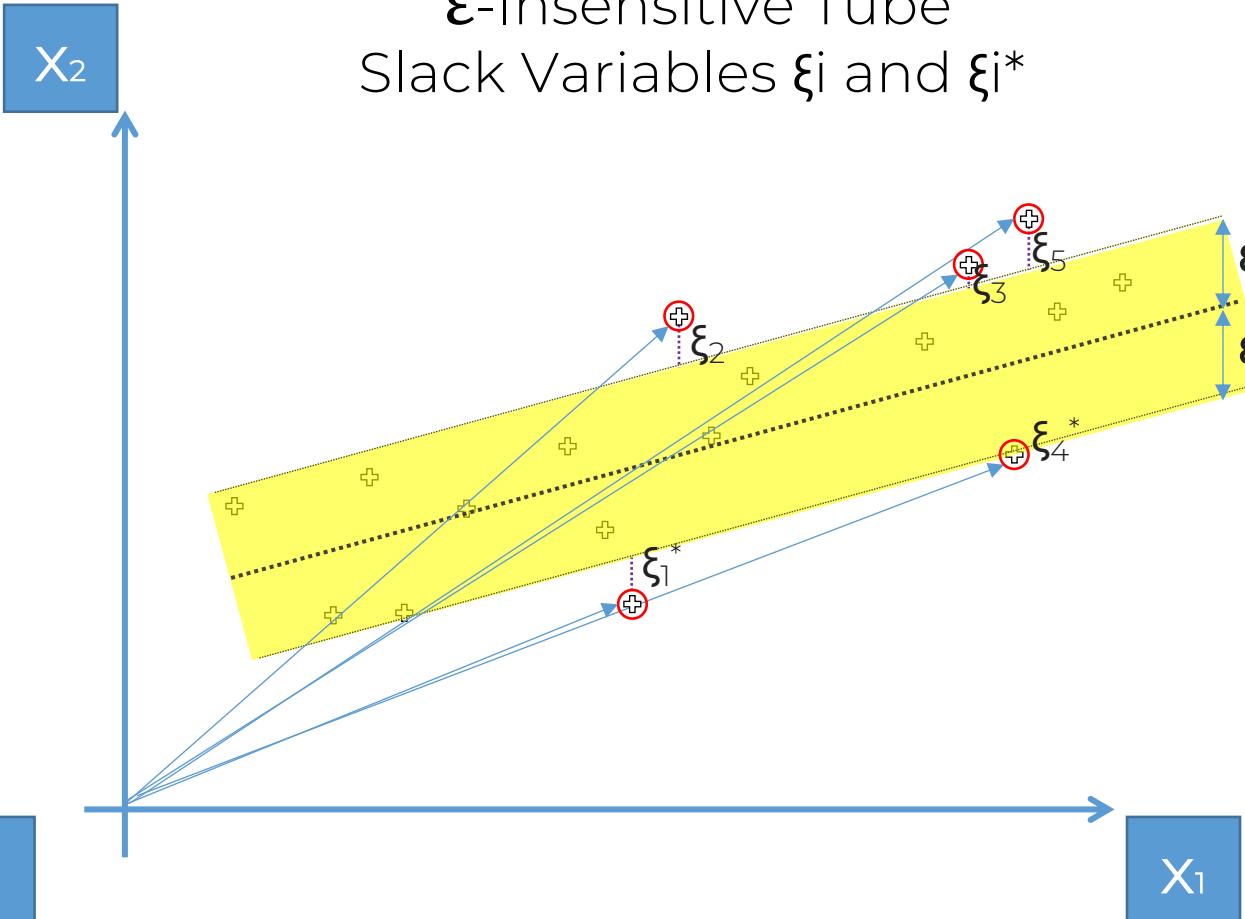
SVR Intuition

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

Ordinary Least Squares
SUM $(y - \hat{y})^2 \rightarrow \min$



ϵ -Insensitive Tube
Slack Variables ξ_i and ξ_i^*



SVR Intuition

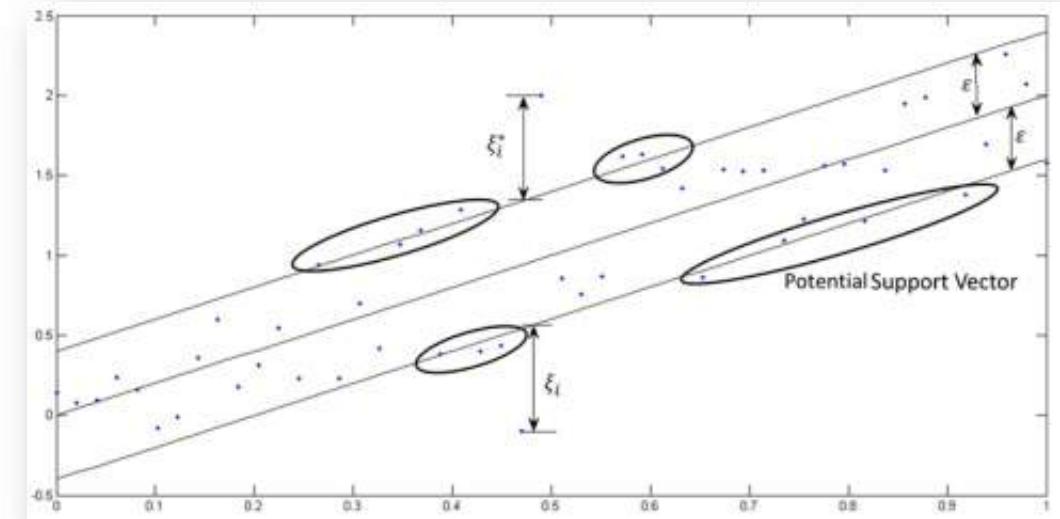
Additional Reading:

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

By Mariette Awad & Rahul Khanna (2015)

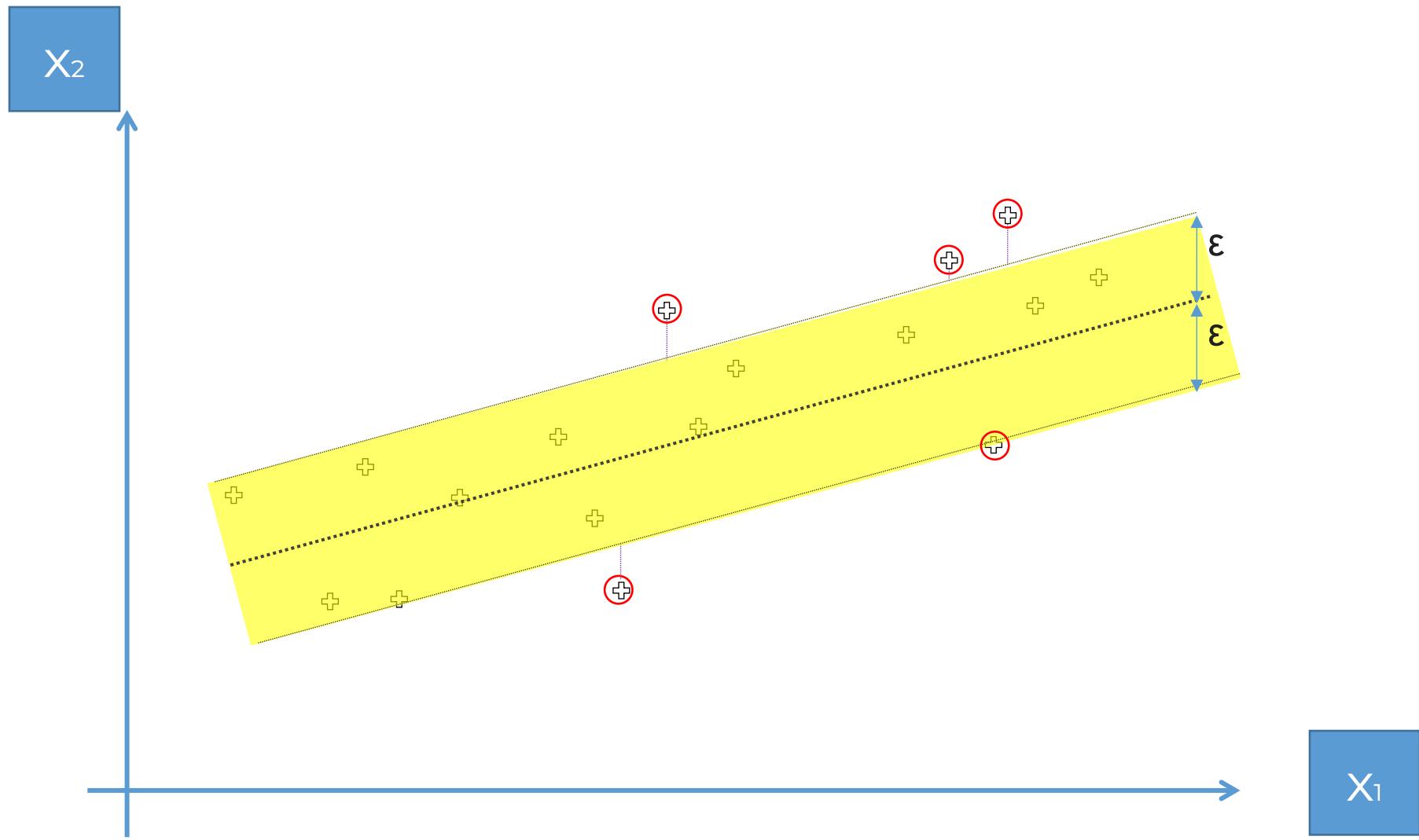
Link:

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



Heads-up about Non-Linear SVR

SVR Intuition



Copy of support_vector_regression.ipynb

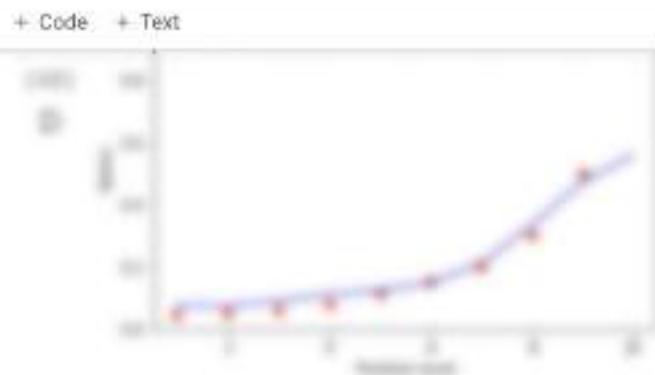
File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings

RAM Disk Editing

Files

- Upload Refresh Mount Drive
- ..
- sample_data
- Position_Salaries.csv

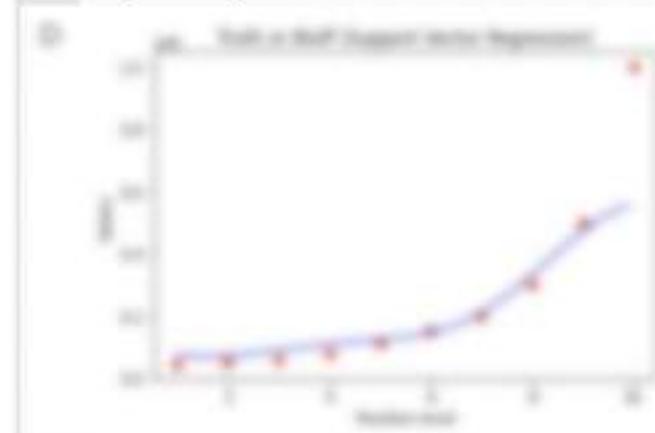


Visualising the SVR results (for higher resolution and smoother curve)

```

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

```



Position_Salaries.csv

Position	1 to 10 of 10 entries Filter	
	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	60000
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

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

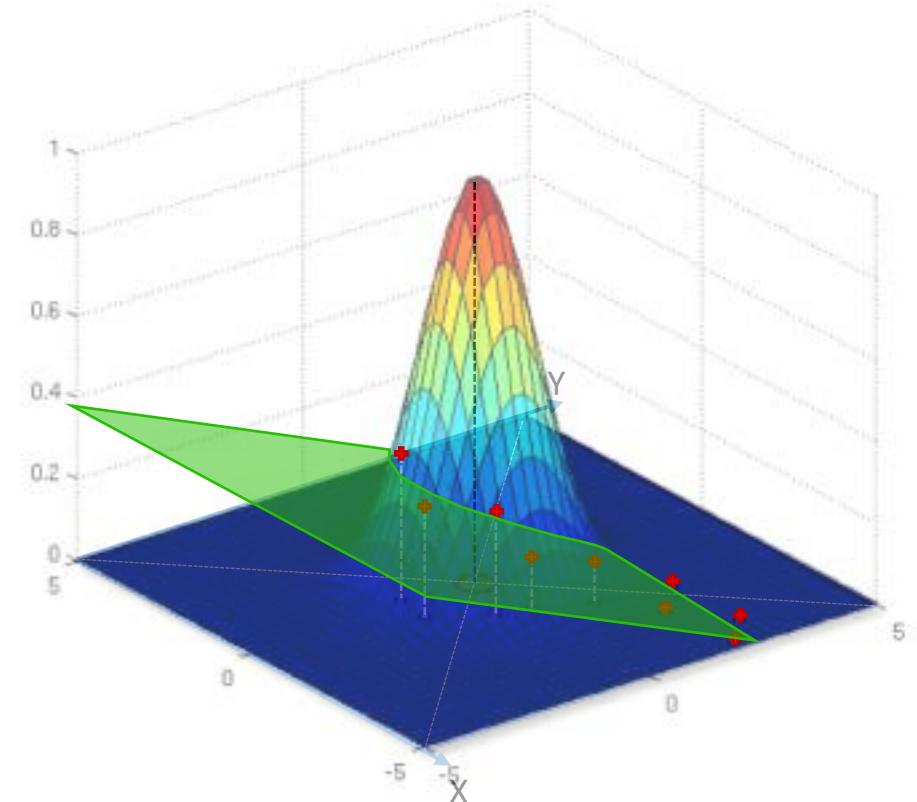
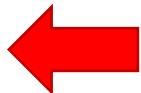
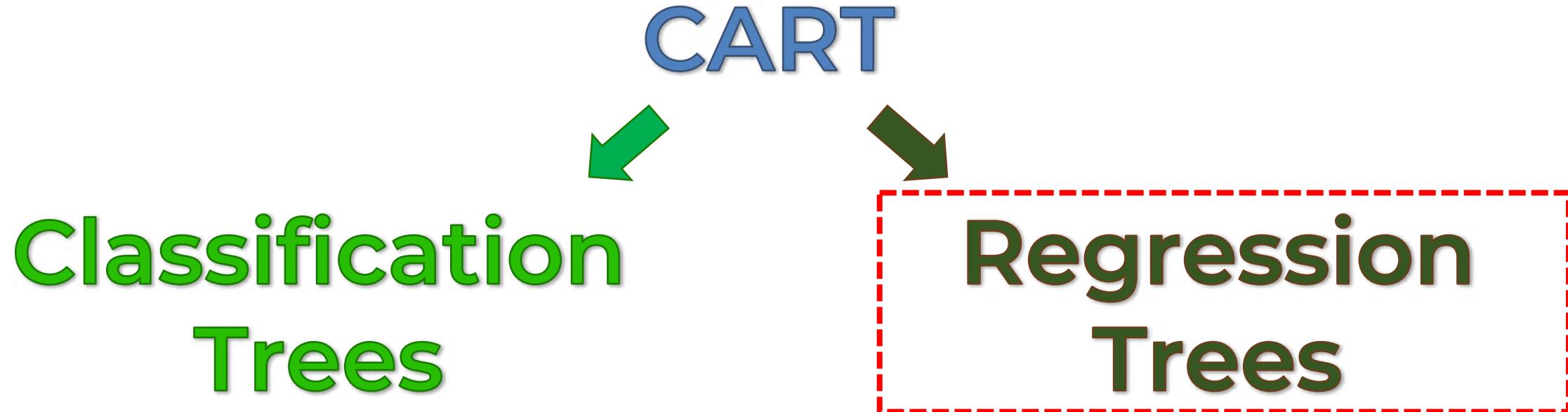


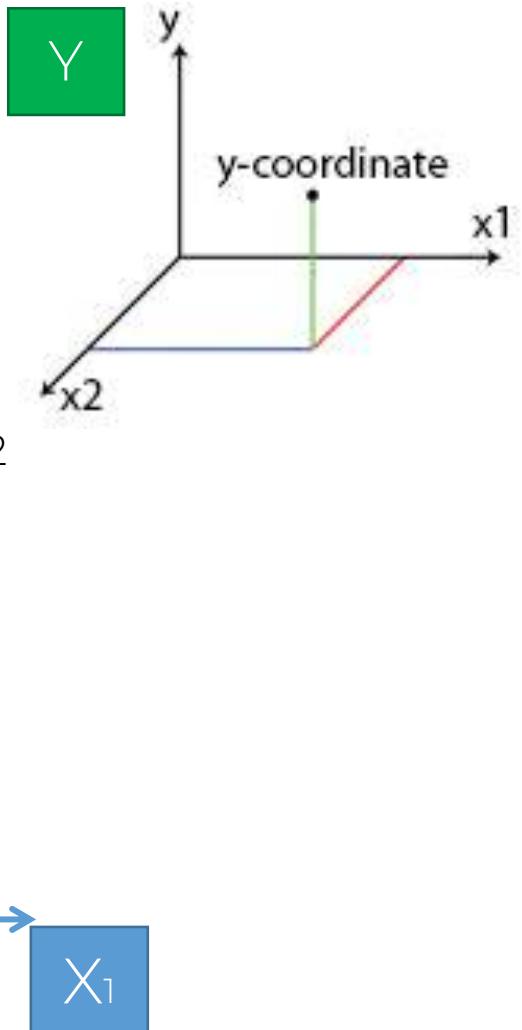
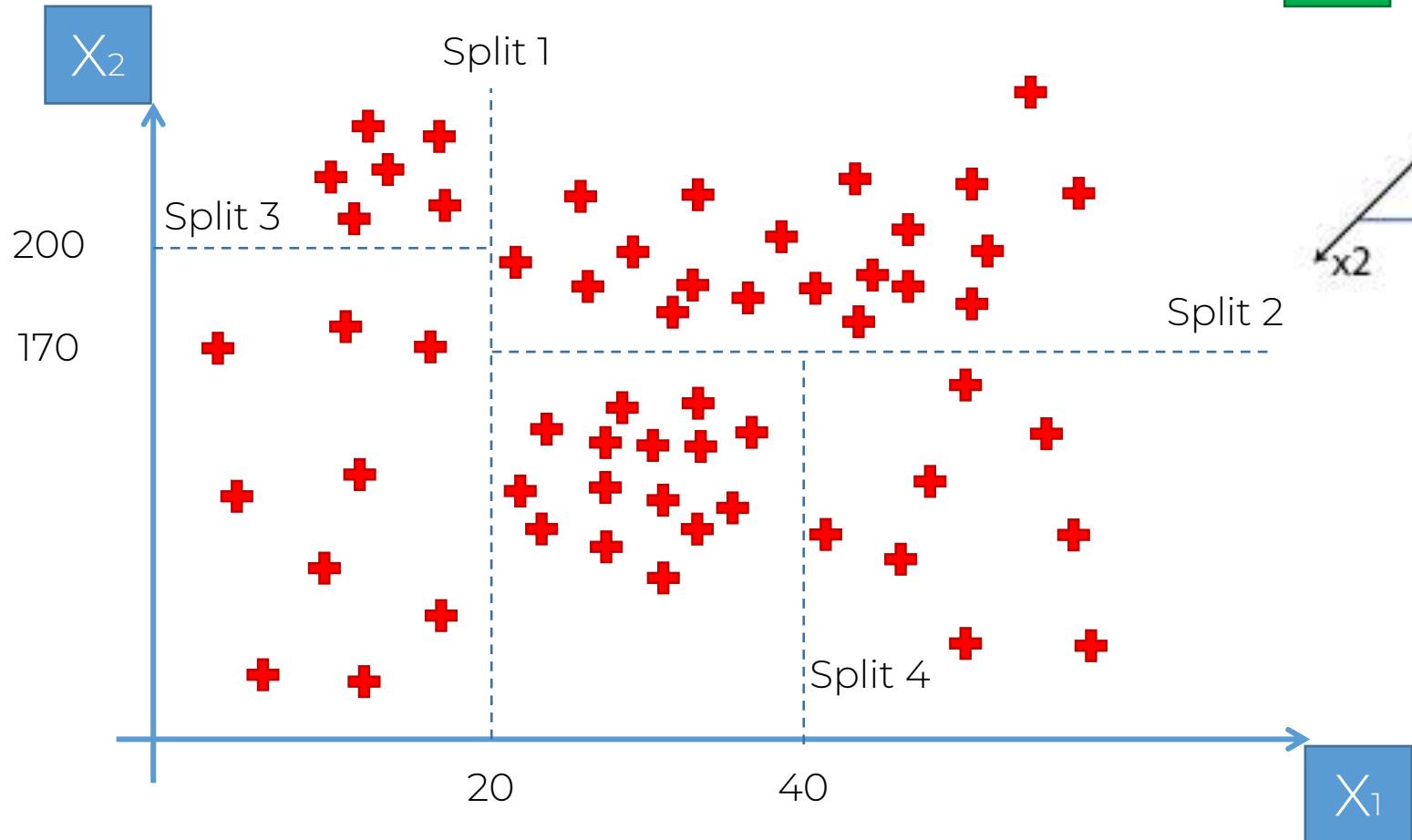
Image source: <http://www.cs.toronto.edu/~duvenaud/cookbook/index.html>

Decision Tree Intuition

Decision Tree Intuition



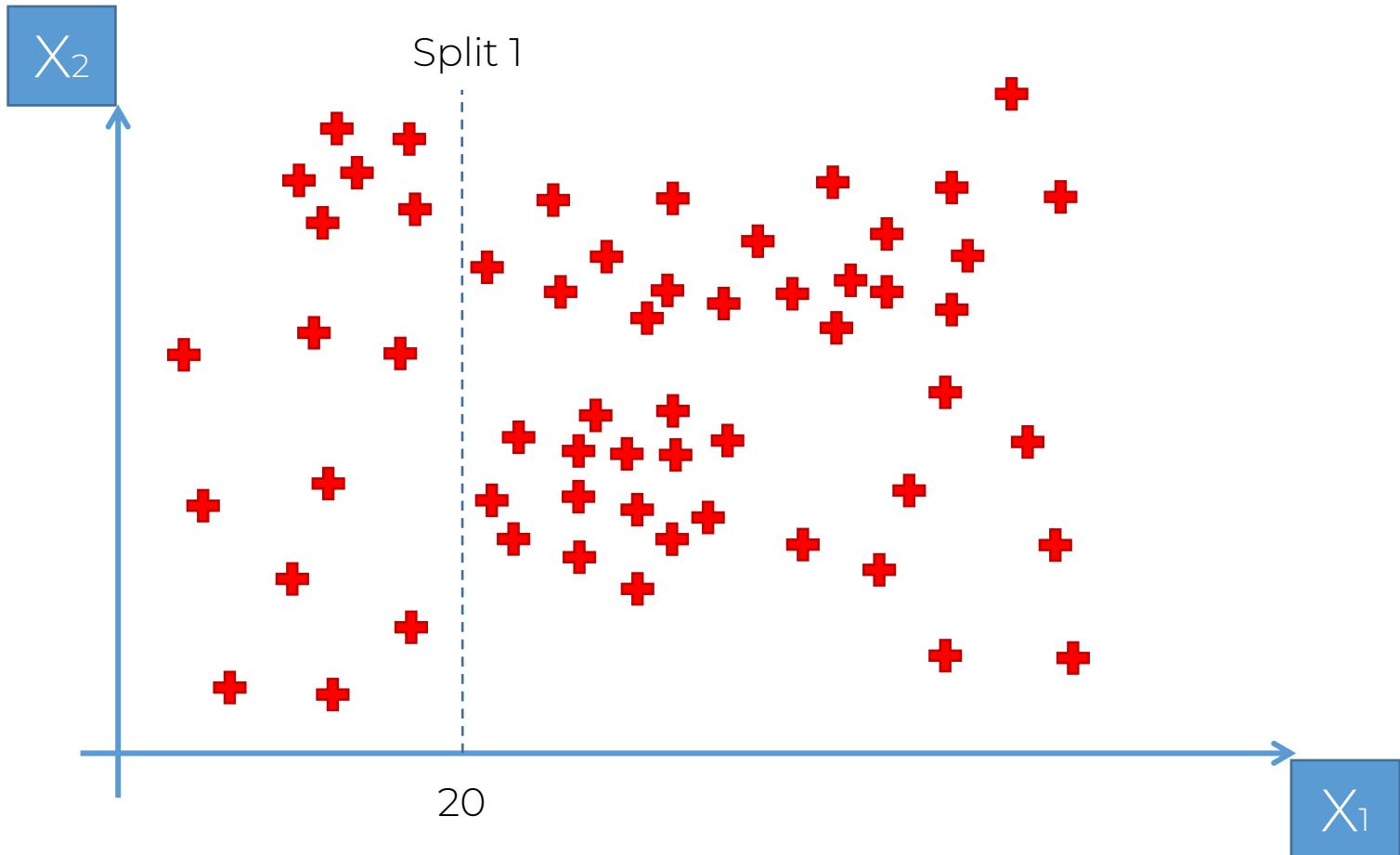
Decision Tree Intuition



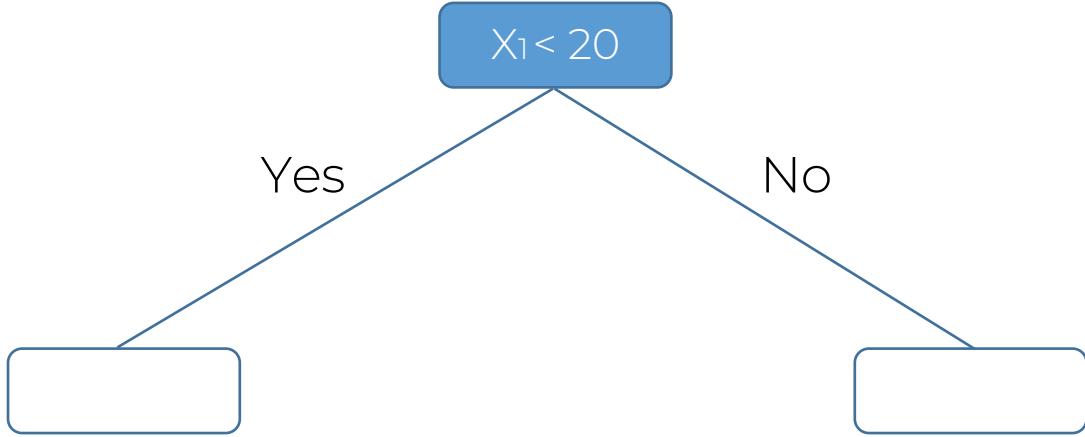
Decision Tree Intuition

Rewind...

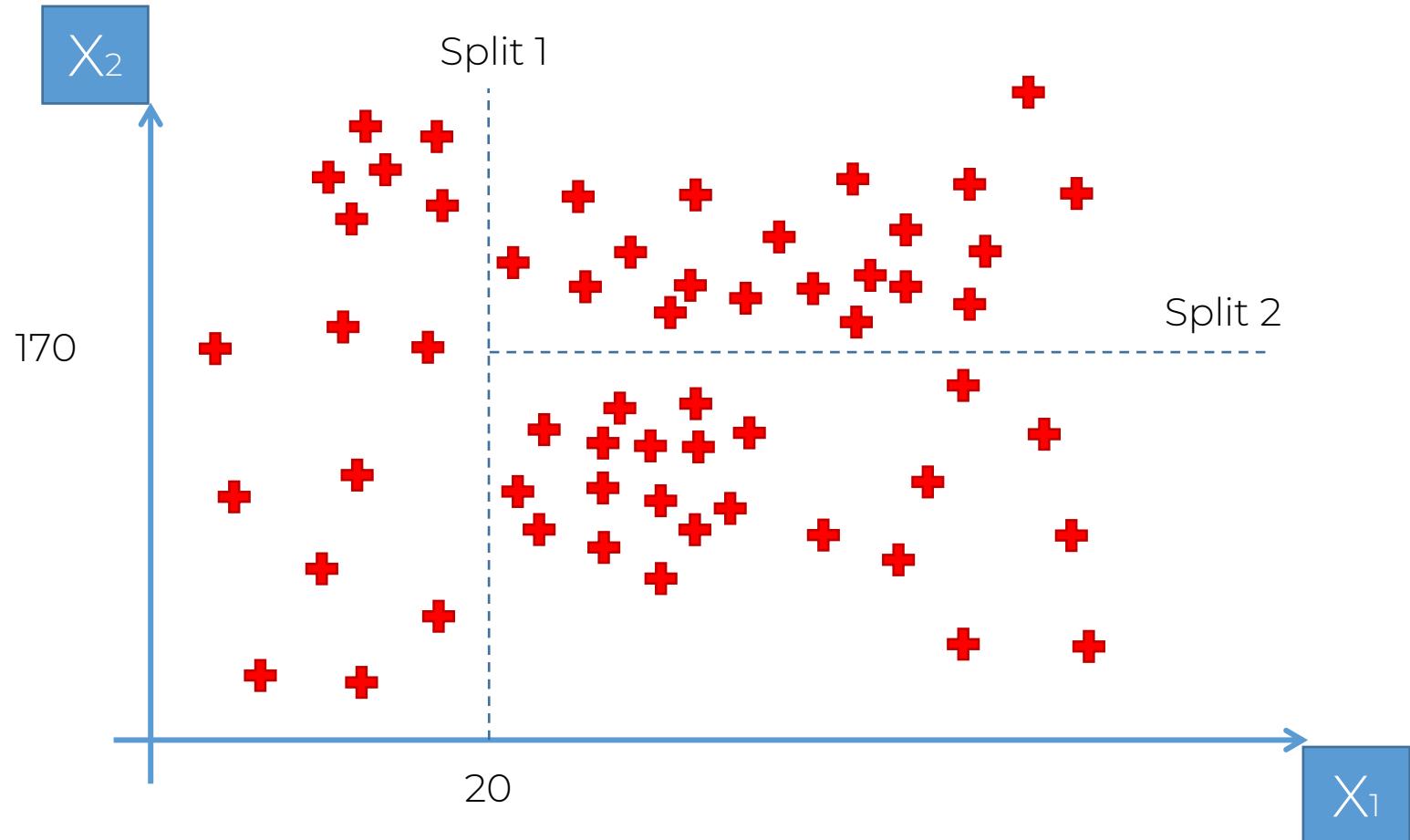
Decision Tree Intuition



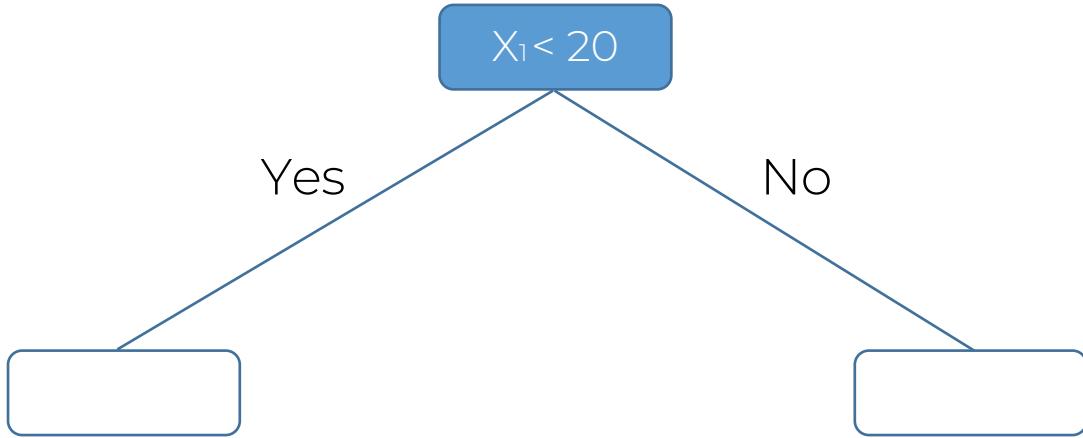
Decision Tree Intuition



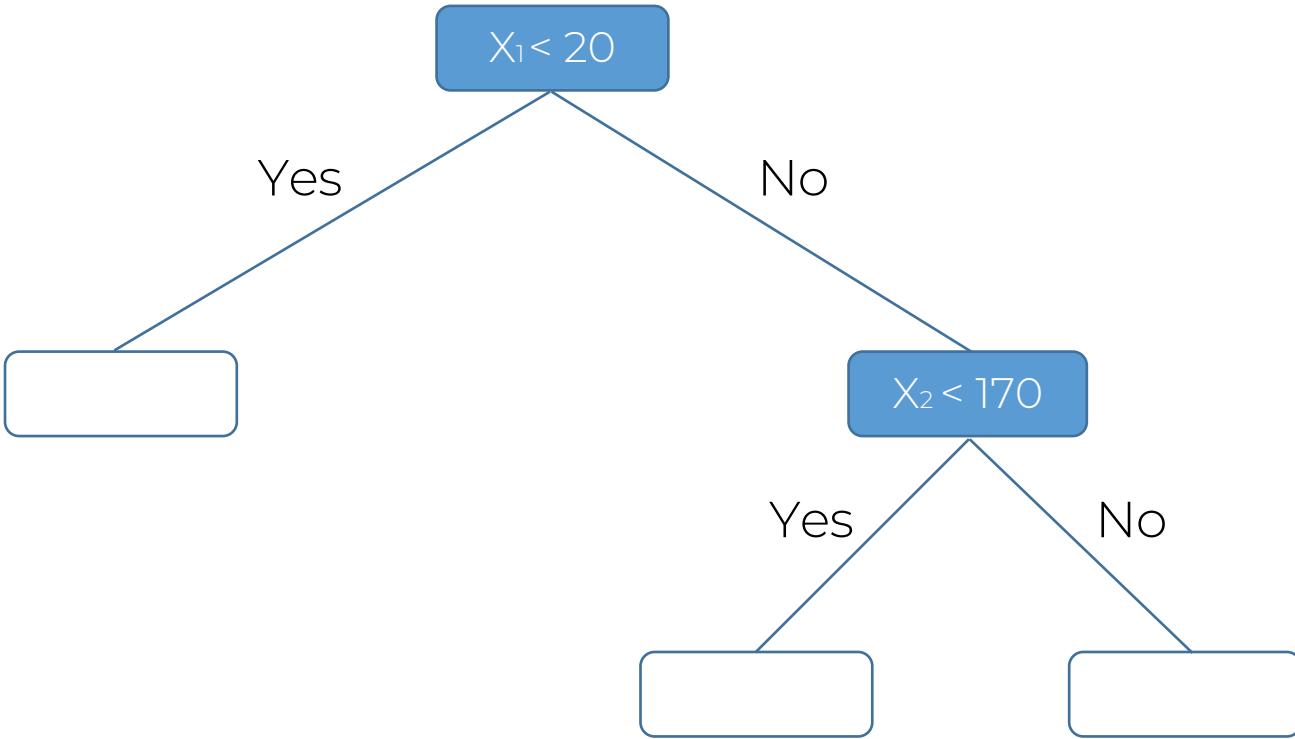
Decision Tree Intuition



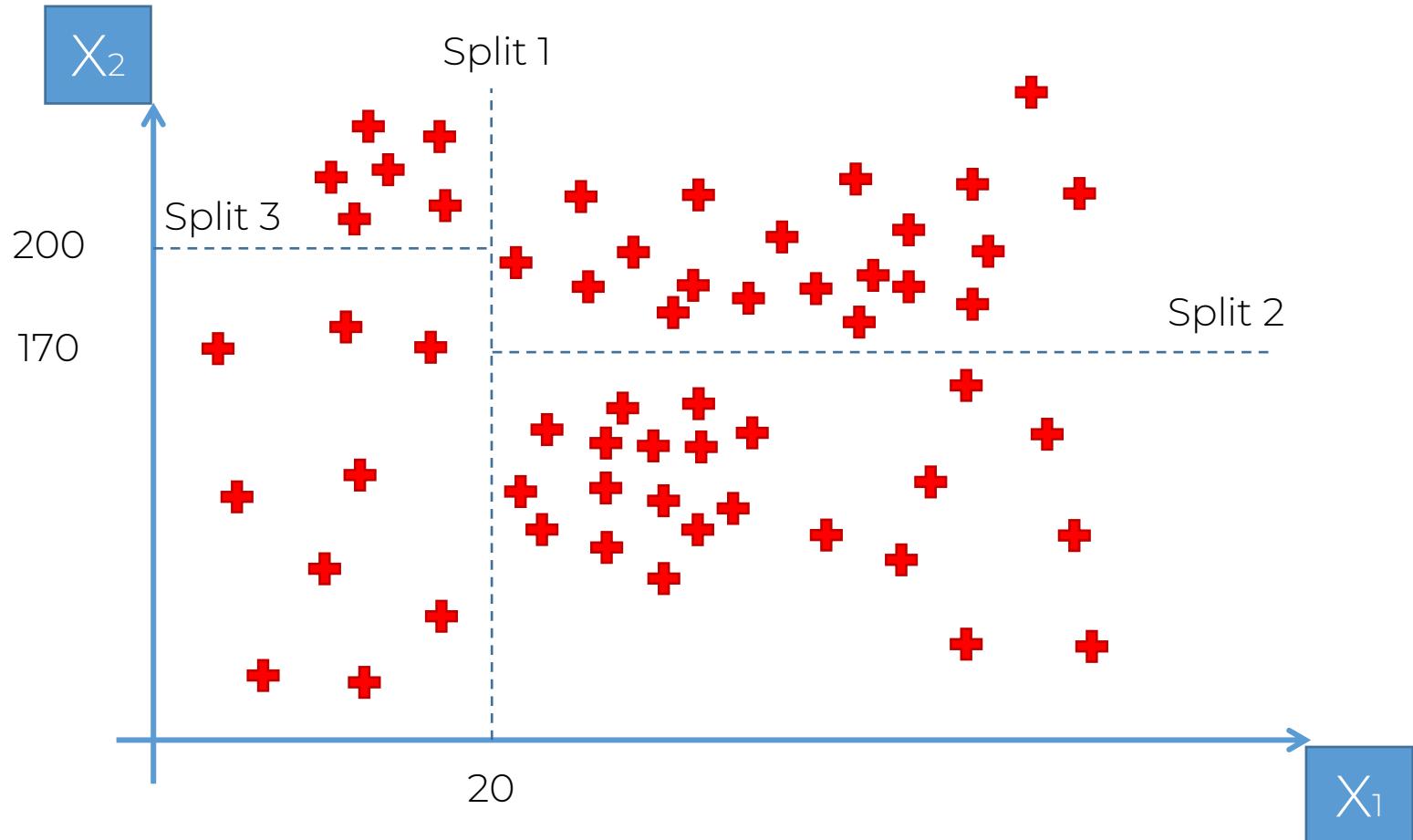
Decision Tree Intuition



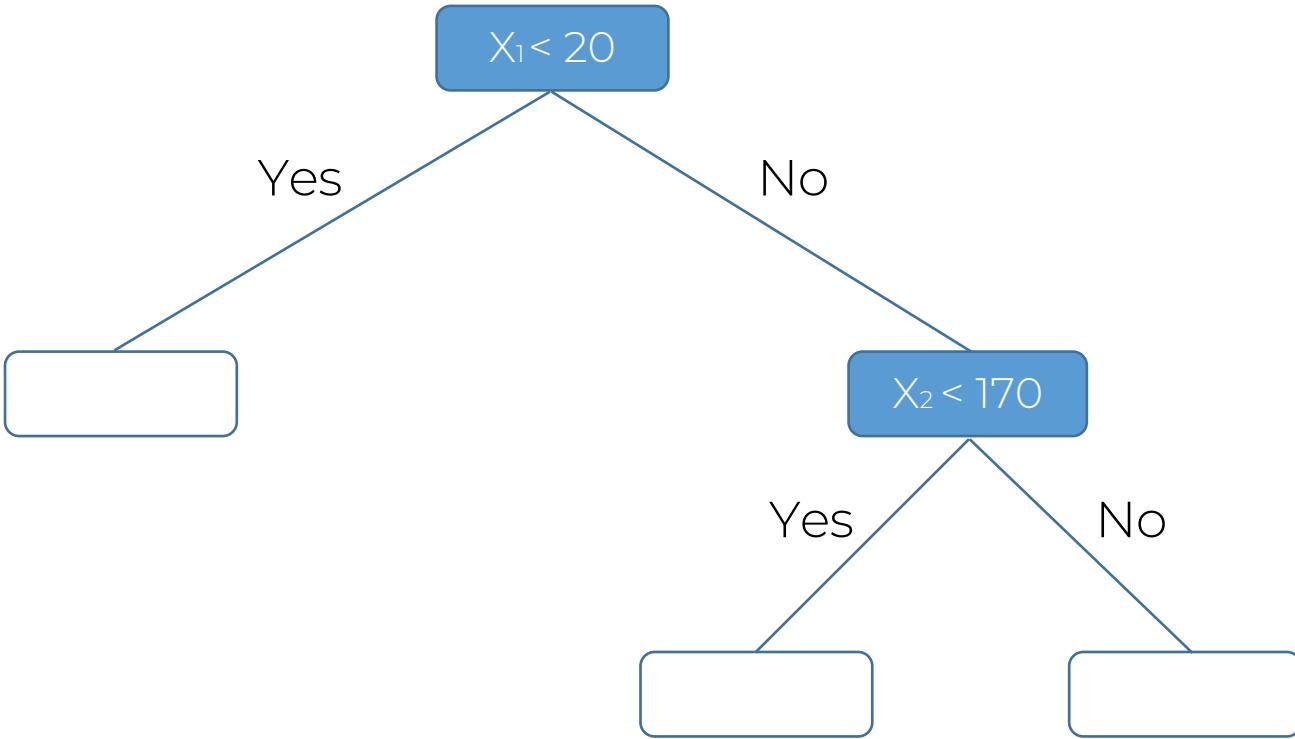
Decision Tree Intuition



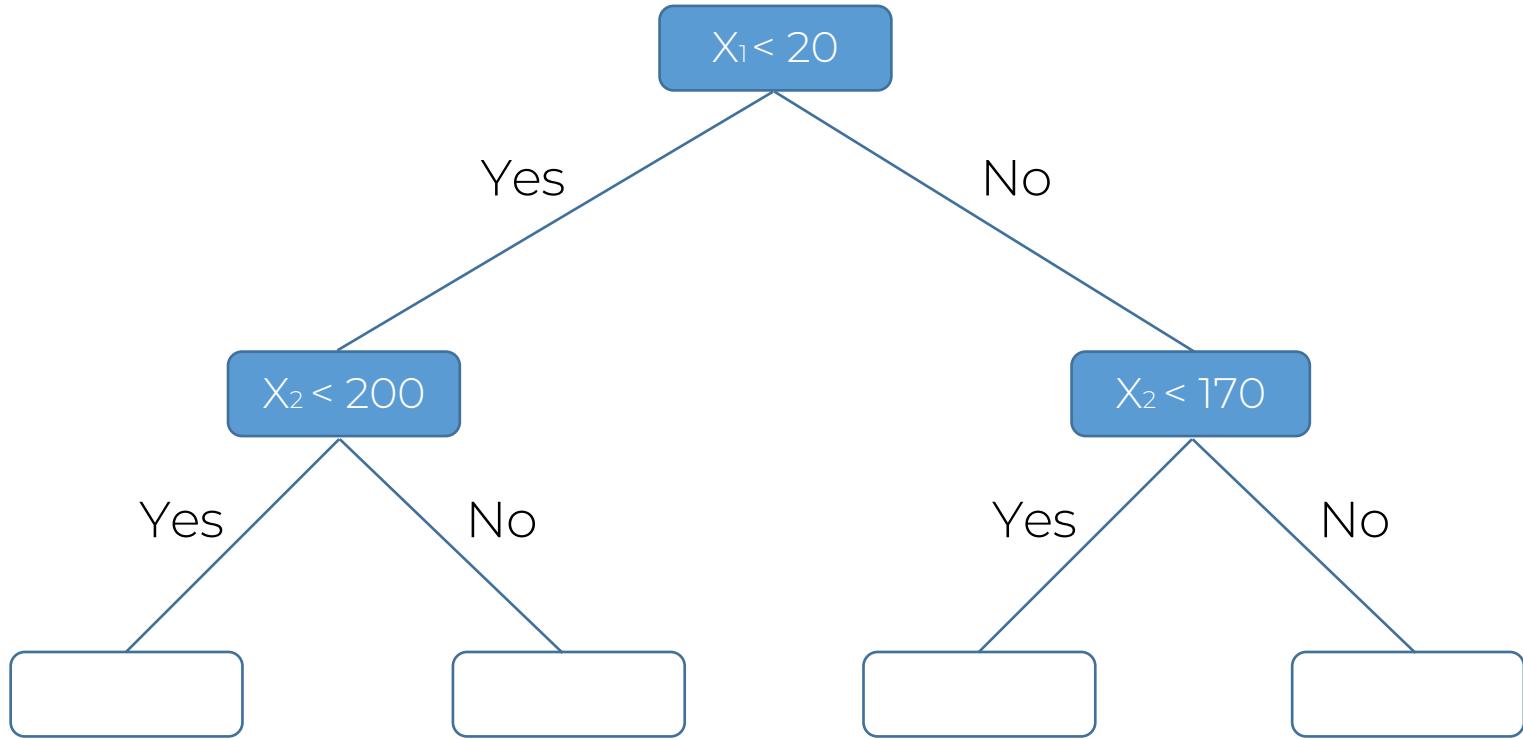
Decision Tree Intuition



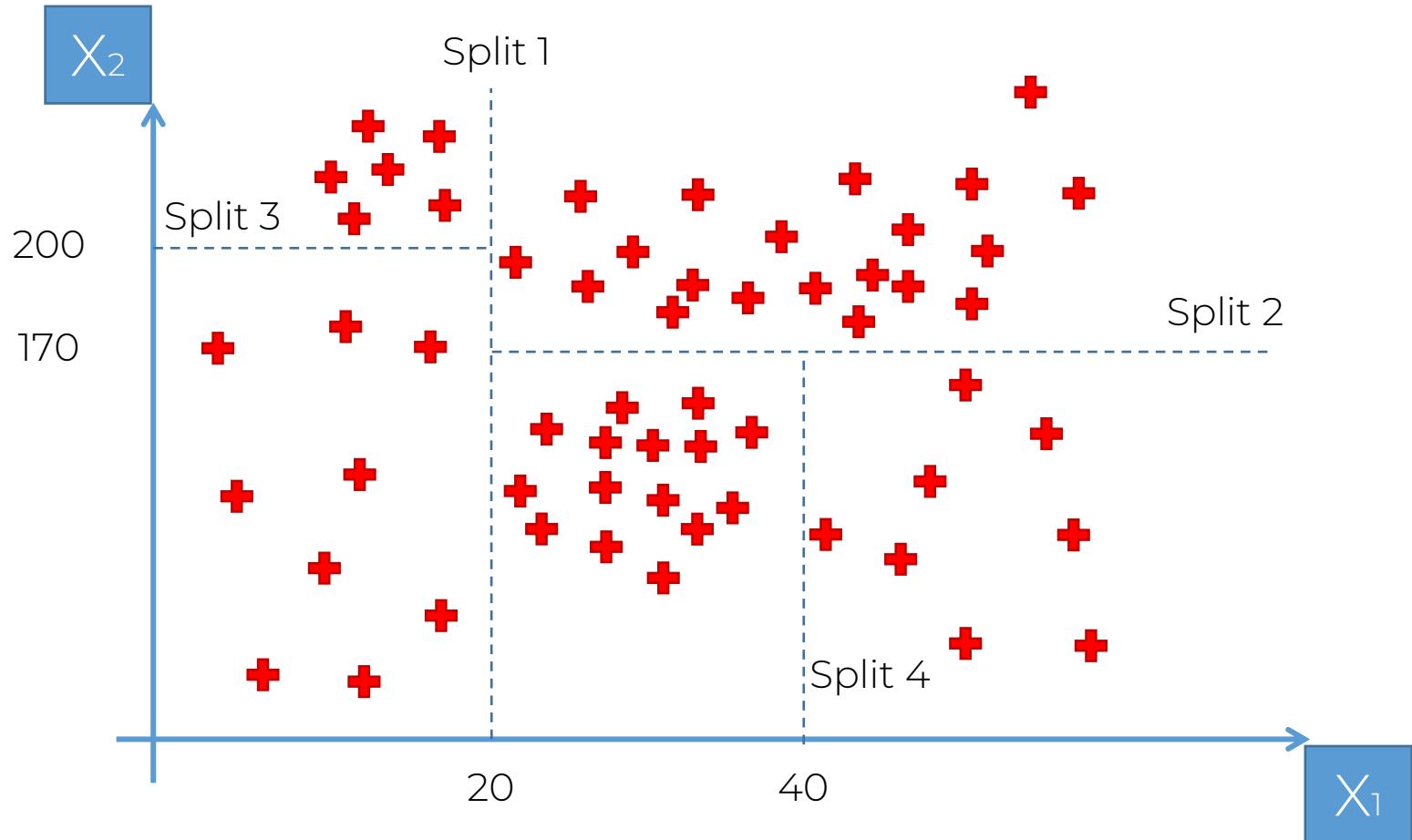
Decision Tree Intuition



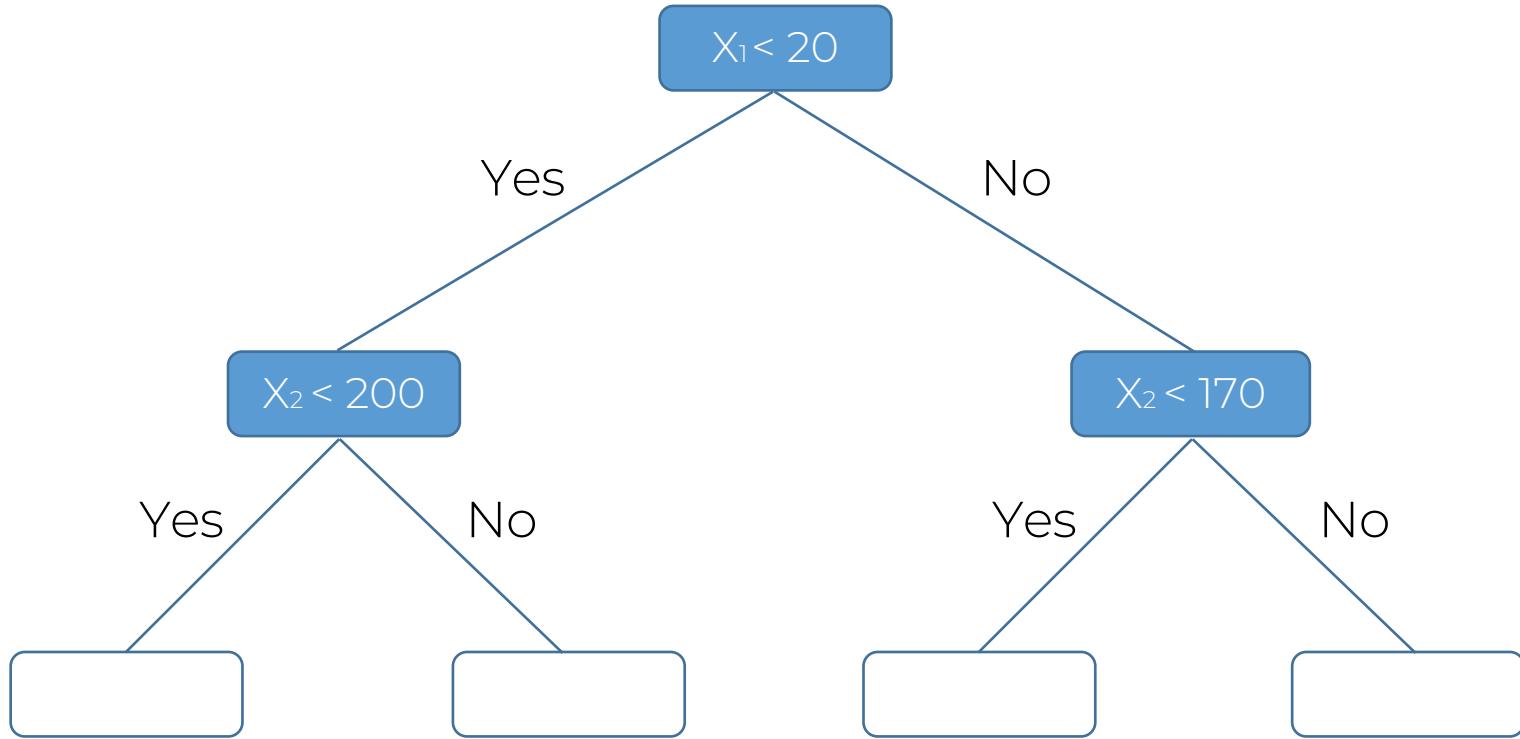
Decision Tree Intuition



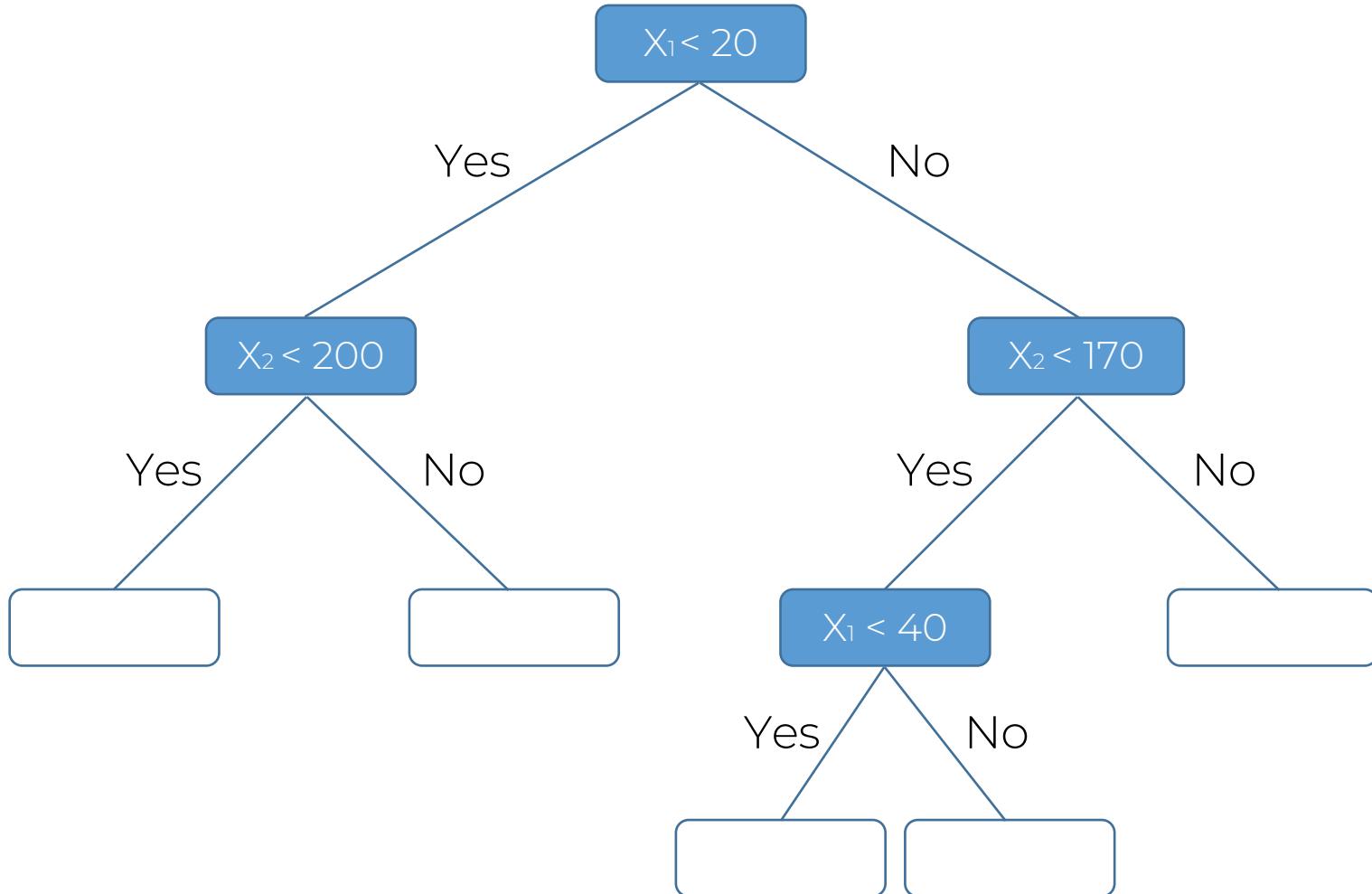
Decision Tree Intuition



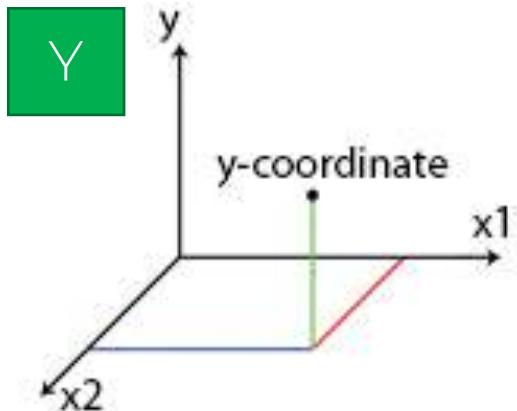
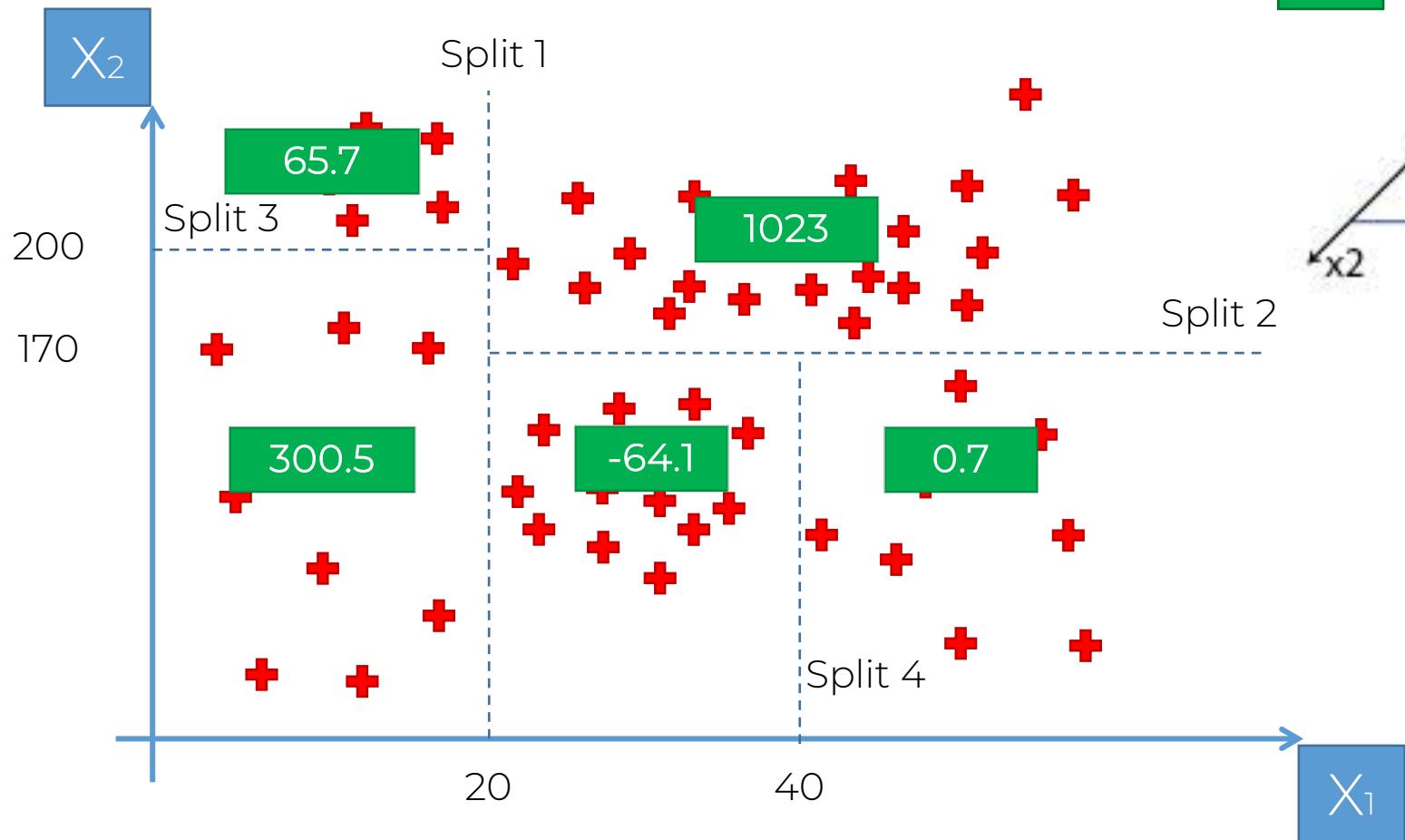
Decision Tree Intuition



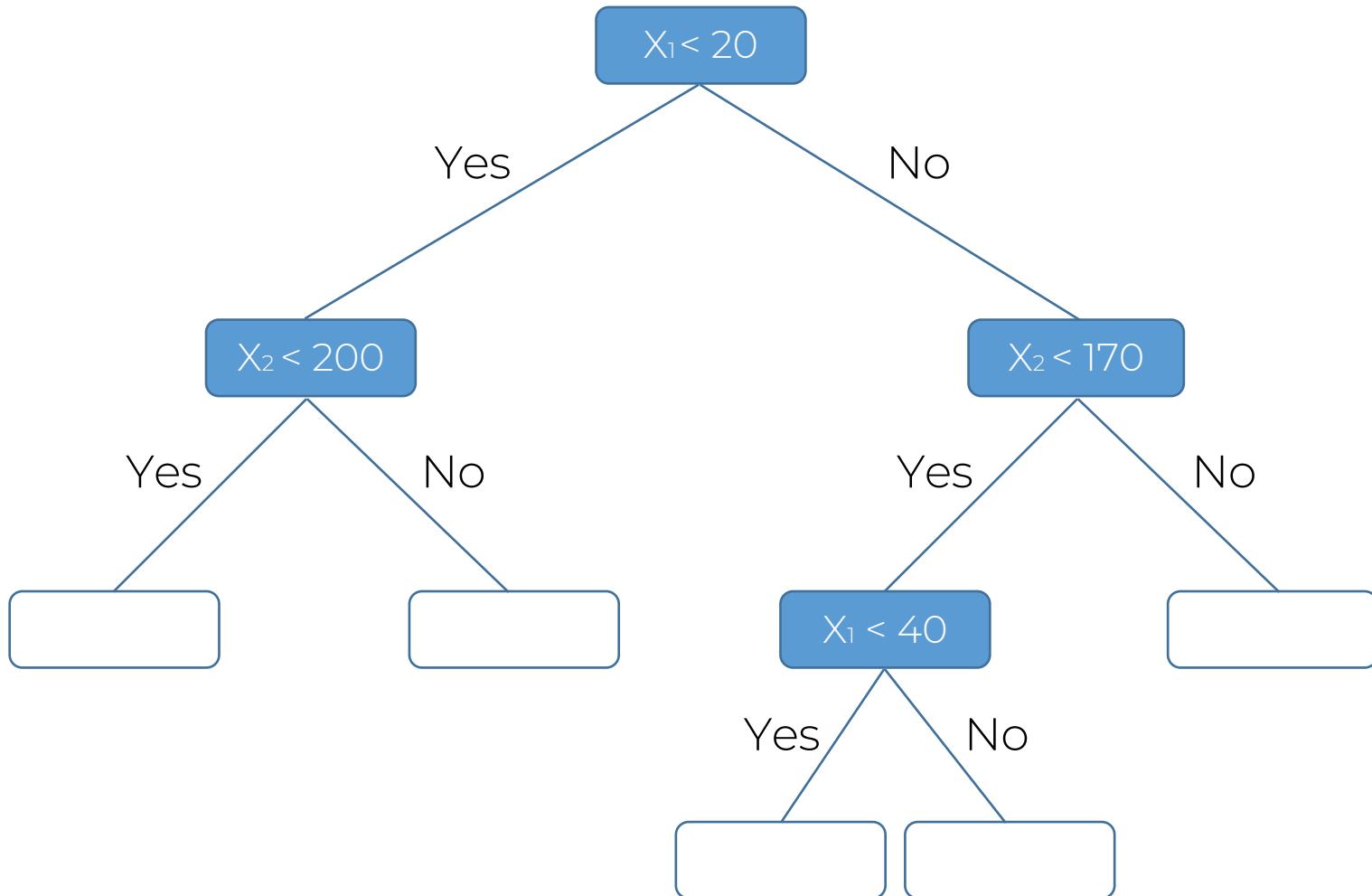
Decision Tree Intuition



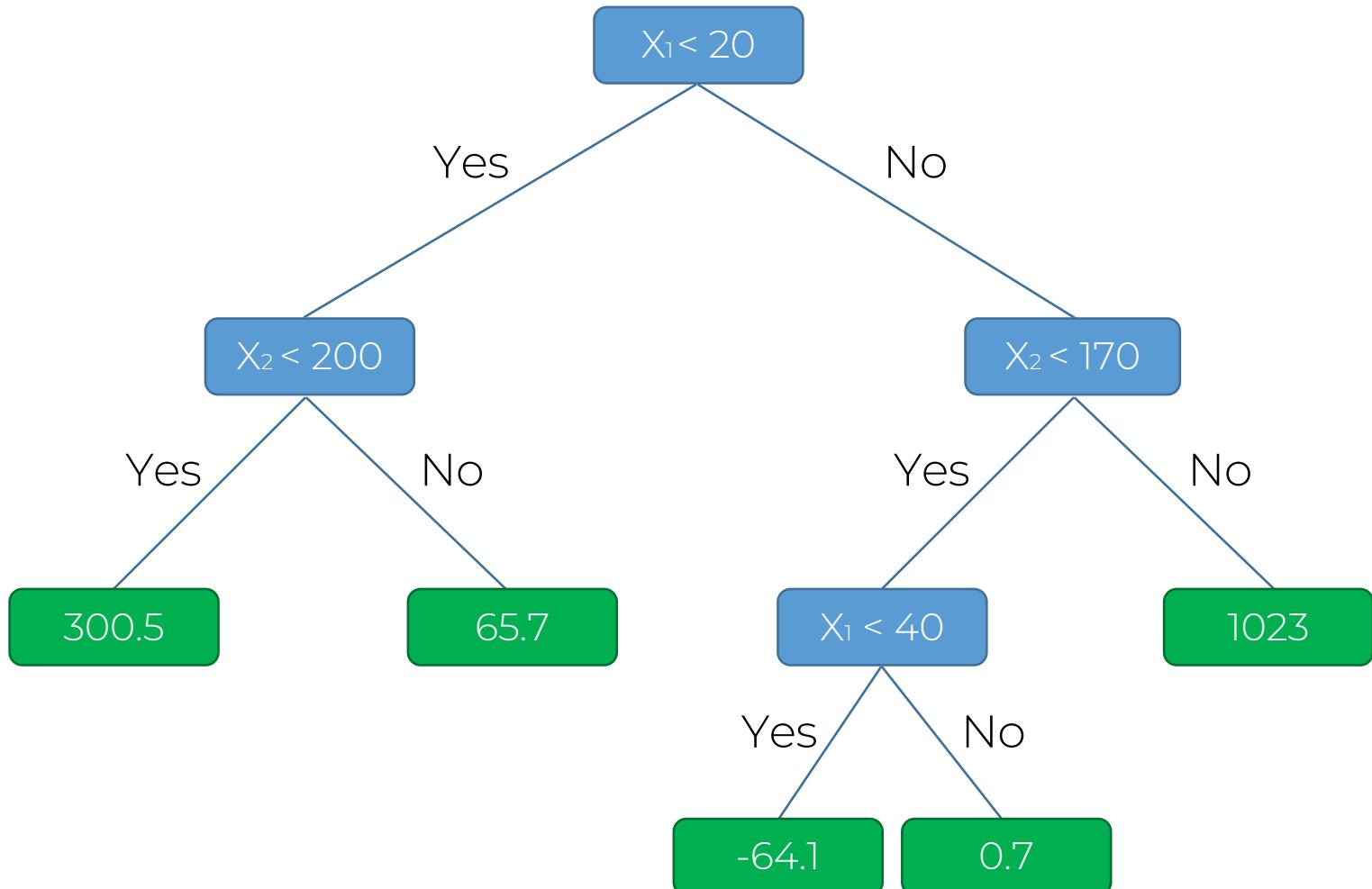
Decision Tree Intuition



Decision Tree Intuition



Decision Tree Intuition



Random Forest Intuition

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.

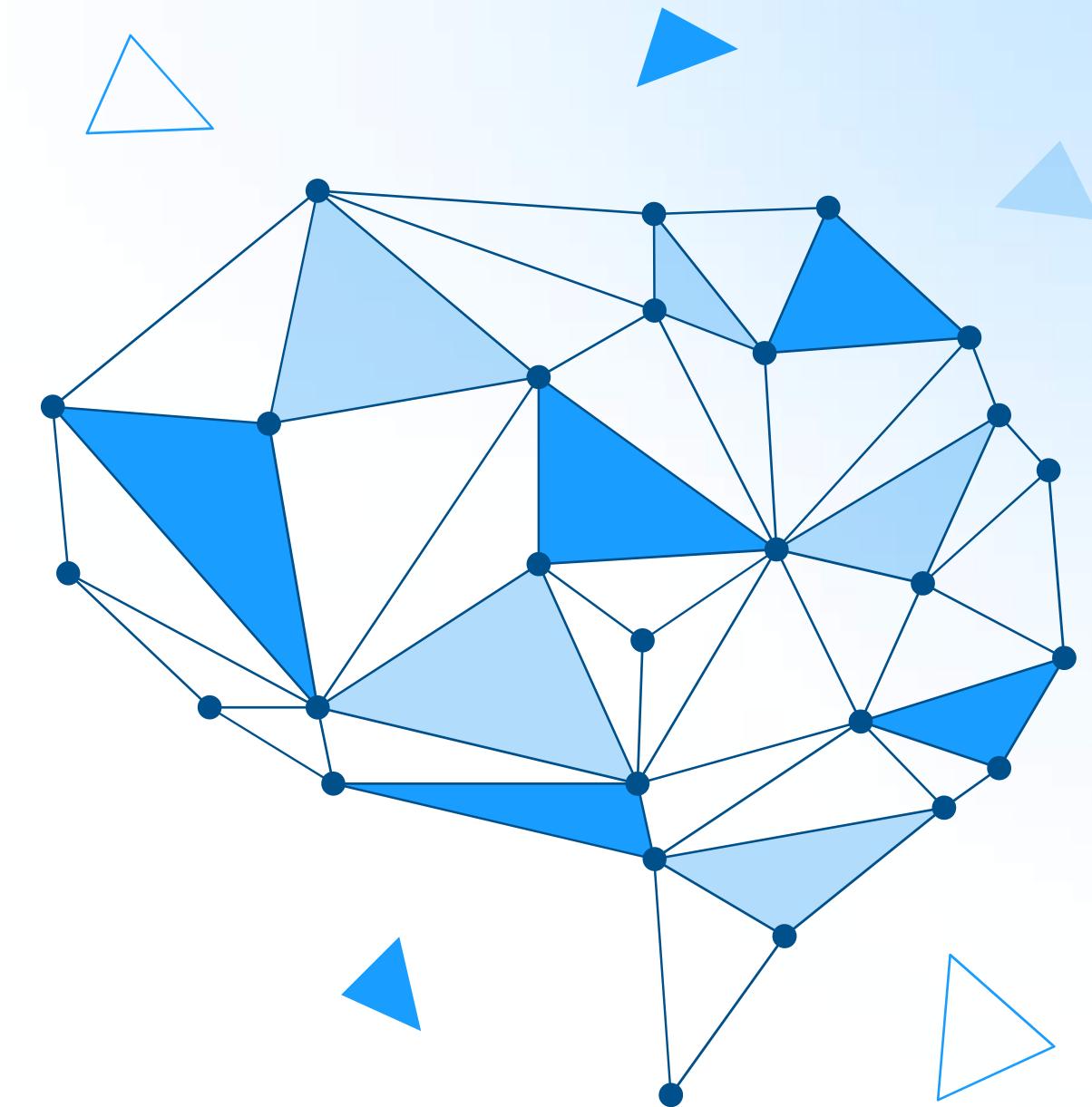


STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2



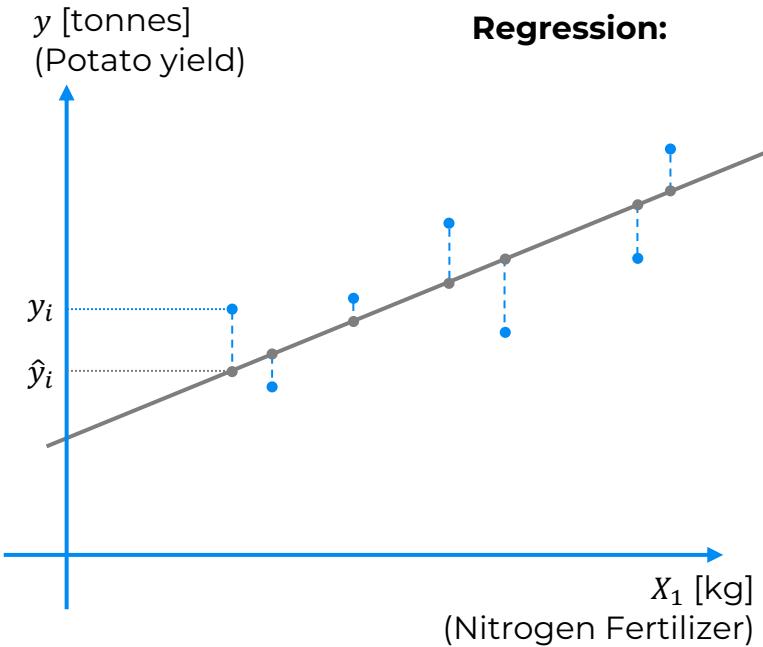
STEP 4: For a new data point, make each one of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values.

R Squared

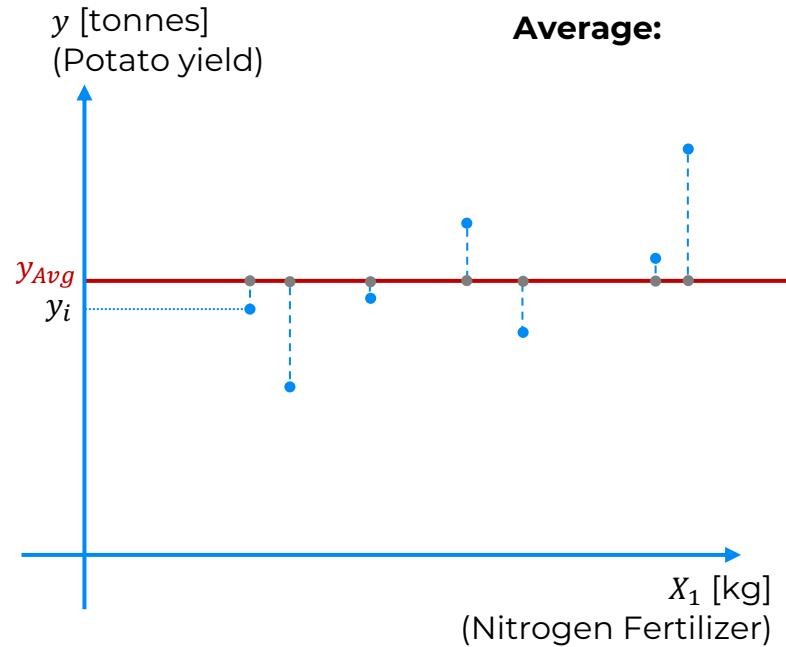




R Squared



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



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

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

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

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

R² – Goodness of fit
(greater is better)

Problem:

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

SS_{tot} doesn't change

SS_{res} will decrease or stay the same

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

(This is because of Ordinary Least Squares: $SS_{res} \rightarrow \text{Min}$)

Solution:

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

k – number of independent variables

n – sample size

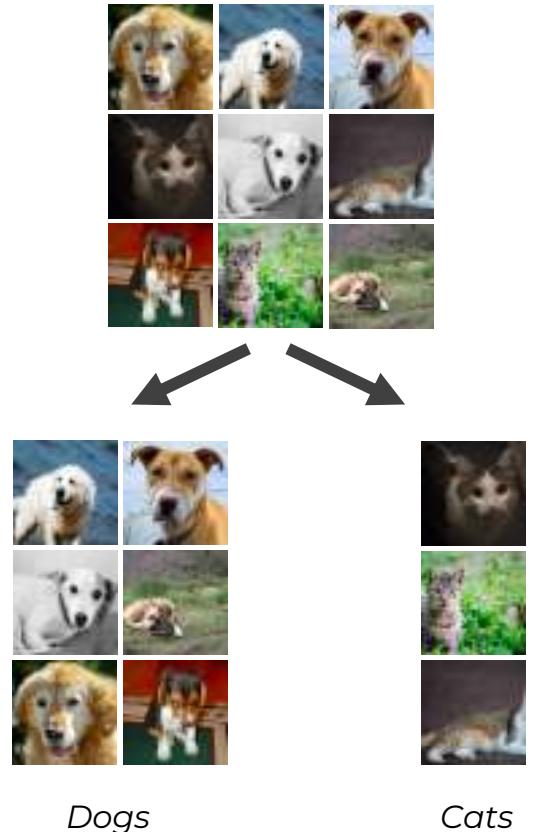
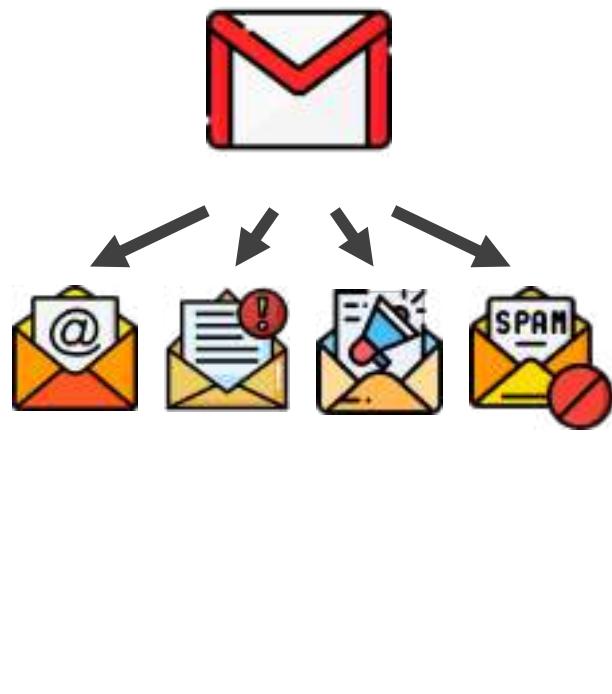
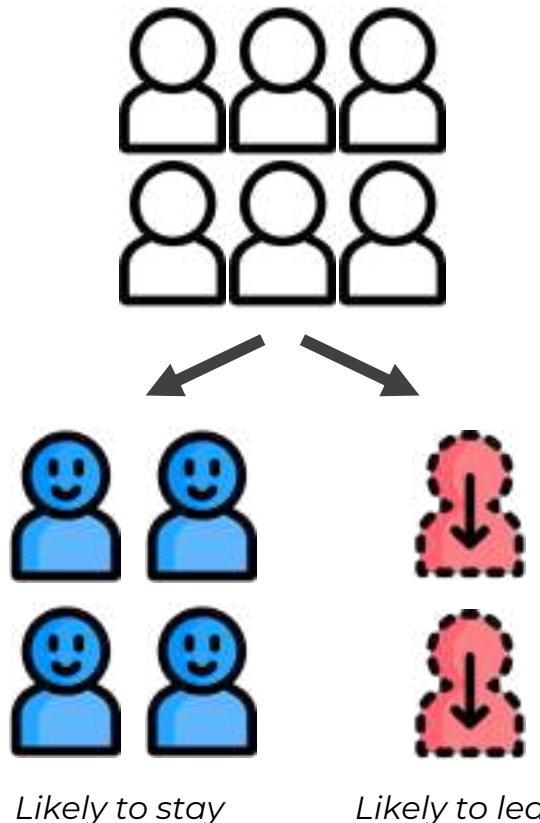


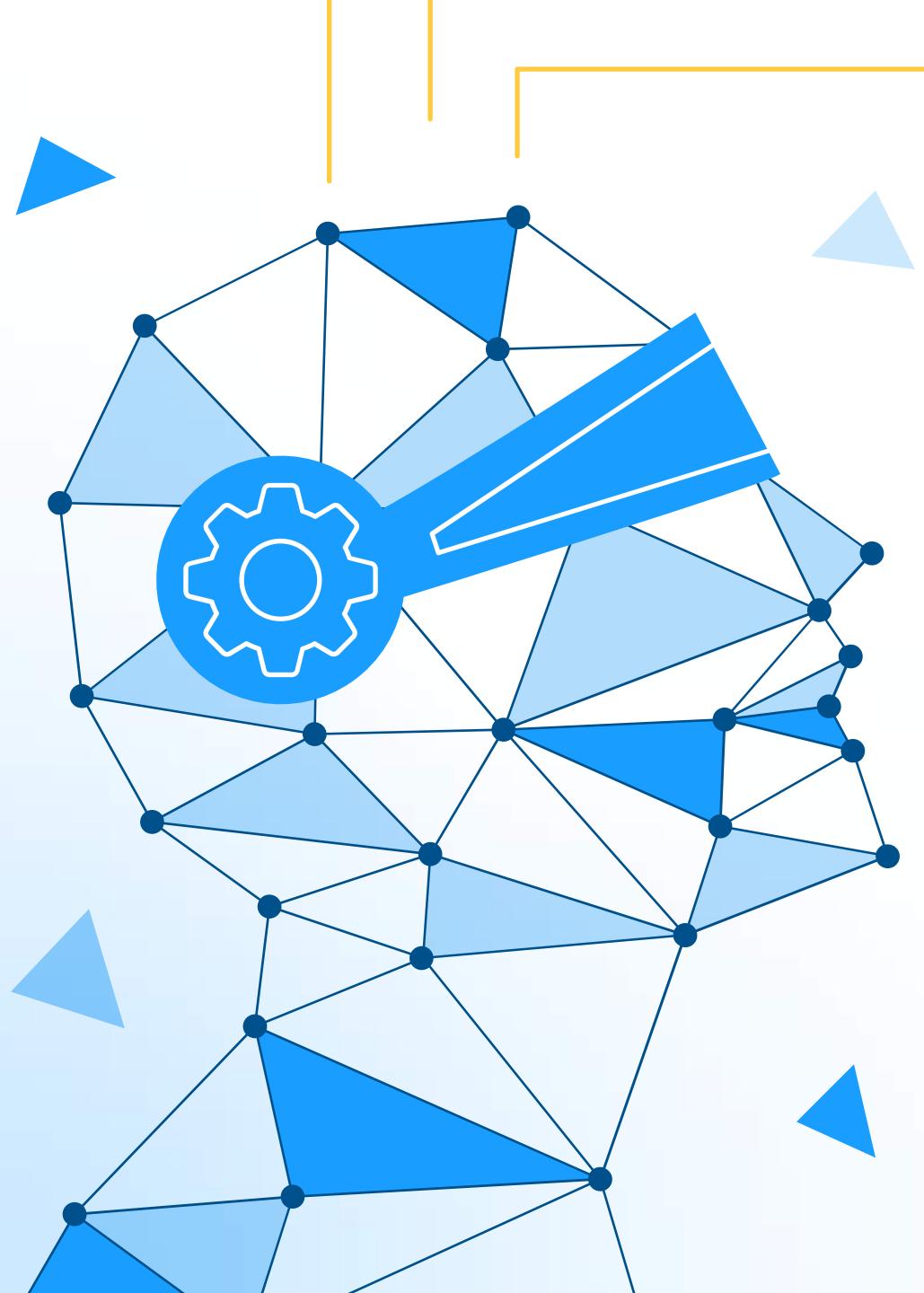


Classification

What is Classification?

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





Logistic Regression

Logistic Regression

Logistic regression: predict a categorical dependent variable from a number of independent variables.

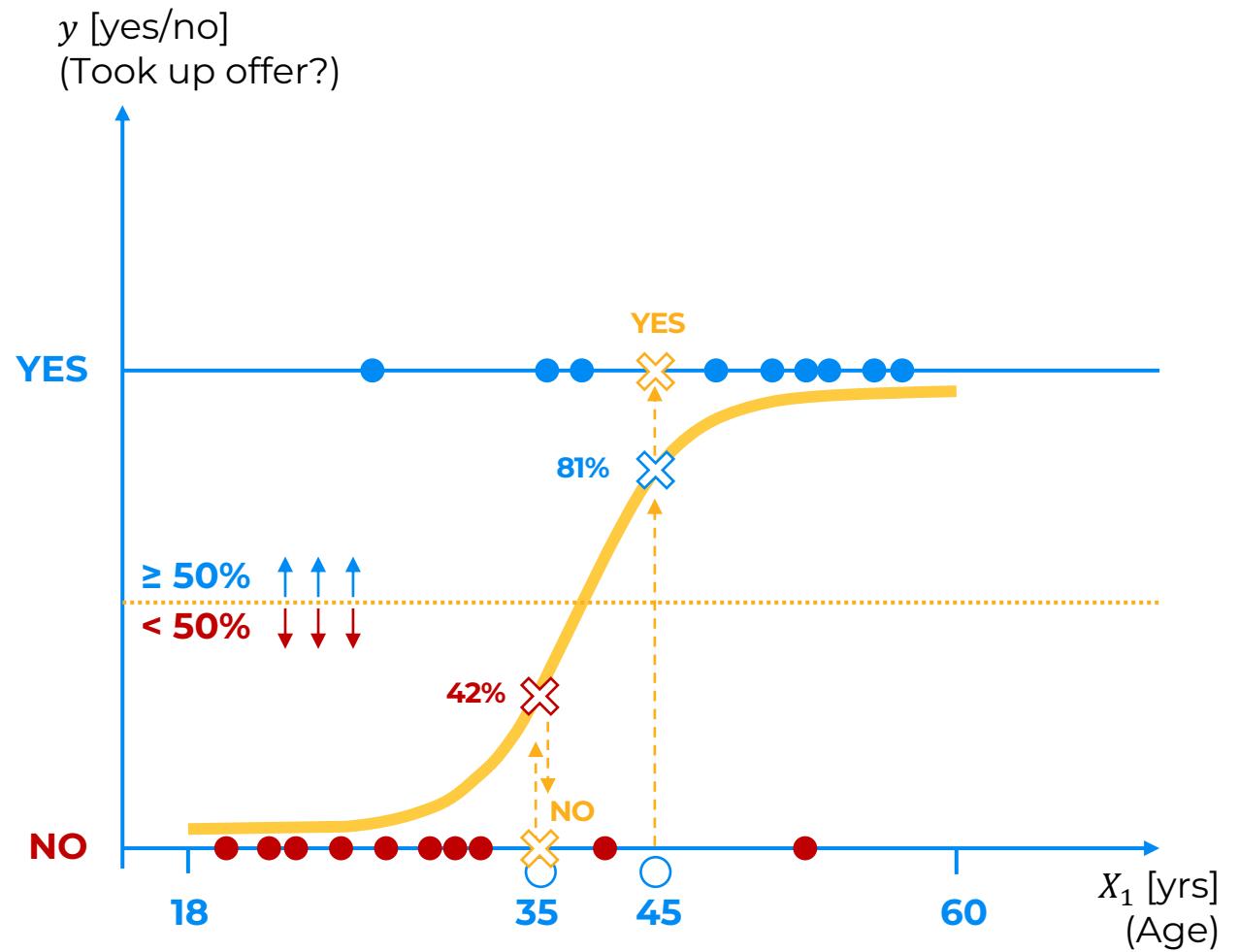


Will purchase health insurance:
Yes / No



Age

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



Logistic Regression



NOT FOR DISTRIBUTION

© SUPERDATASCIENCE

www.superdatascience.com

Will purchase
health insurance:
Yes / No

~



Age



Income



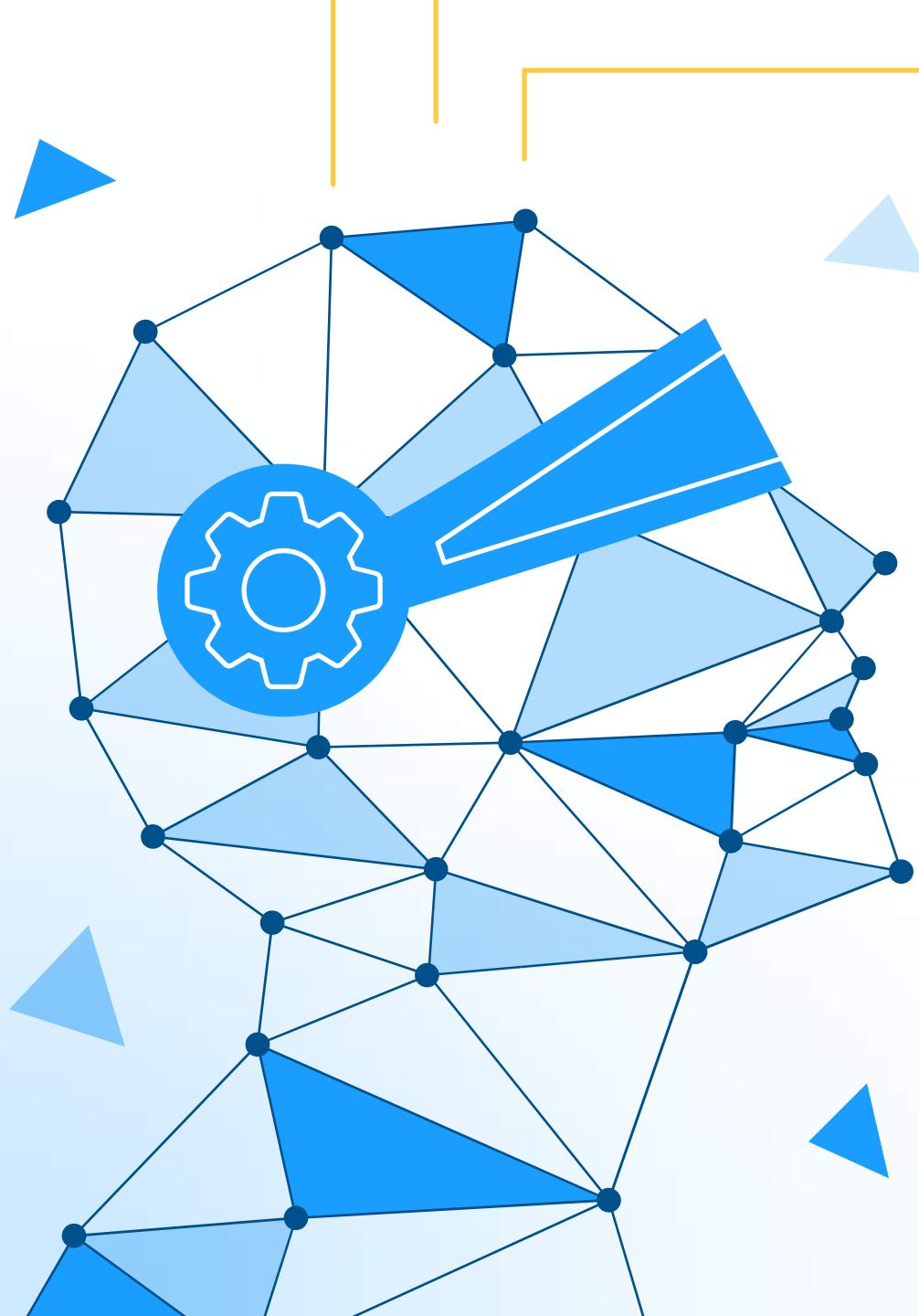
Level of
Education



Family or
Single

$$\ln \frac{p}{1-p} = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X_4$$





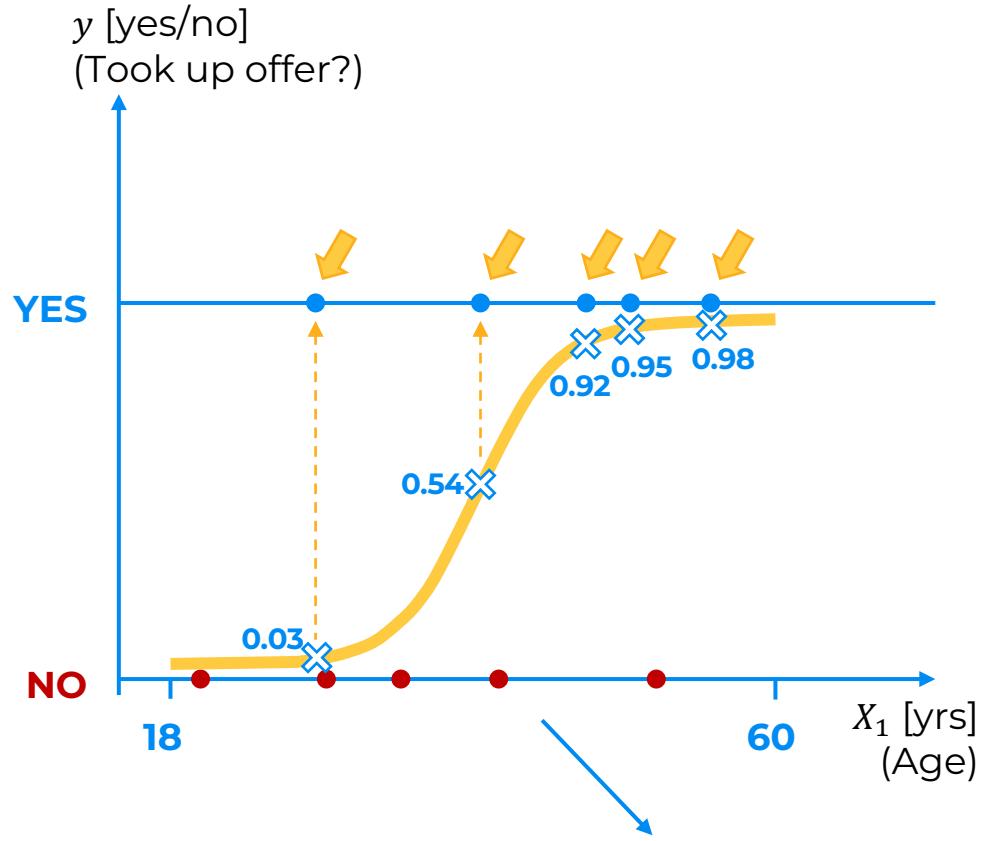
Maximum Likelihood

Maximum Likelihood



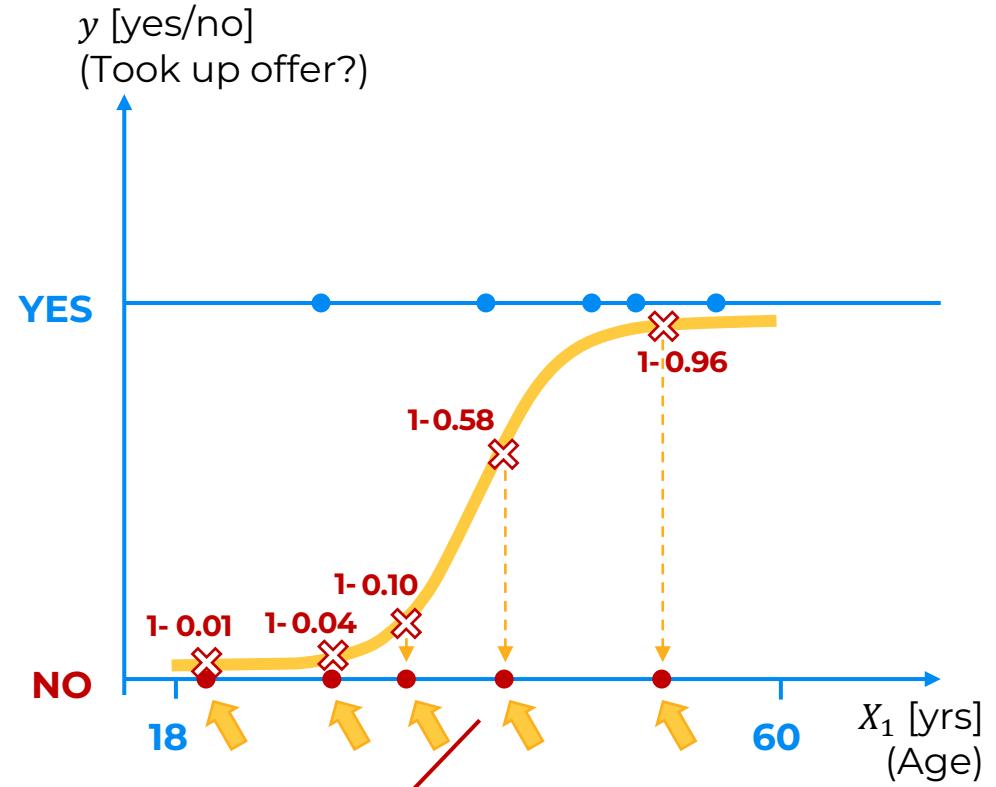
NOT FOR DISTRIBUTION

© SUPERDATASCIENCE

www.superdatascience.com

$$\text{Likelihood} = 0.03 \times 0.54 \times 0.92 \times 0.95 \times 0.98 \times (1 - 0.01) \times (1 - 0.04) \times (1 - 0.10) \times (1 - 0.58) \times (1 - 0.96)$$

$$\text{Likelihood} = \mathbf{0.00019939}$$



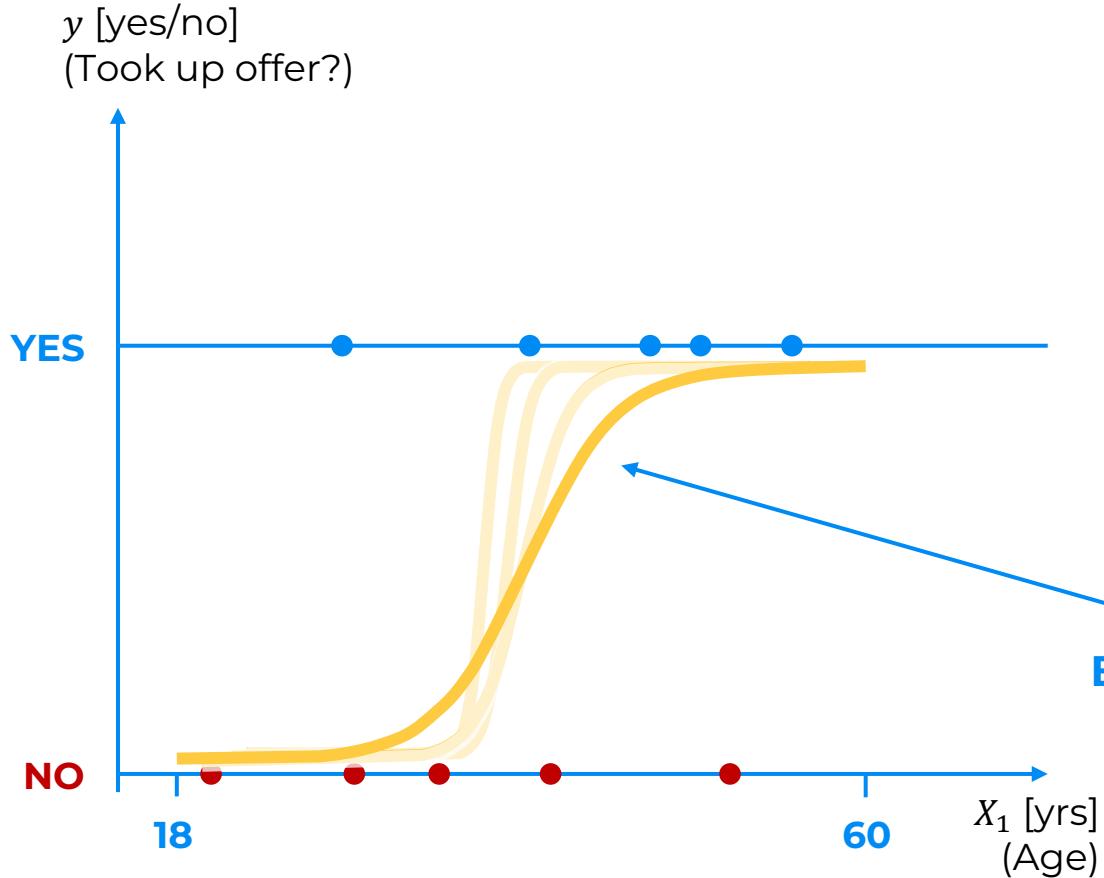
$$\begin{aligned} &1 - 0.01 \\ &1 - 0.04 \\ &1 - 0.10 \\ &1 - 0.58 \\ &1 - 0.96 \end{aligned}$$

Maximum Likelihood



NOT FOR DISTRIBUTION

© SUPERDATASCIENCE

www.superdatascience.com

Likelihood = 0.00007418

Likelihood = 0.00012845

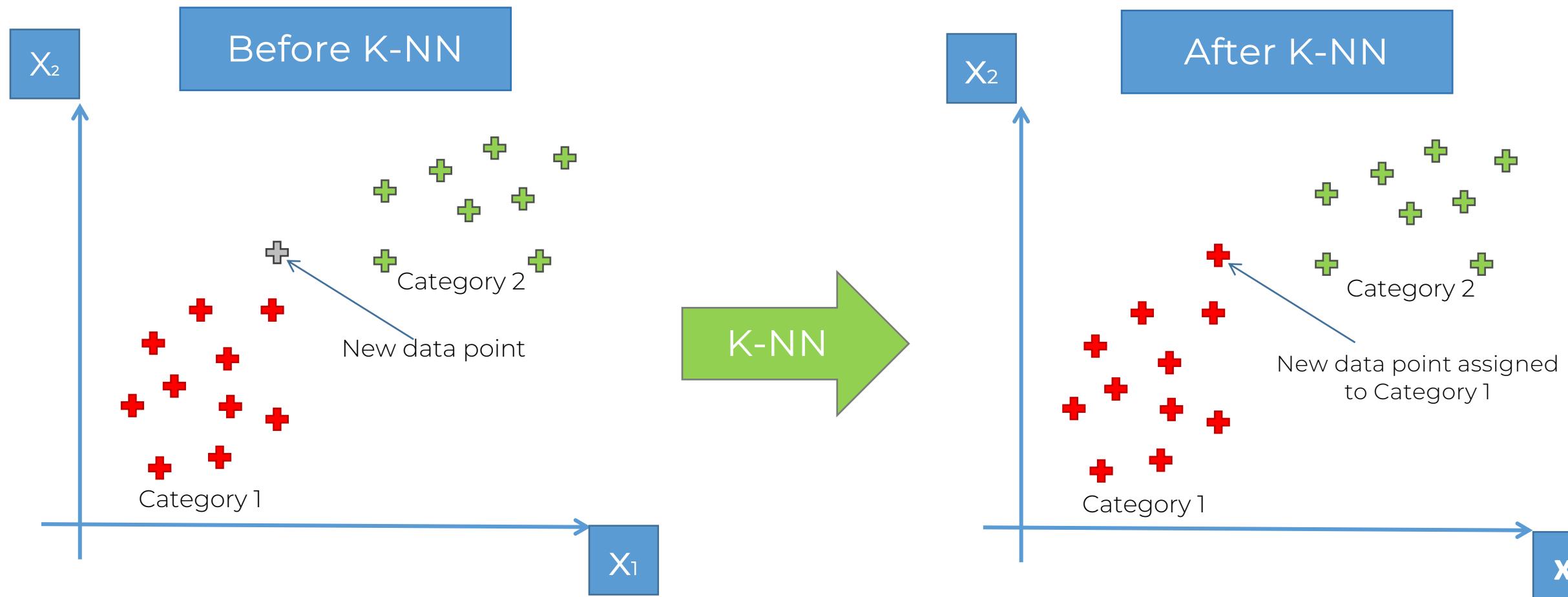
Likelihood = 0.00016553

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



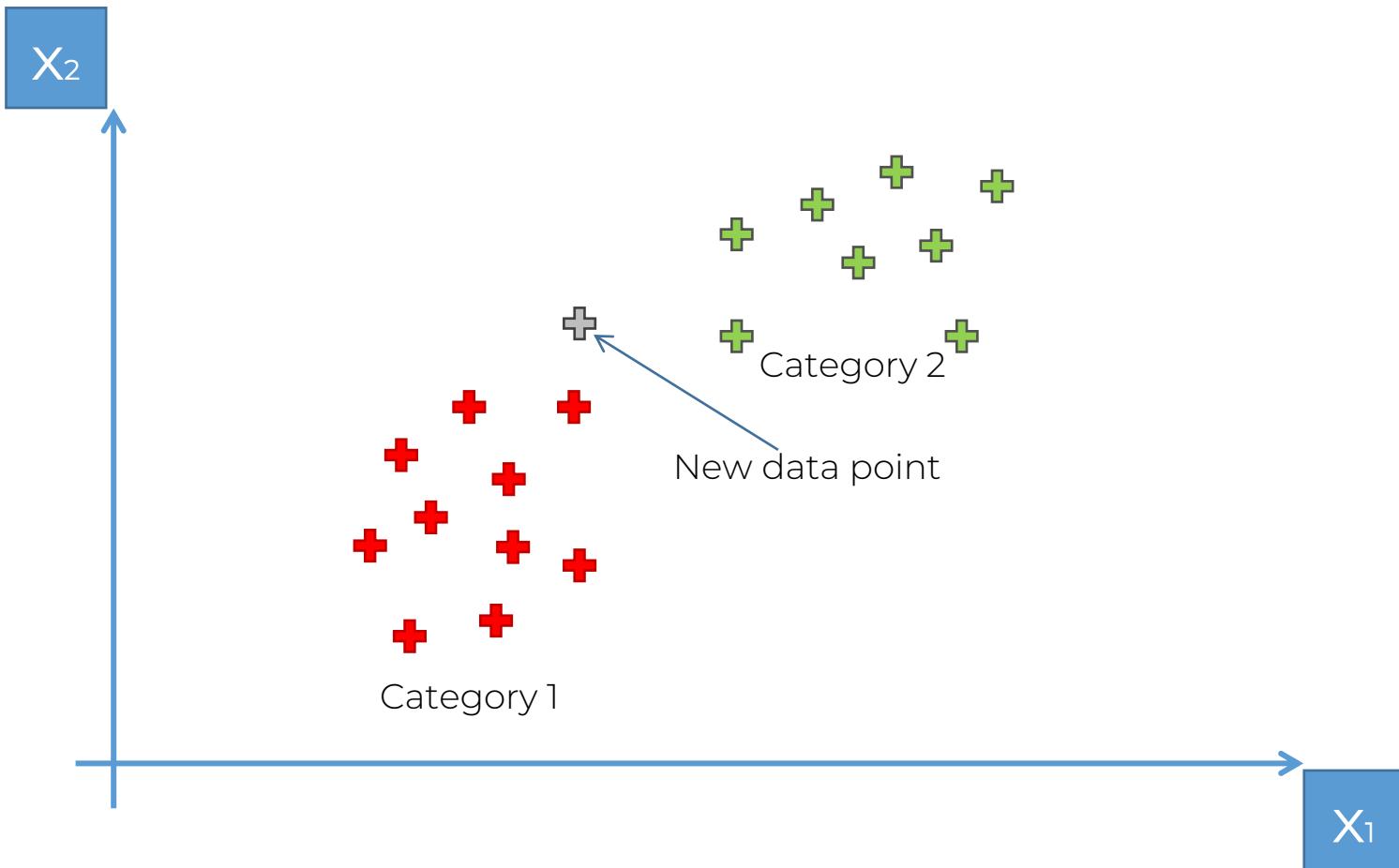
STEP 4: Assign the new data point to the category where you counted the most neighbors



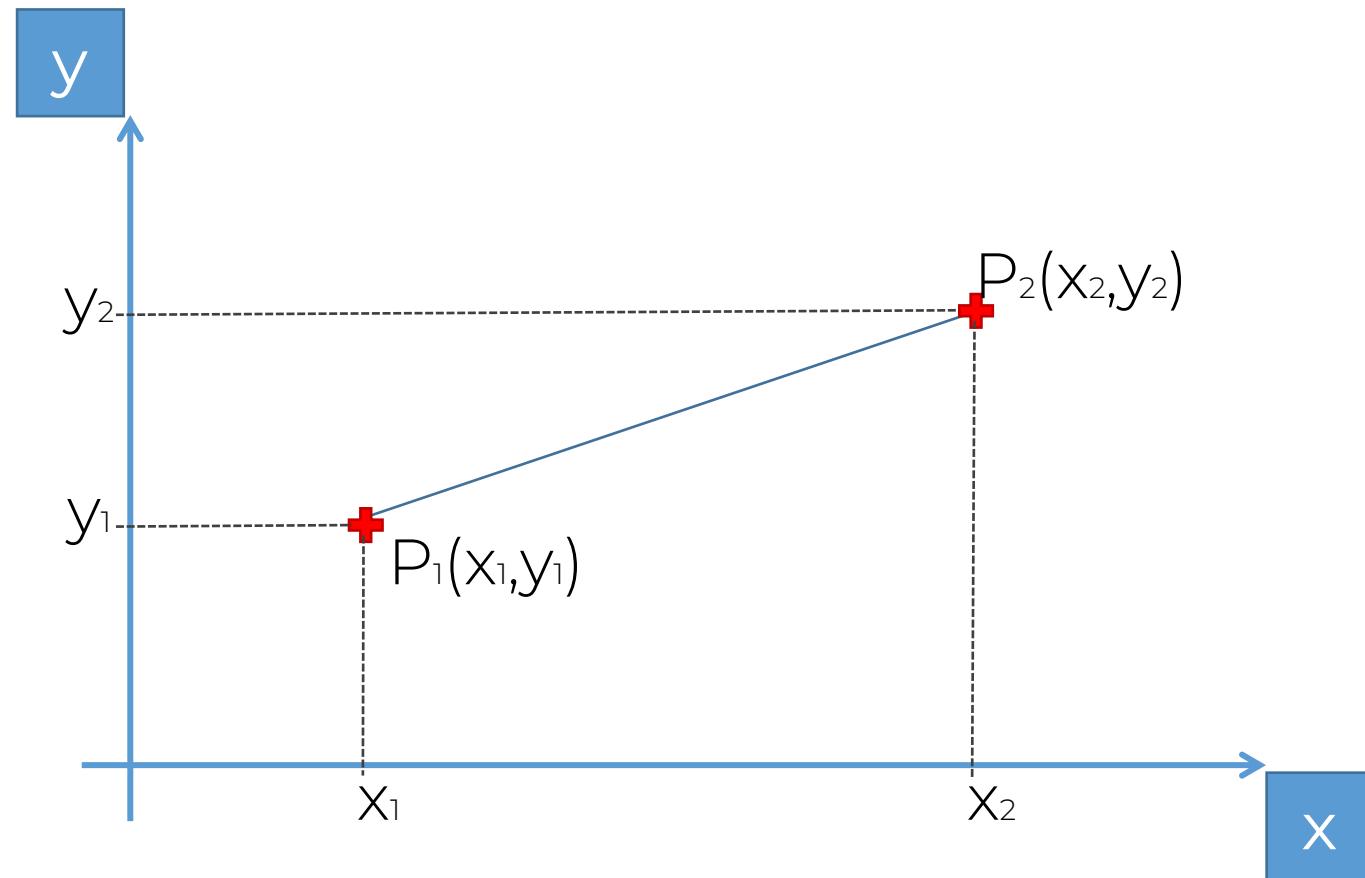
Your Model is Ready

K-NN algorithm

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

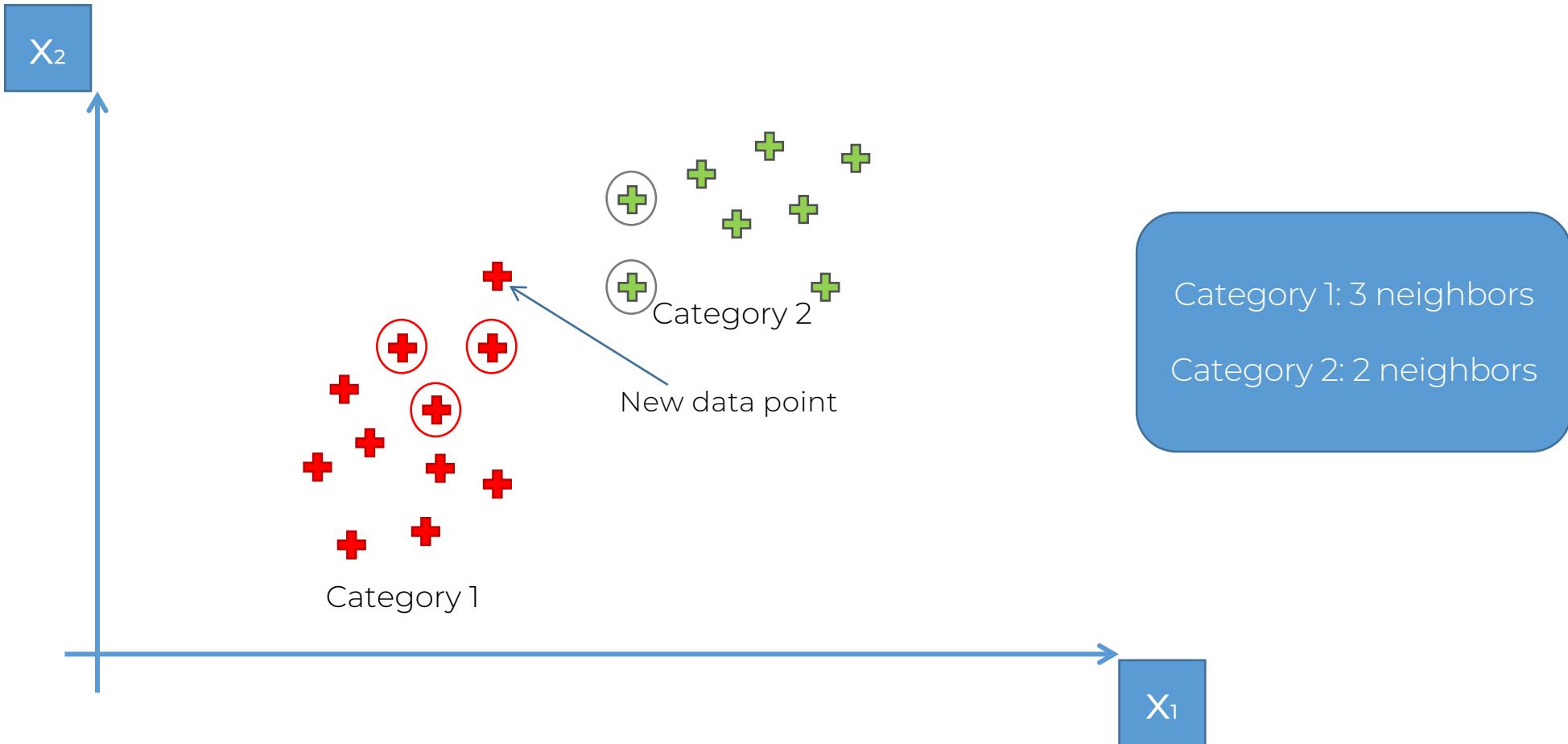


Euclidean Distance



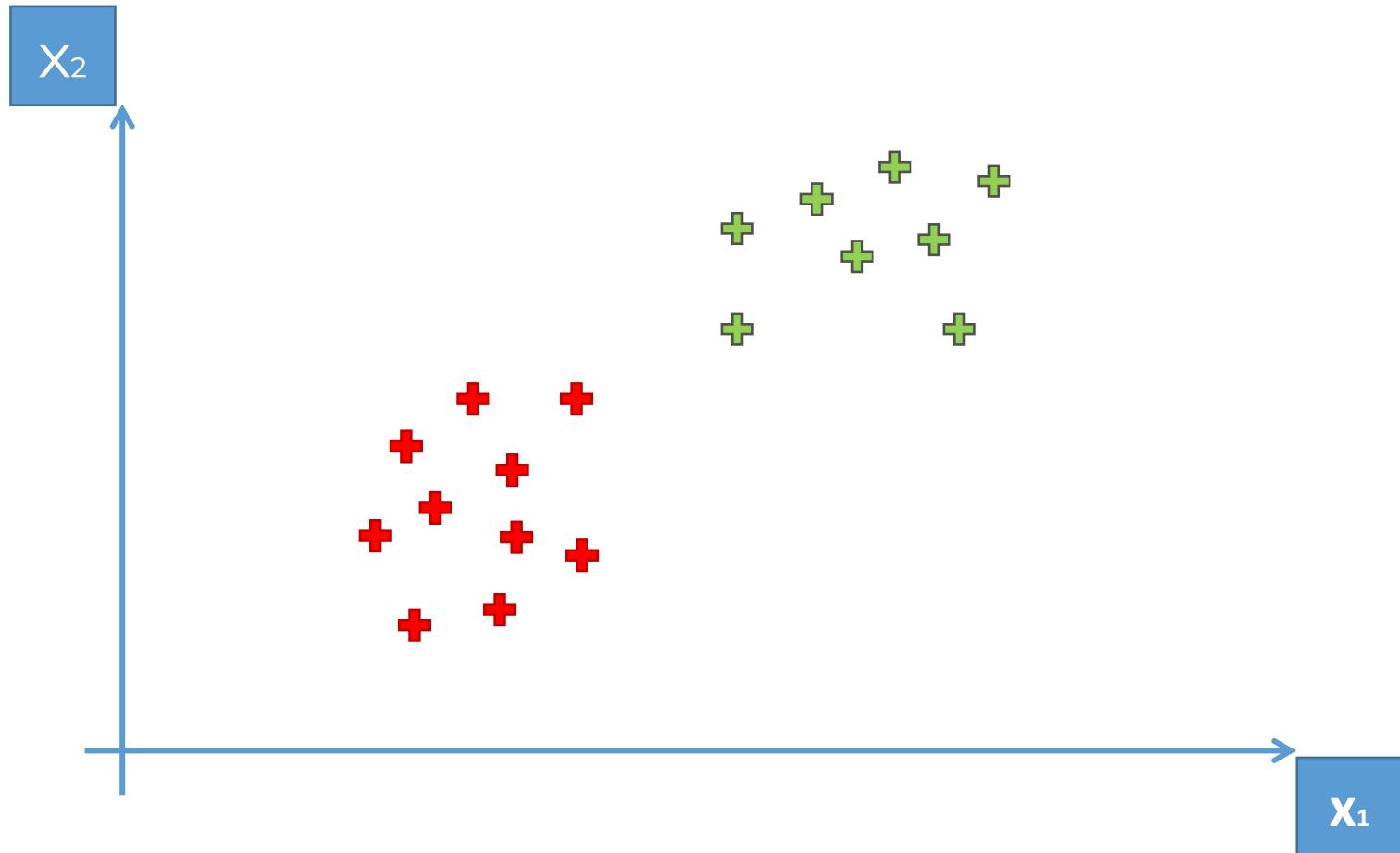
$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-NN algorithm

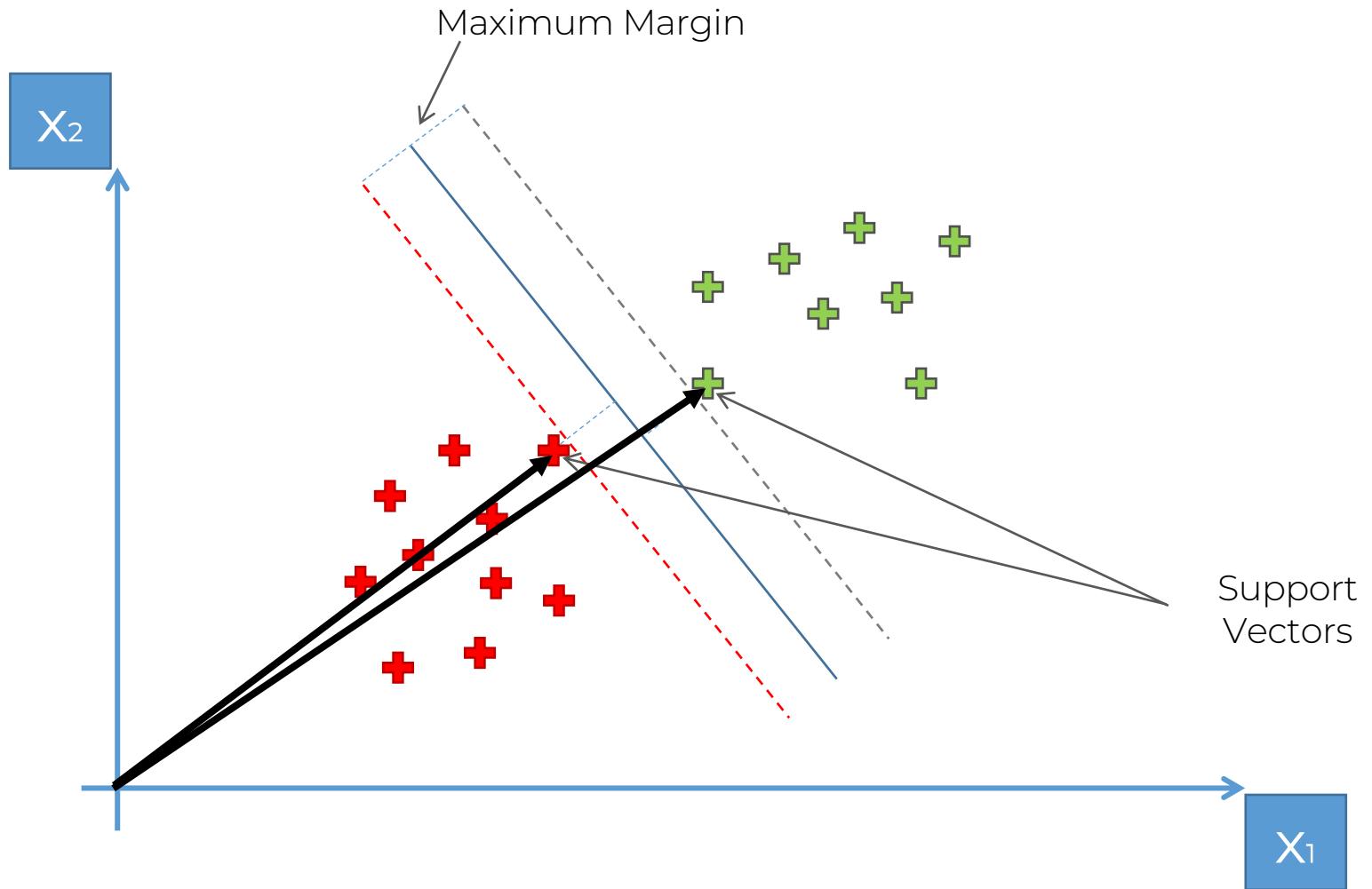


SVM Intuition

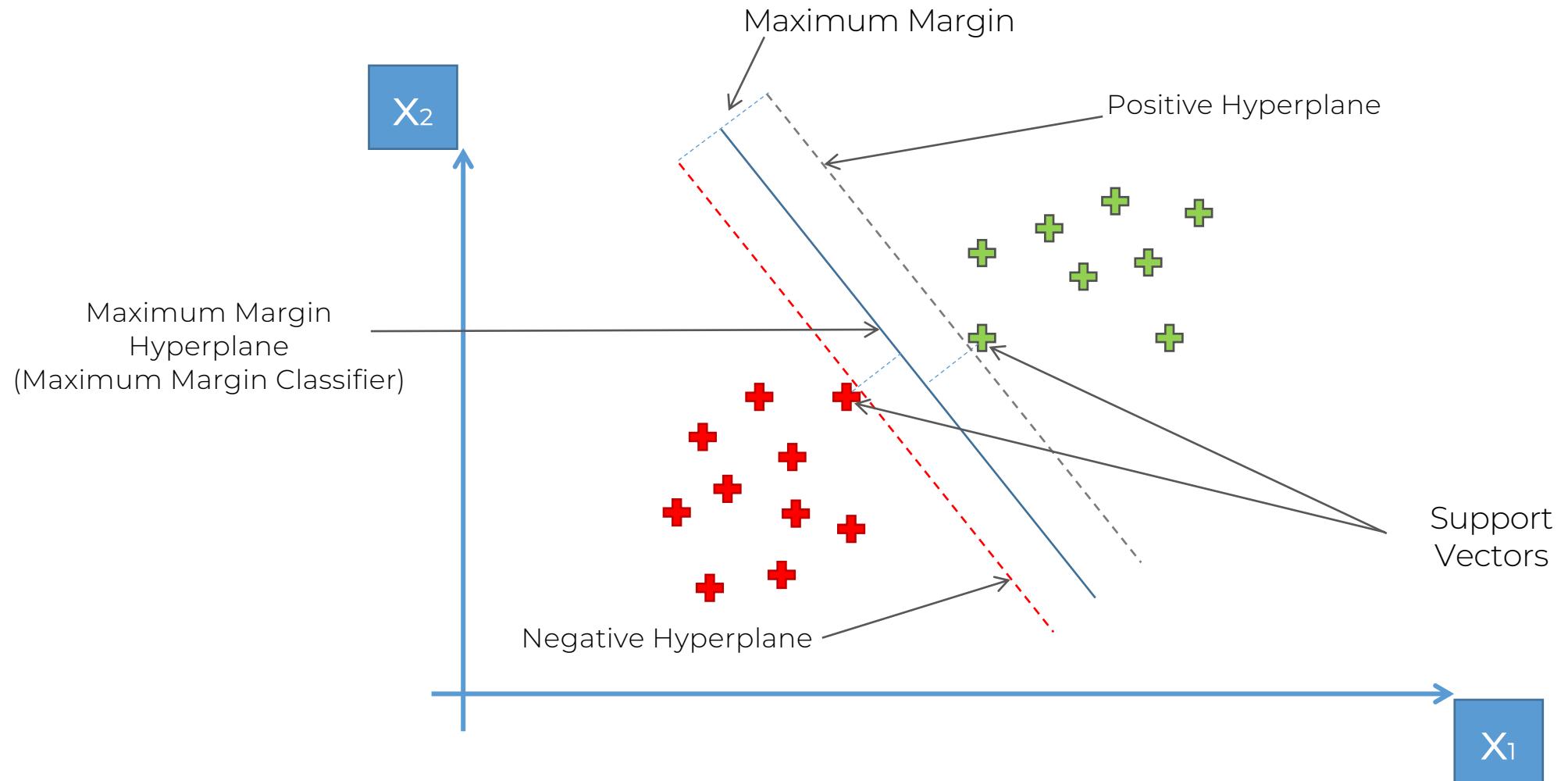
How to separate these points ?



Support Vectors

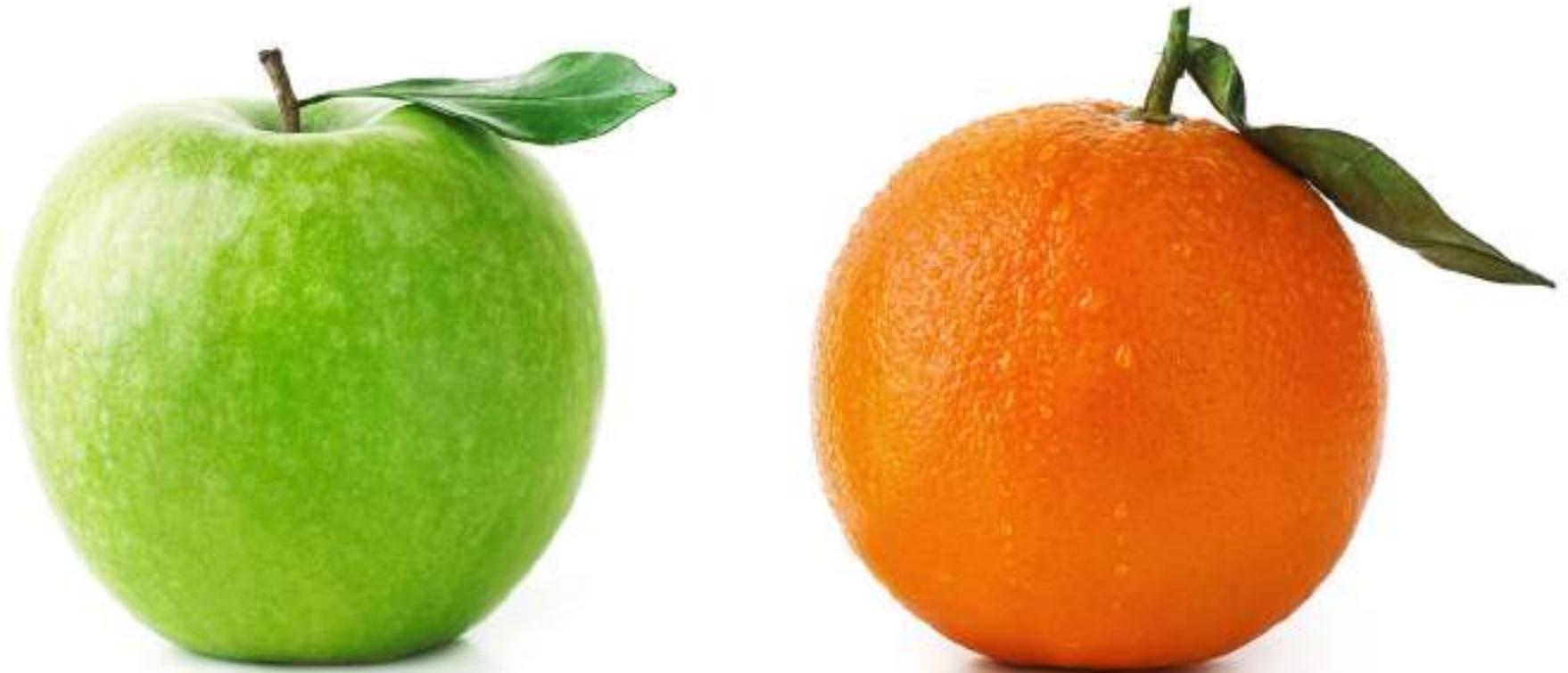


Hyperplanes

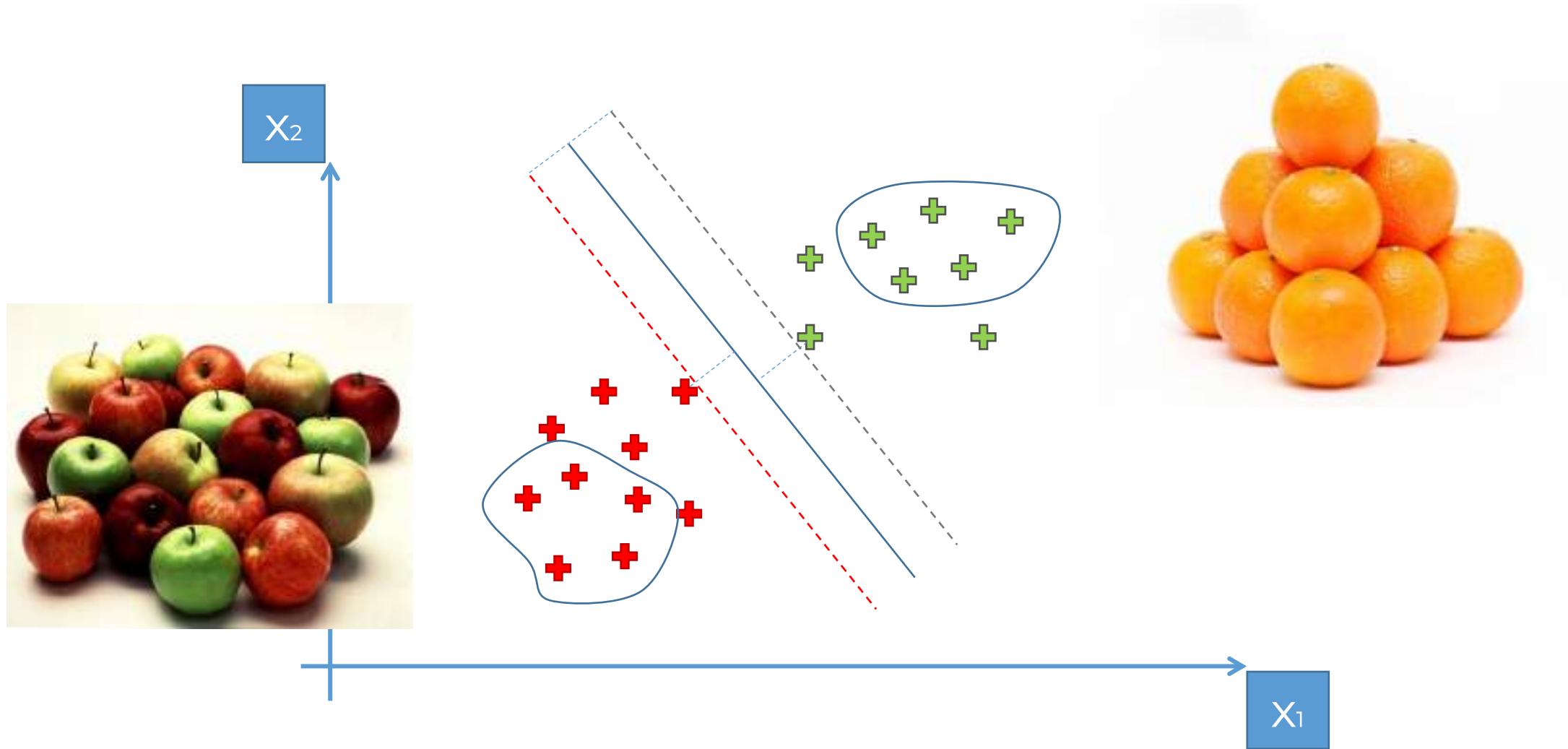


What's So Special About SVMs?

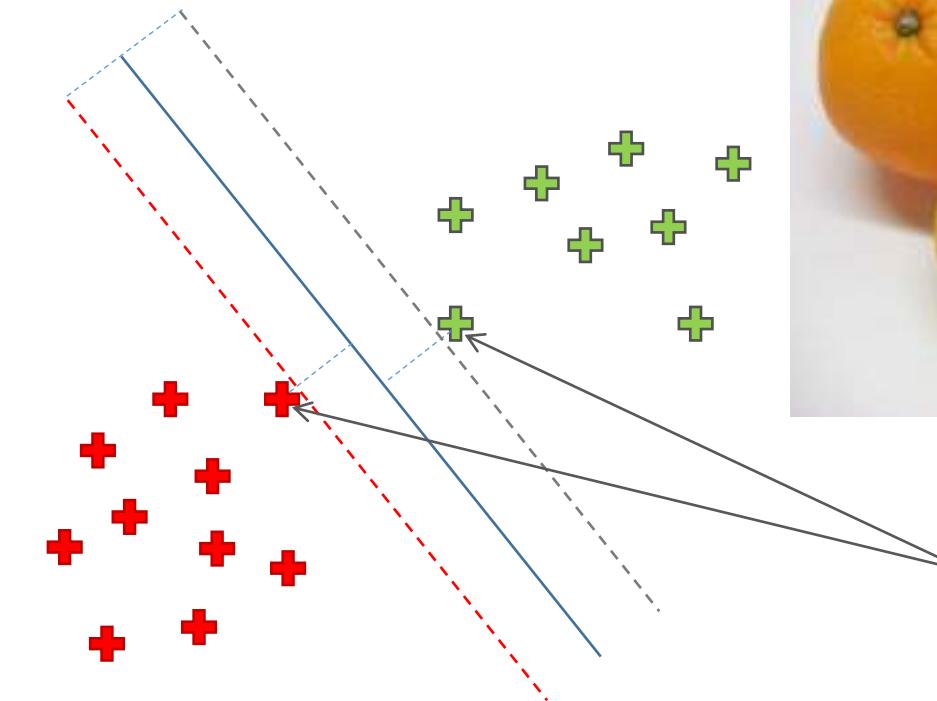
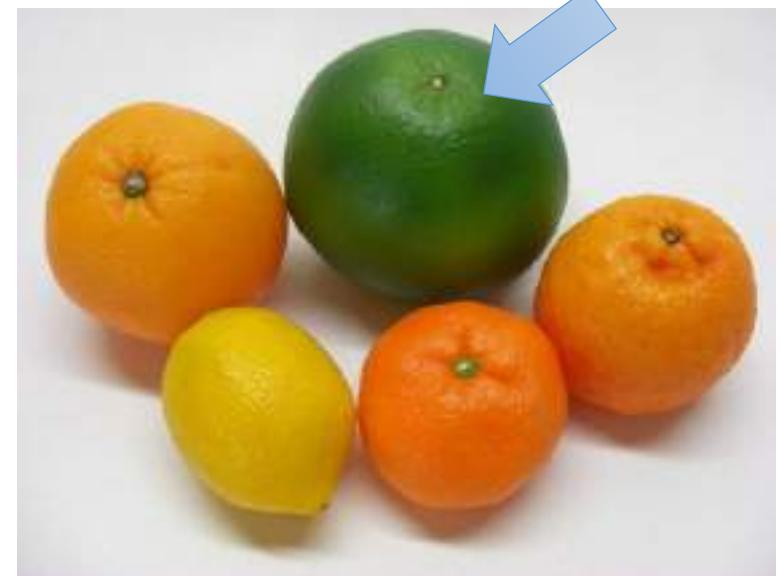
What's So Special About SVMs?



What's So Special About SVMs?



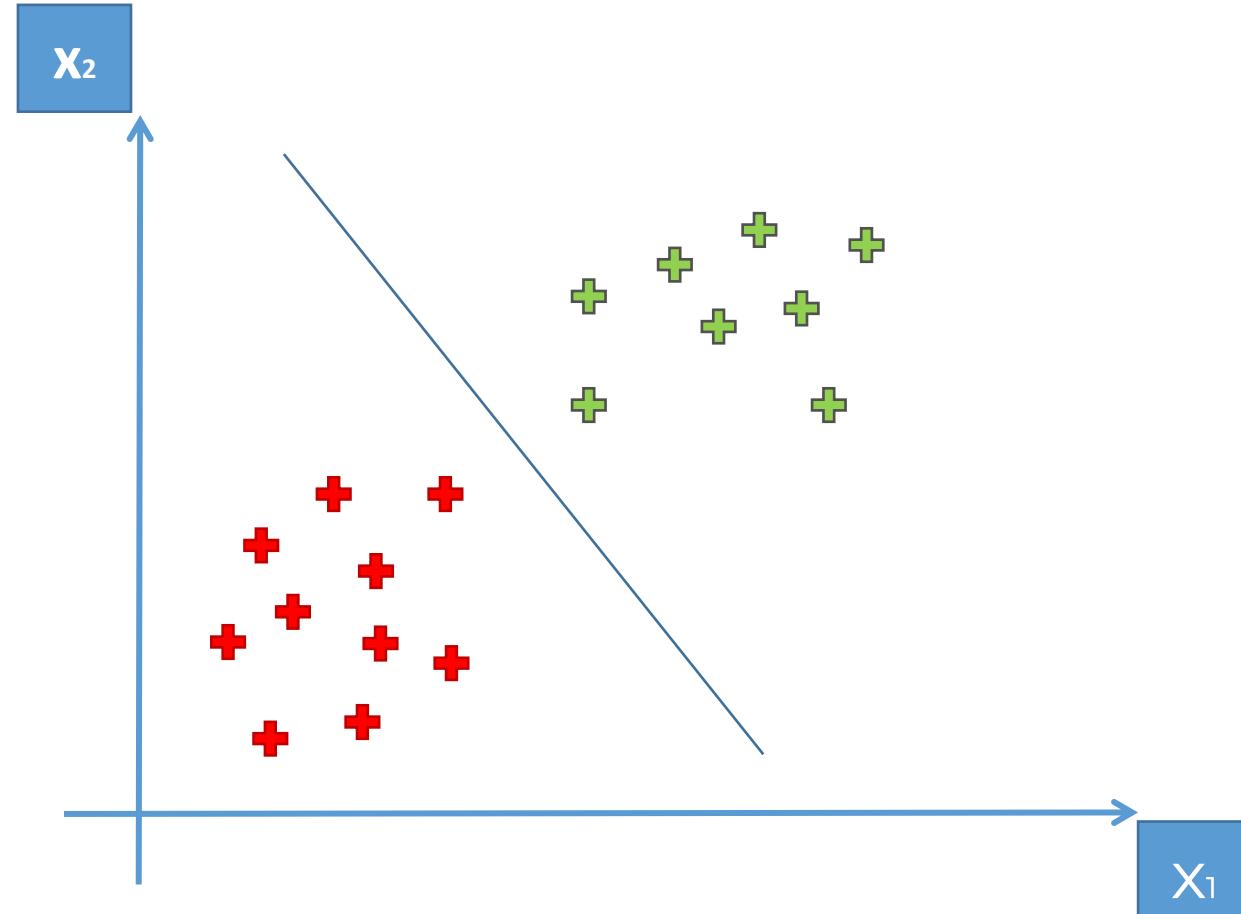
What's So Special About SVMs?

 X_2 Support
Vectors X_1 

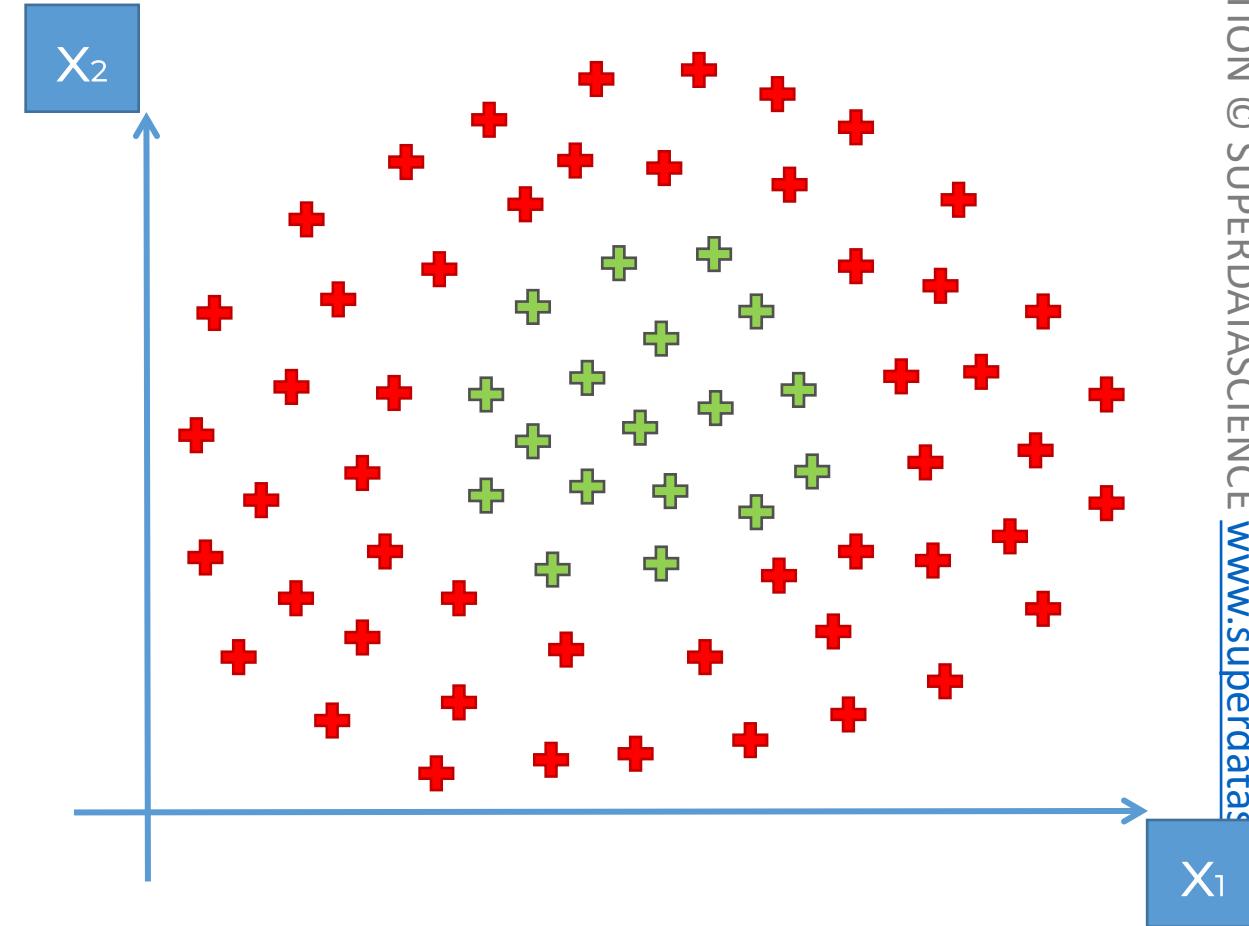
Kernel SVM Intuition

Linear Separability

Linearly Separable



Not Linearly Separable



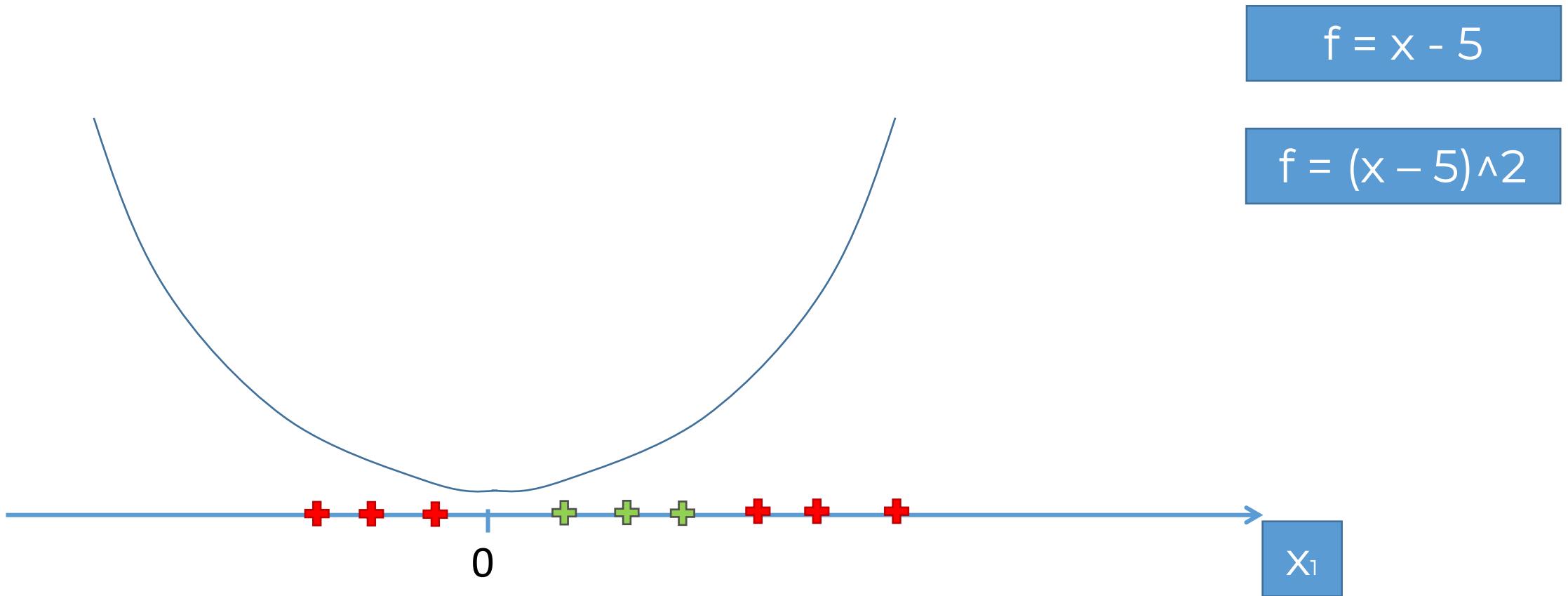
A Higher-Dimensional Space

Mapping to a Higher Dimension

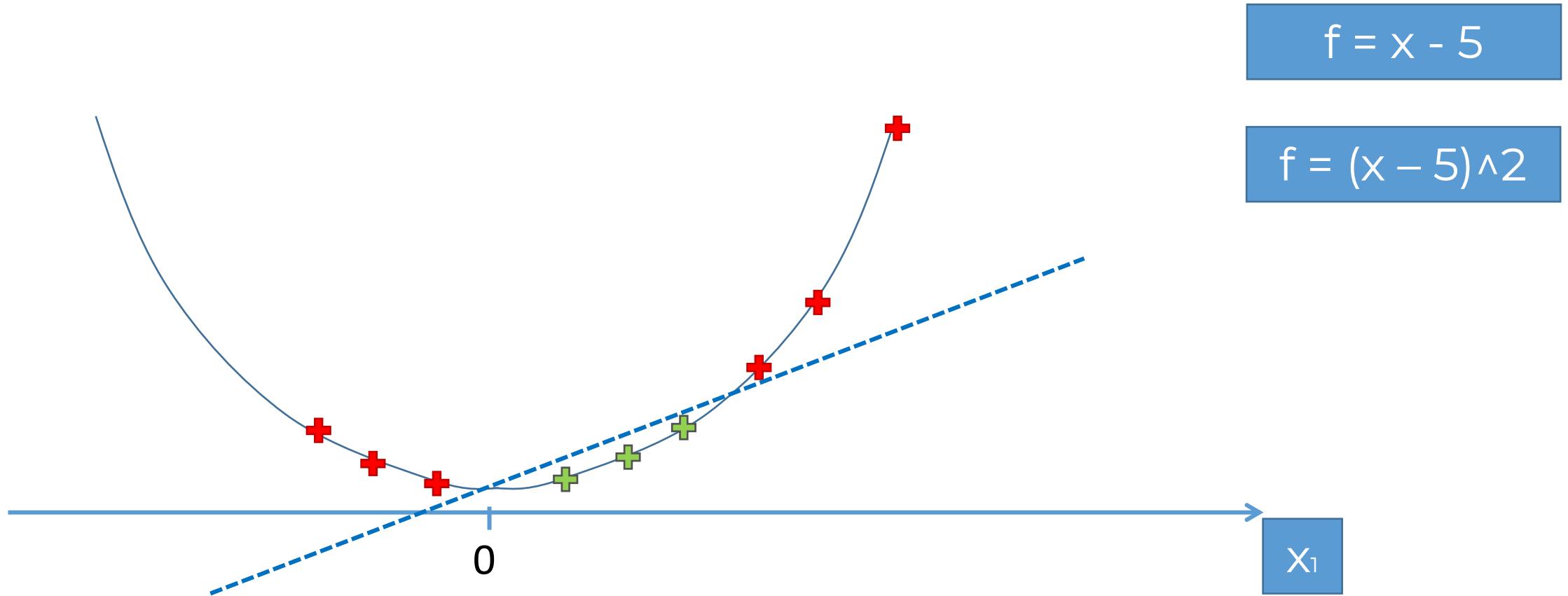
$$f = x - 5$$



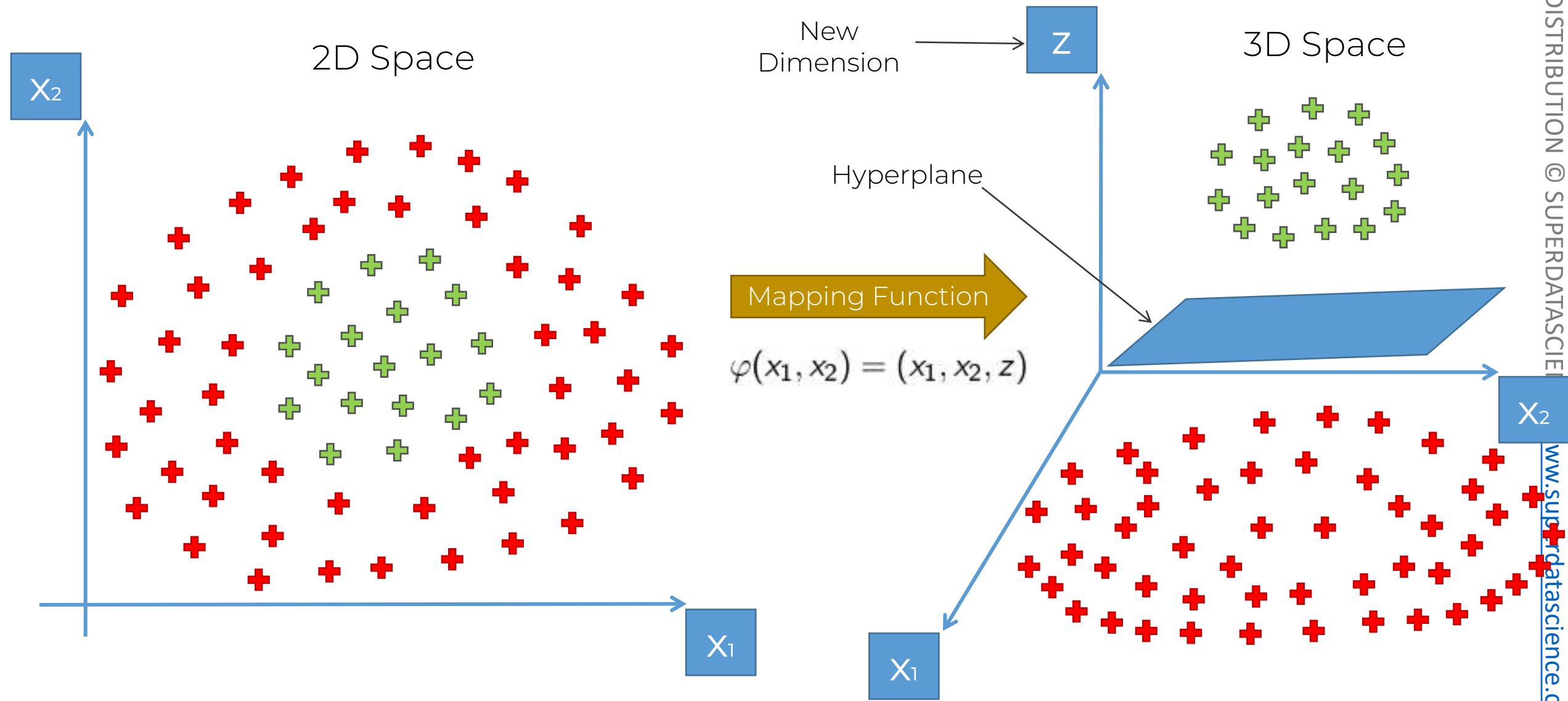
Mapping to a Higher Dimension



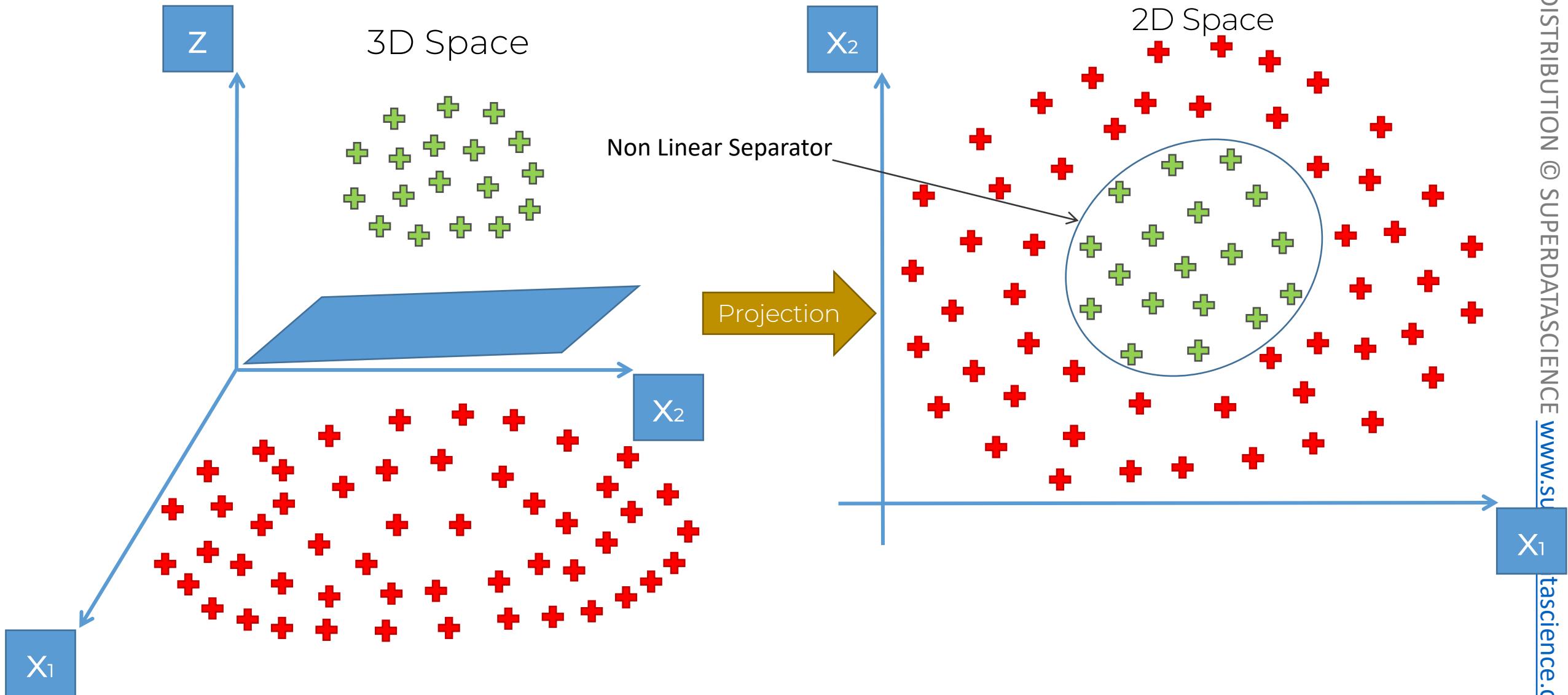
Mapping to a Higher Dimension



Mapping to a Higher Dimension



Projecting back to 2D Space



But there is a catch...

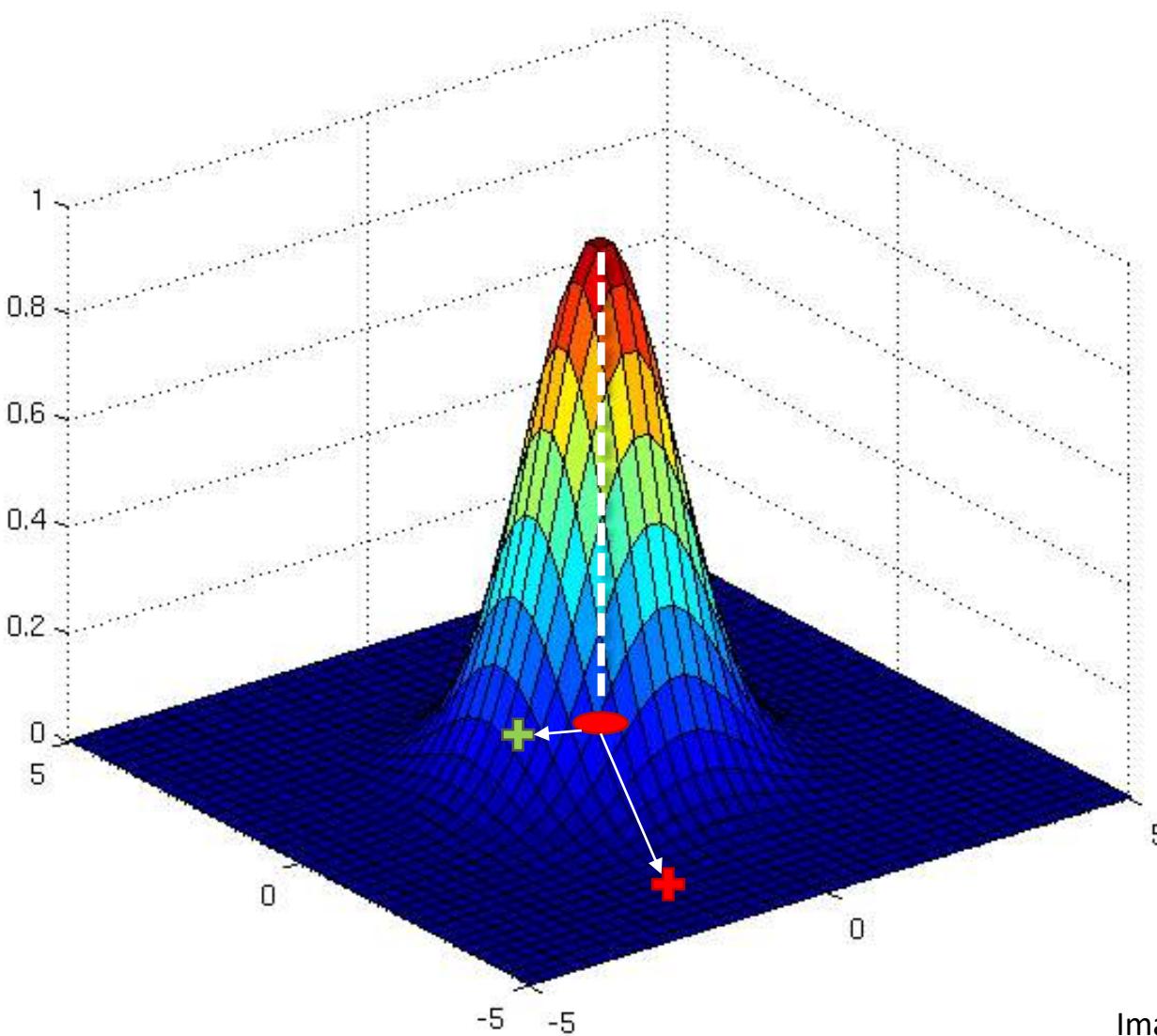
Mapping to a Higher Dimensional Space
can be highly compute-intensive

The Kernel Trick

The Gaussian RBF Kernel

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

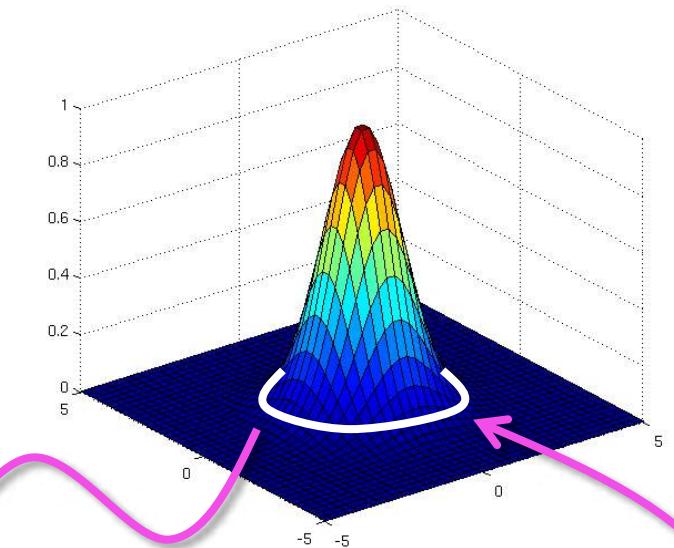
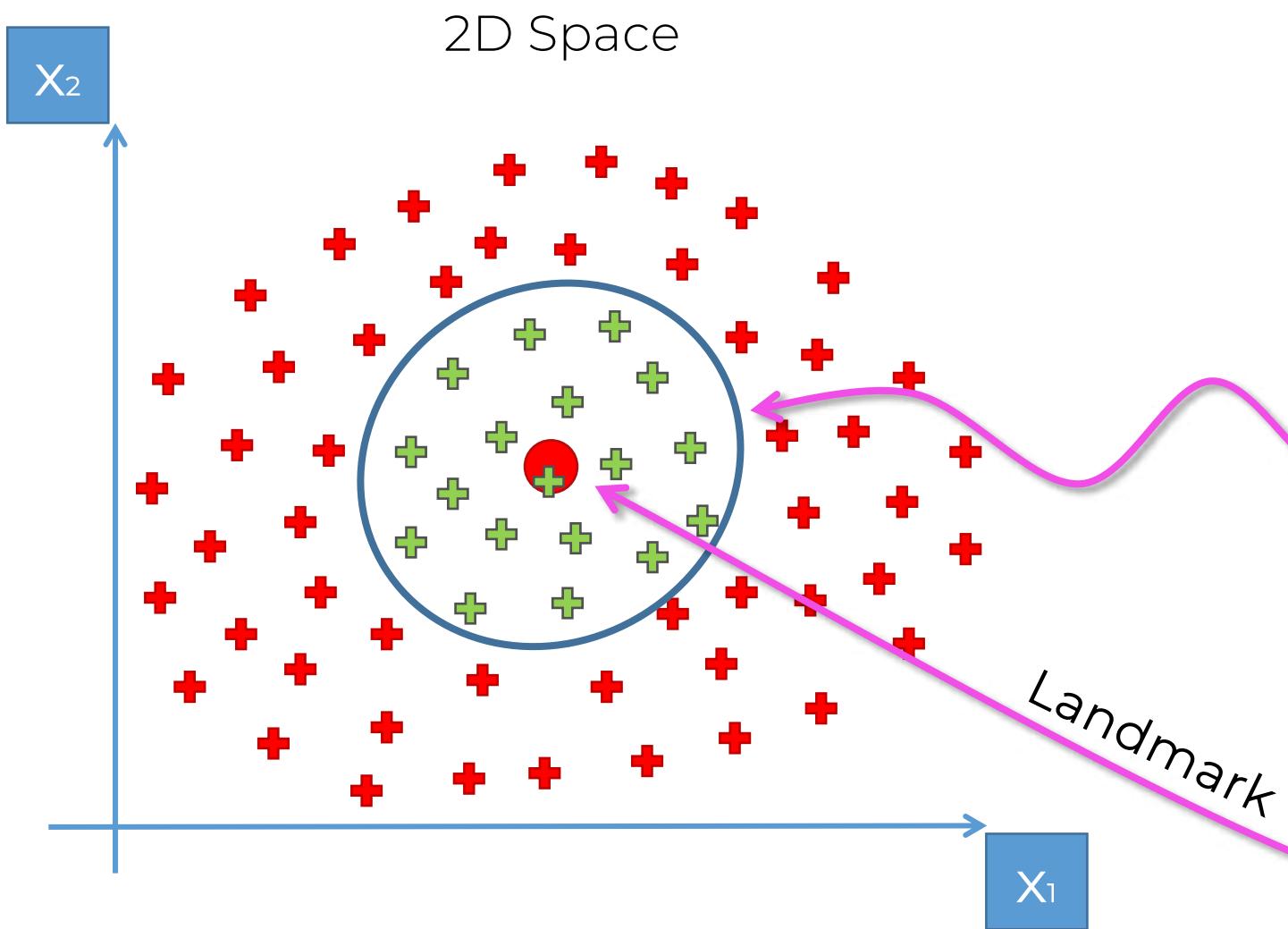
The Gaussian RBF Kernel



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

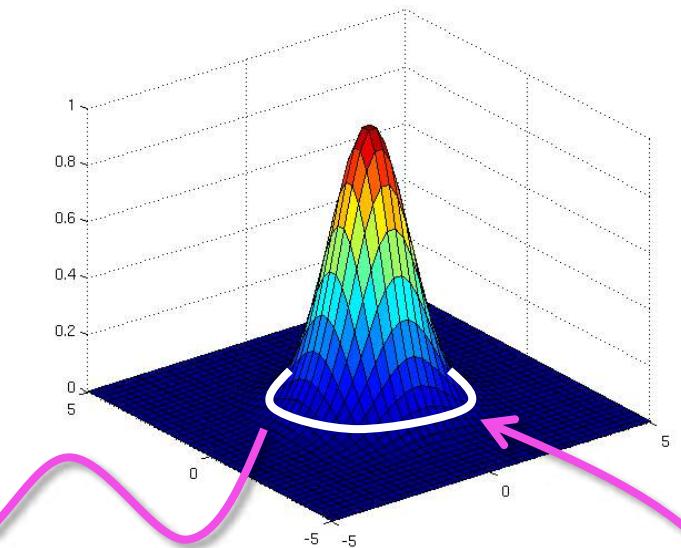
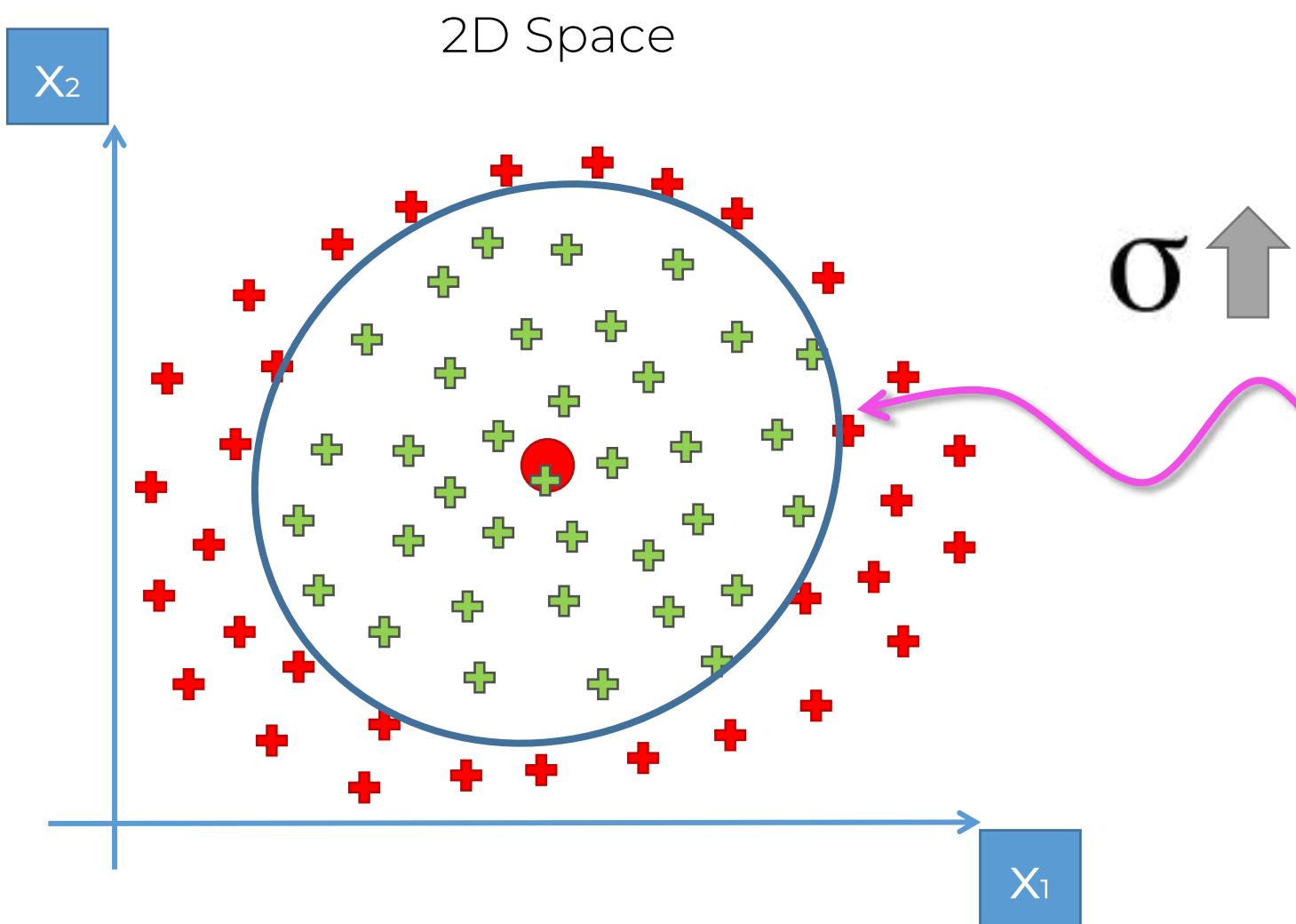
Image source: <http://www.cs.toronto.edu/~duvenaud/cookbook/index.html>

The Gaussian RBF Kernel



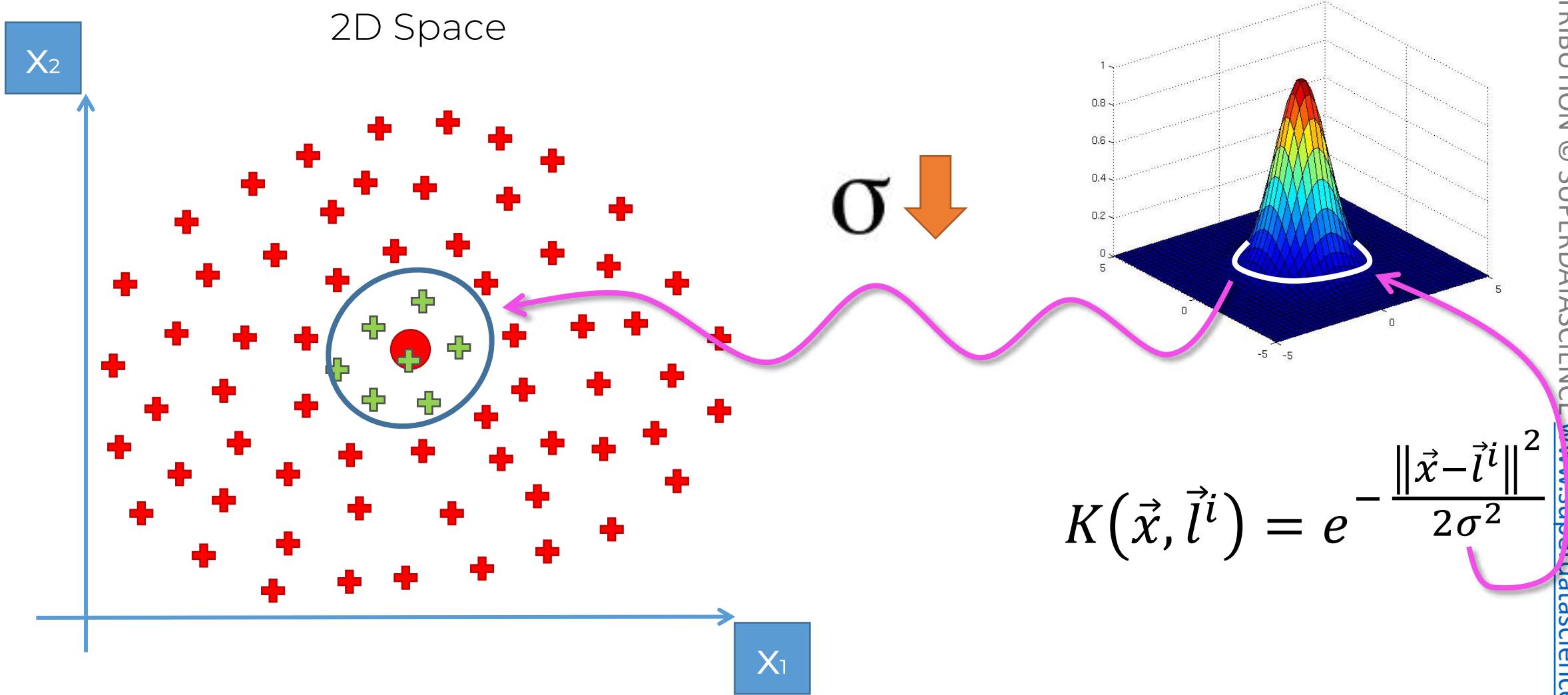
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel

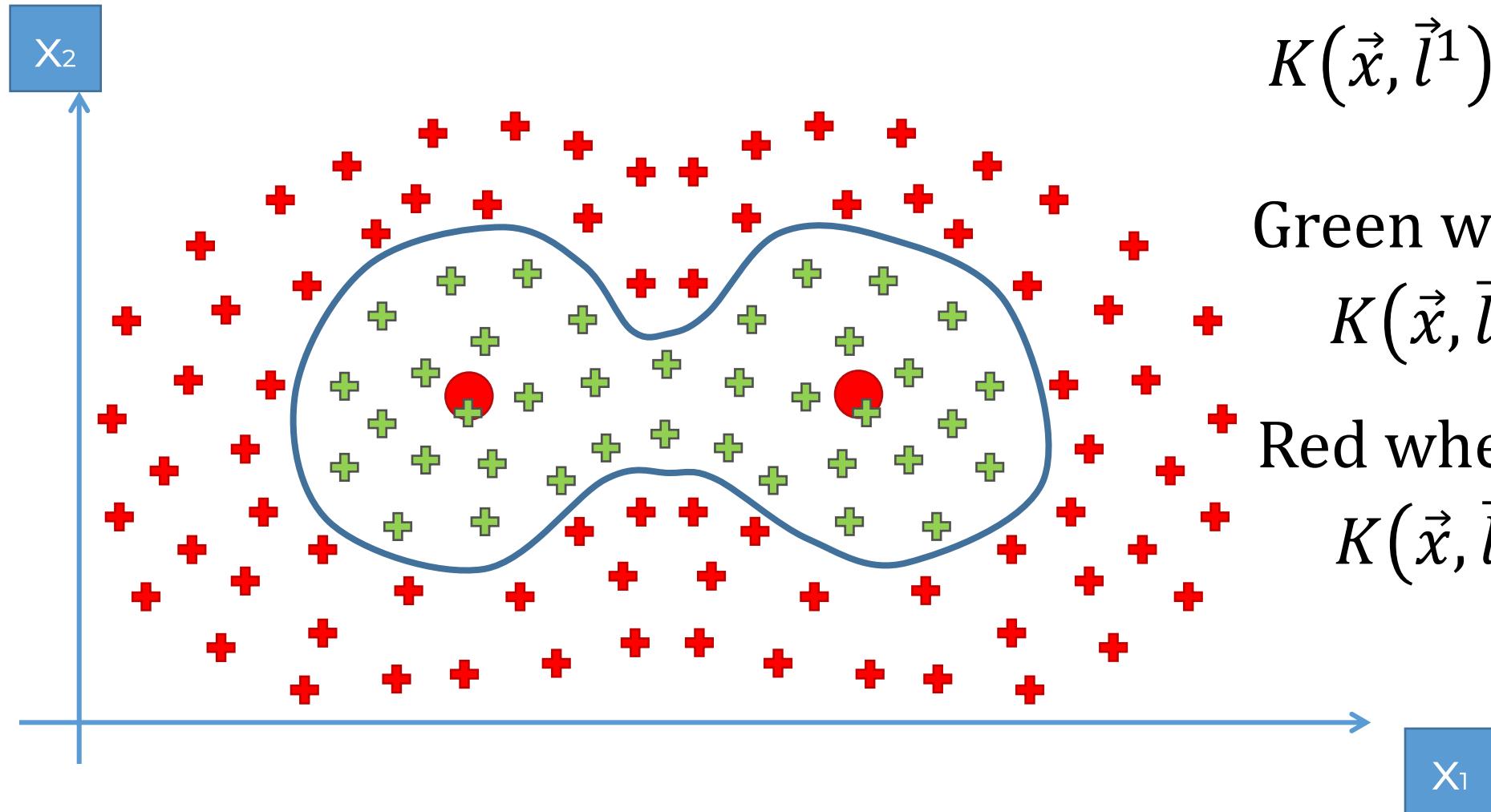


$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel

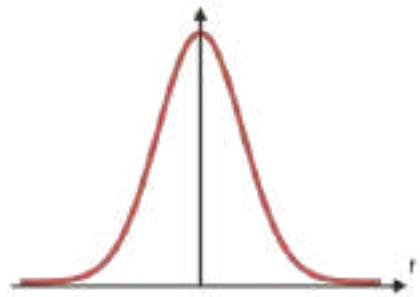


The Gaussian RBF Kernel



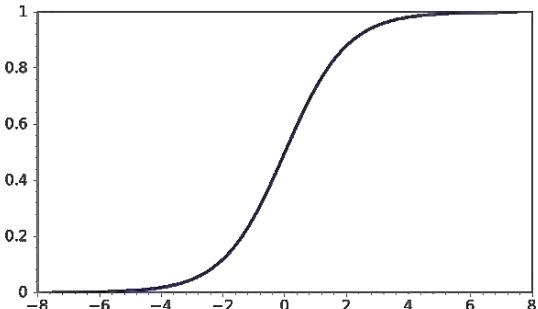
Types of Kernel Functions

Types of Kernel Functions



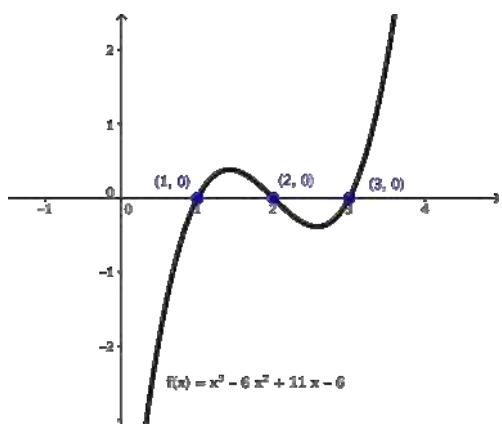
Gaussian RBF Kernel

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



Sigmoid Kernel

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



Polynomial Kernel

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

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

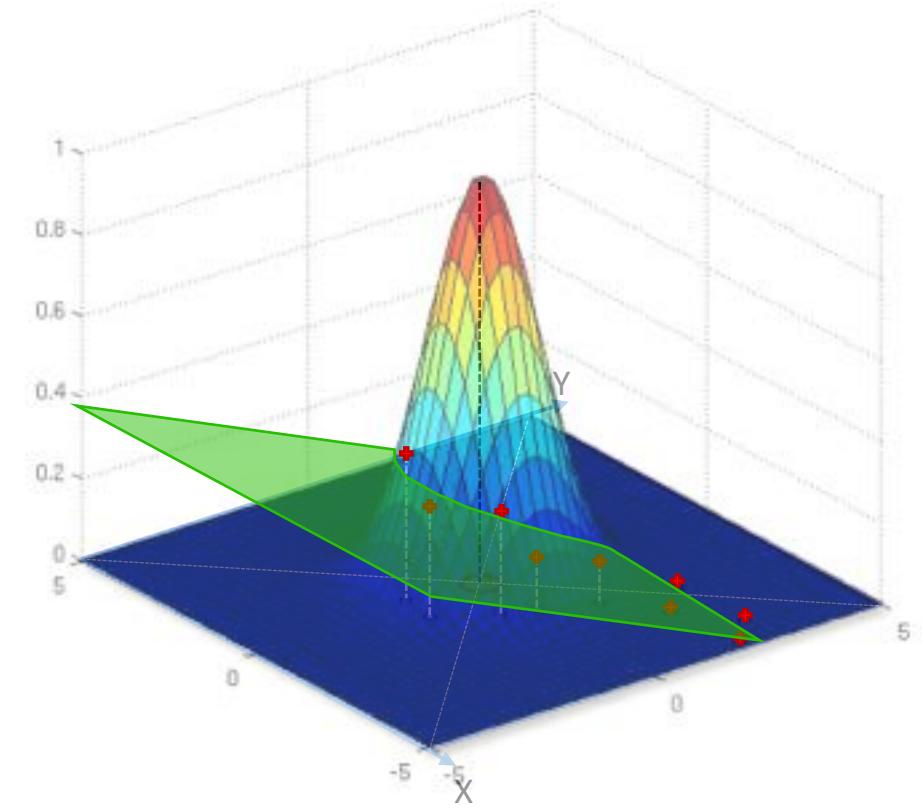
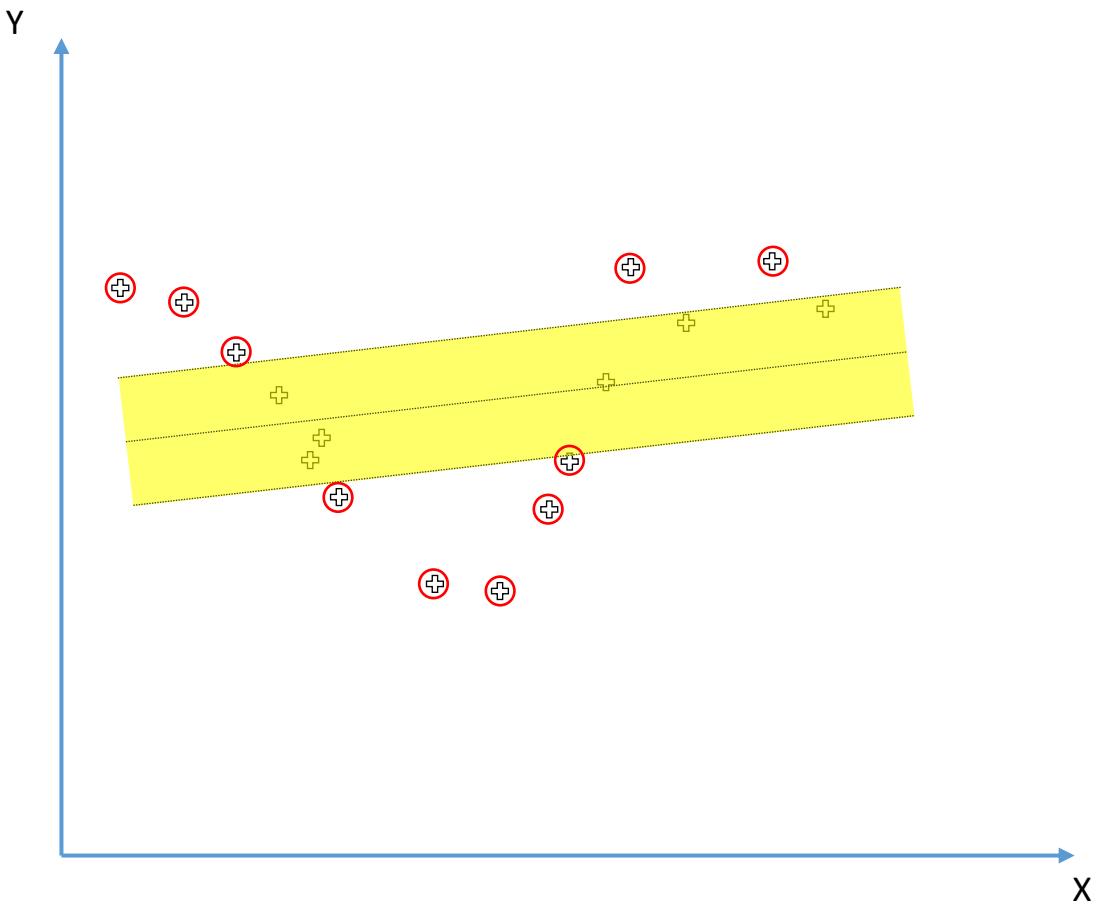


Image source: <http://www.cs.toronto.edu/~duvenaud/cookbook/index.html>

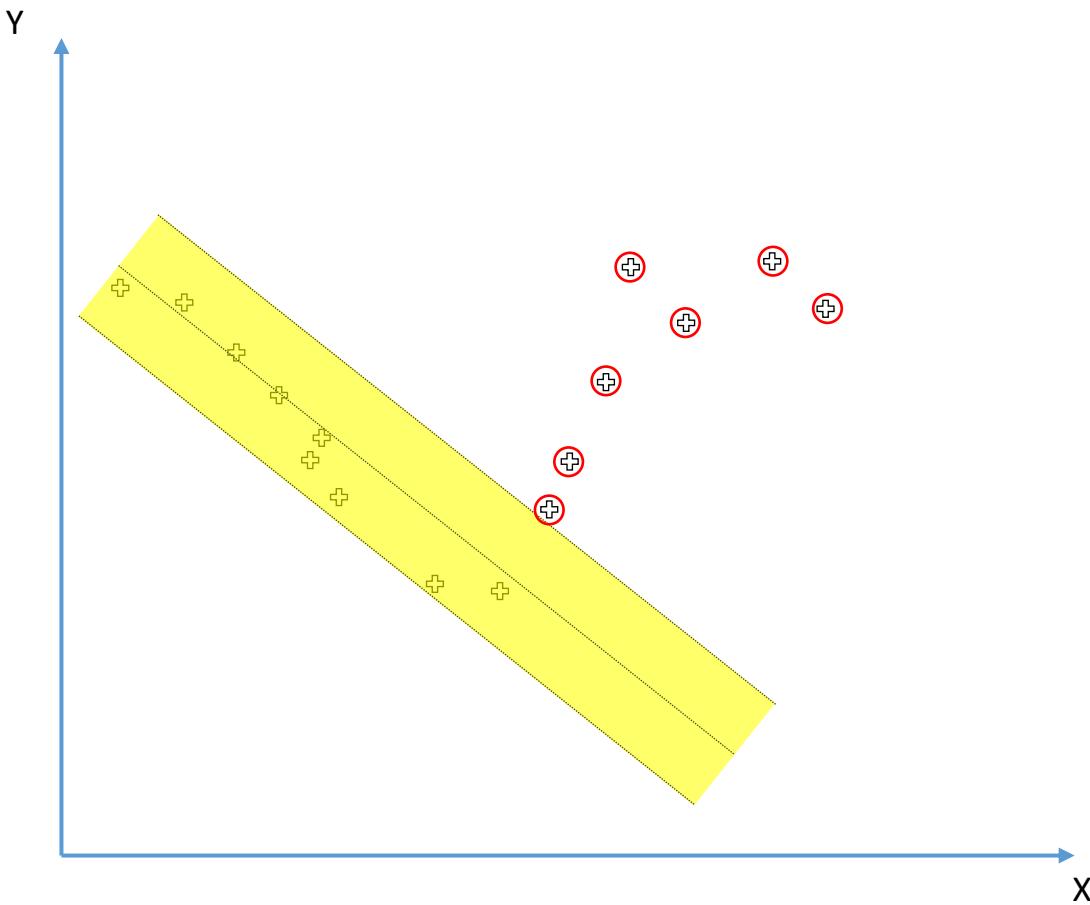
Non-Linear SVR



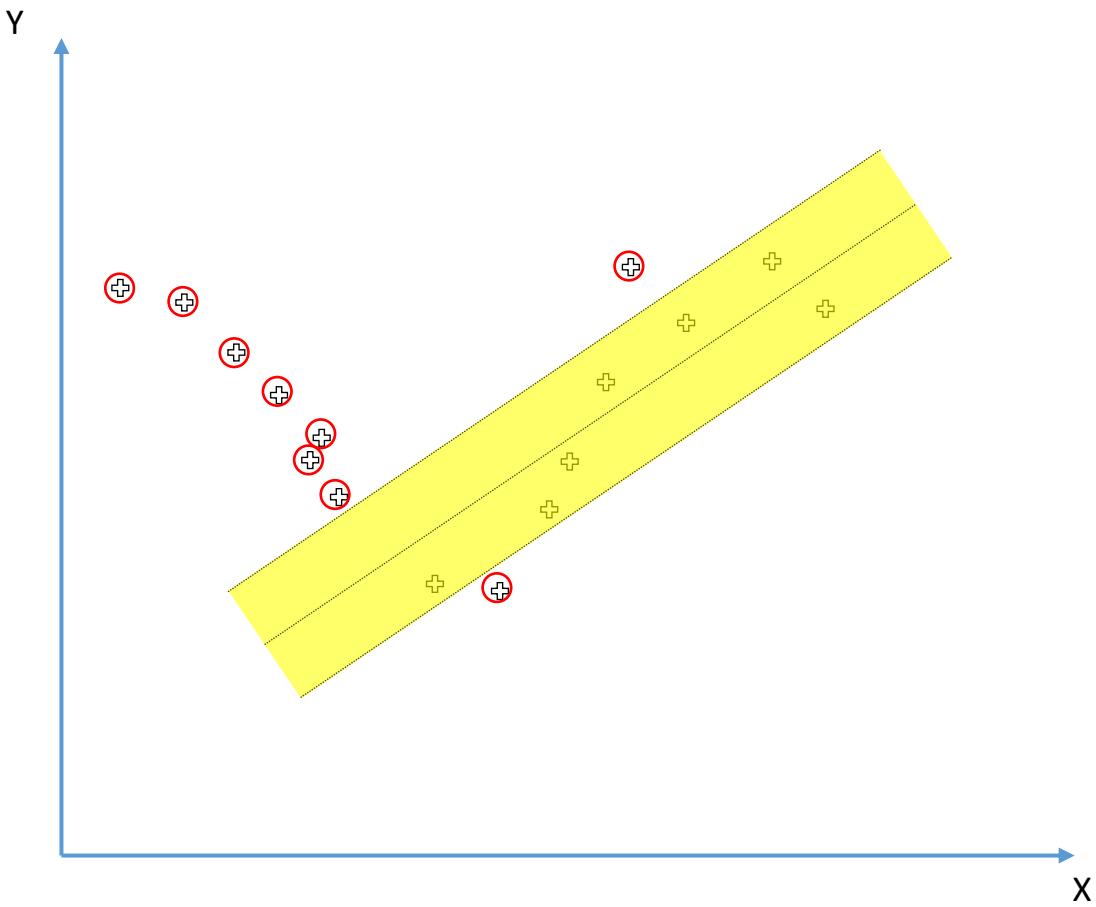
Non-Linear SVR



Non-Linear SVR



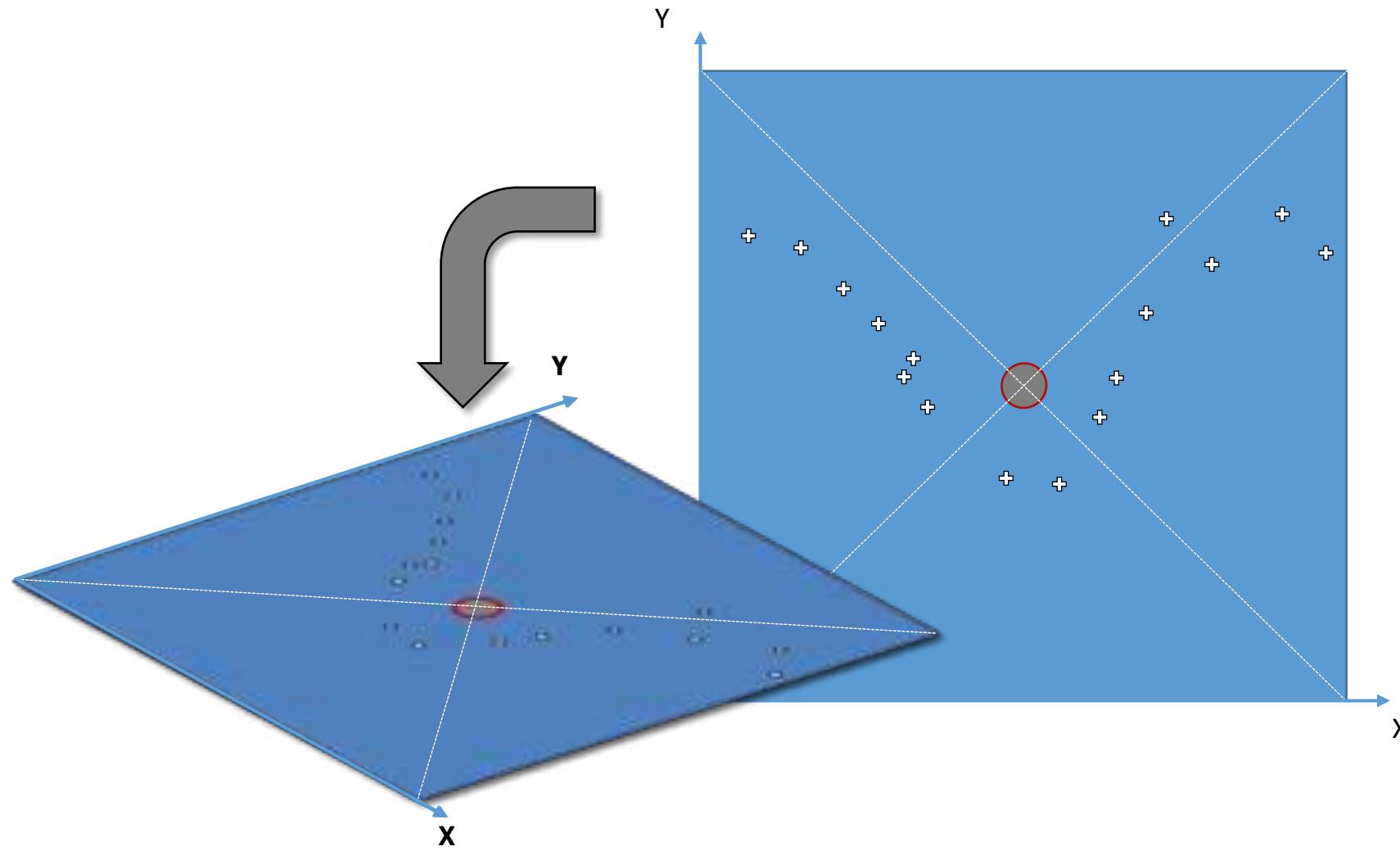
Non-Linear SVR



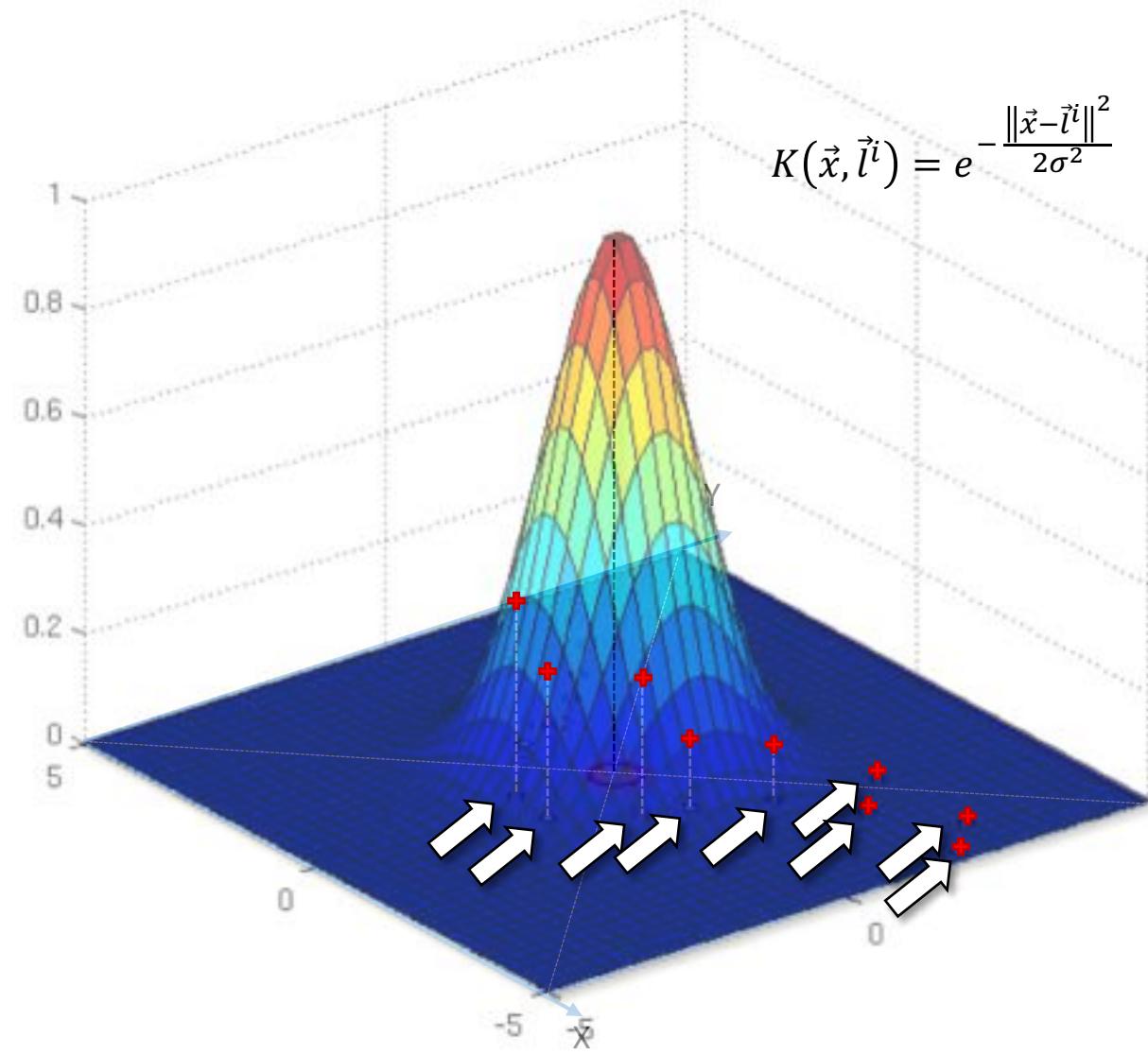
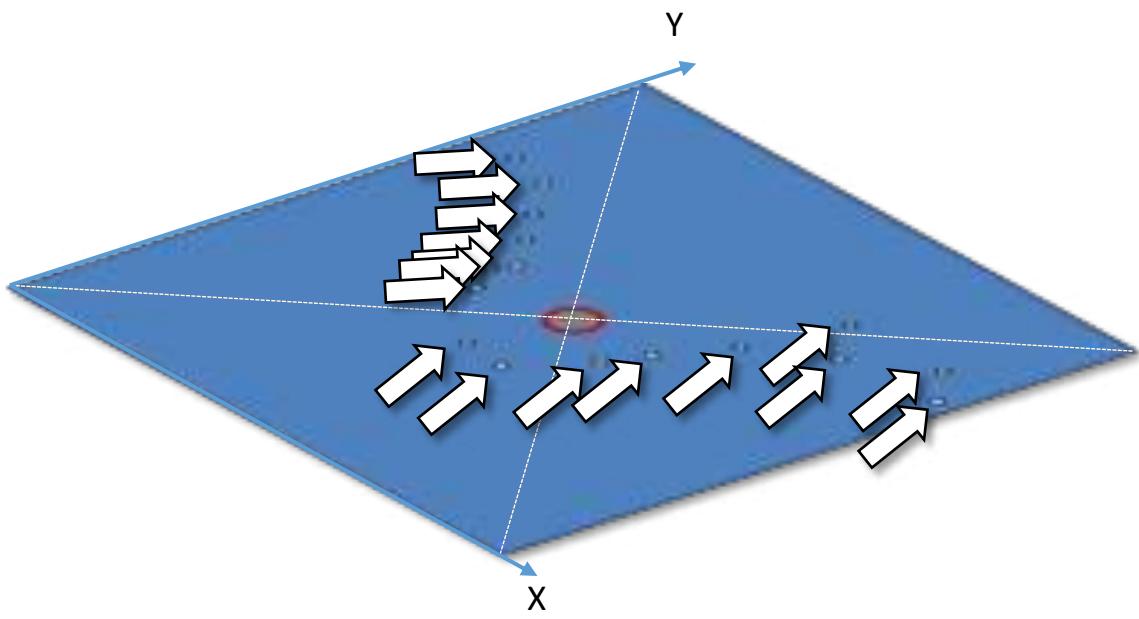
Non-Linear SVR



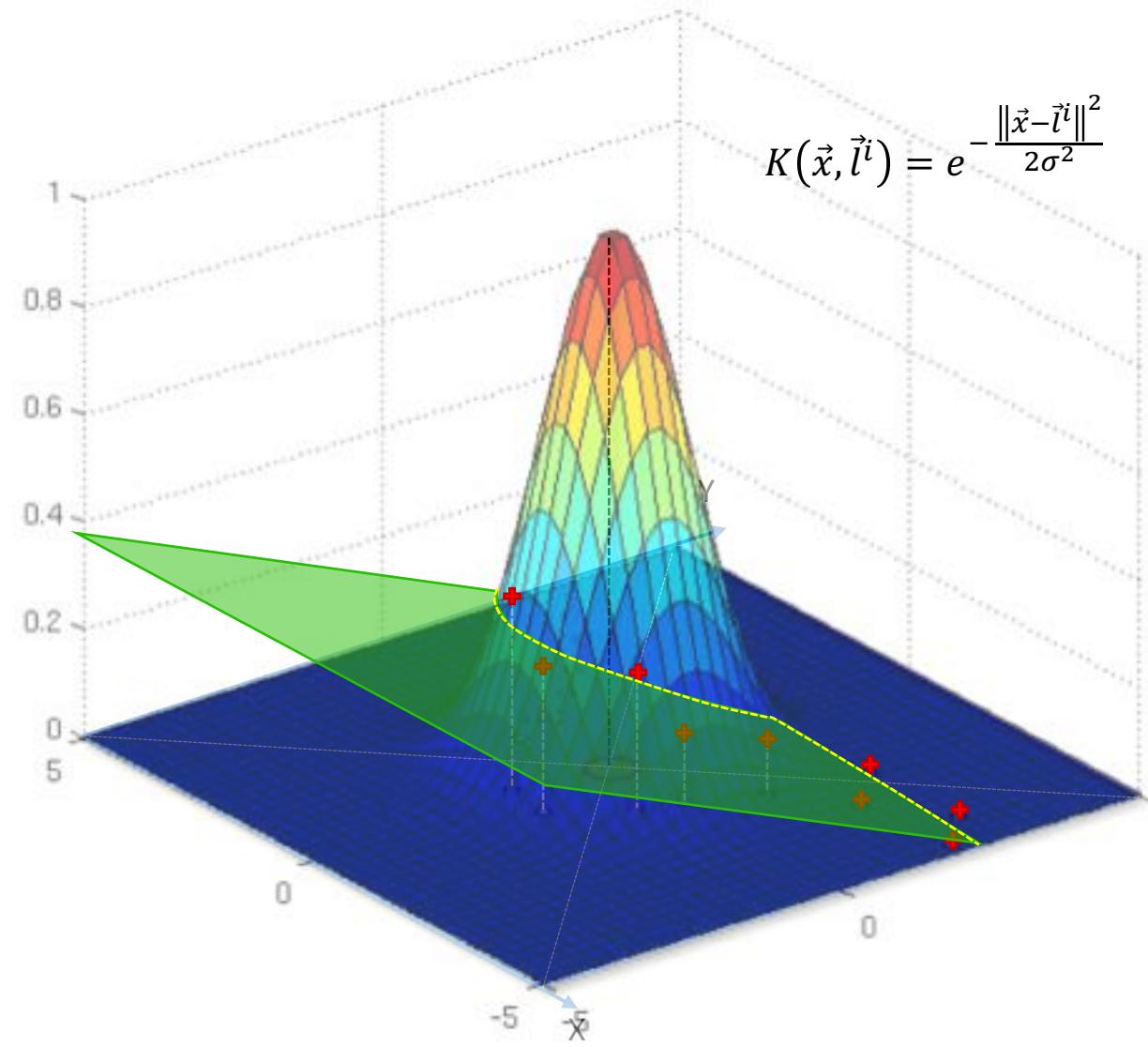
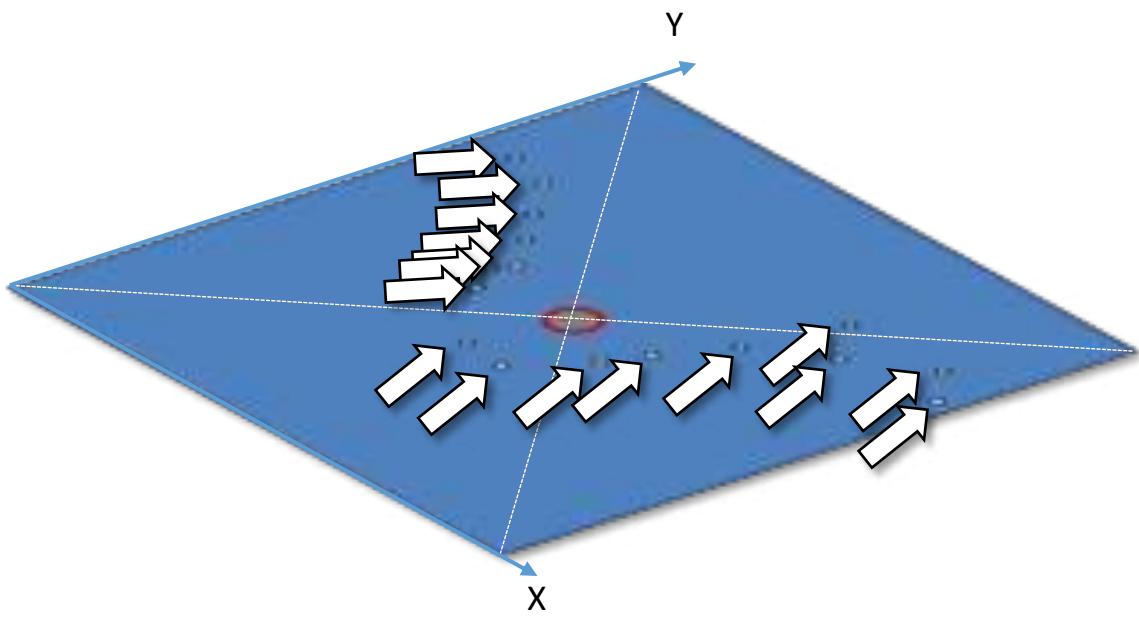
Non-Linear SVR



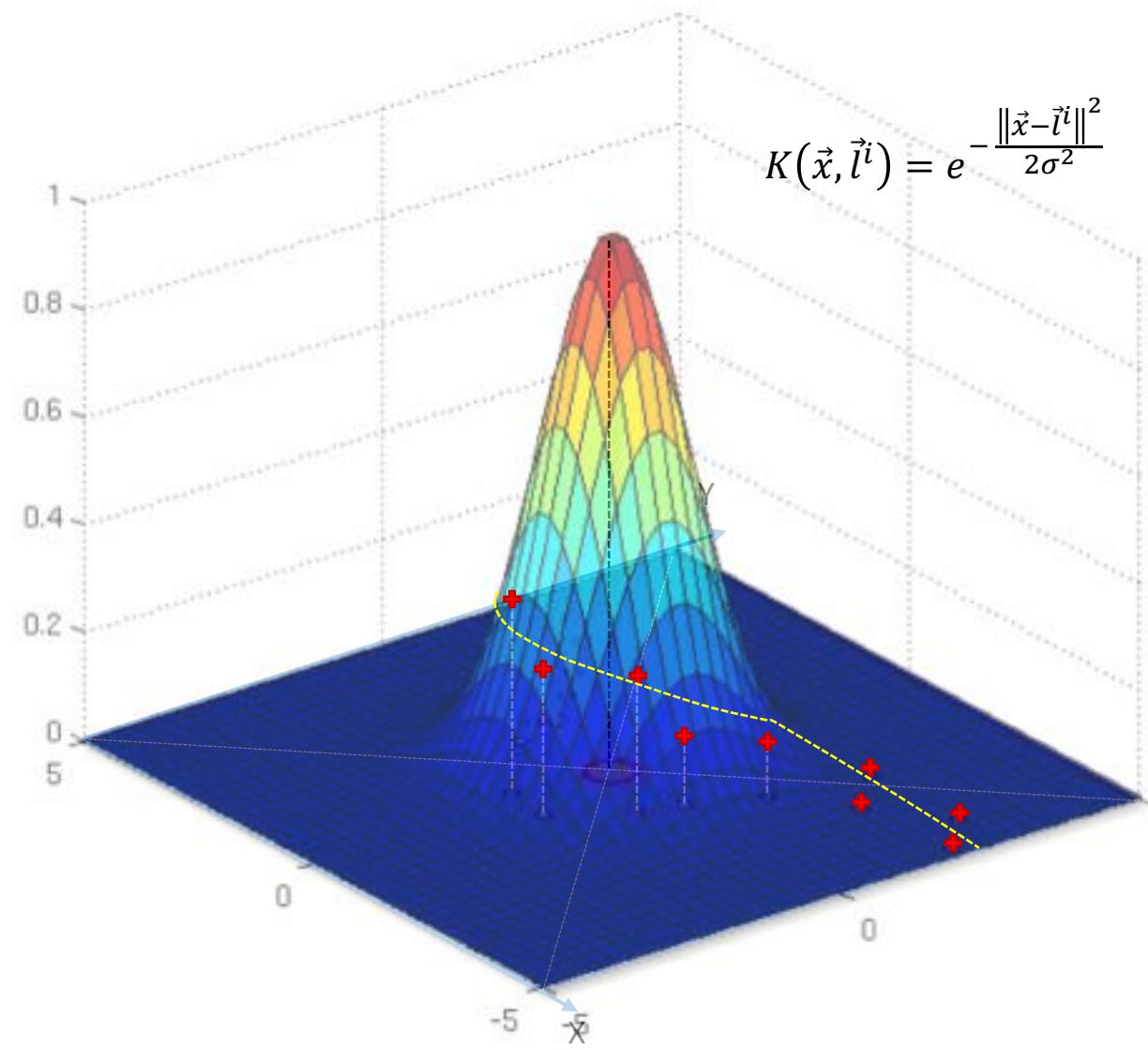
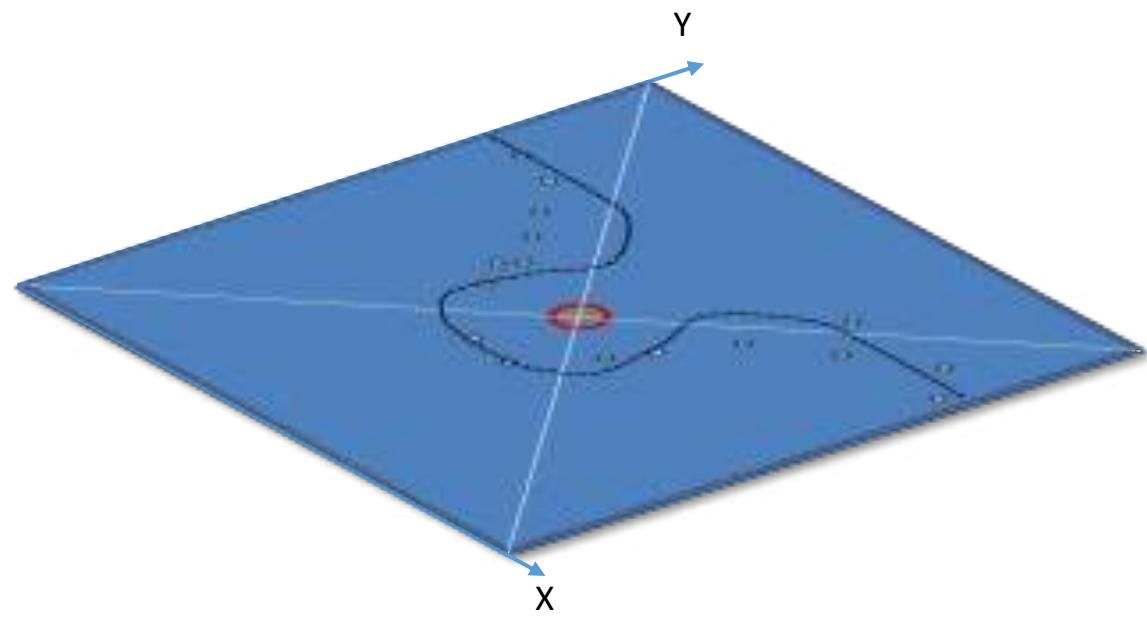
Non-Linear SVR



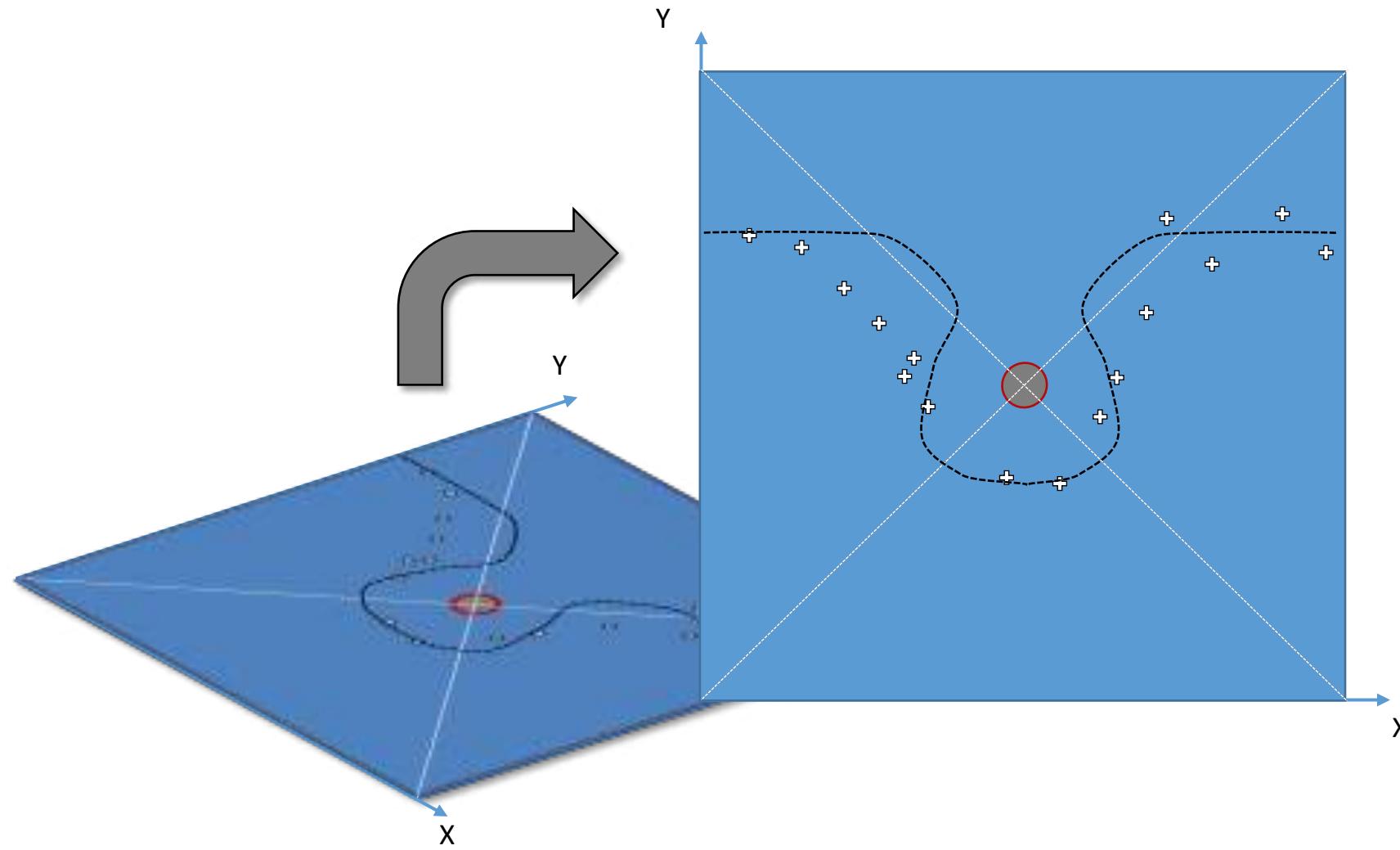
Non-Linear SVR



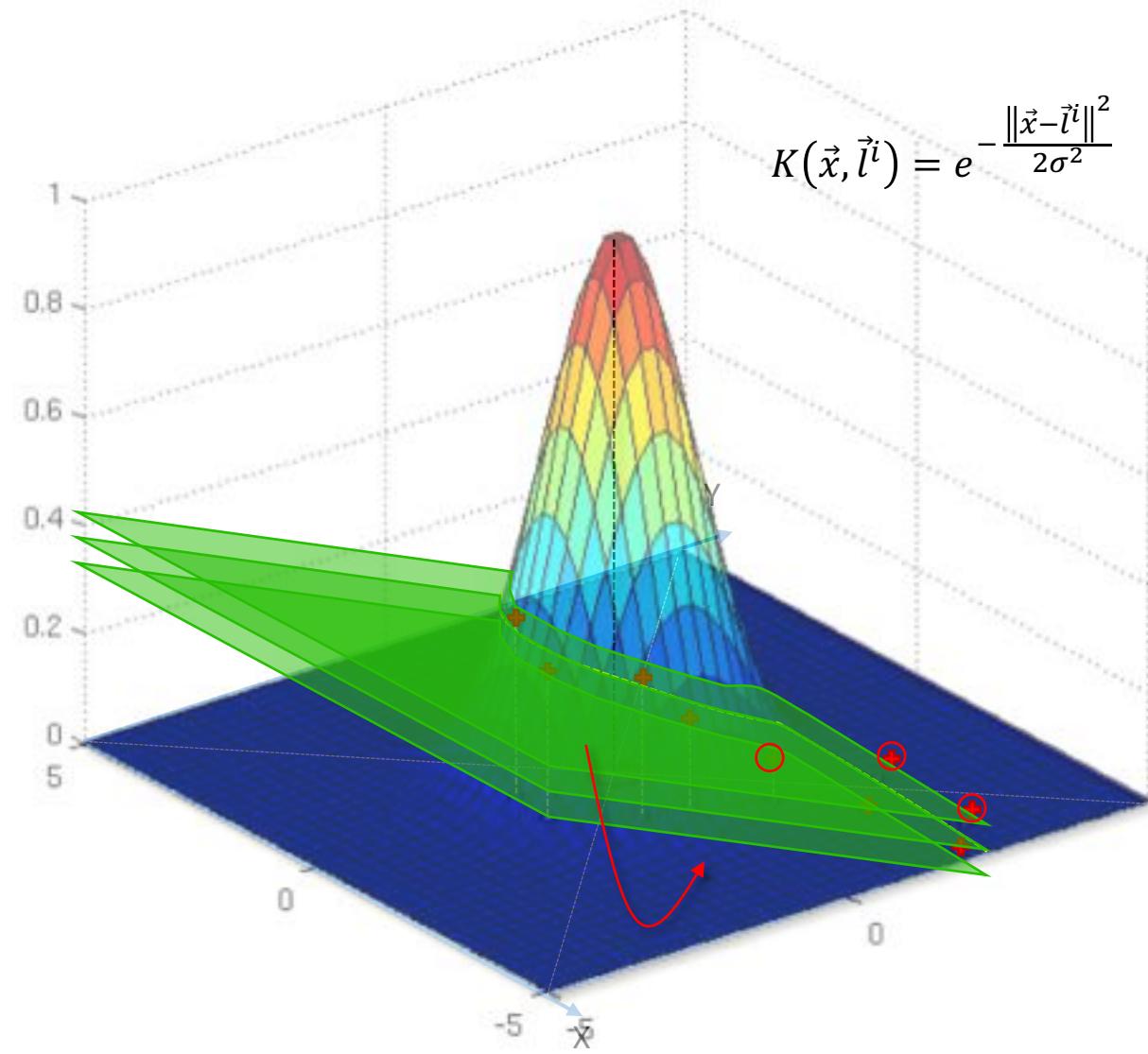
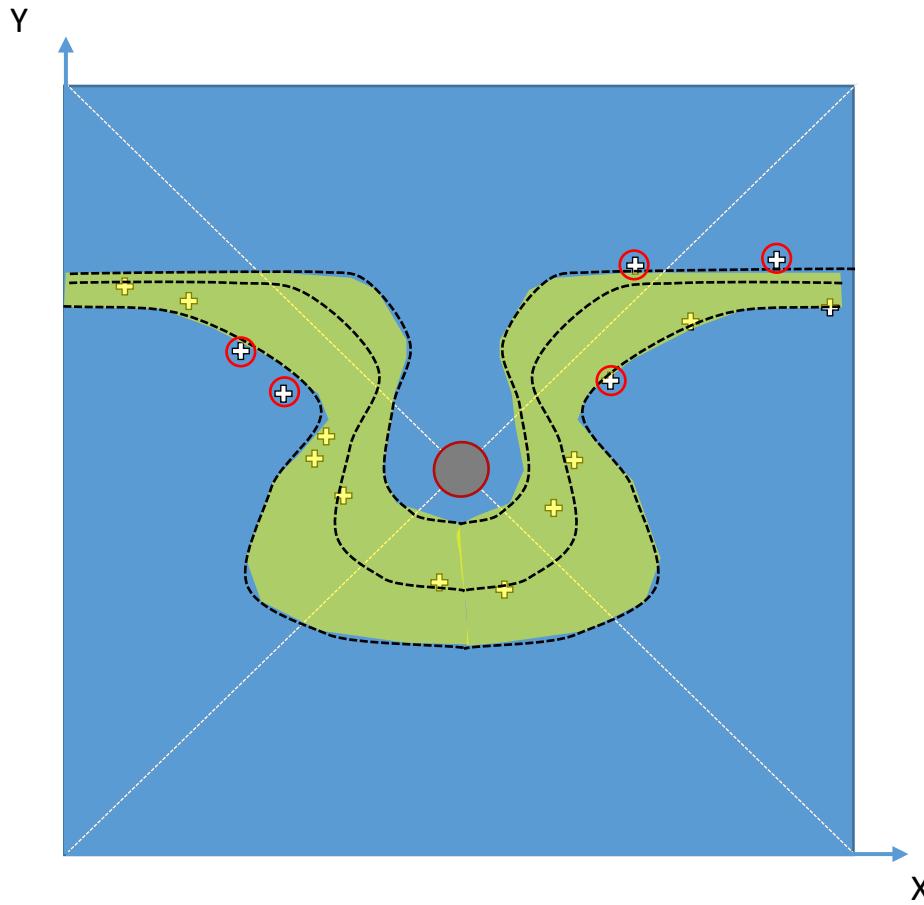
Non-Linear SVR



Non-Linear SVR



Non-Linear SVR



Bayes' Theorem

Bayes Theorem



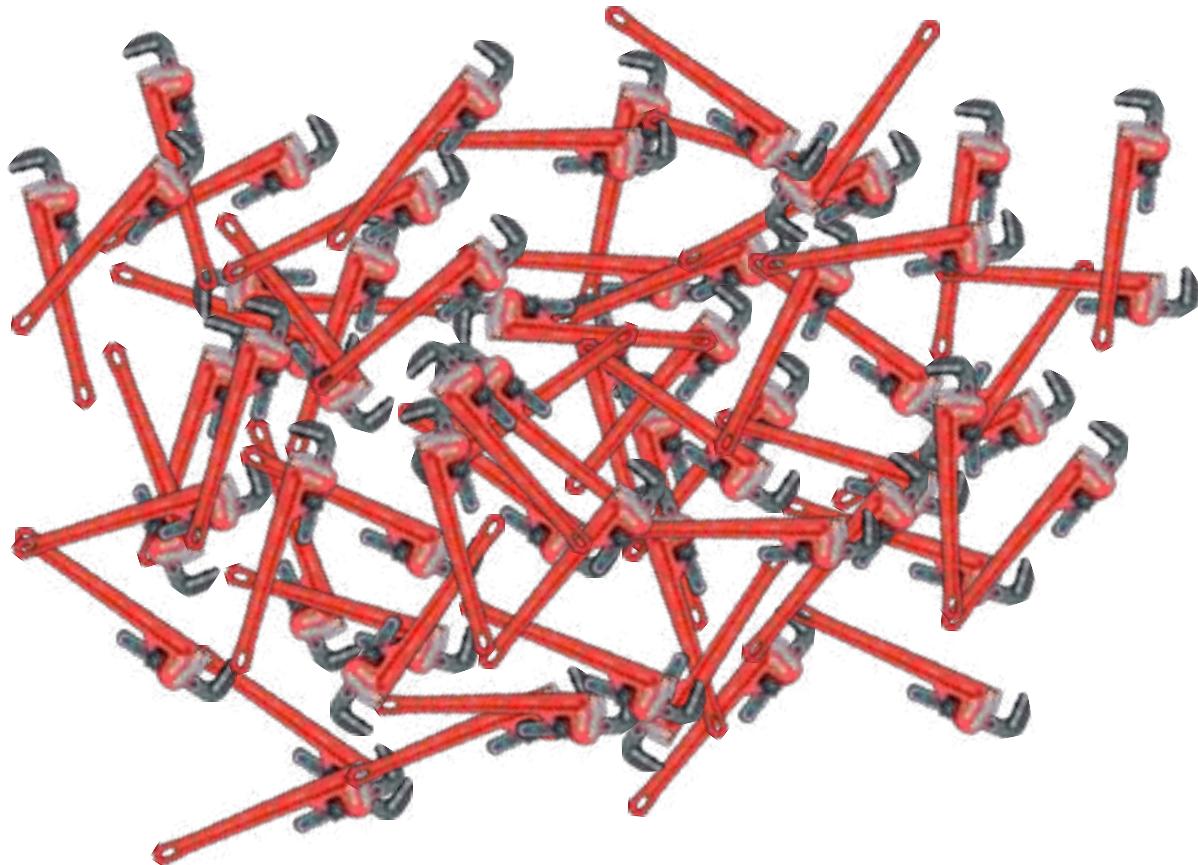
m1 m1



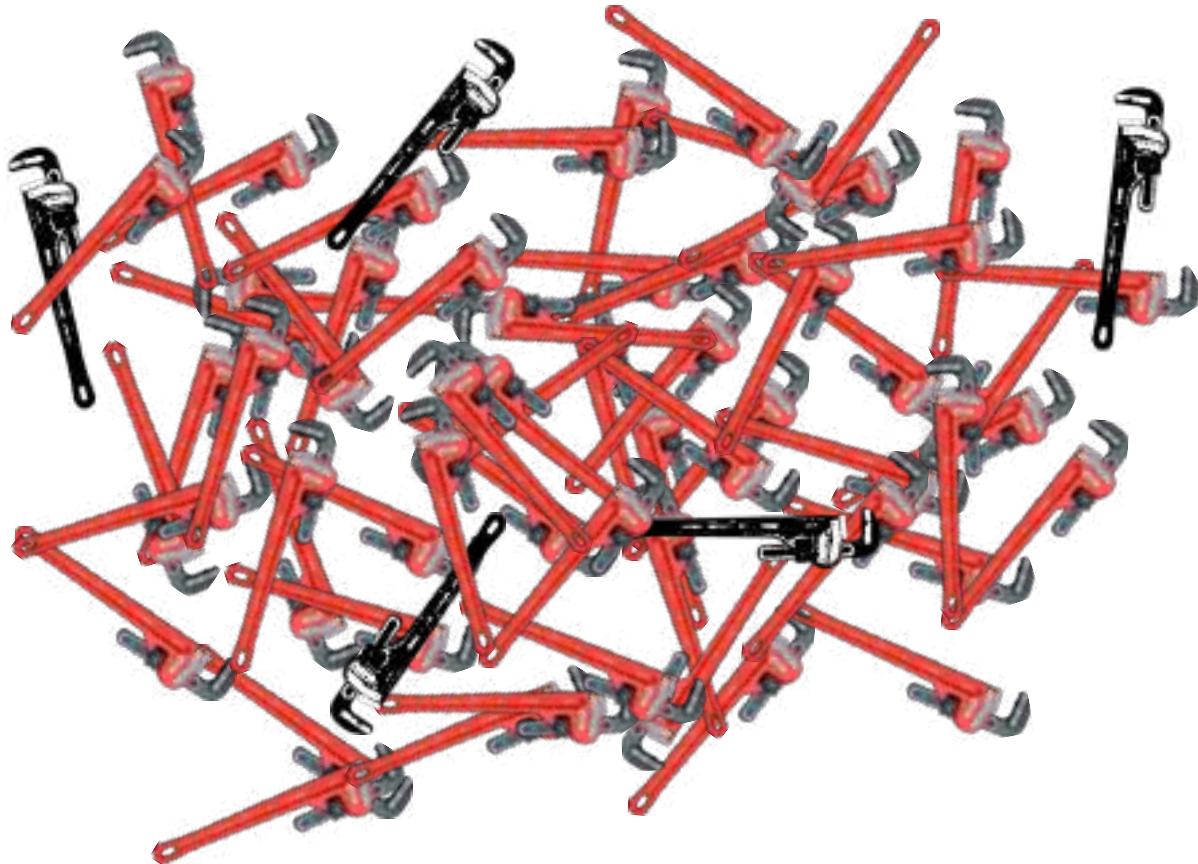
m2 m2



Bayes Theorem



Bayes Theorem



Bayes Theorem

What's the probability?



m2



Bayes Theorem

What's the probability?



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

Mach2: 20 wrenches / hr

$$\rightarrow P(\text{Mach1}) = 30/50 = 0.6$$

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

Out of all produced parts:

We can SEE that 1% are defective

$$\rightarrow P(\text{Defect}) = 1\%$$

Out of all defective parts:

We can SEE that 50% came from mach1

And 50% came from mach2

$$\rightarrow P(\text{Mach1} | \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

Question:

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

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

Bayes Theorem

Mach1: 30 wrenches / hr

Mach2: 20 wrenches / hr

$$\cancel{\rightarrow P(\text{Mach1}) = 30/50 = 0.6}$$

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

Out of all produced parts:

We can SEE that 1% are defective

$$\rightarrow P(\text{Defect}) = 1\%$$

Out of all defective parts:

We can SEE that 50% came from mach1

And 50% came from mach2

$$\cancel{\rightarrow P(\text{Mach1} | \text{Defect}) = 50\%}$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

Question:

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

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

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 = ?

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

$$\rightarrow P(\text{Defect}) = 1\%$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

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 = ?

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

$$\rightarrow P(\text{Defect}) = 1\%$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

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

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 = ?

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

$$\rightarrow P(\text{Defect}) = 1\%$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

$$P(\text{Defect} | \text{Mach2}) = \frac{0.5 * 0.01}{0.4}$$

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 = ?

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

$$\rightarrow P(\text{Defect}) = 1\%$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

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

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:

$$P(\text{Defect} \mid \text{Mach1}) = ?$$

Bayes Theorem

Mach1: 30 wrenches / hr

Mach2: 20 wrenches / hr

$$\rightarrow P(\text{Mach1}) = 30/50 = 0.6$$

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

Out of all produced parts:

We can SEE that 1% are defective

$$\rightarrow P(\text{Defect}) = 1\%$$

Out of all defective parts:

We can SEE that 50% came from mach1

And 50% came from mach2

$$\rightarrow P(\text{Mach1} | \text{Defect}) = 50\%$$

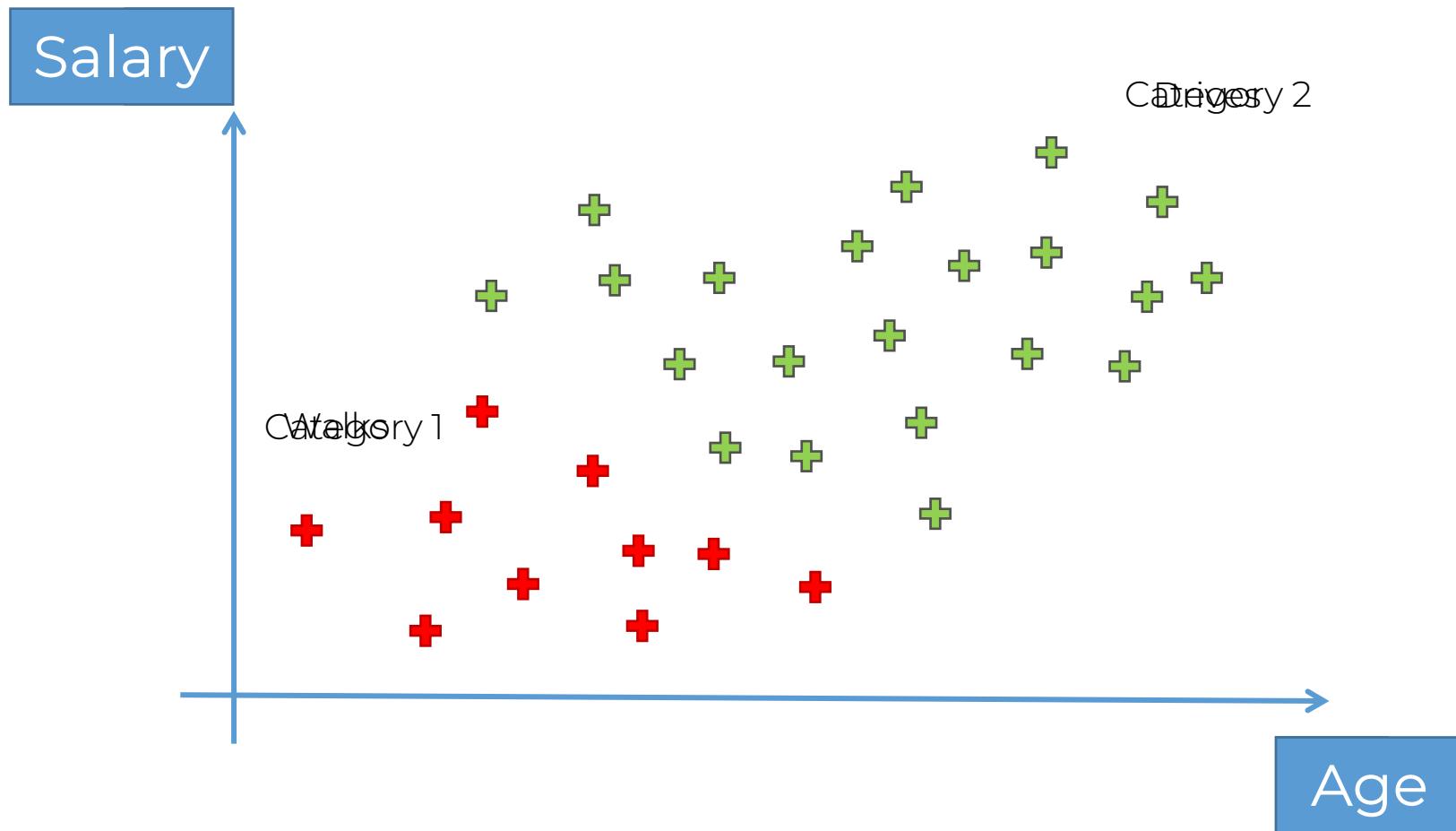
$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

Naïve Bayes Classifier Intuition

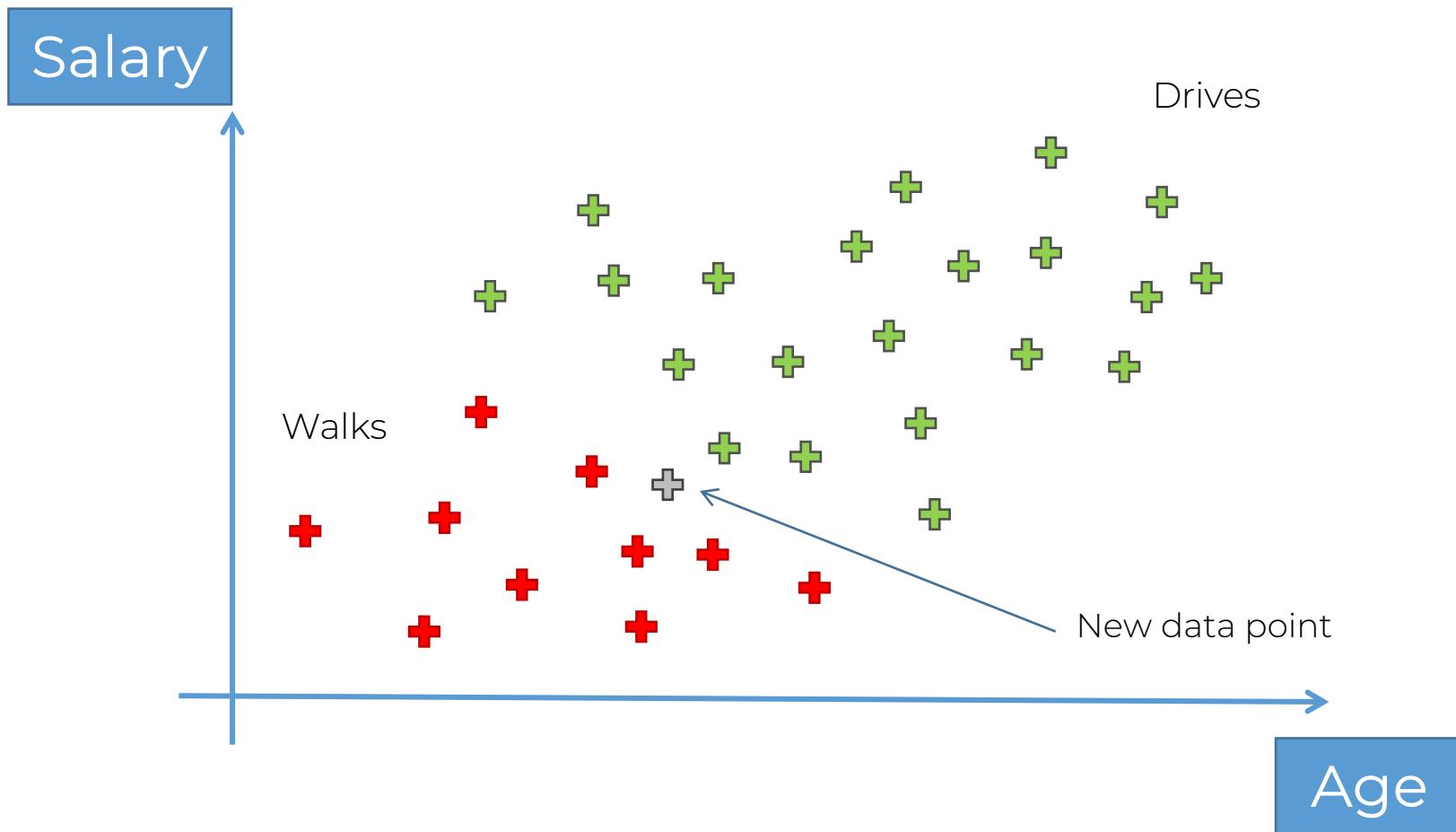
Naïve Bayes

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

Naïve Bayes



Naïve Bayes



Naïve Bayes

Plan of Attack

Naïve Bayes

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

Step 1

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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

#2 Marginal Likelihood

Step 2

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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

#2 Marginal Likelihood

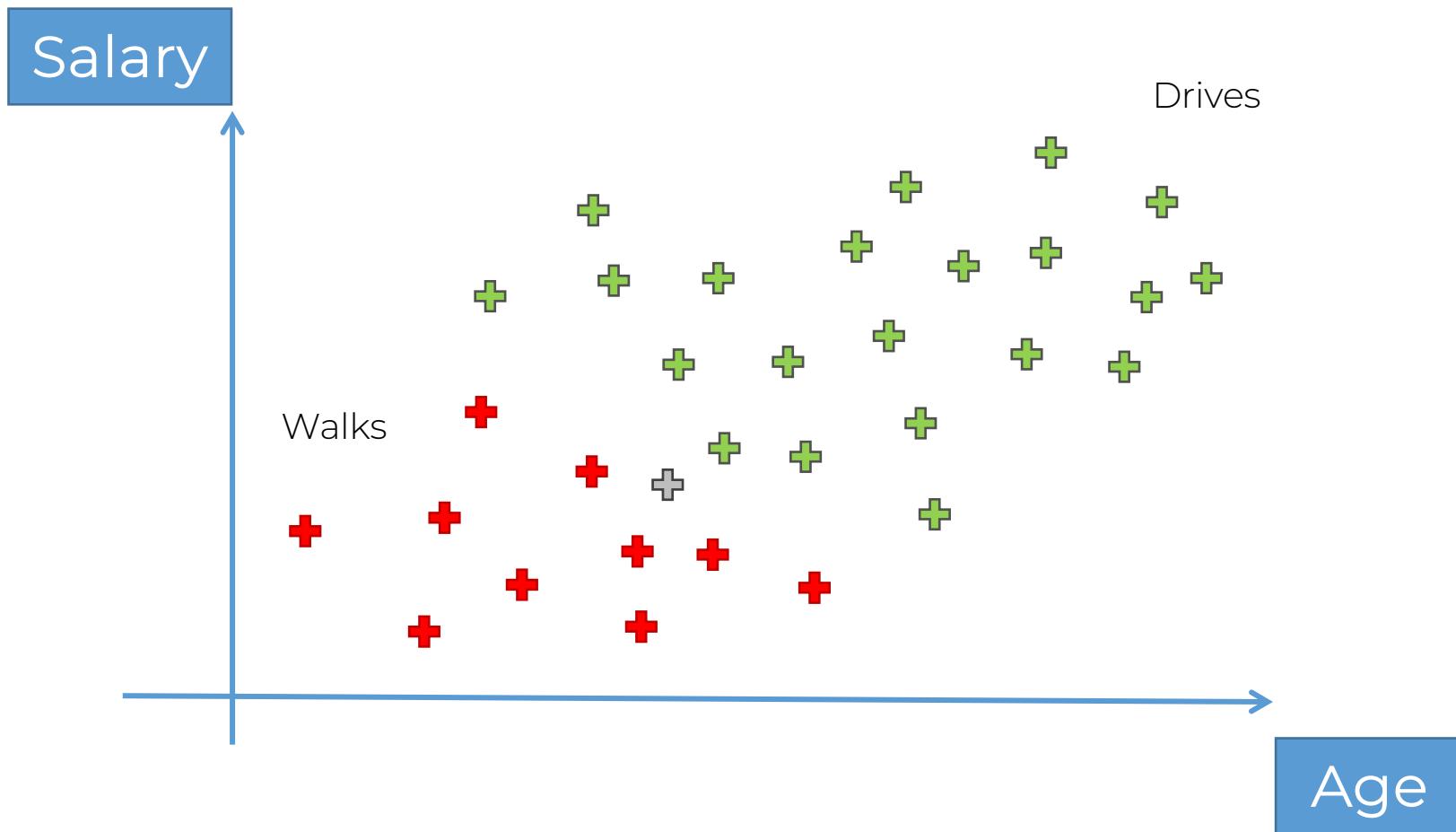
Step 3

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

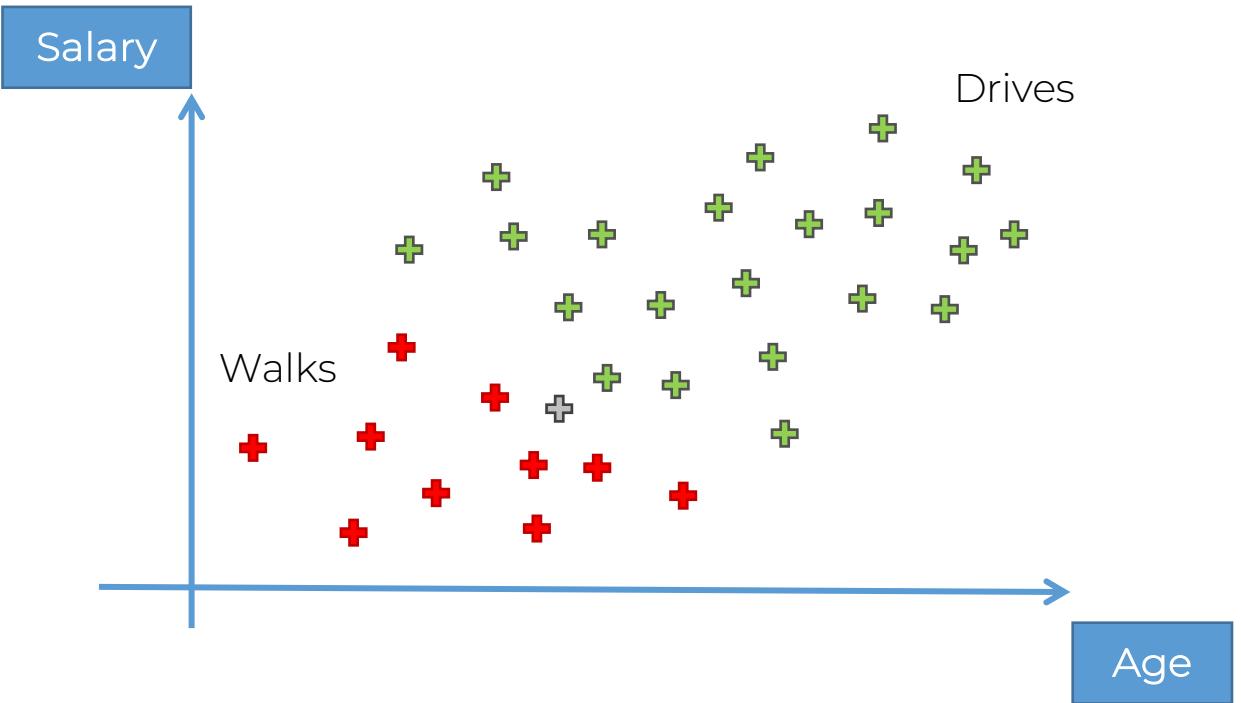
Naïve Bayes

Ready?

Naïve Bayes: Step 1



Naïve Bayes: Step 1



#1. $P(\text{Walks})$

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

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

Naïve Bayes: Step 1

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

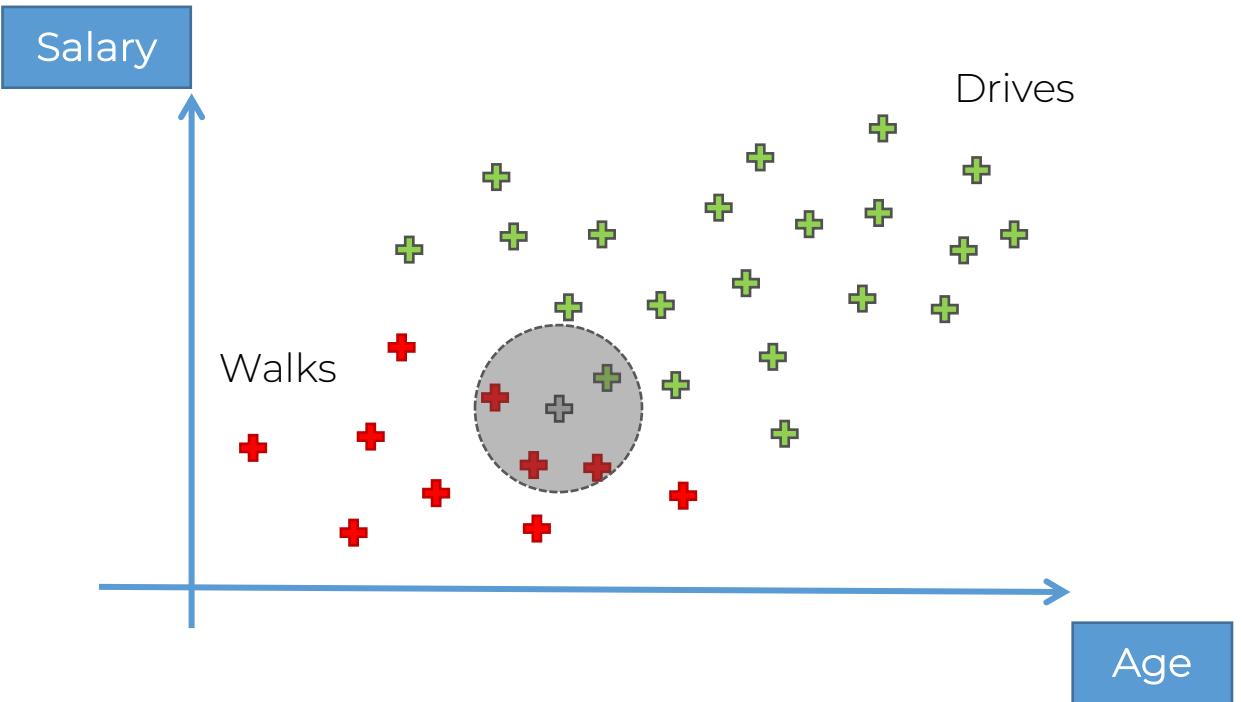
The diagram illustrates the components of the Naïve Bayes formula:

- #4 Posterior Probability
- #3 Likelihood
- #1 Prior Probability (represented by a green checkmark)
- #2 Marginal Likelihood (enclosed in a red oval)

Arrows point from the labels to their corresponding terms in the formula:

- An arrow points from #4 Posterior Probability to the term $P(Walks|X)$.
- An arrow points from #3 Likelihood to the term $P(X|Walks)$.
- An arrow points from #1 Prior Probability to the term $P(Walks)$.
- An arrow points from #2 Marginal Likelihood to the term $P(X)$.

Naïve Bayes: Step 1

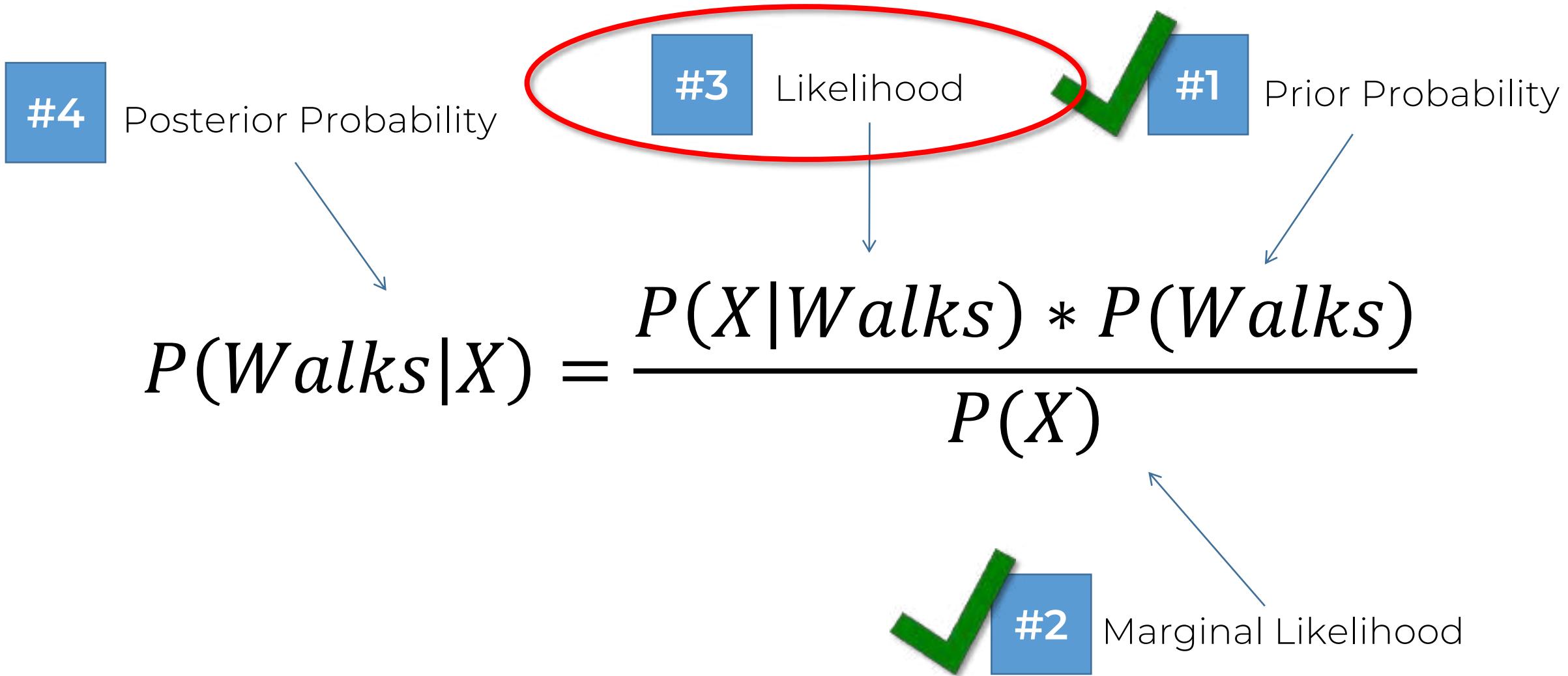


#2. $P(X)$

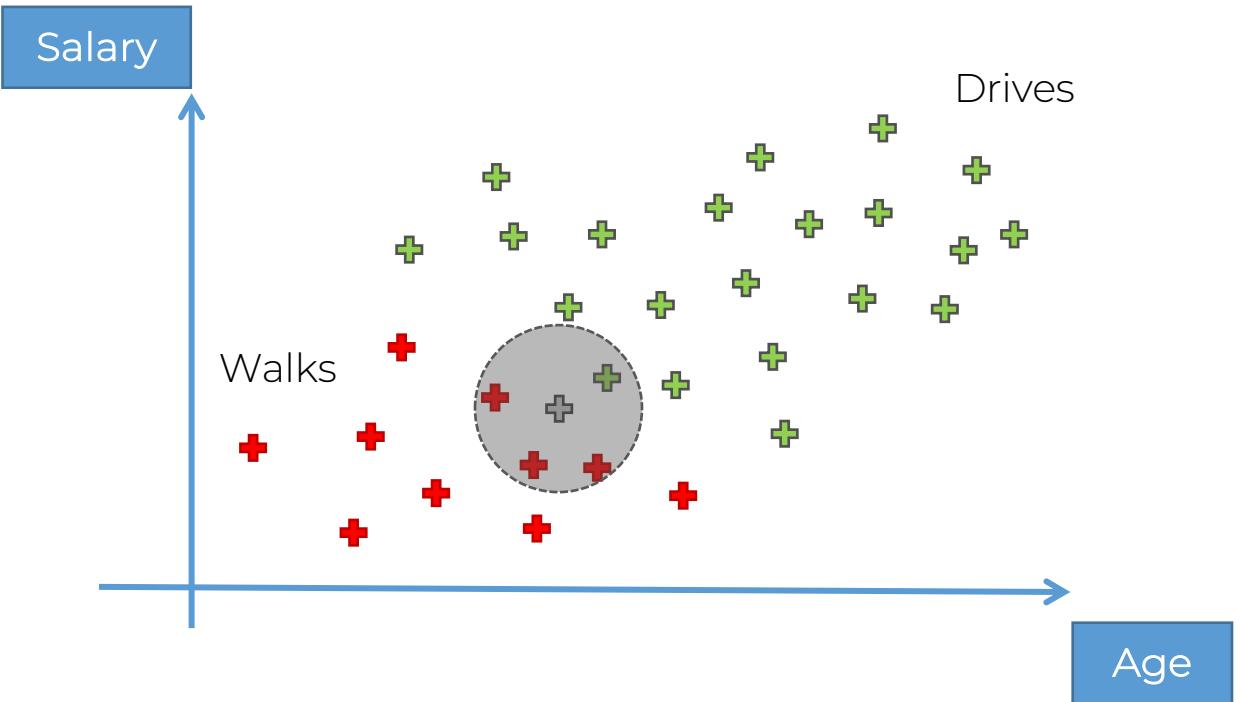
$$P(X) = \frac{\text{Number of Similar Observations}}{\text{Total Observations}}$$

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

Naïve Bayes: Step 1



Naïve Bayes: Step 1

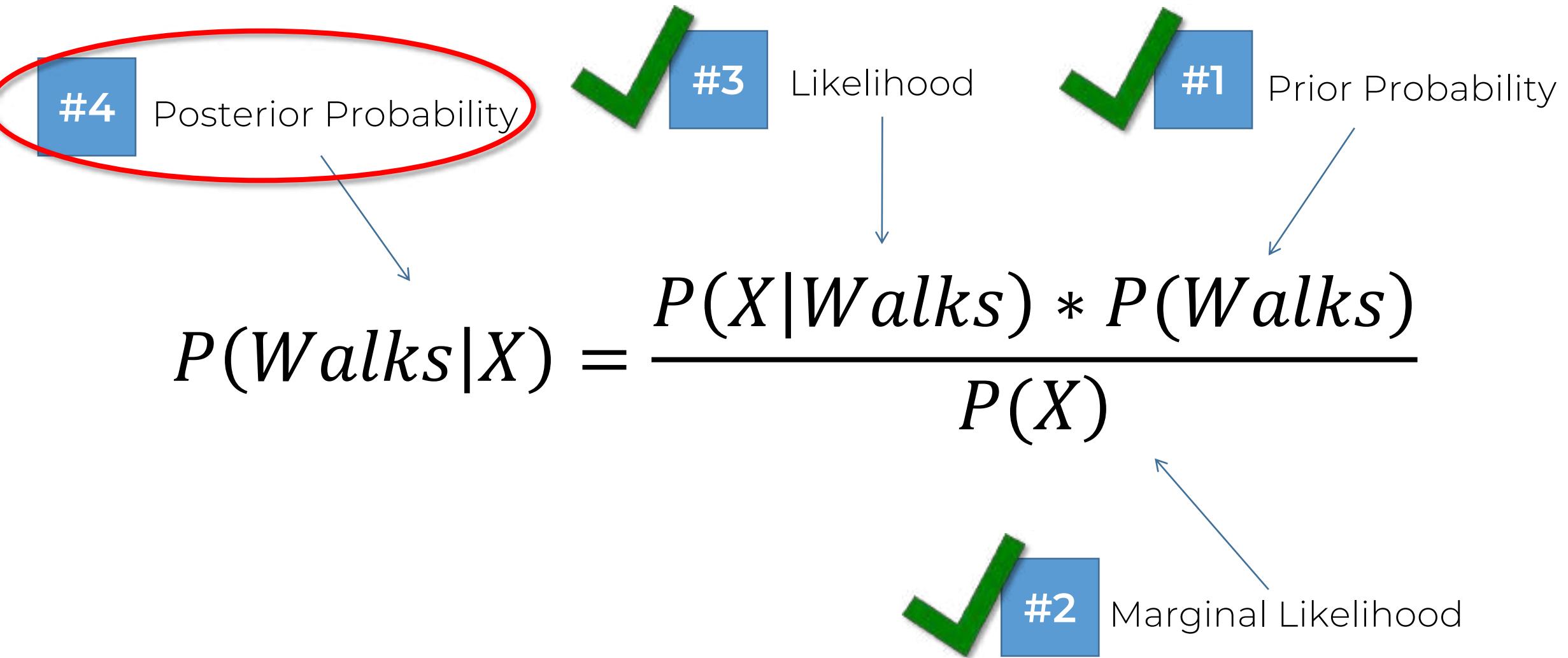


#3. $P(X|Walks)$

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

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

Naïve Bayes: Step 1



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

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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

#2 Marginal Likelihood

Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes

Step 2 - Done.

Step 3

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

Step 3

0.75 v. s. 0.25

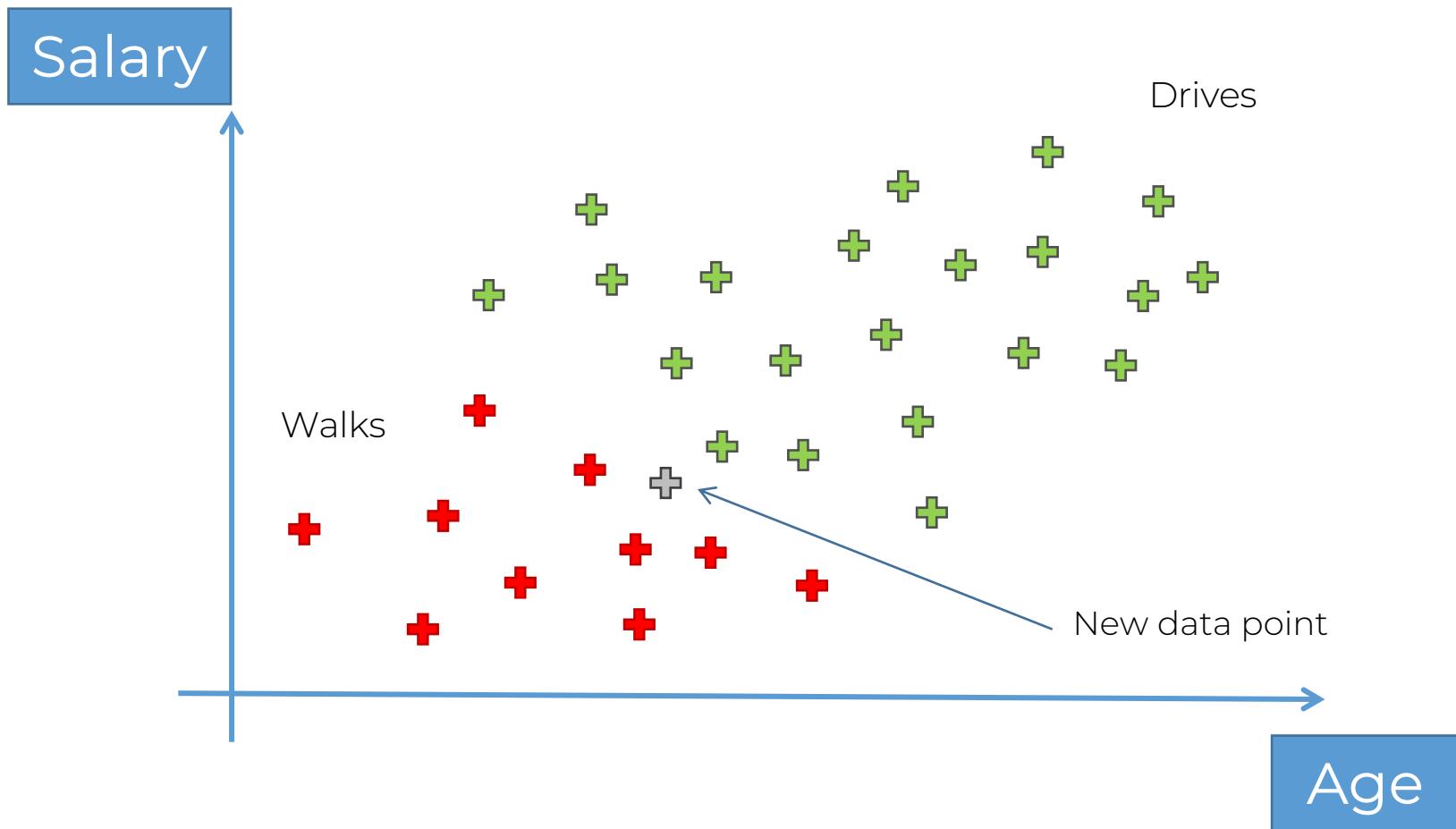
Step 3

$$0.75 > 0.25$$

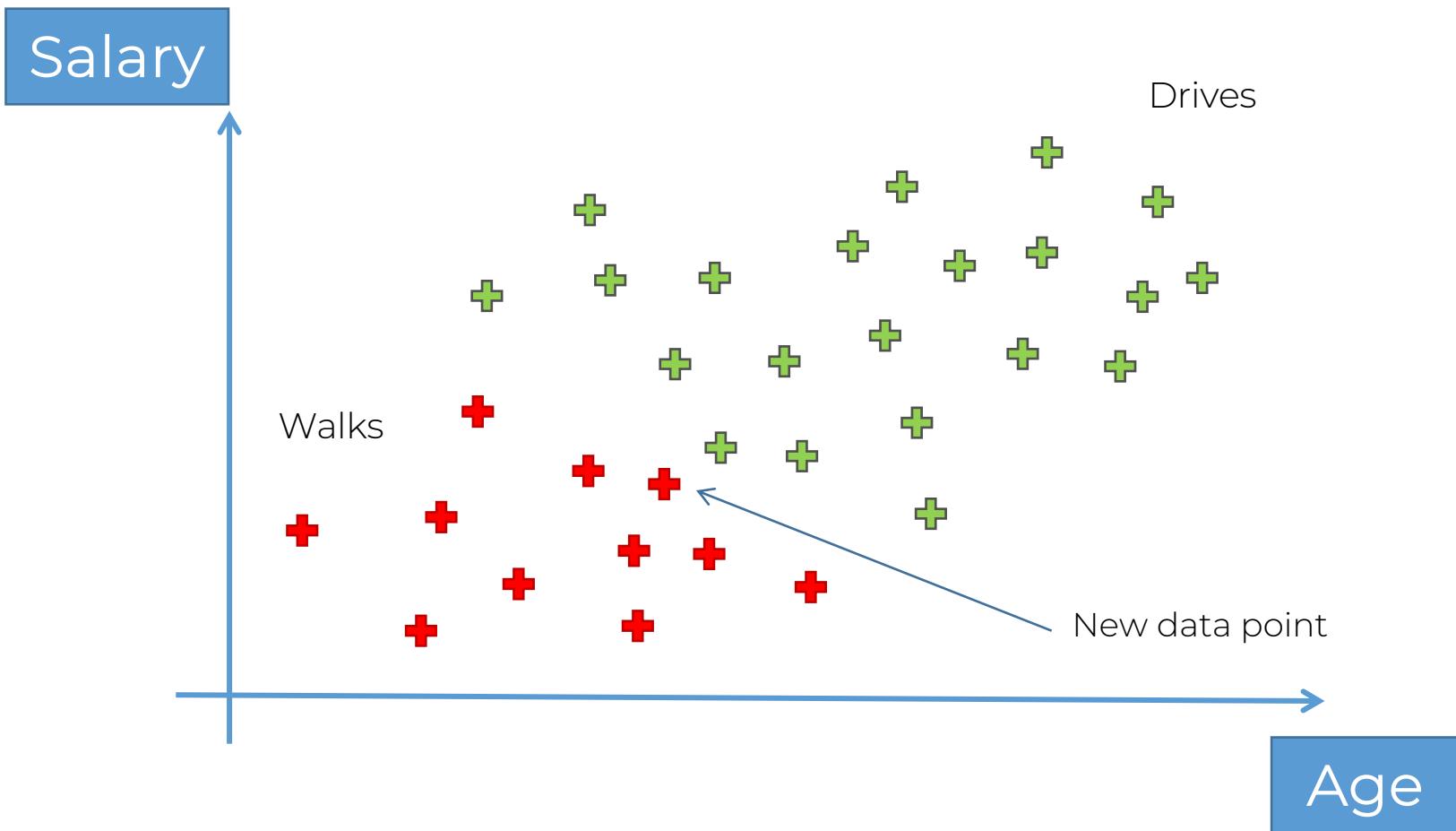
Step 3

$$P(\text{Walks}|X) > P(\text{Drives}|X)$$

Naïve Bayes



Naïve Bayes



Naïve Bayes Classifier Intuition (Challenge Reveal)

Step 2

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

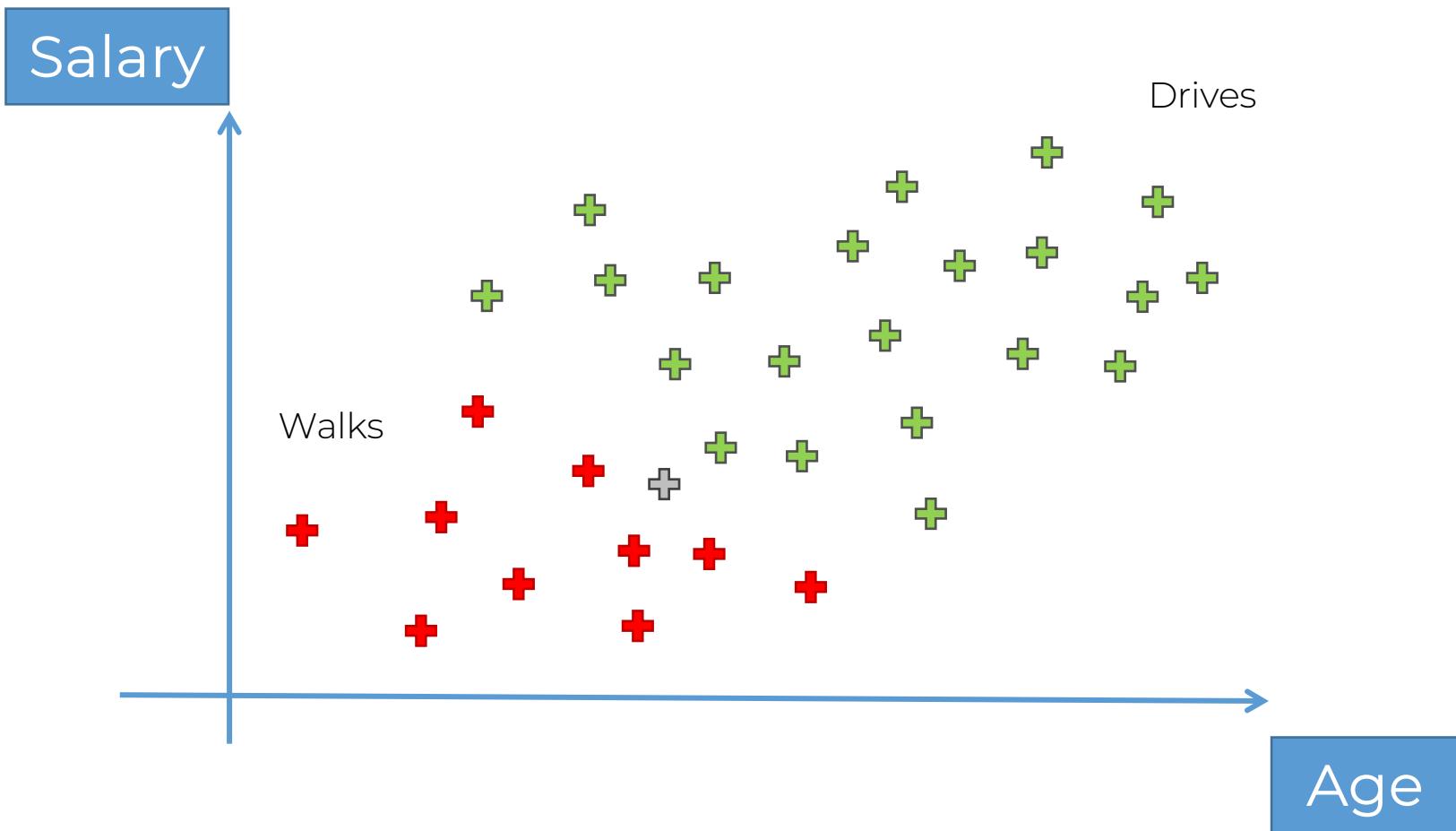
#4 Posterior Probability

#3 Likelihood

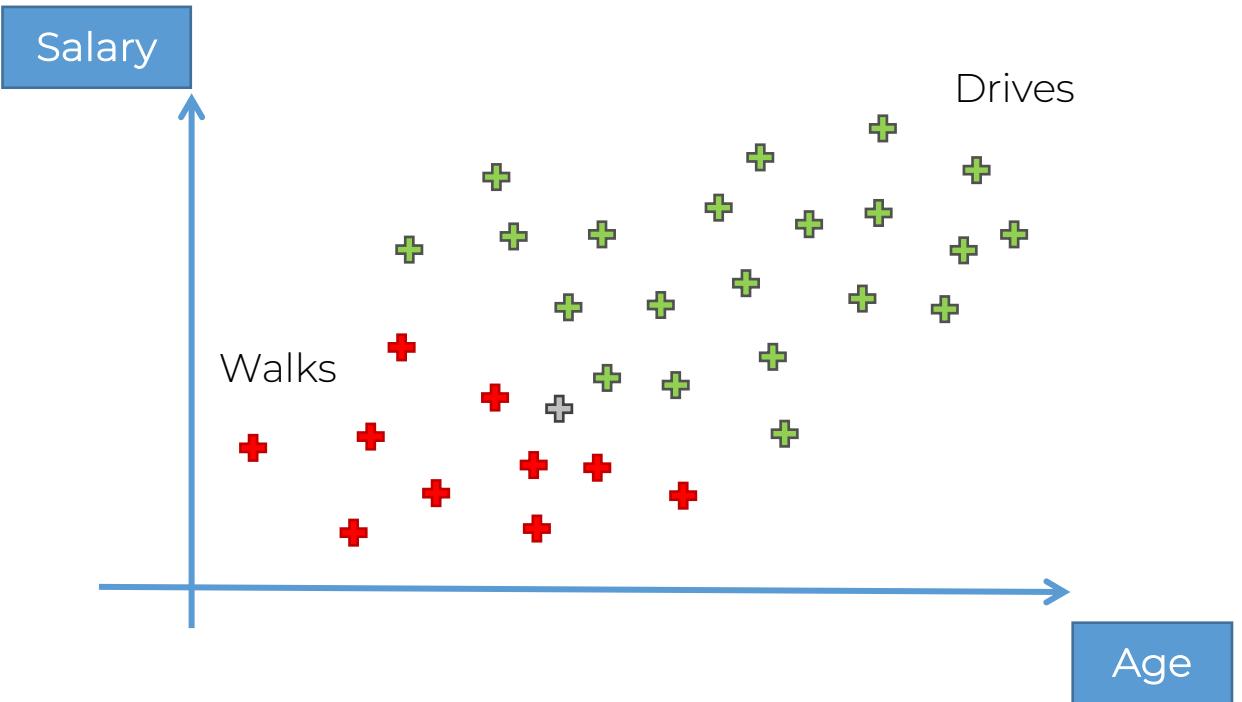
#1 Prior Probability

#2 Marginal Likelihood

Naïve Bayes: Step 2



Naïve Bayes: Step 2



#1. $P(\text{Drives})$

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

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

Naïve Bayes: Step 2

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

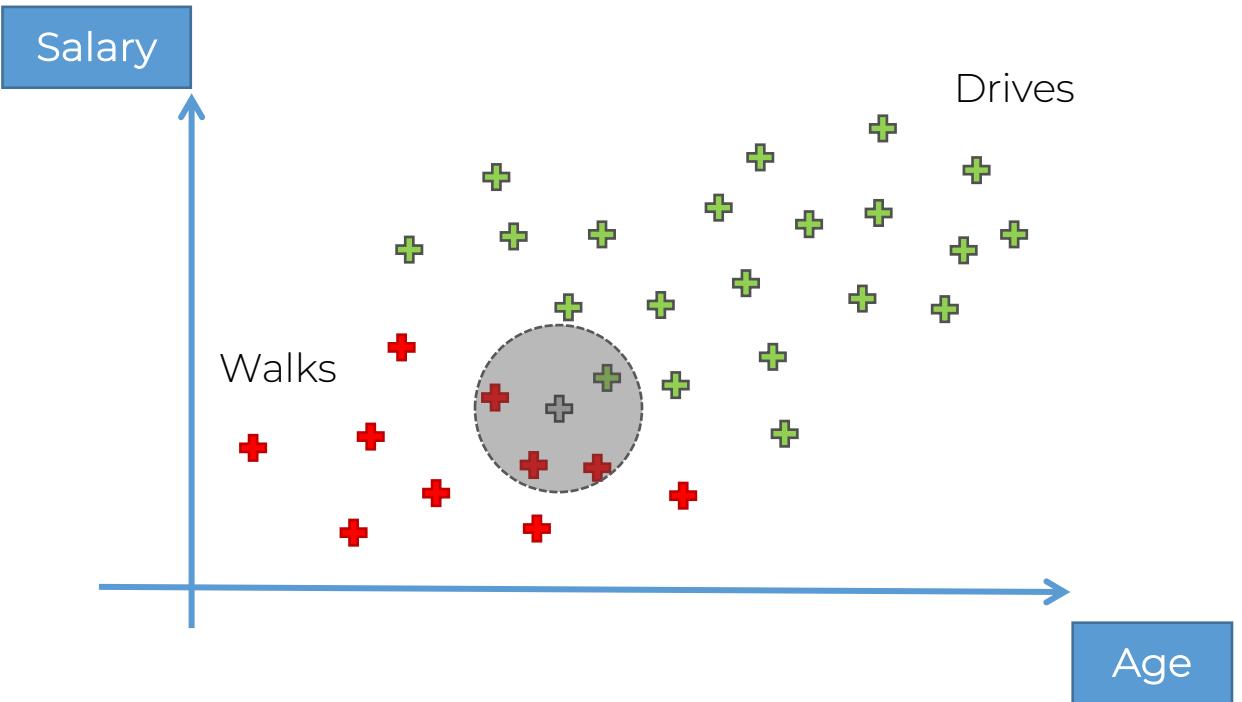
Diagram illustrating the components of the Naïve Bayes formula:

- #4 Posterior Probability
- #3 Likelihood
- #1 Prior Probability
- #2 Marginal Likelihood (circled in red)

Arrows point from the labels to their corresponding terms in the formula:

- #4 Posterior Probability points to $P(\\text{Drives}|X)$
- #3 Likelihood points to $P(X|\\text{Drives})$
- #1 Prior Probability points to $P(\\text{Drives})$
- #2 Marginal Likelihood points to $P(X)$

Naïve Bayes: Step 2



#2. $P(X)$

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

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

Naïve Bayes: Step 2

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

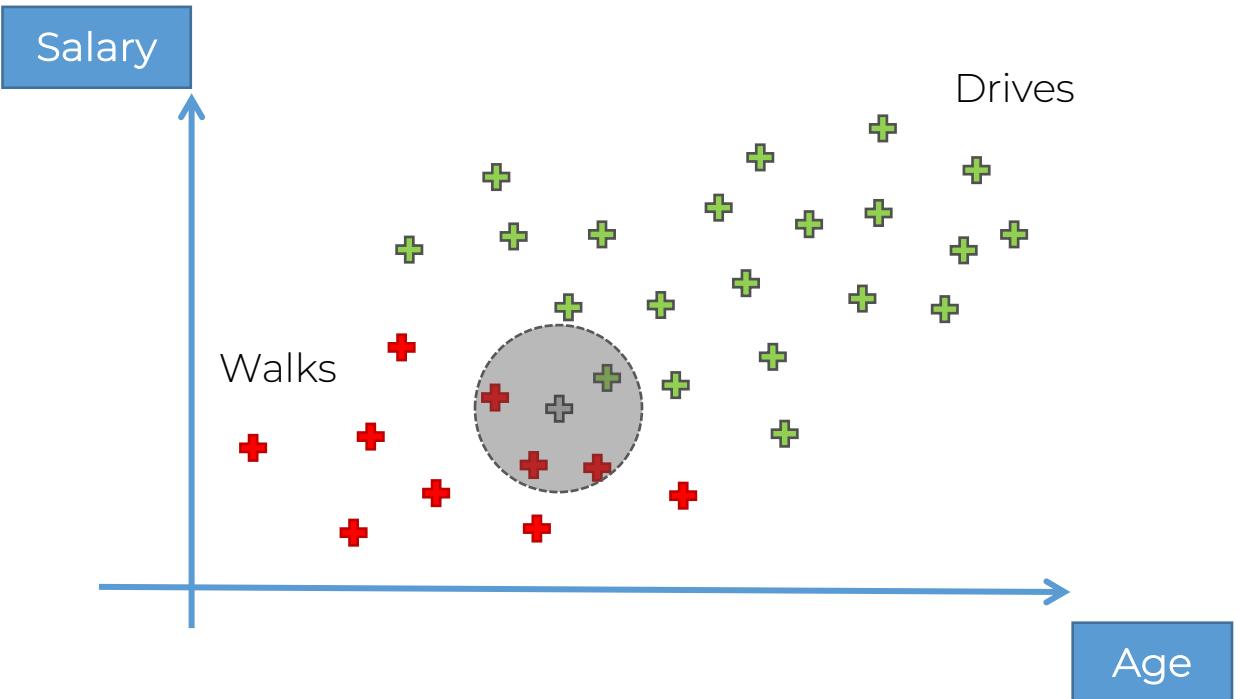
Diagram illustrating the components of the Naïve Bayes formula:

- #4 Posterior Probability
- #3 Likelihood (circled in red)
- #1 Prior Probability
- #2 Marginal Likelihood

Arrows point from the labels to their corresponding terms in the equation:

- #4 Posterior Probability points to $P(\\text{Drives}|X)$
- #3 Likelihood points to $P(X|\\text{Drives})$
- #1 Prior Probability points to $P(\\text{Drives})$
- #2 Marginal Likelihood points to $P(X)$

Naïve Bayes: Step 2



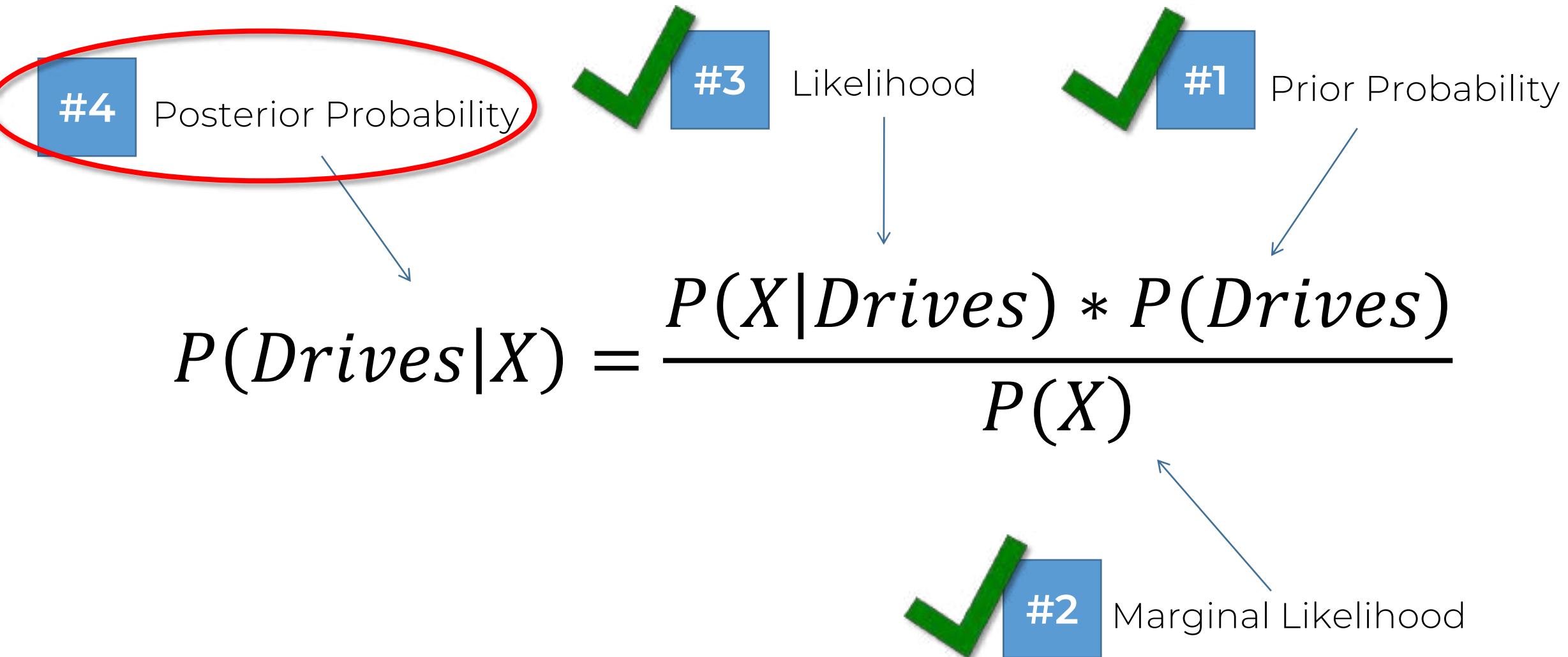
#3. $P(X|Drives)$

Number of Similar Observations Among those who Walk

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

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

Naïve Bayes: Step 2



Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

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

#2 Marginal Likelihood

Naïve Bayes

Step 2 - Done.

Naïve Bayes Classifier Additional Comments

Naïve Bayes

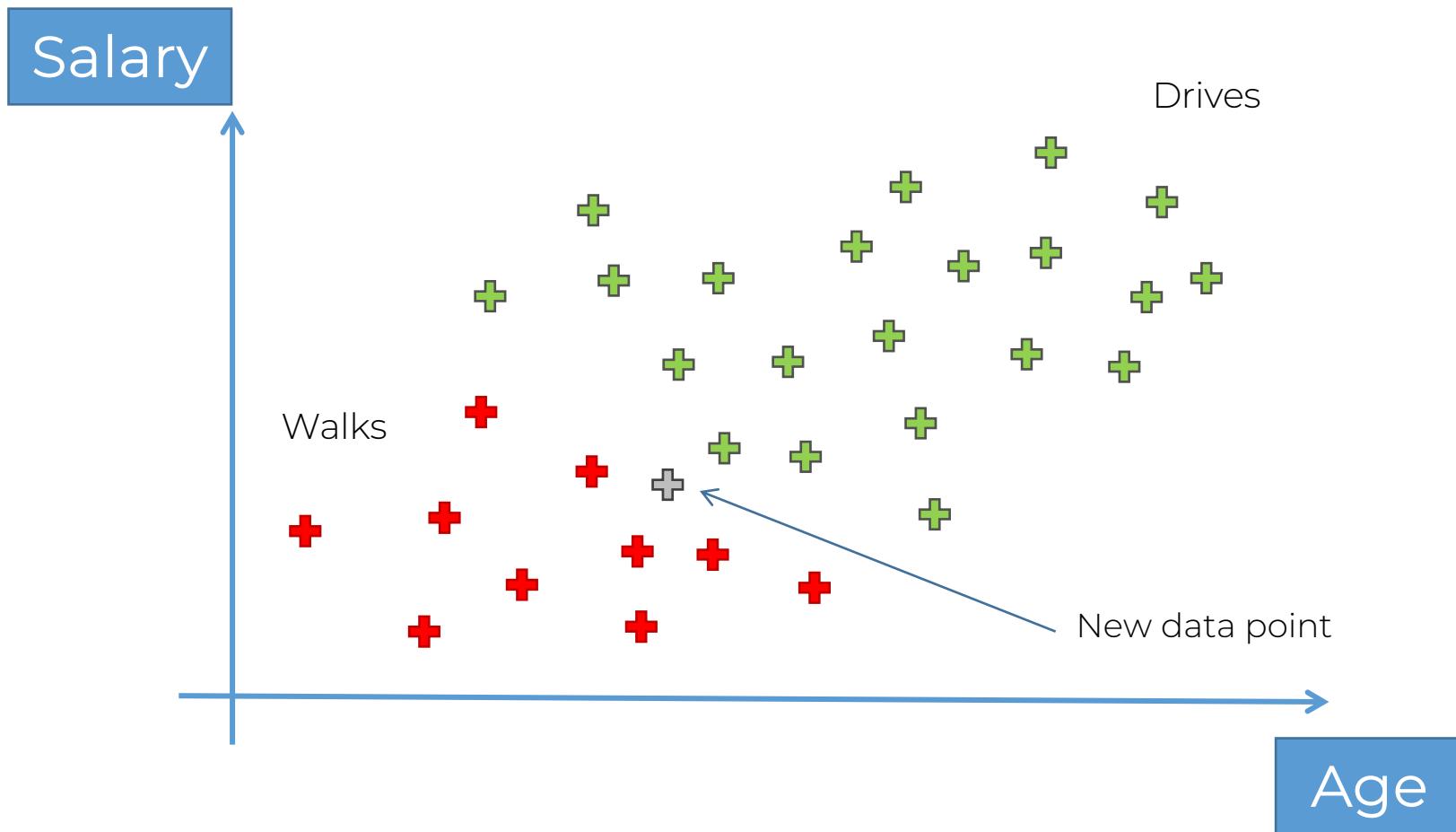
1. Q: Why “Naïve”?
2. $P(X)$
3. More than 2 features

Naïve Bayes

Q: Why “Naïve”?

A: Independence assumption

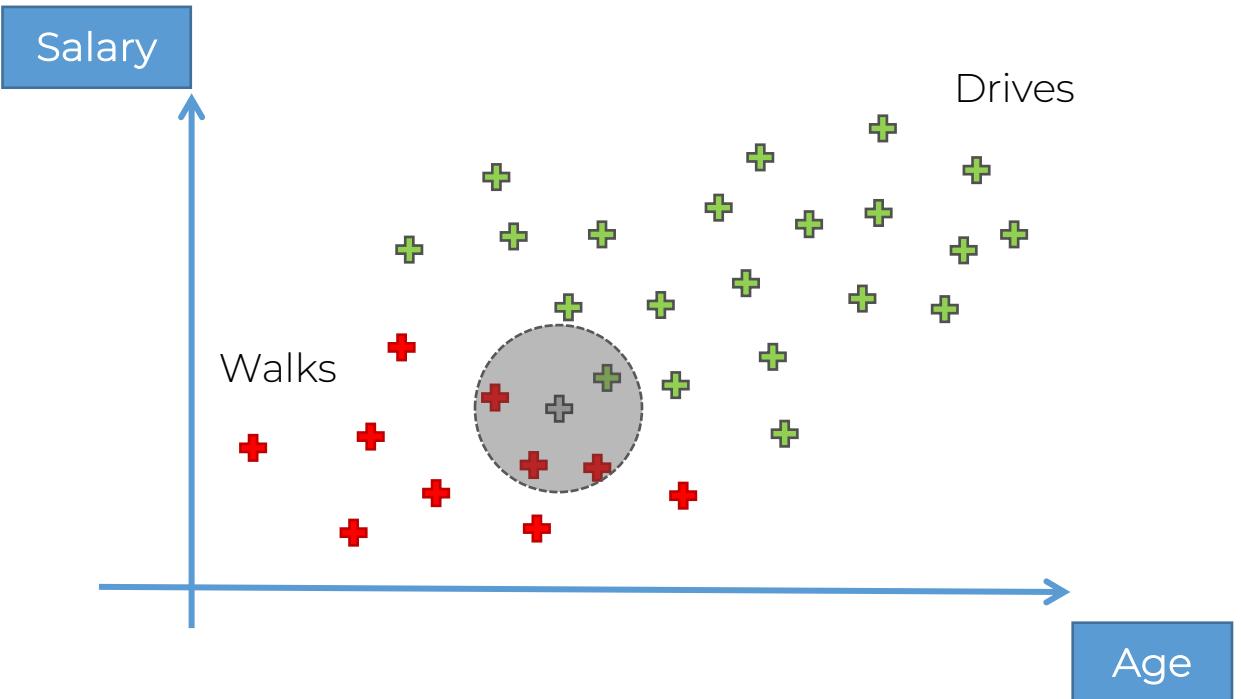
Naïve Bayes



Naïve Bayes

$P(X)$

Naïve Bayes: Step 2



#2. $P(X)$

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

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

NOTE: Same both times

Step 1

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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

#2 Marginal Likelihood

Step 2

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

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

#2 Marginal Likelihood

Step 3

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

Step 3

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

Naïve Bayes

More than 2 classes

Step 3

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

Step 3

0.75 v. s. 0.25

Step 3

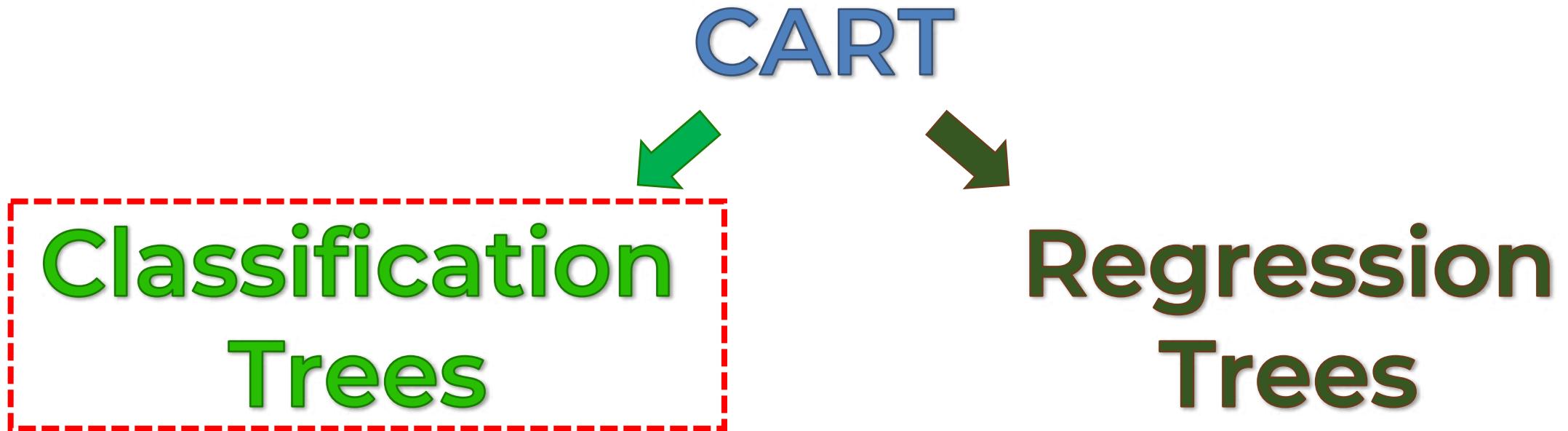
$$0.75 > 0.25$$

Step 3

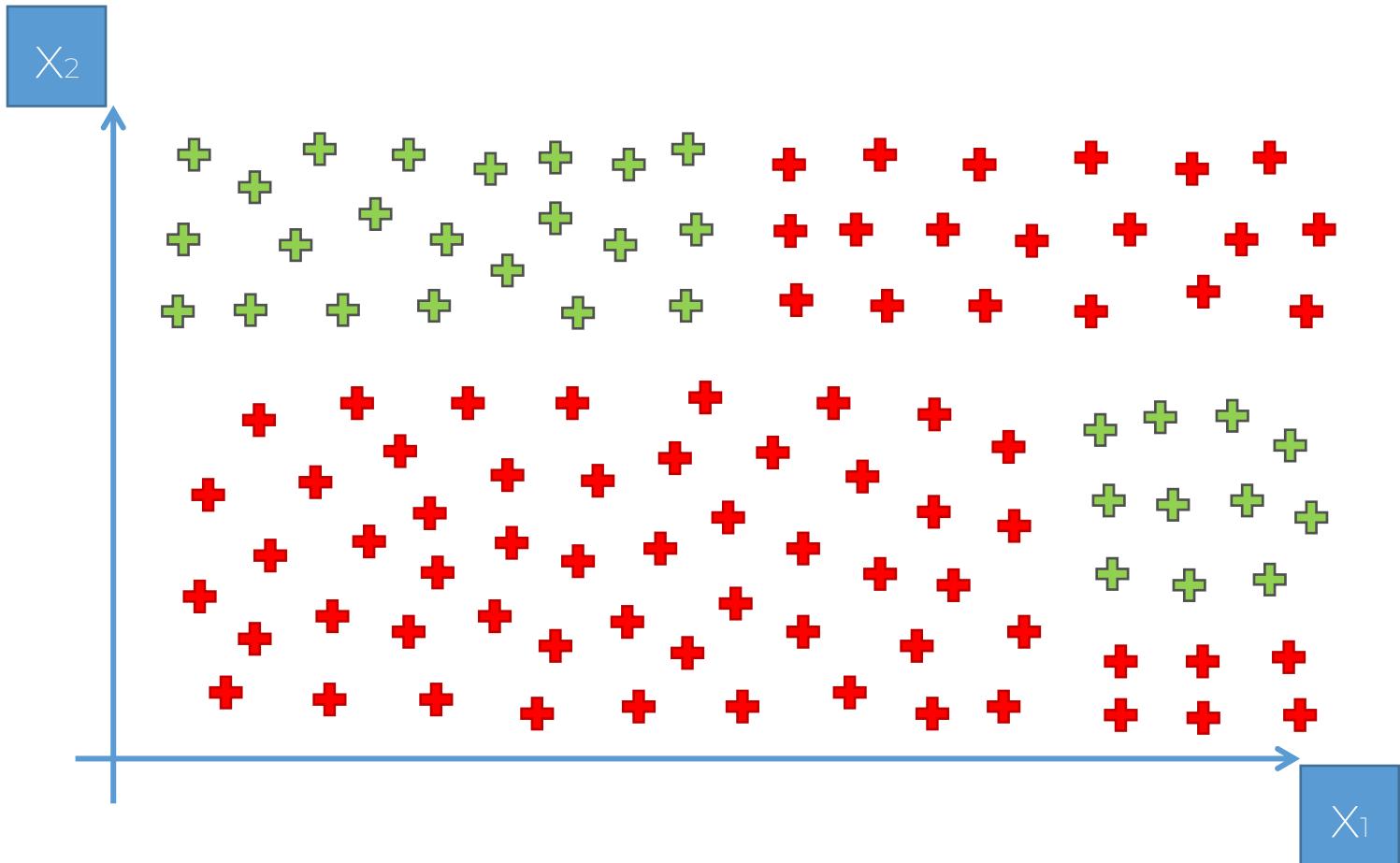
$$P(\text{Walks}|X) > P(\text{Drives}|X)$$

Decision Tree Intuition

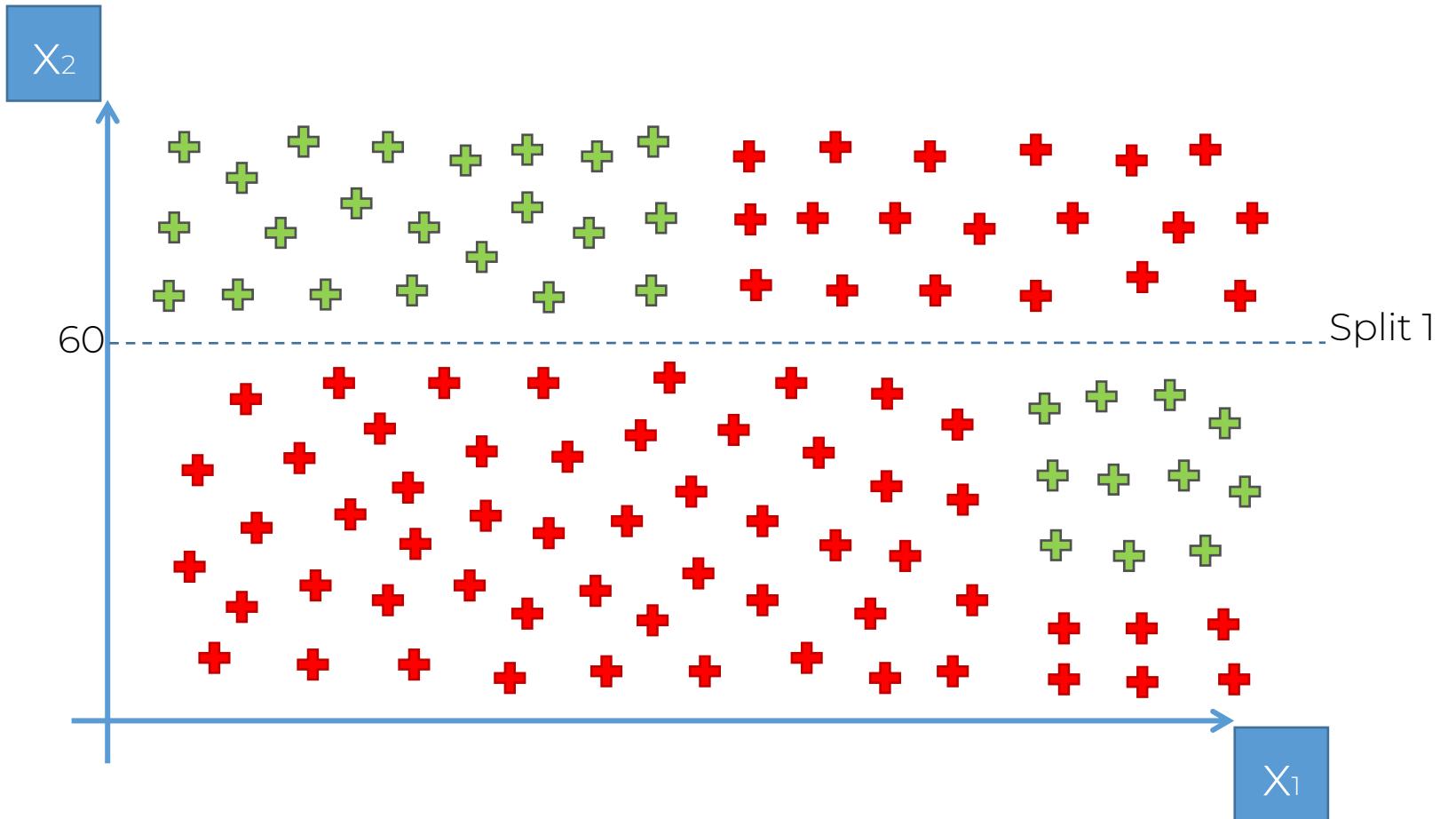
Decision Tree Intuition



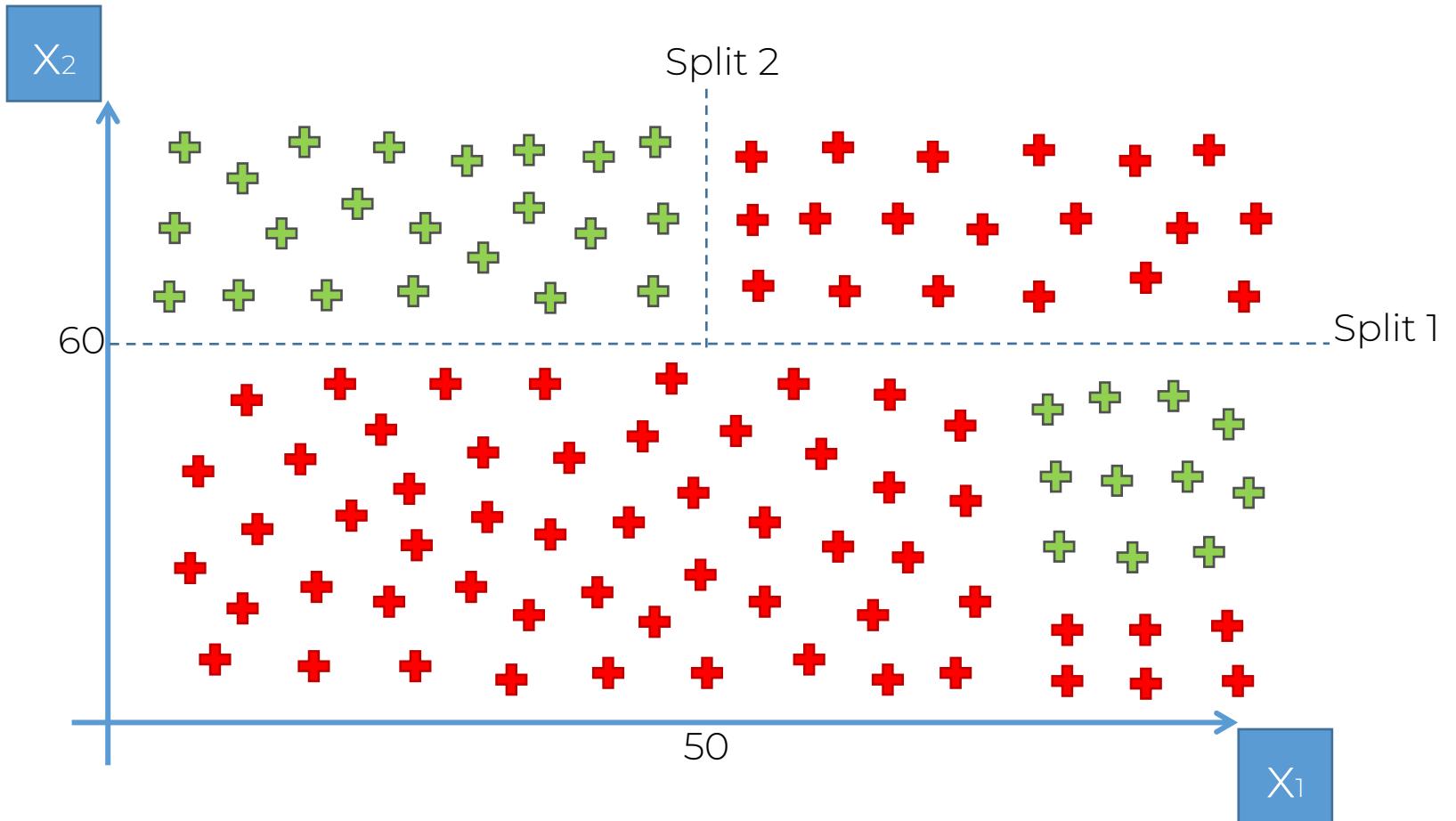
Decision Tree Intuition



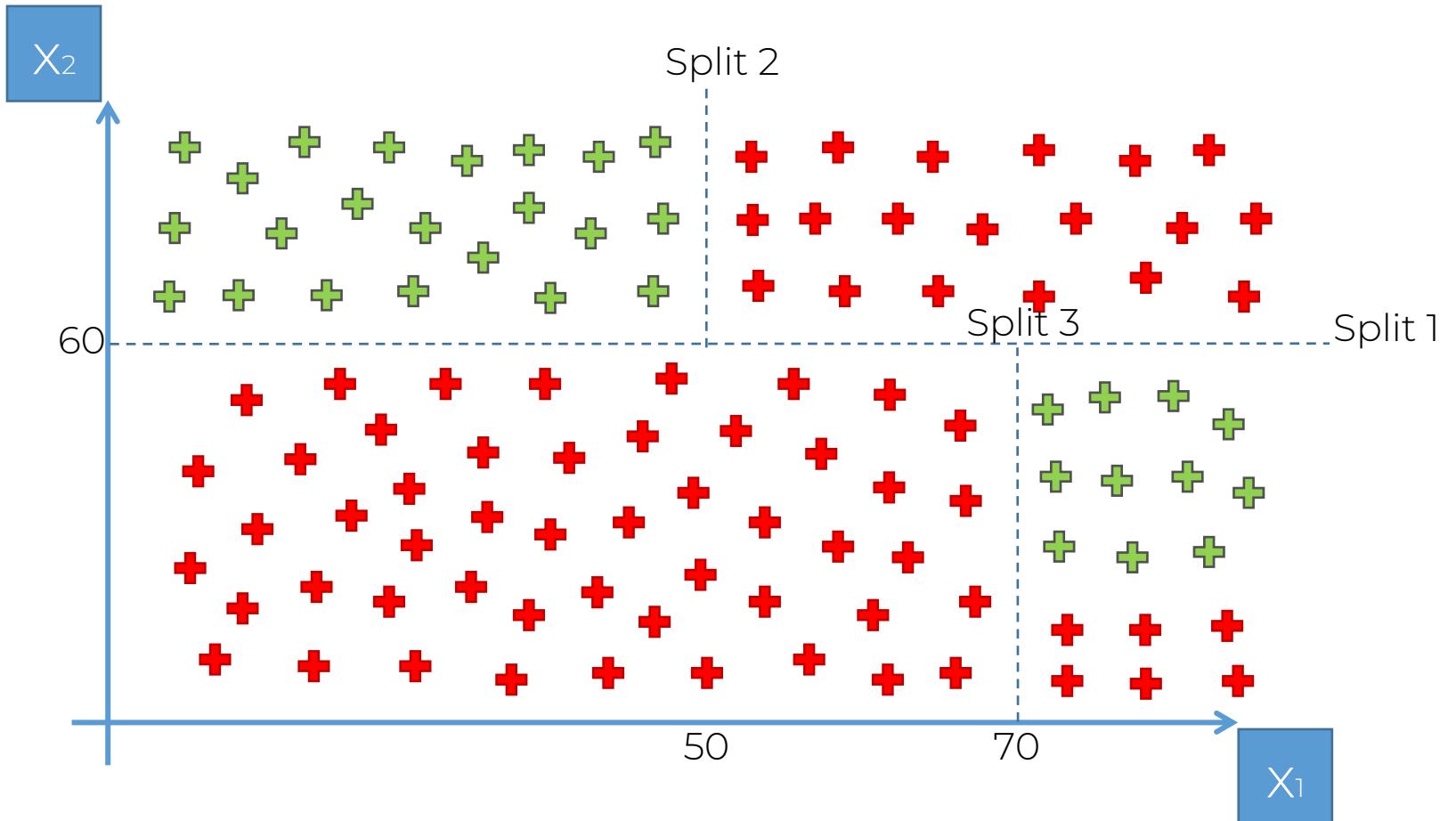
Decision Tree Intuition



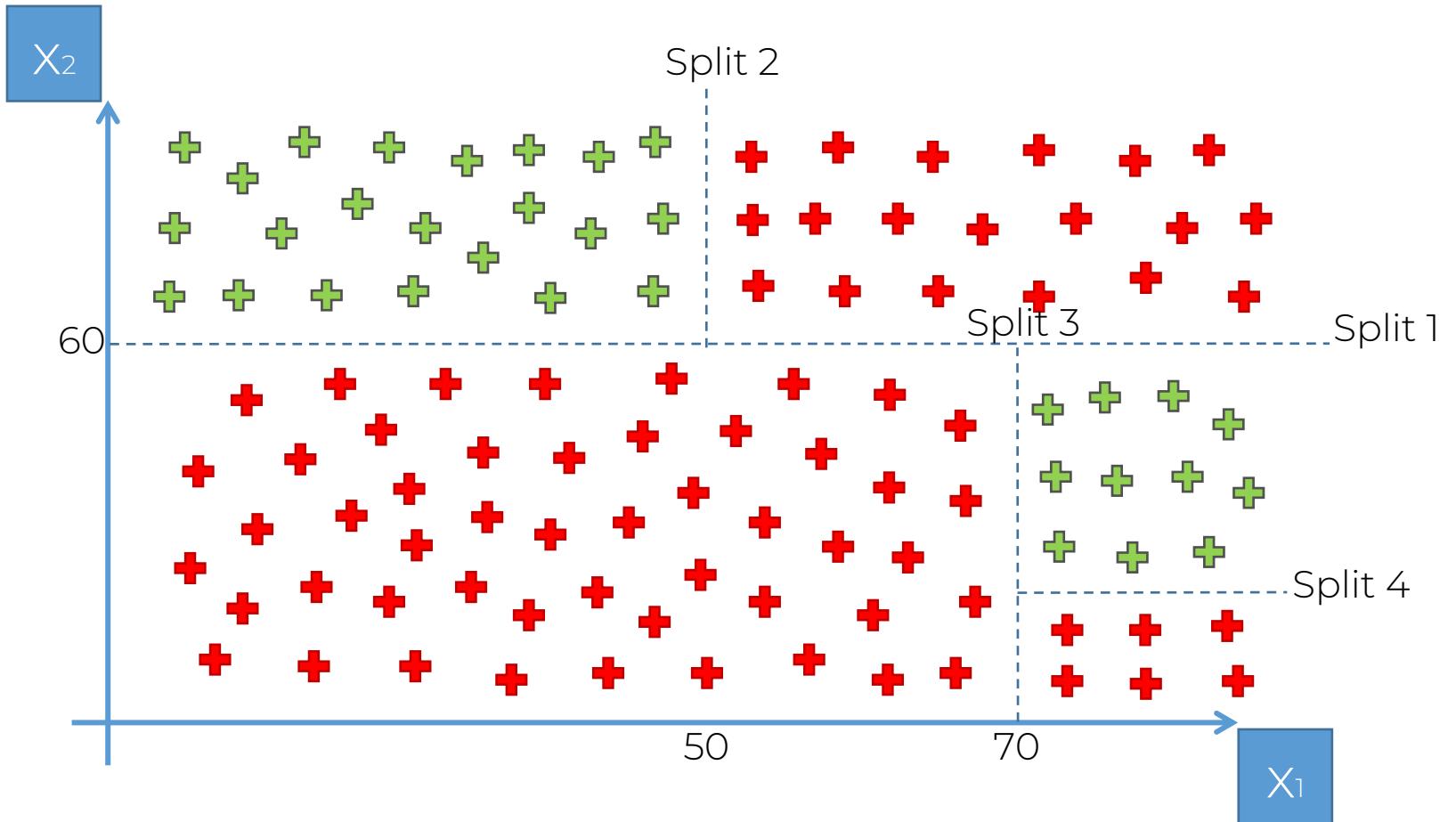
Decision Tree Intuition



Decision Tree Intuition



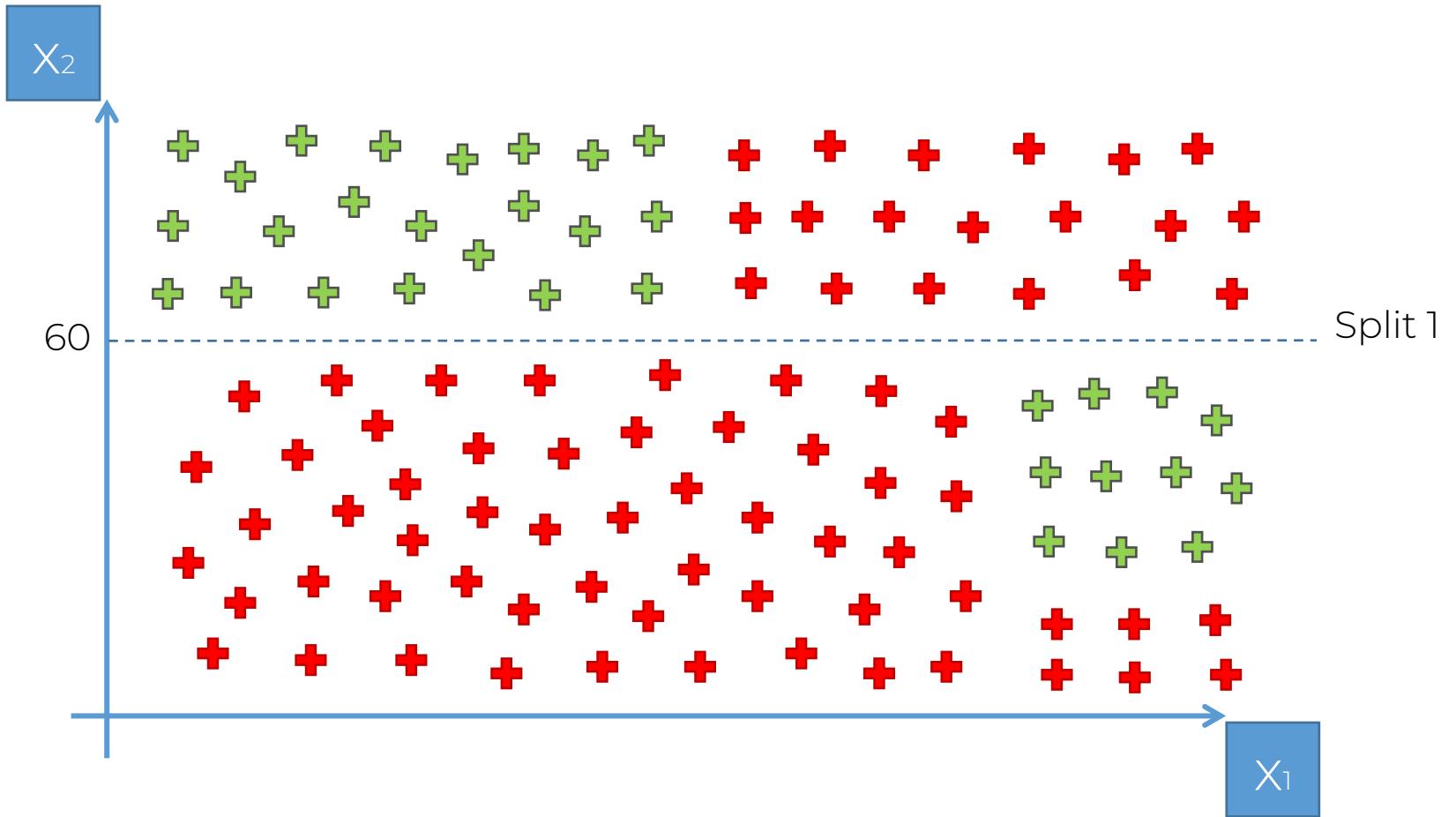
Decision Tree Intuition



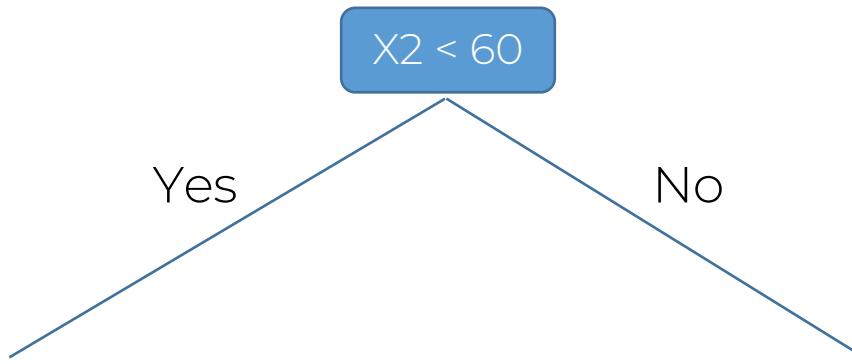
Decision Tree Intuition

Rewind...

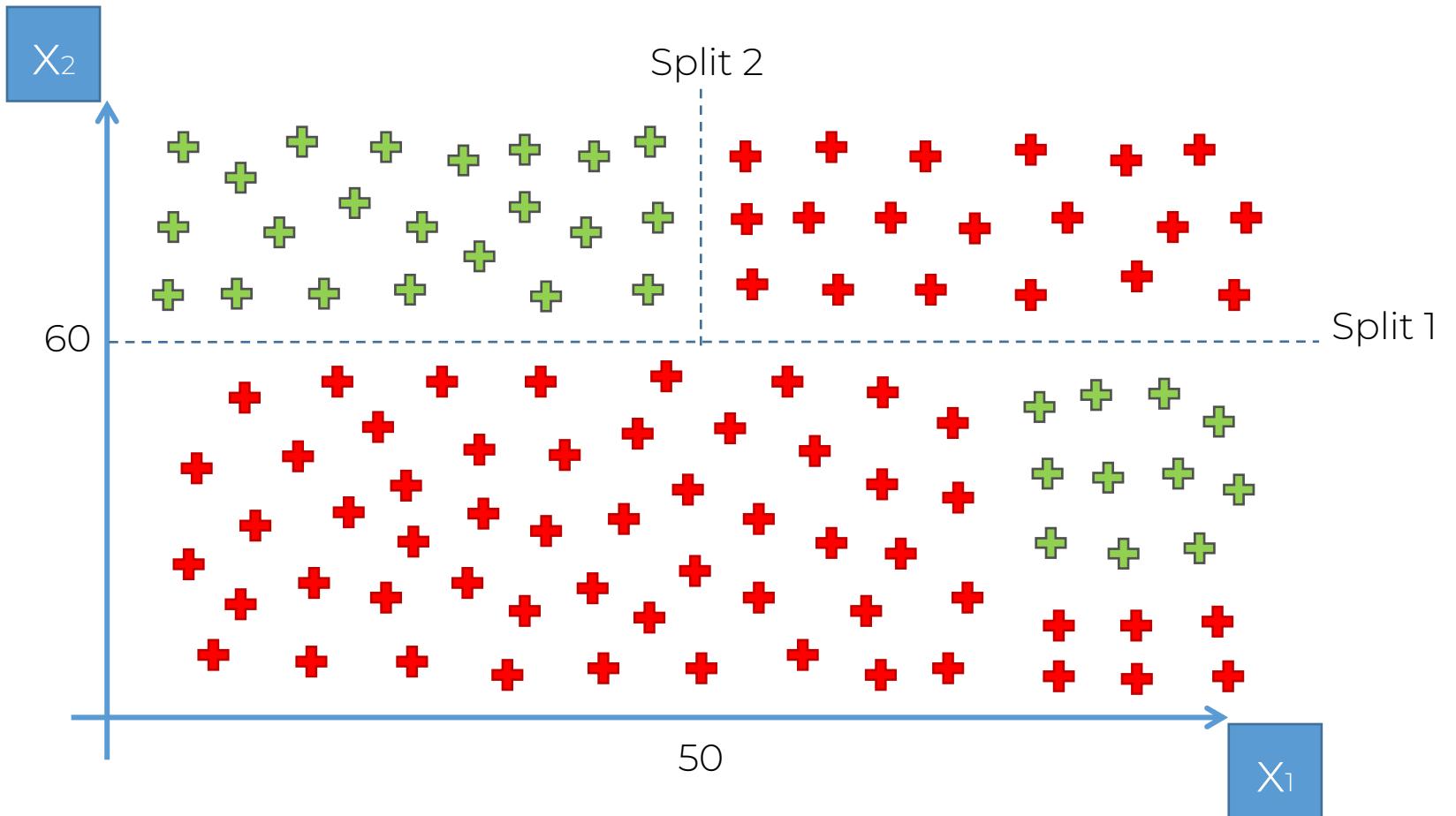
Split 1



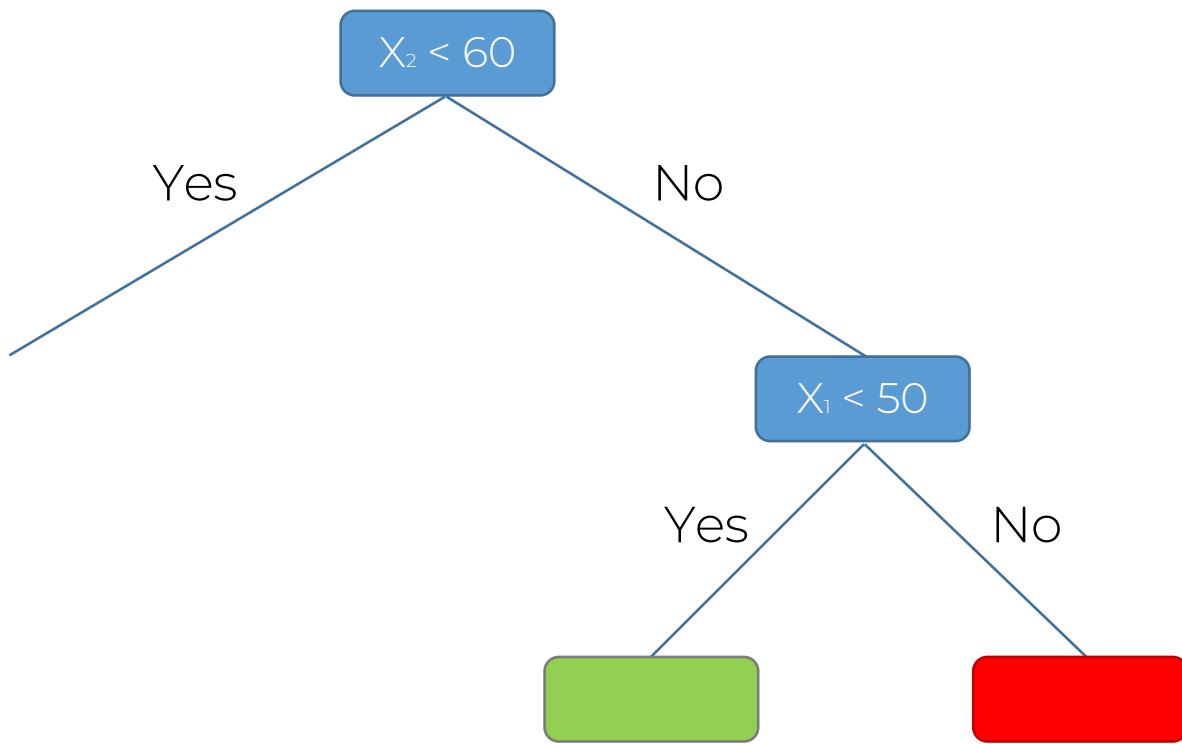
Split 1



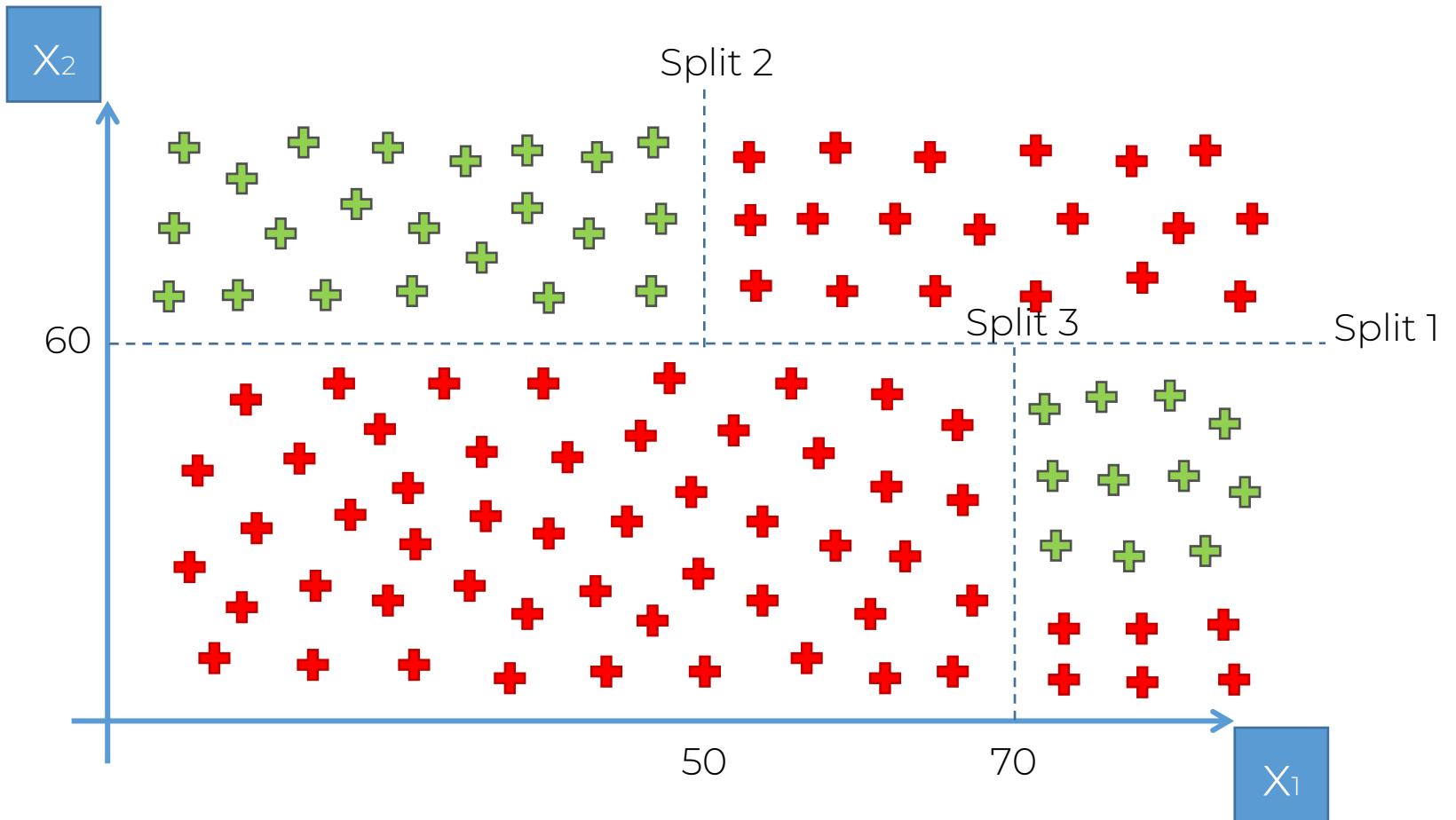
Split 2



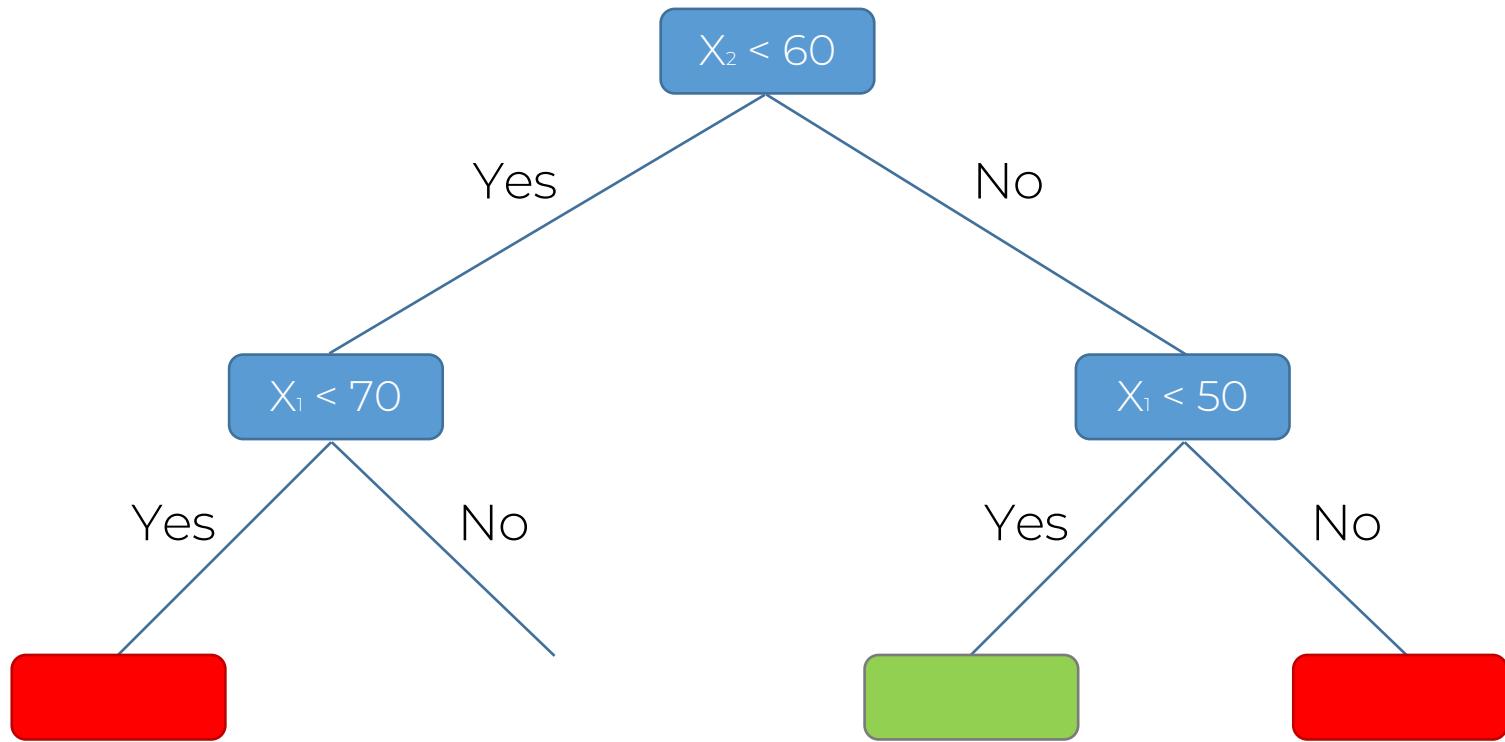
Split 2



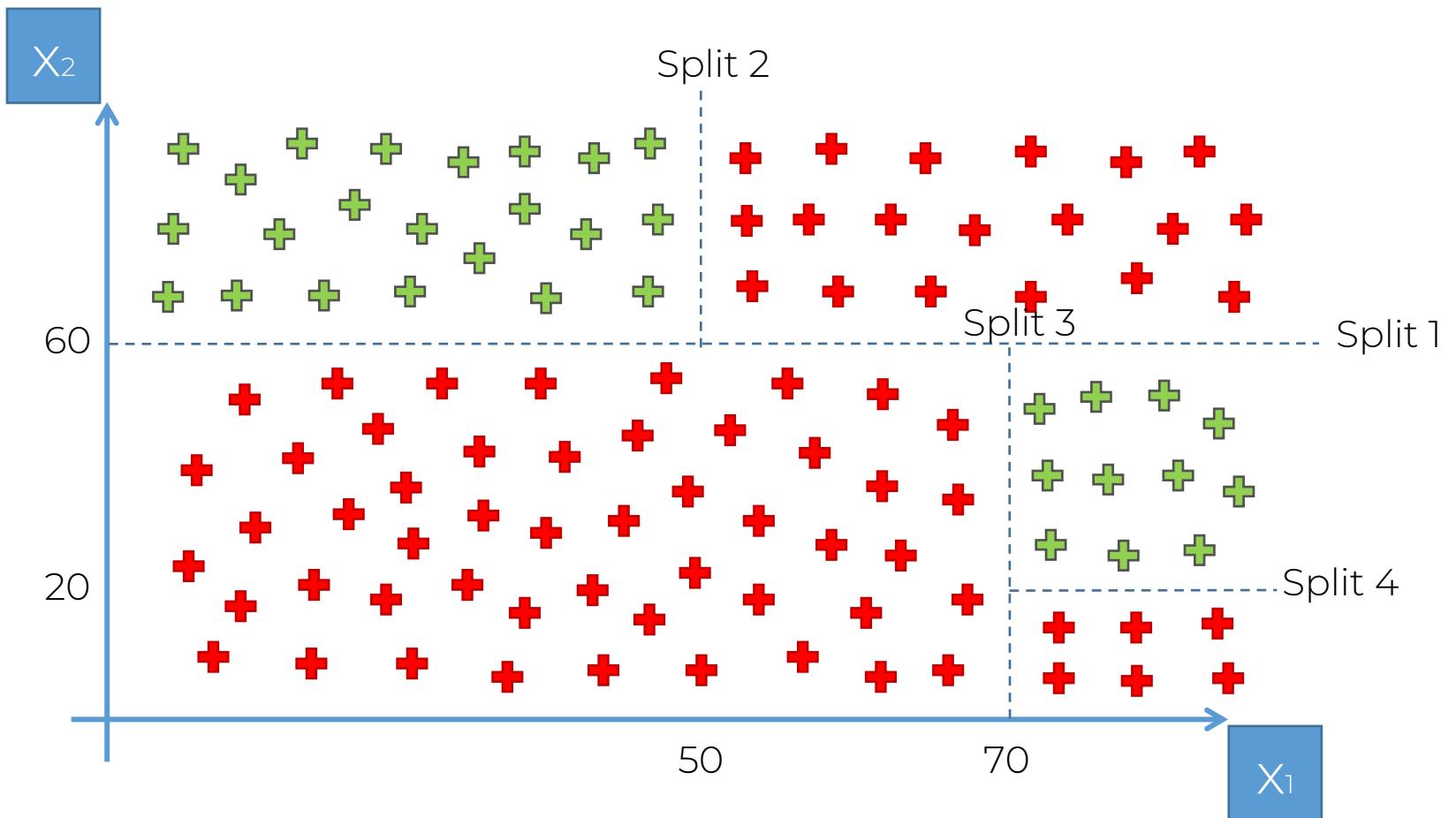
Split 3



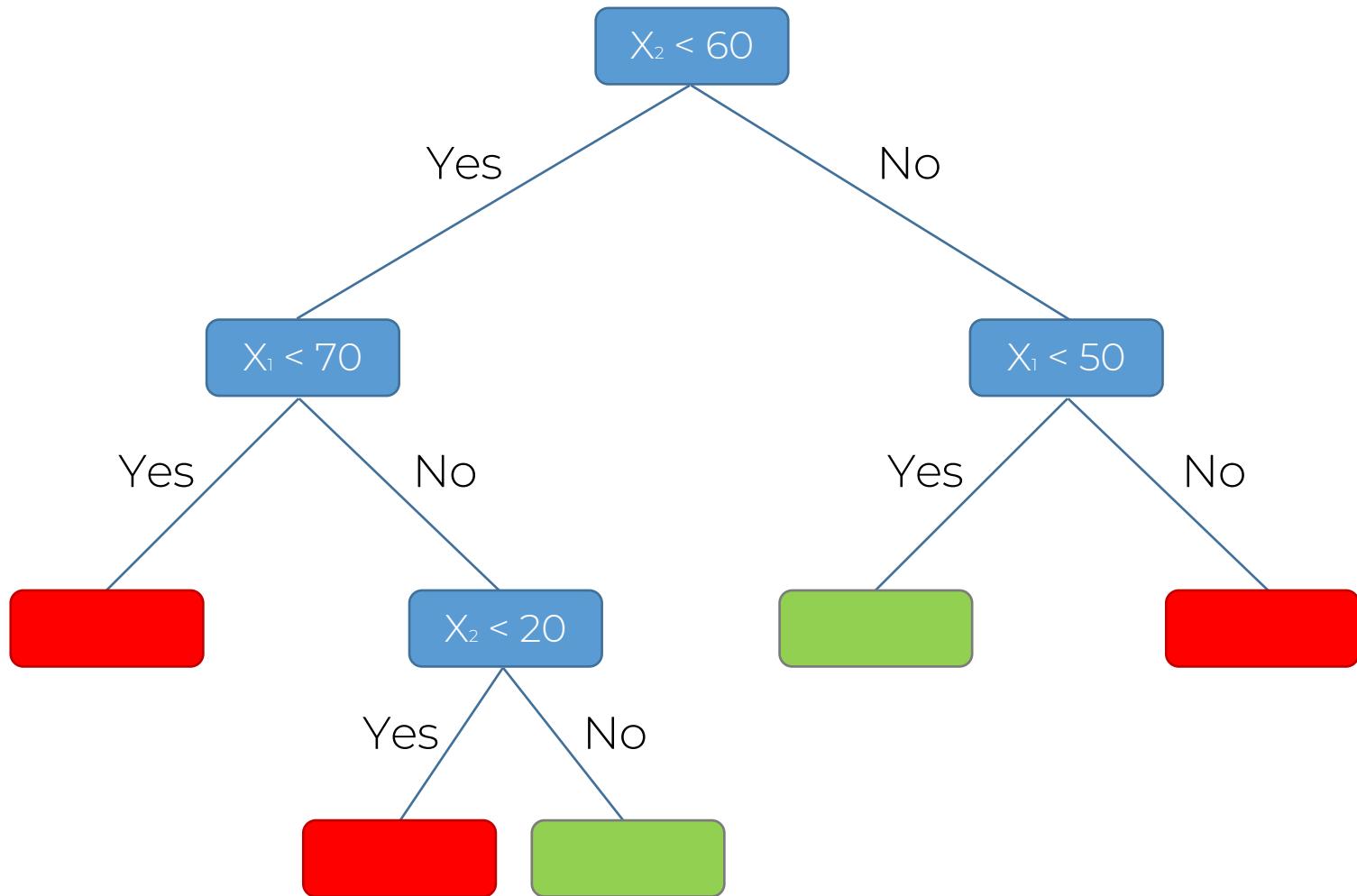
Split 3



Split 4



Split 4



Decision Trees

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

Random Forest Intuition

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.



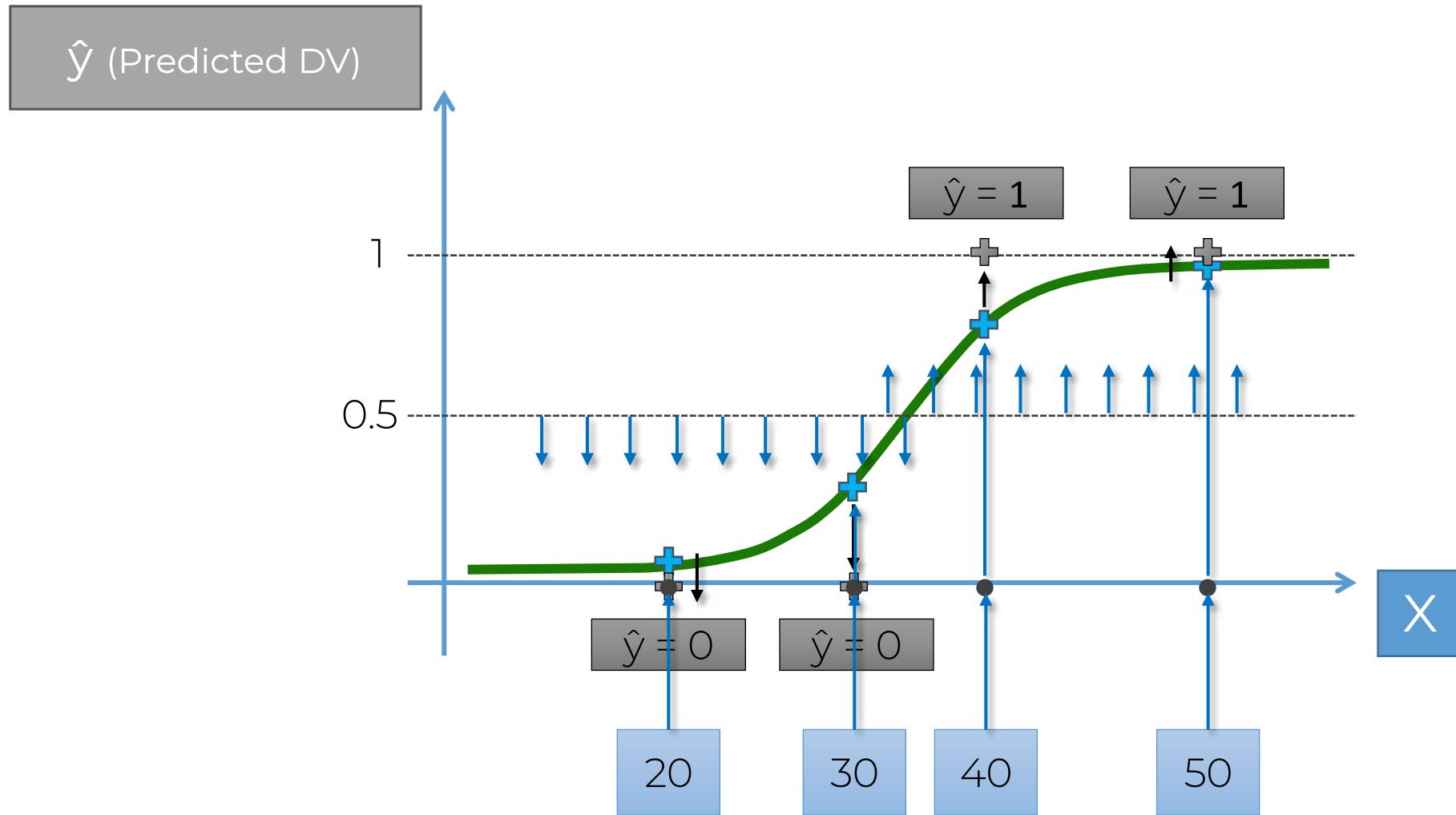
STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2



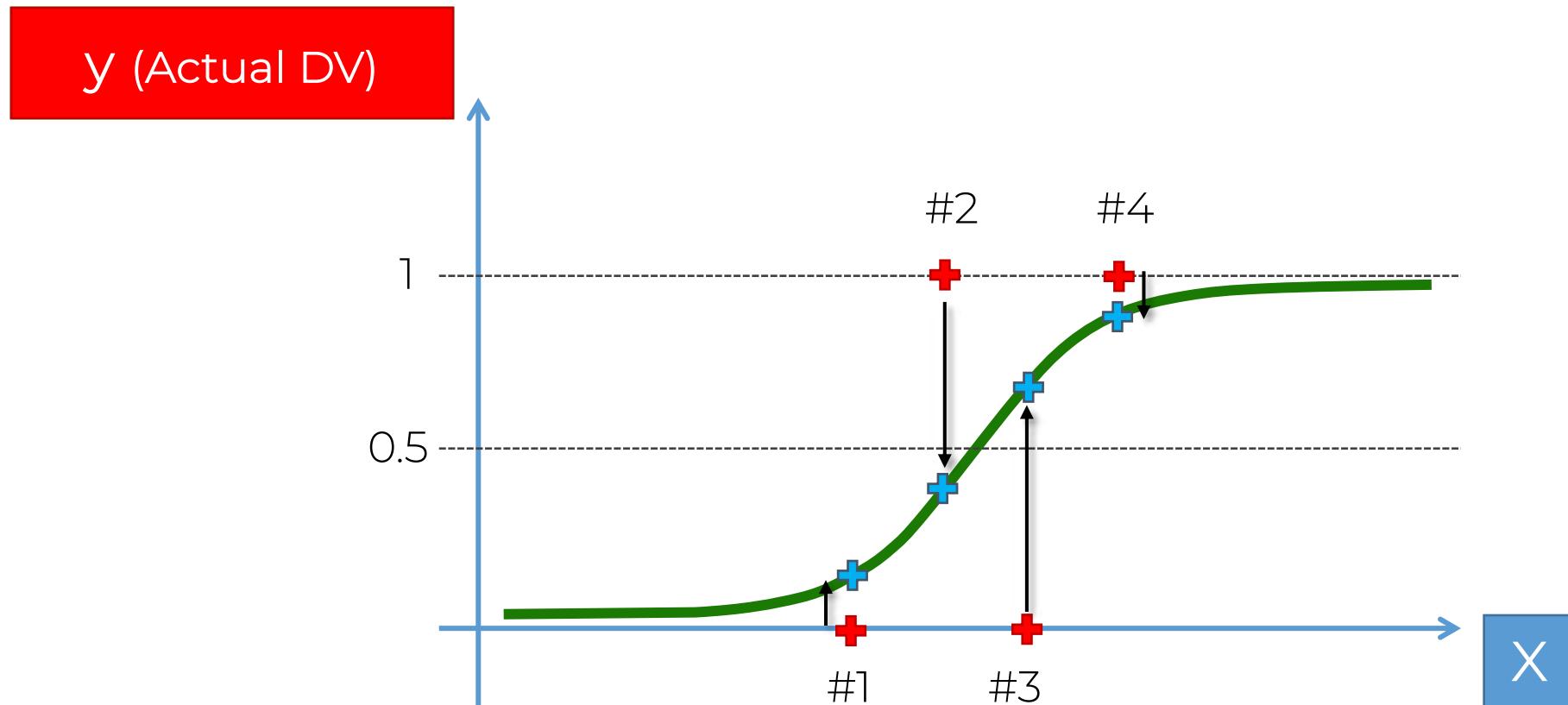
STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

False Positives False Negatives

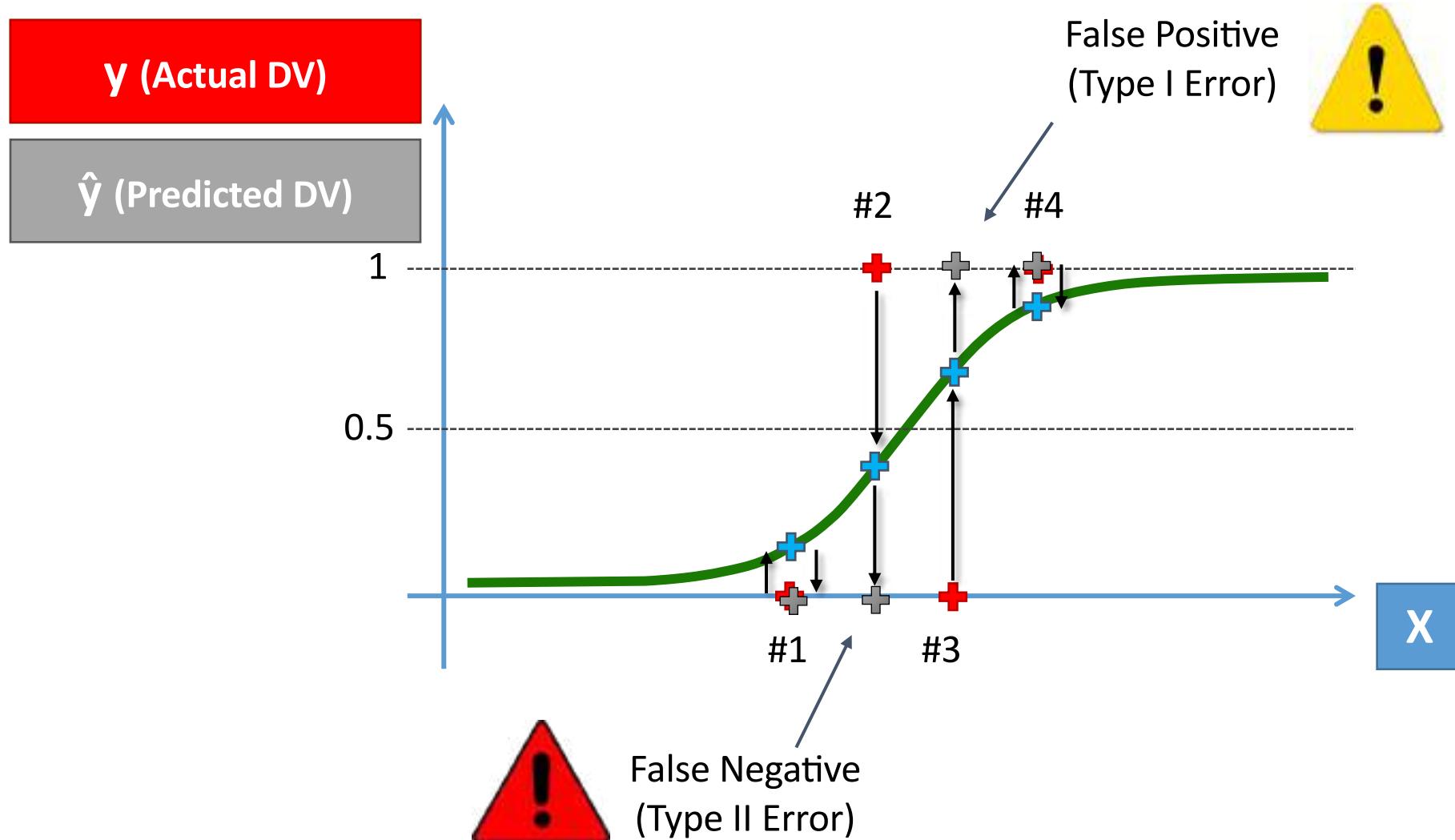
False Positives & Negatives



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) points to the bottom-left cell (Actual POS, Predicted NEG).

Type I Error (False Positives) points to the bottom-right cell (Actual NEG, Predicted POS).

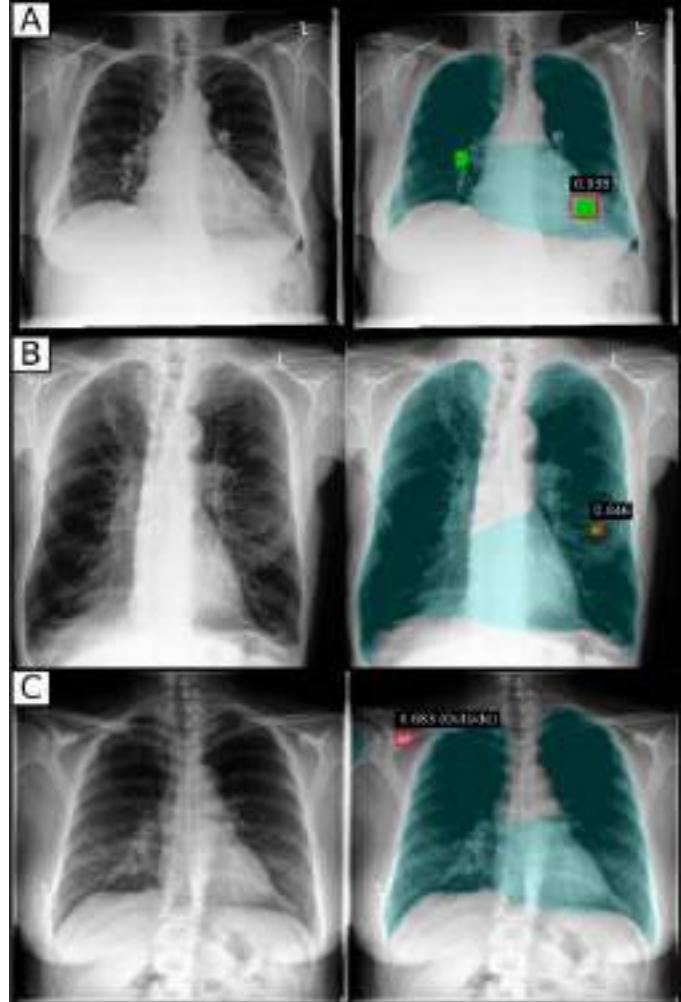


Image source: [nature.com](https://www.nature.com)

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>

The diagram illustrates four different ways to construct a confusion matrix based on the labels '0' and '1'. Each matrix is a 2x2 grid where the columns represent 'Actual Label' and the rows represent 'Predicted Label'.

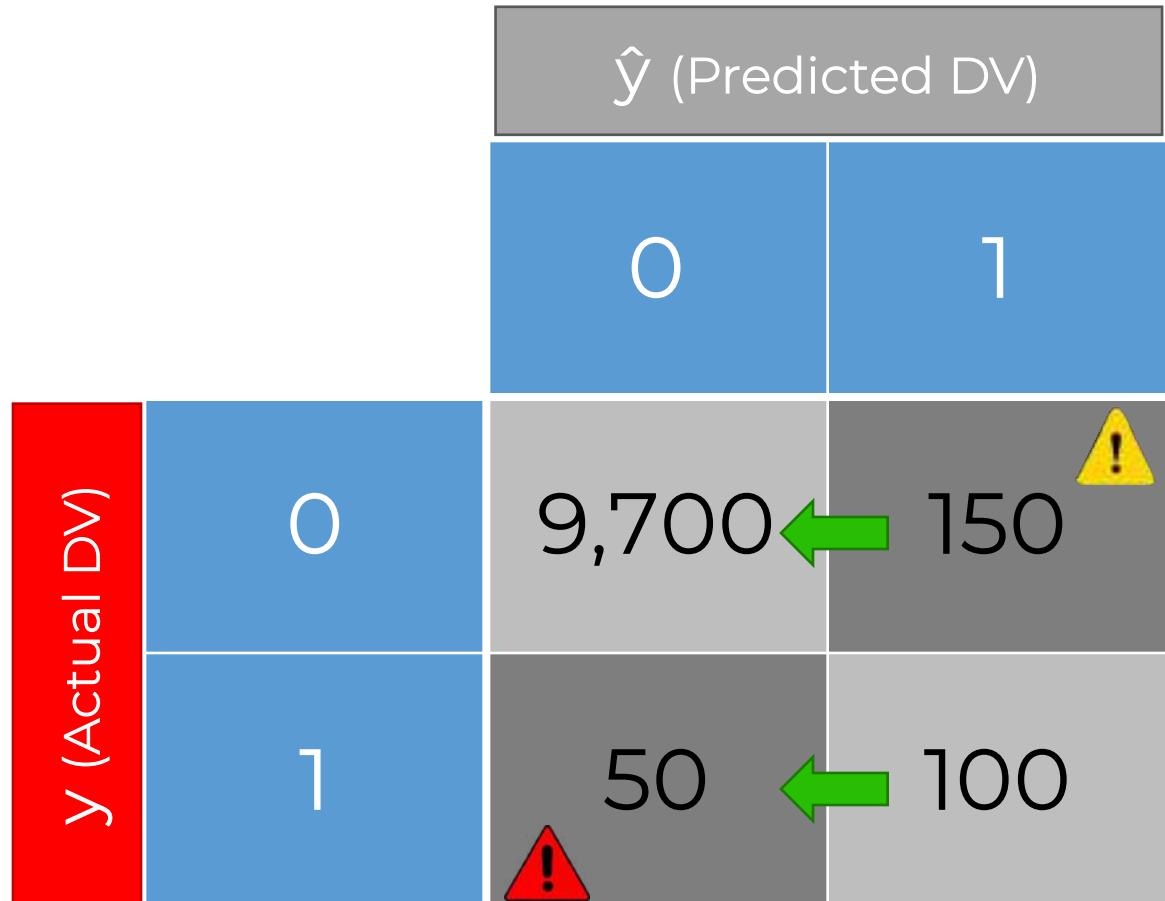
- A)** Actual Label | 1 0
Predicted Label | 1 TP FP
| 0 FN TN
- B)** Actual Label | 0 1
Predicted Label | 0 TN FN
| 1 FP TP
- C)** Predicted Label | 1 0
Actual Label | 1 TP FN
| 0 FP TN
- D)** Predicted Label | 0 1
Actual Label | 0 TN FP
| 1 FN TP

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



Accuracy Paradox

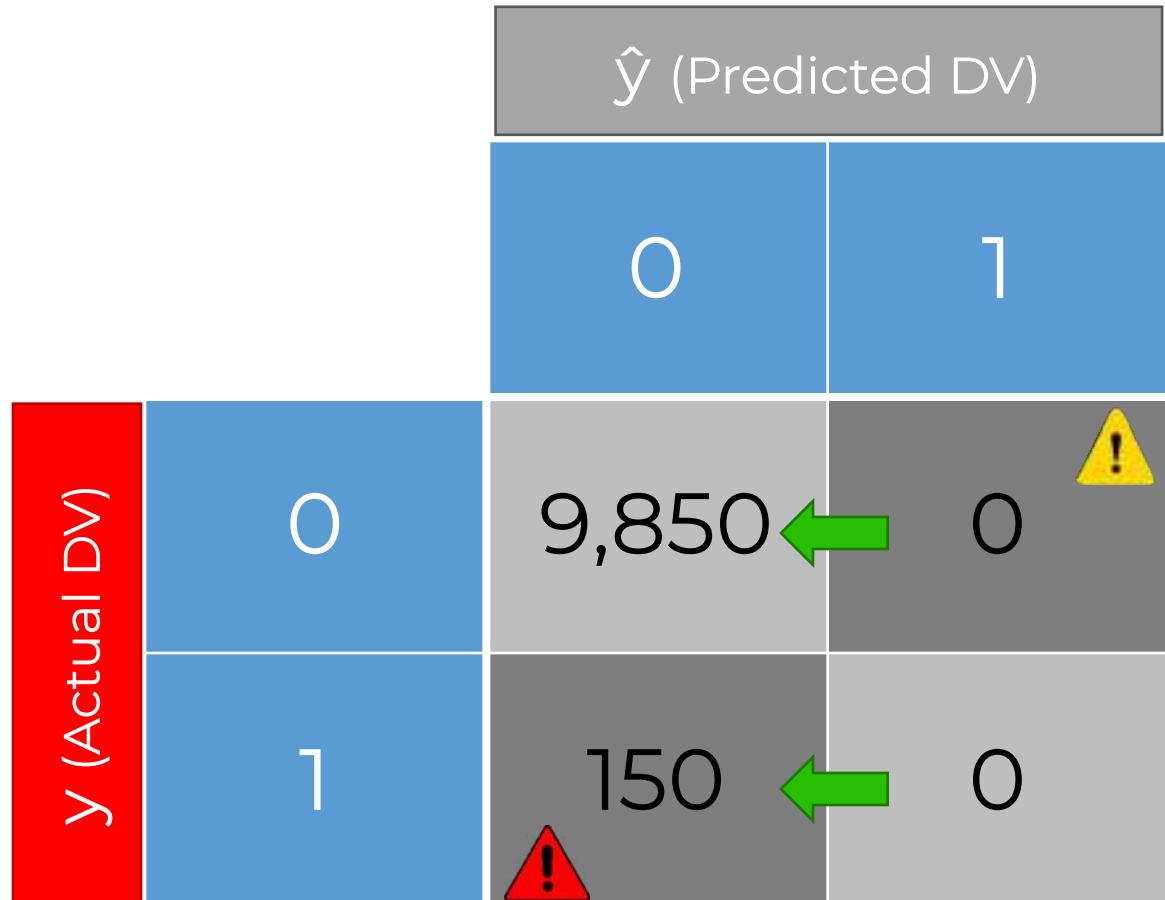
Accuracy Paradox



Scenario 1:

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

Accuracy Paradox



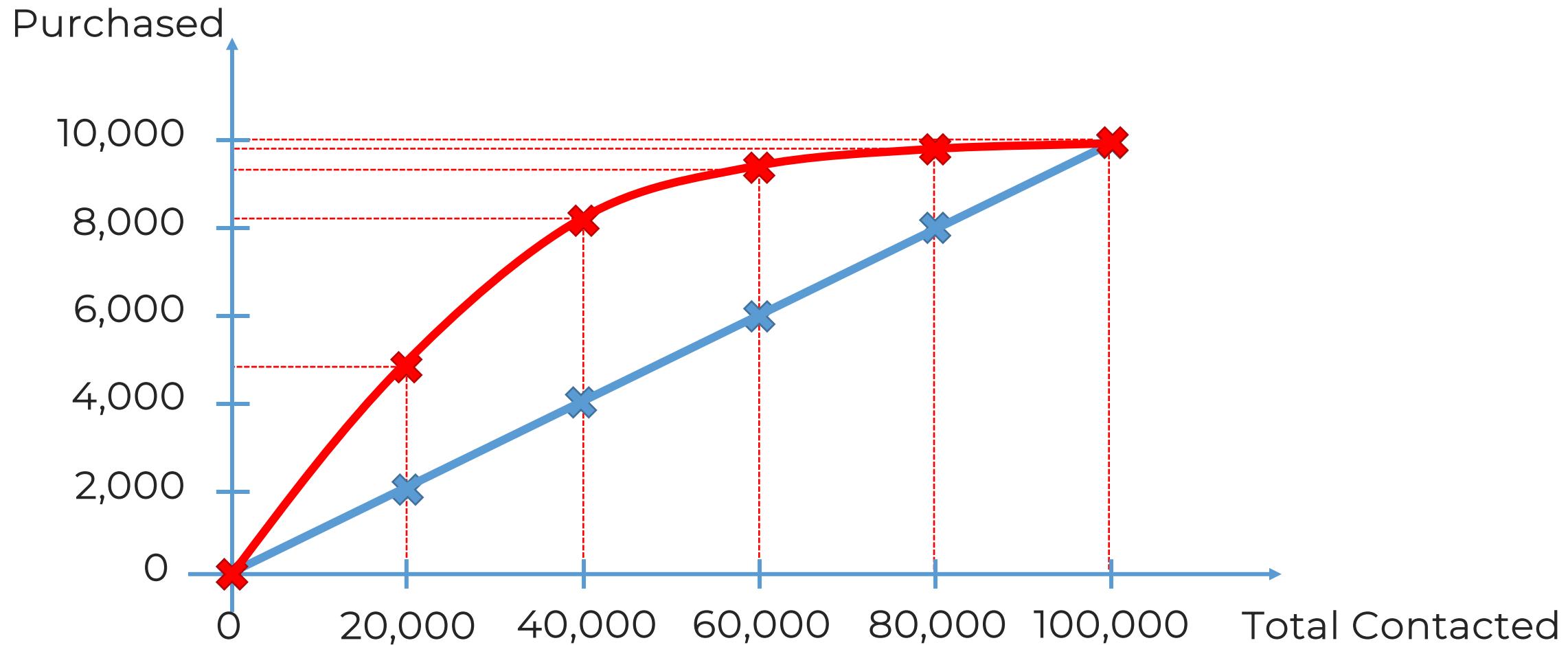
Scenario 1:

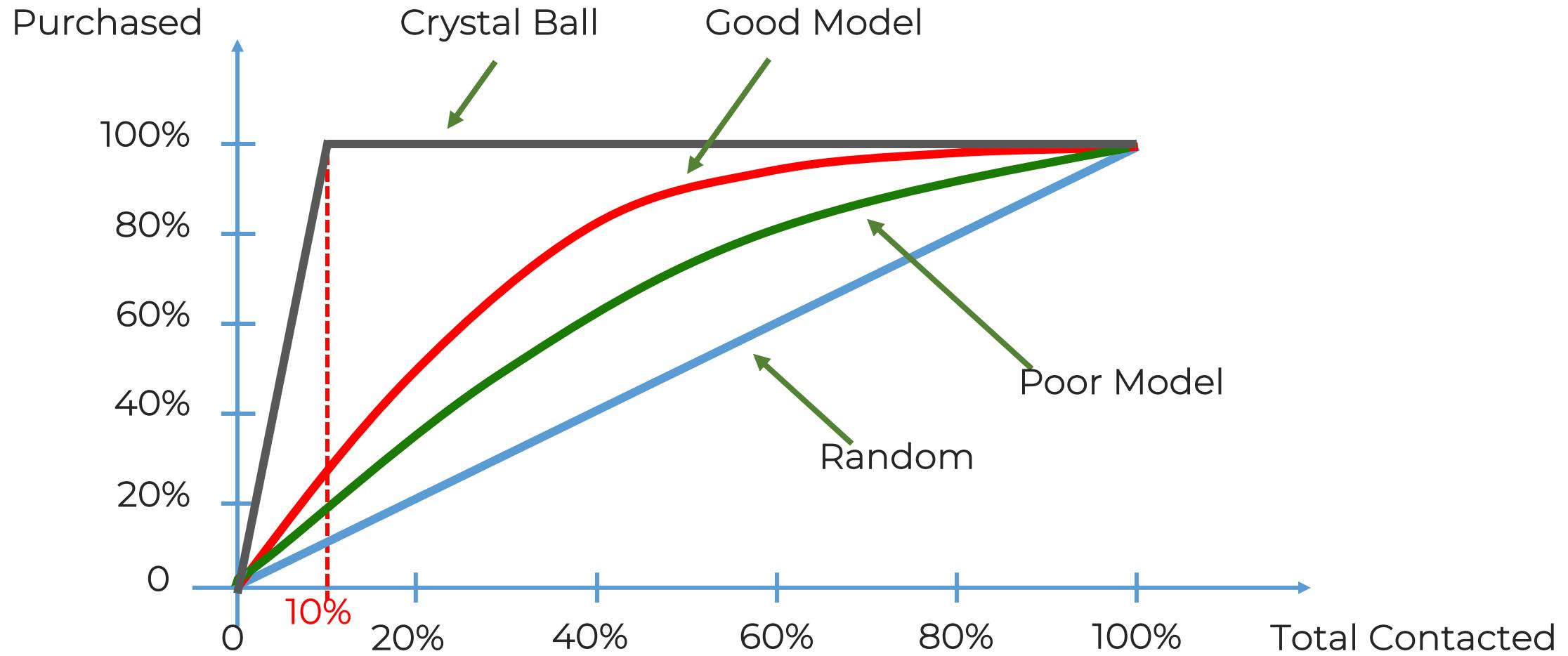
Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

Scenario 2:

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

Cumulative Accuracy Profile



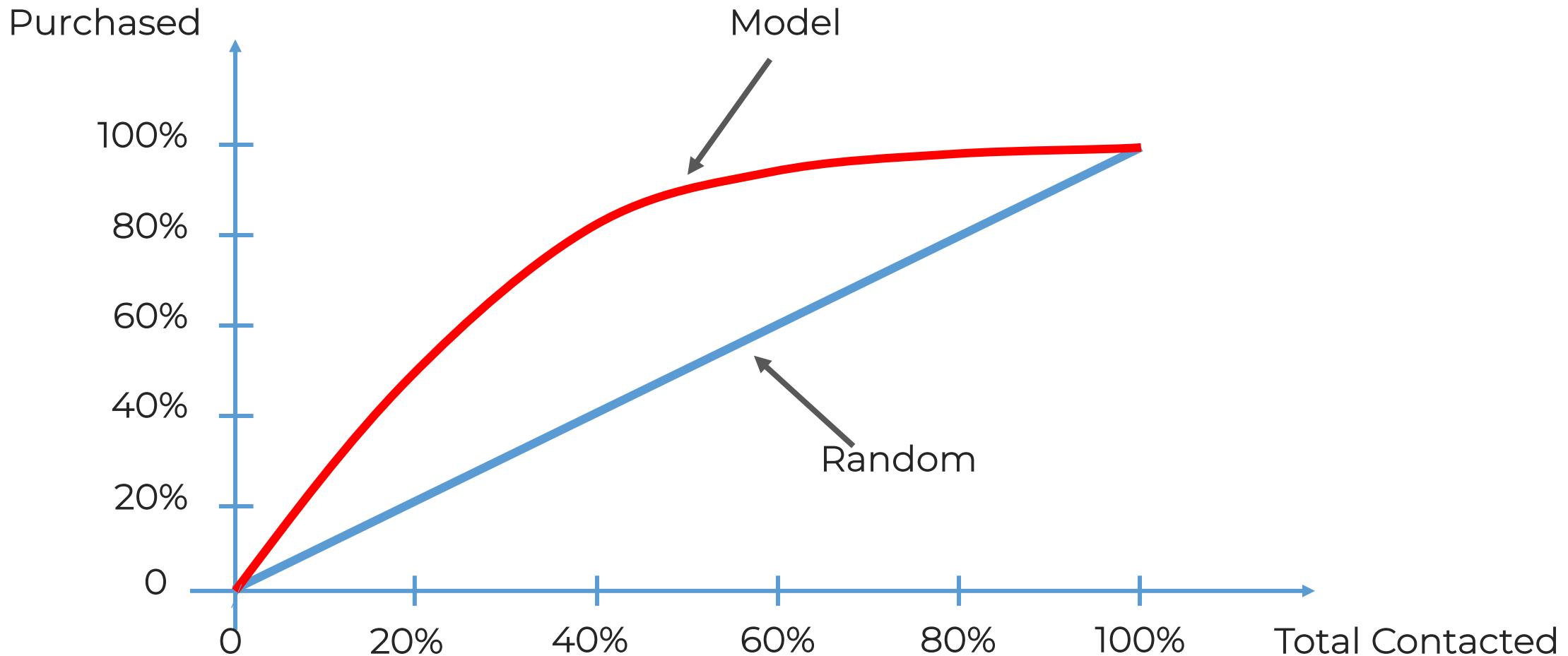


CAP

Note:

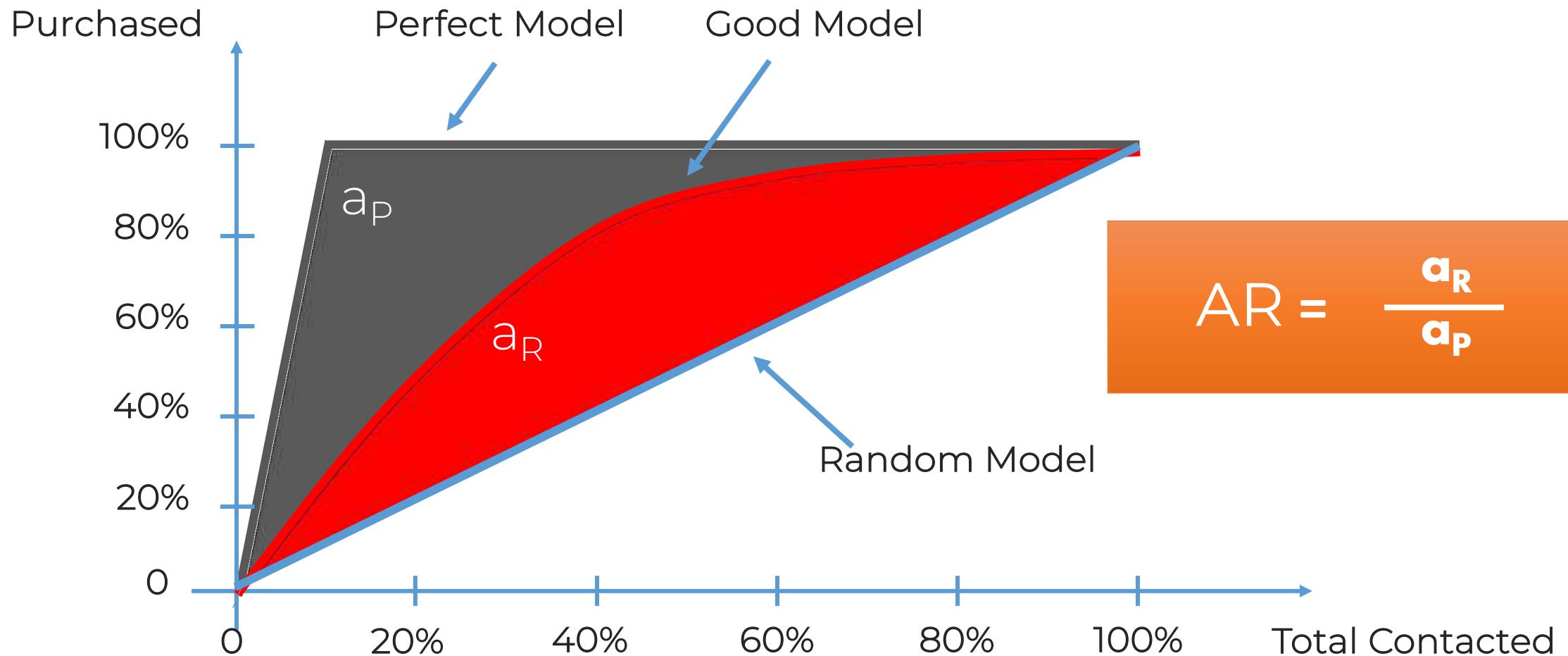
CAP = Cumulative Accuracy Profile

ROC = Receiver Operating Characteristic



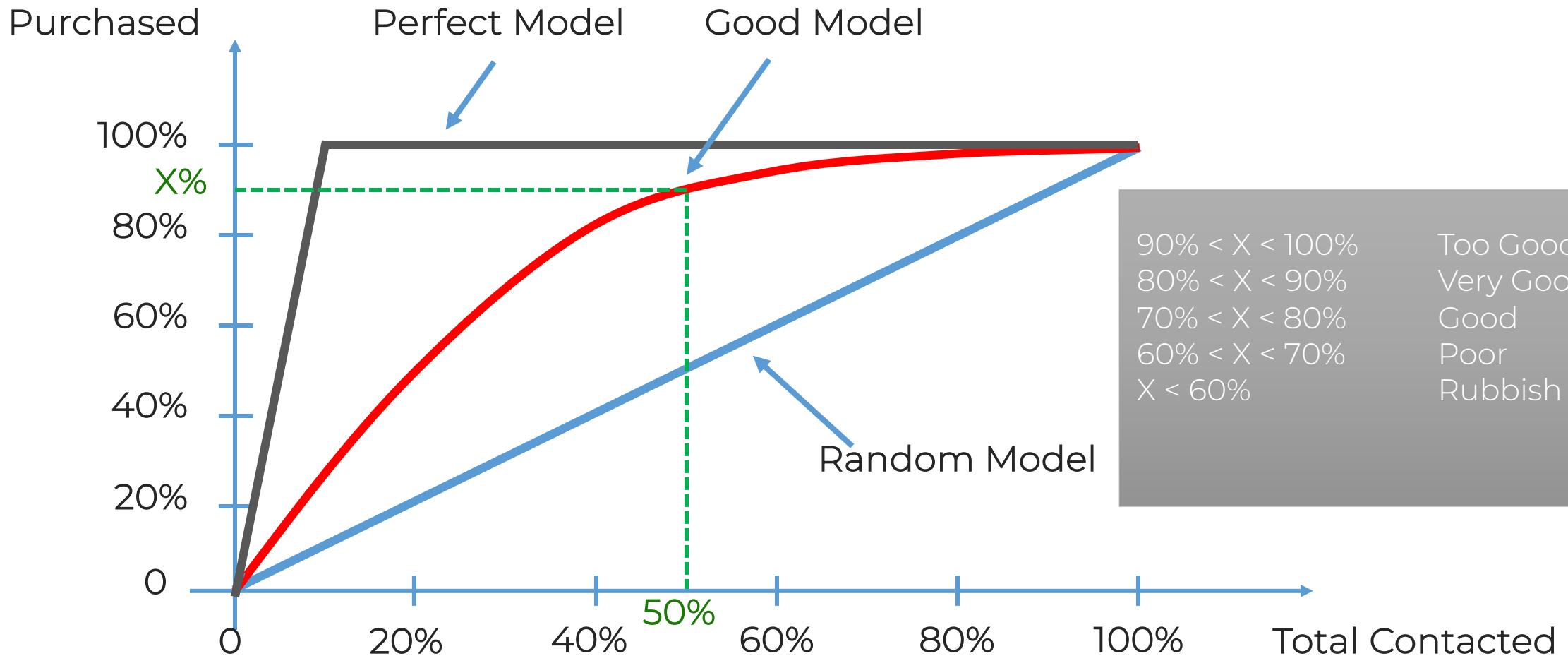
CAP Analysis

CAP Analysis



CAP Analysis

NOT FOR DISTRIBUTION ©



Clustering



What is Clustering?



*Clustering – grouping
unlabelled data*

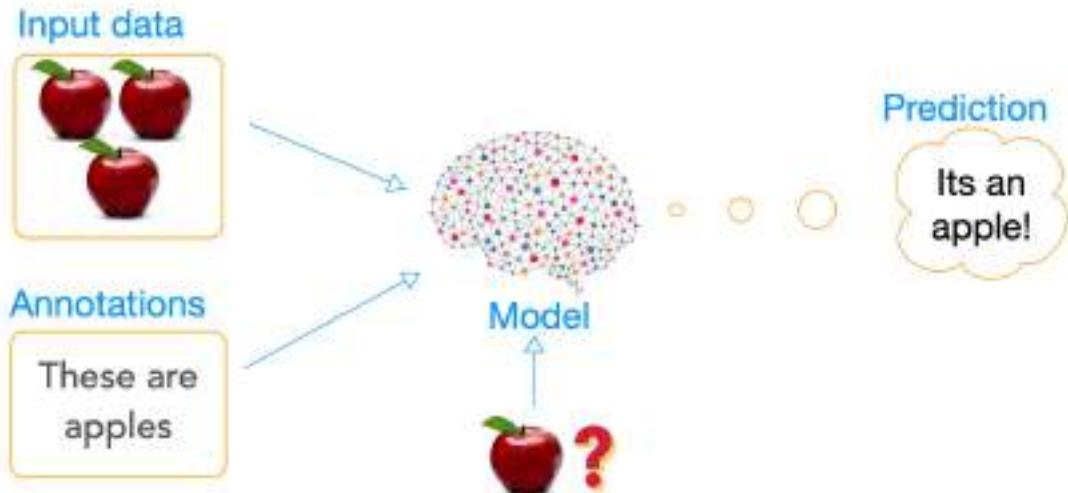
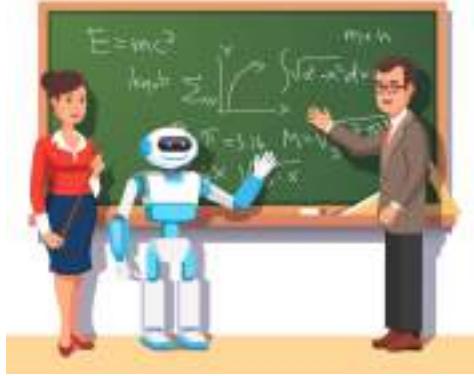




What is Clustering?

Supervised Learning

(e.g. Regression, Classification)



Unsupervised Learning

(e.g. Clustering)

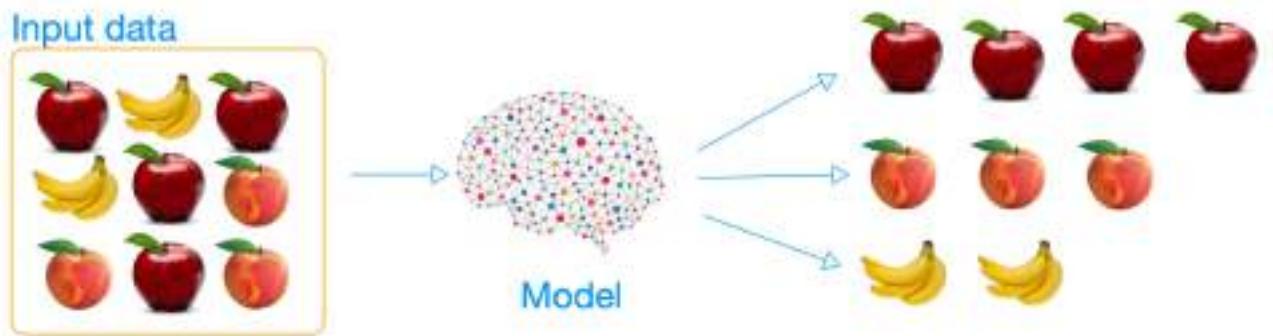
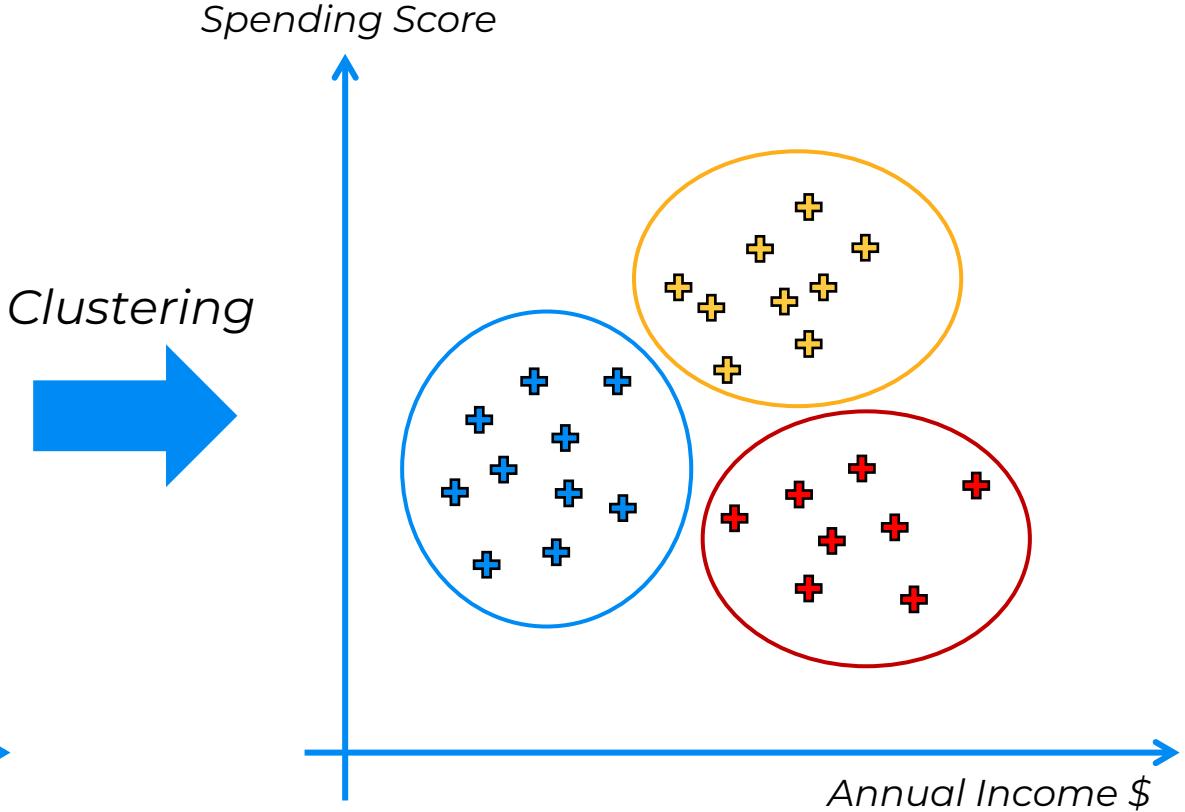
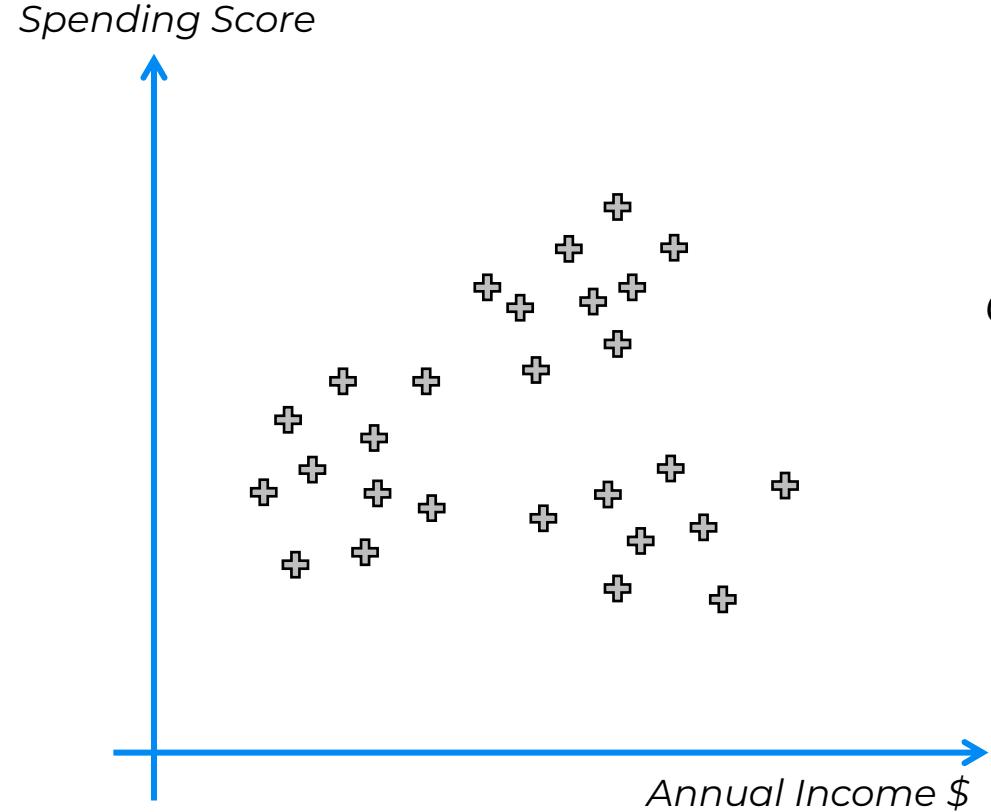


Image source: mdpi.com/2073-8994/10/12/734





What is Clustering?



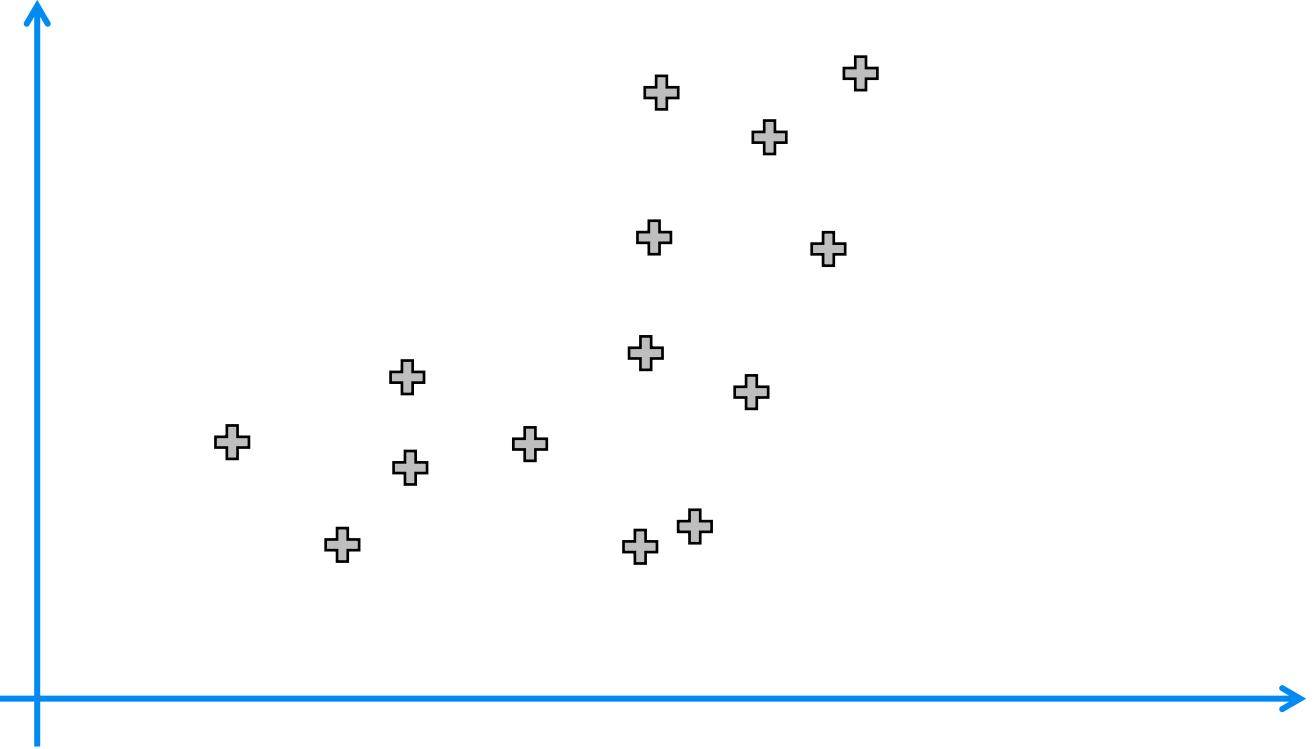
K-Means Clustering





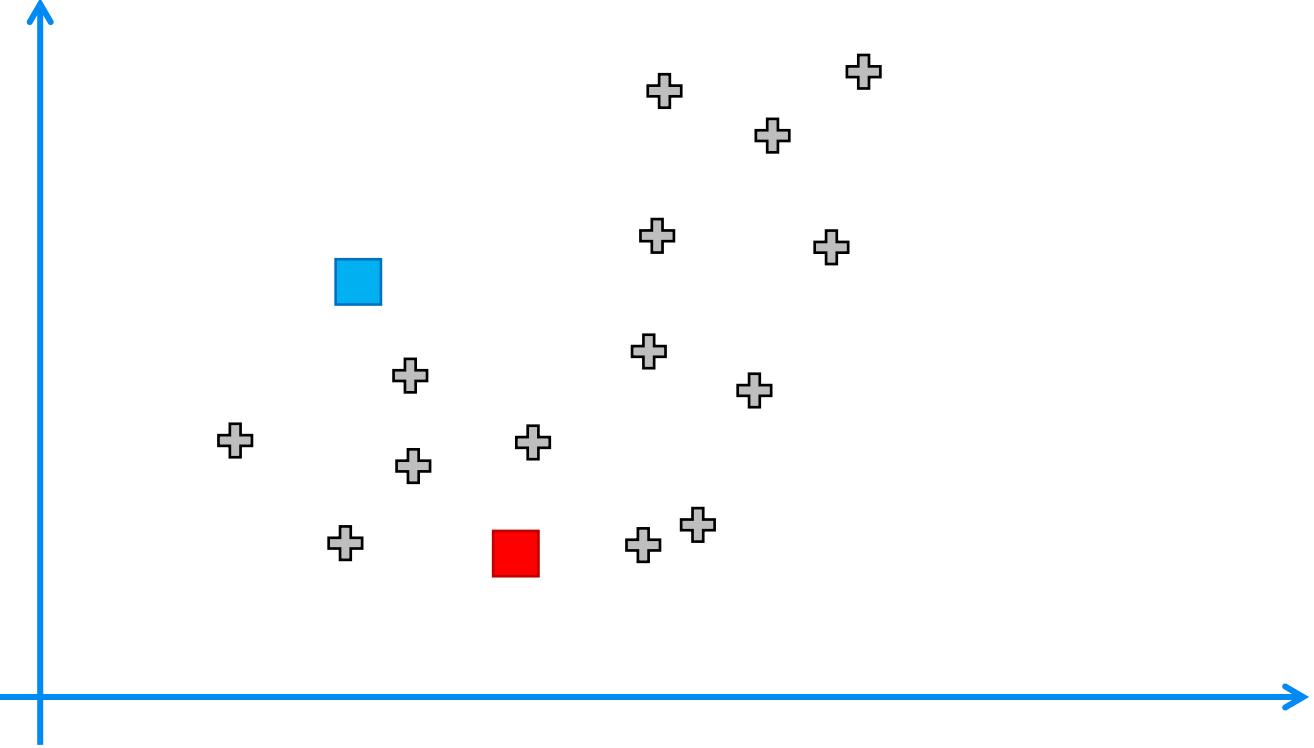
K-Means Clustering

NOT FOR DISTRIBUTION © SUPERDATASCIENCE www.superdatascience.com



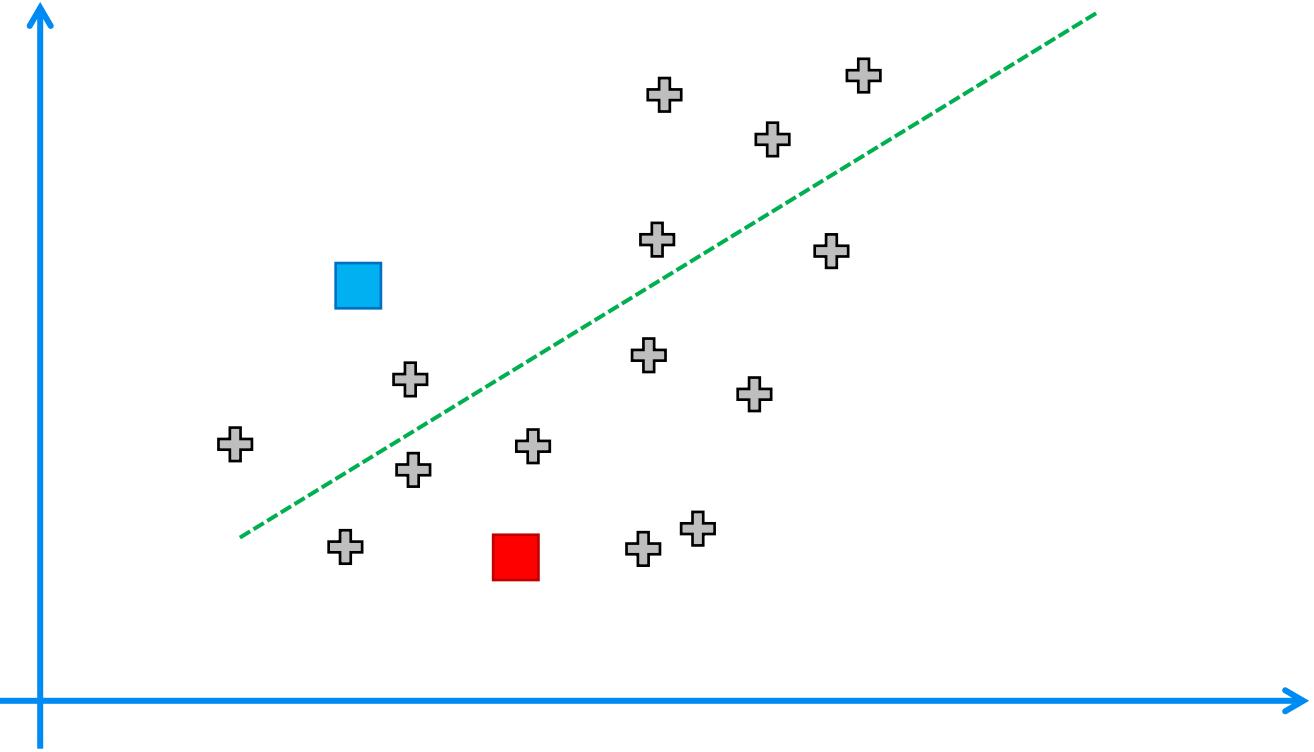


K-Means Clustering



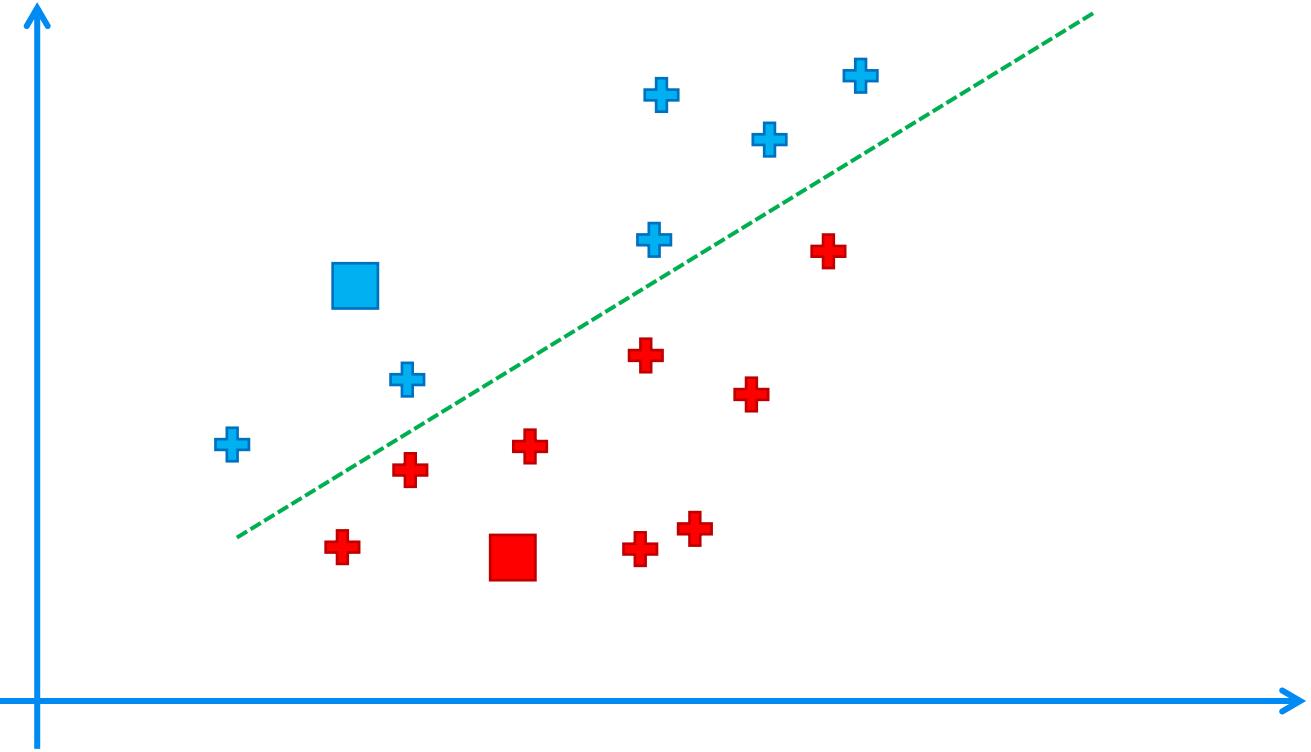


K-Means Clustering



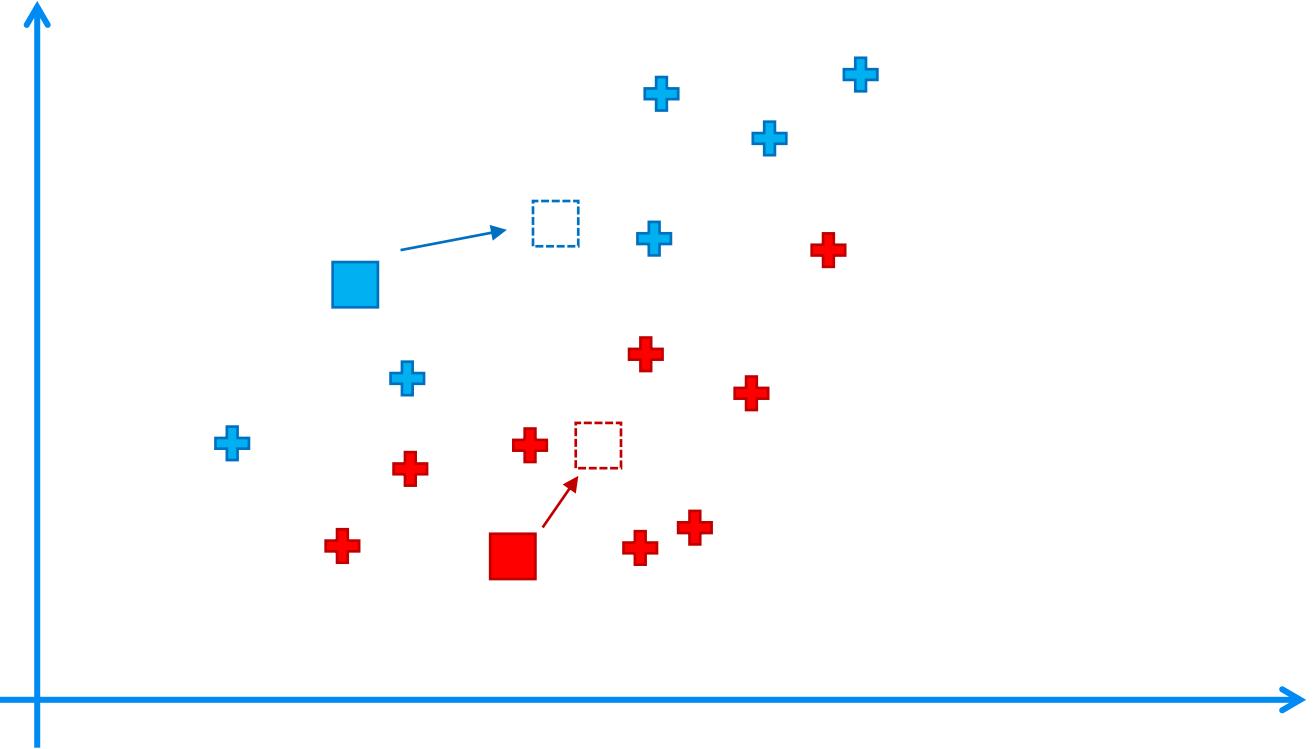


K-Means Clustering



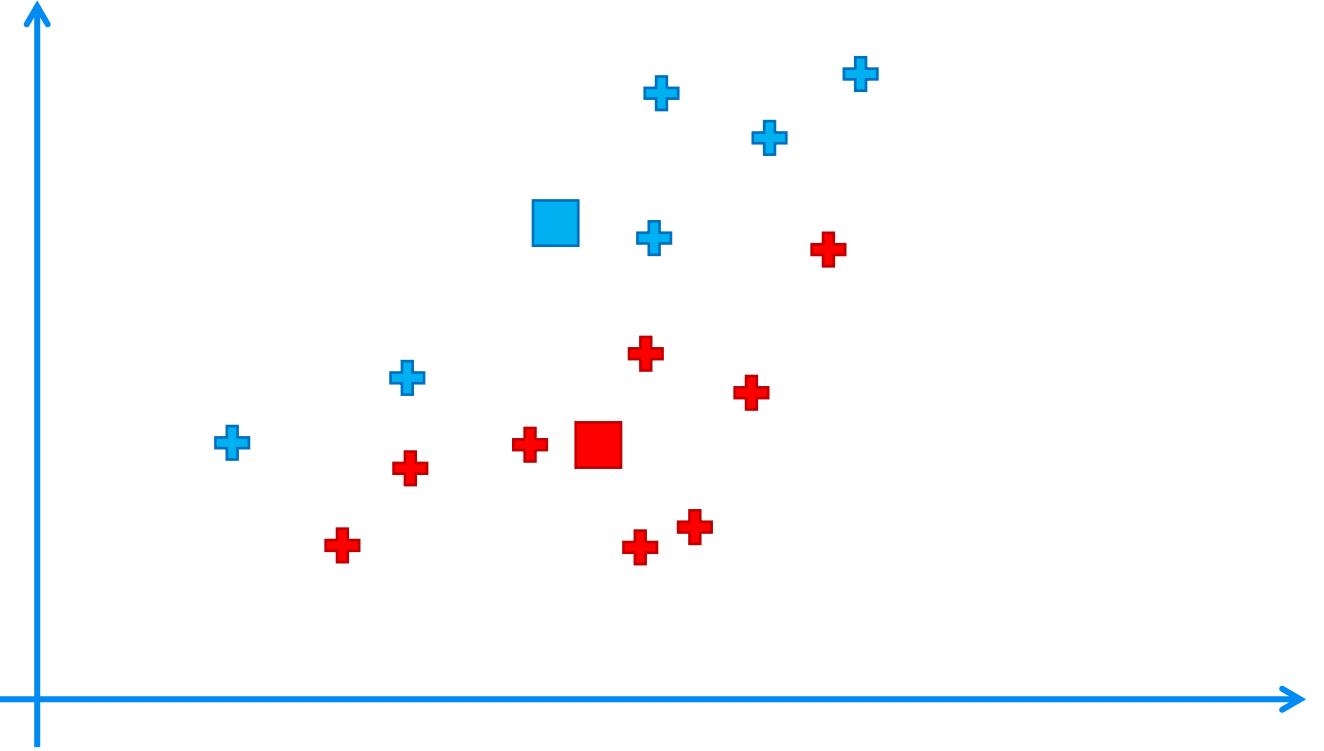


K-Means Clustering



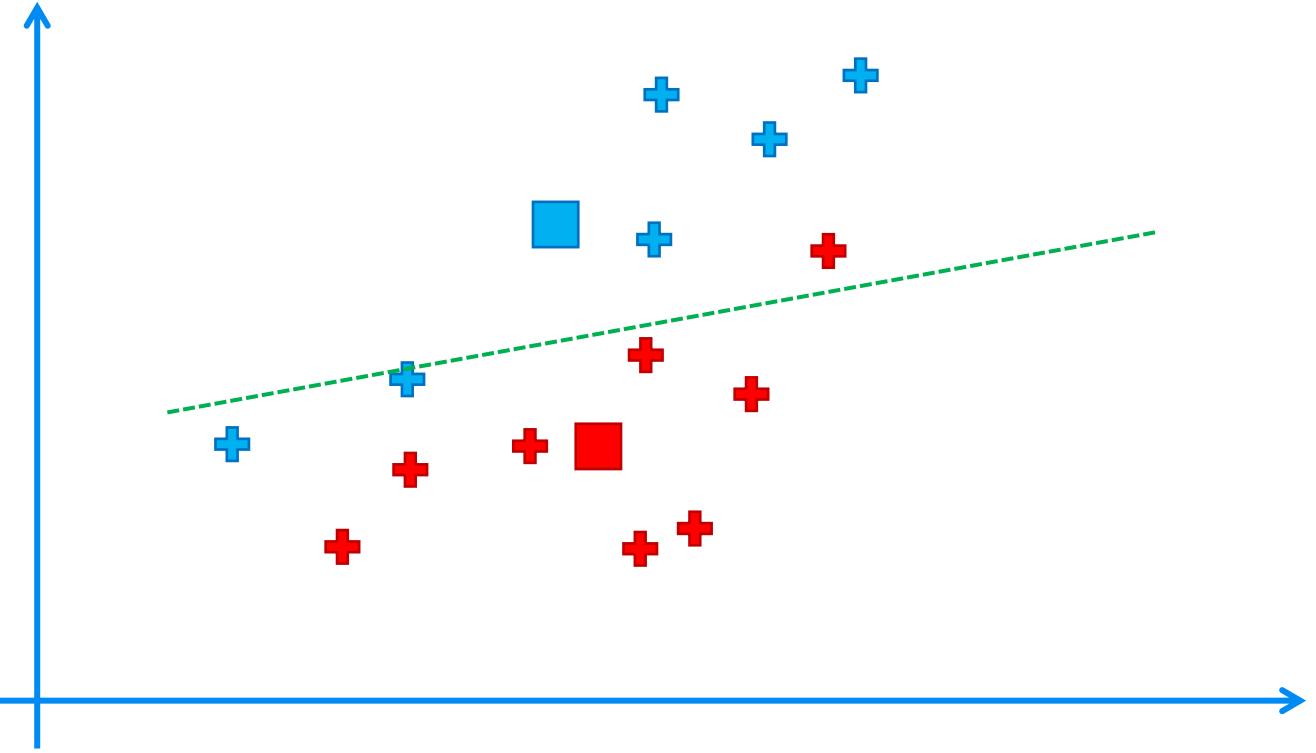


K-Means Clustering



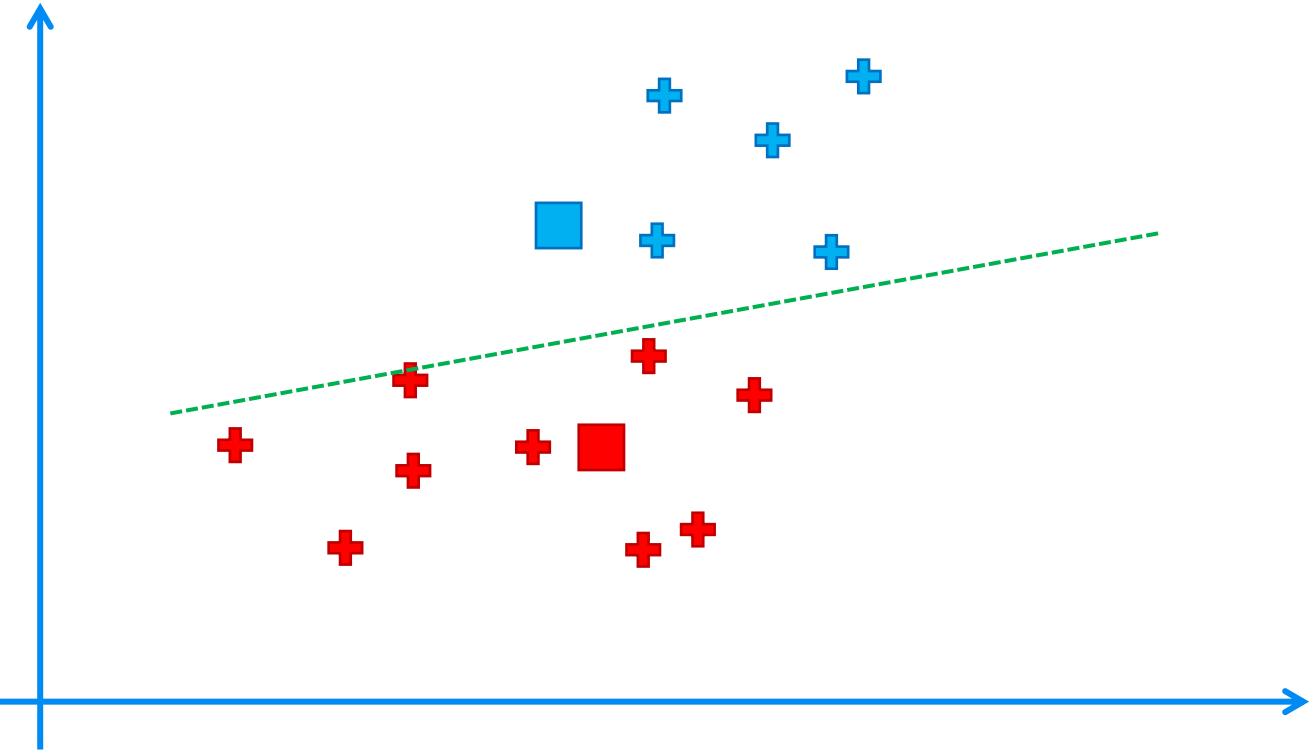


K-Means Clustering



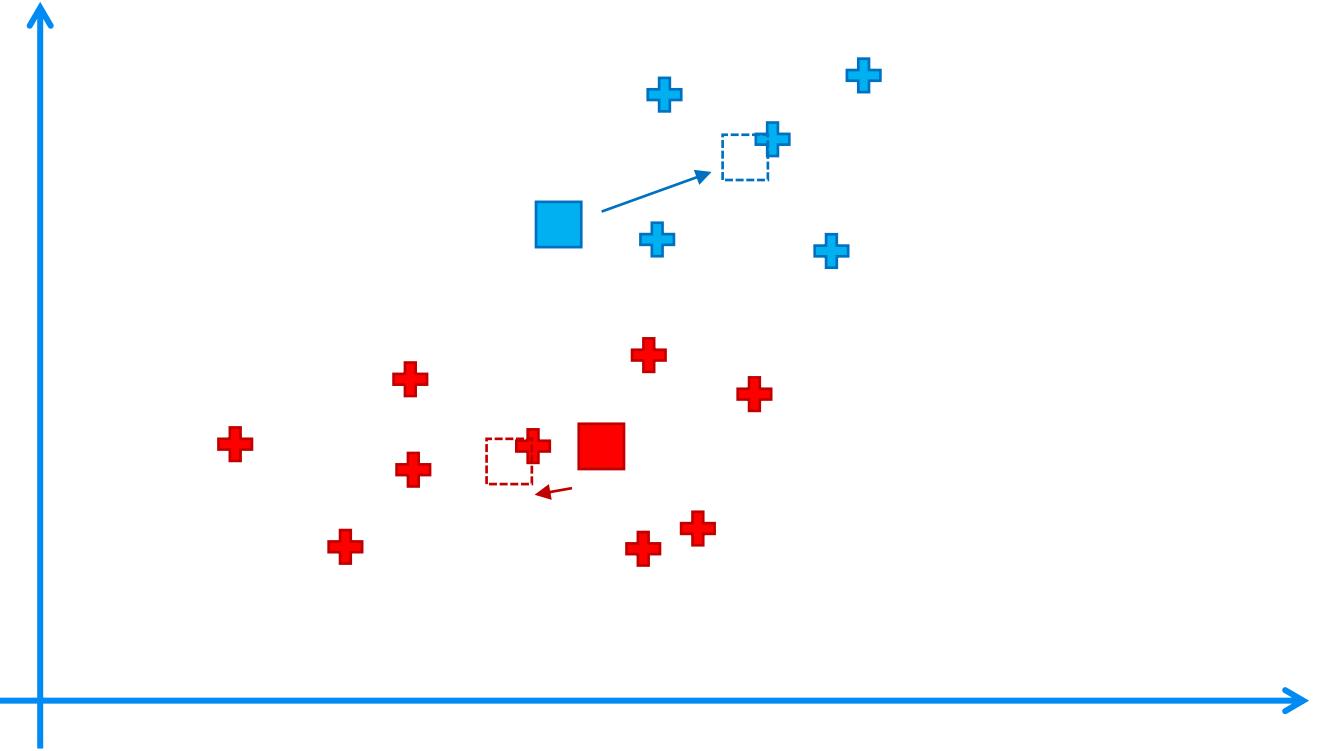


K-Means Clustering





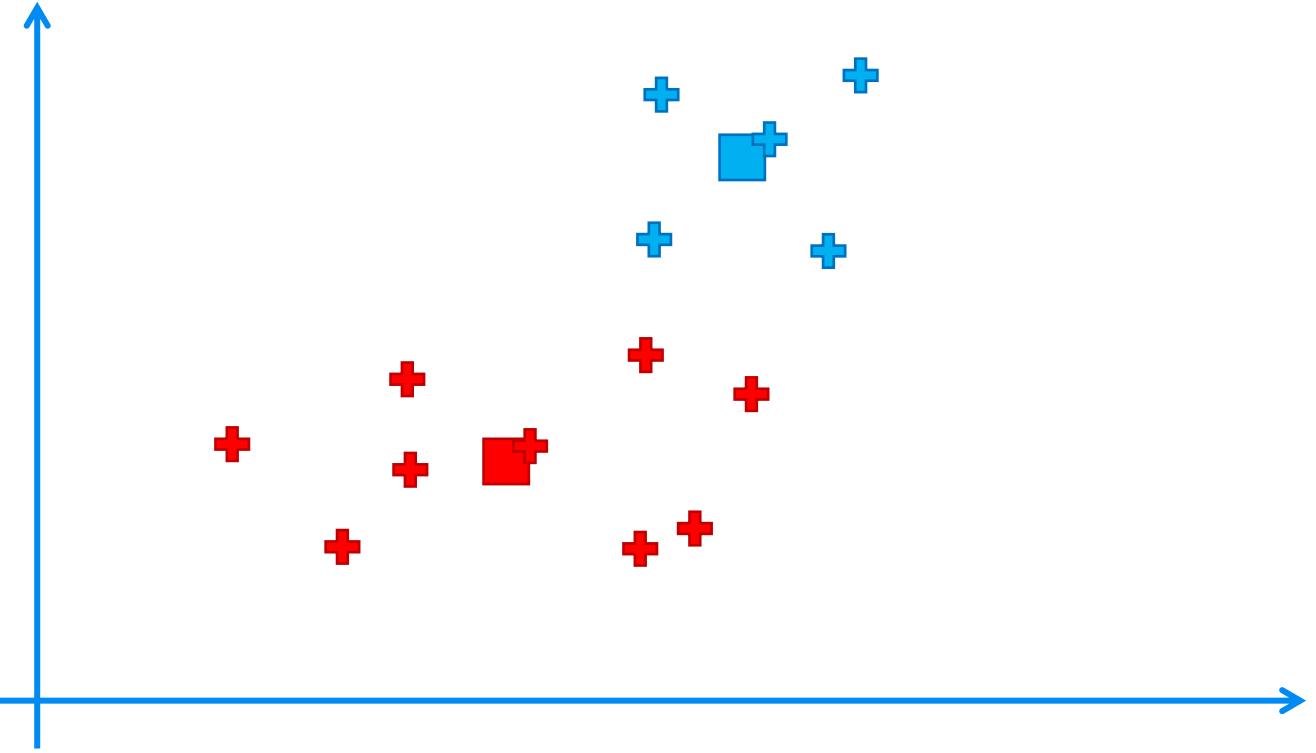
K-Means Clustering





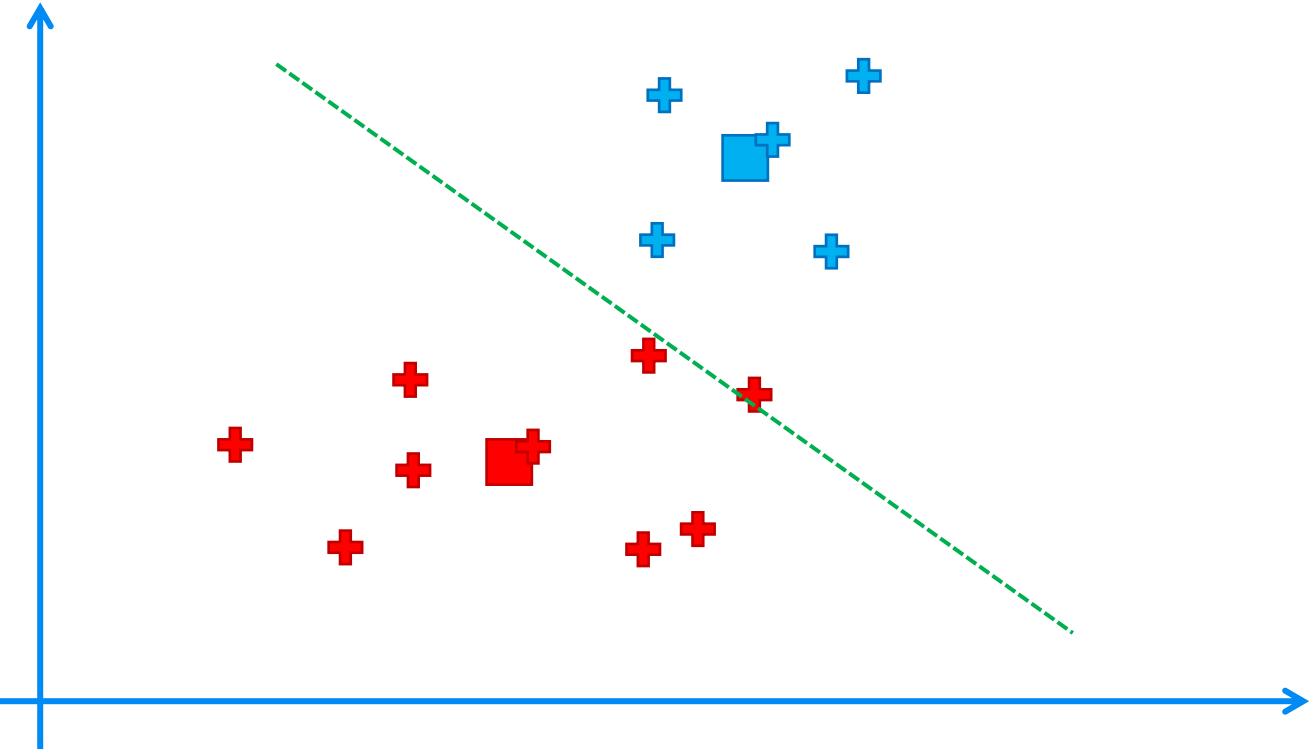
K-Means Clustering

NOT FOR DISTRIBUTION © SUPERDATASCIENCE www.superdatascience.com



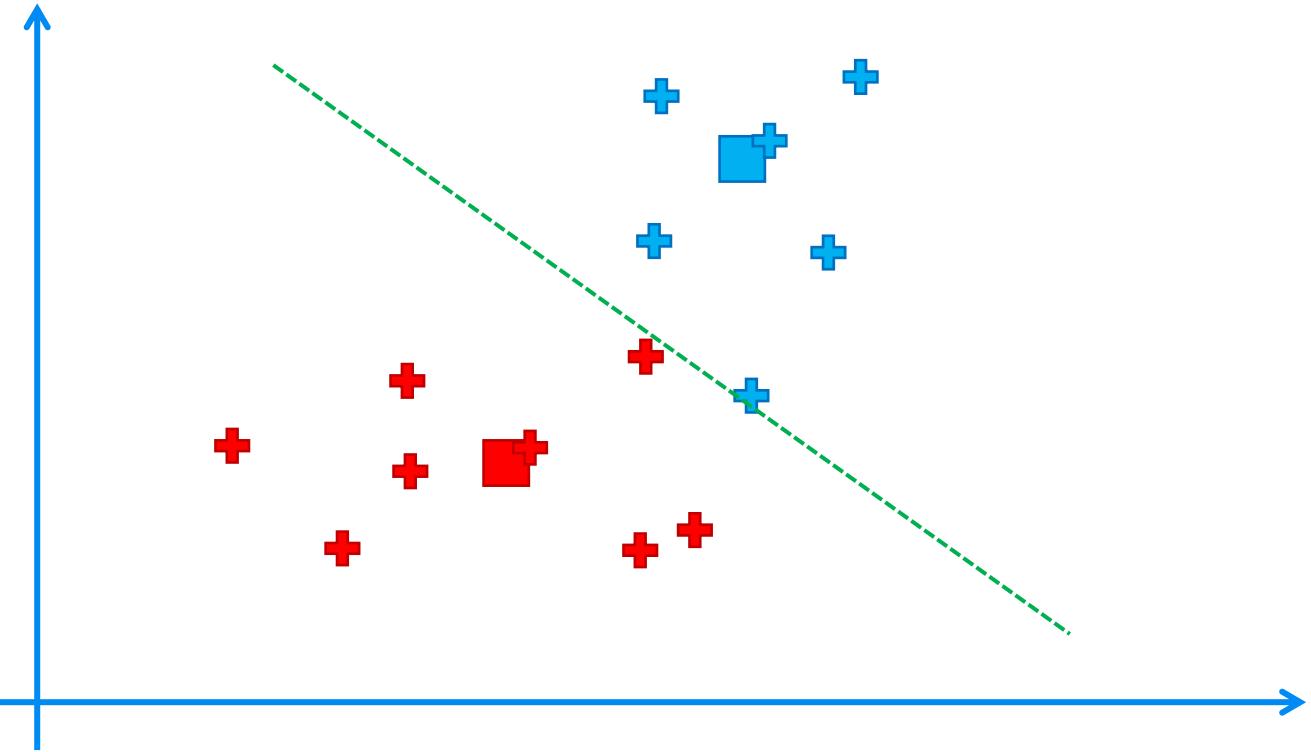


K-Means Clustering





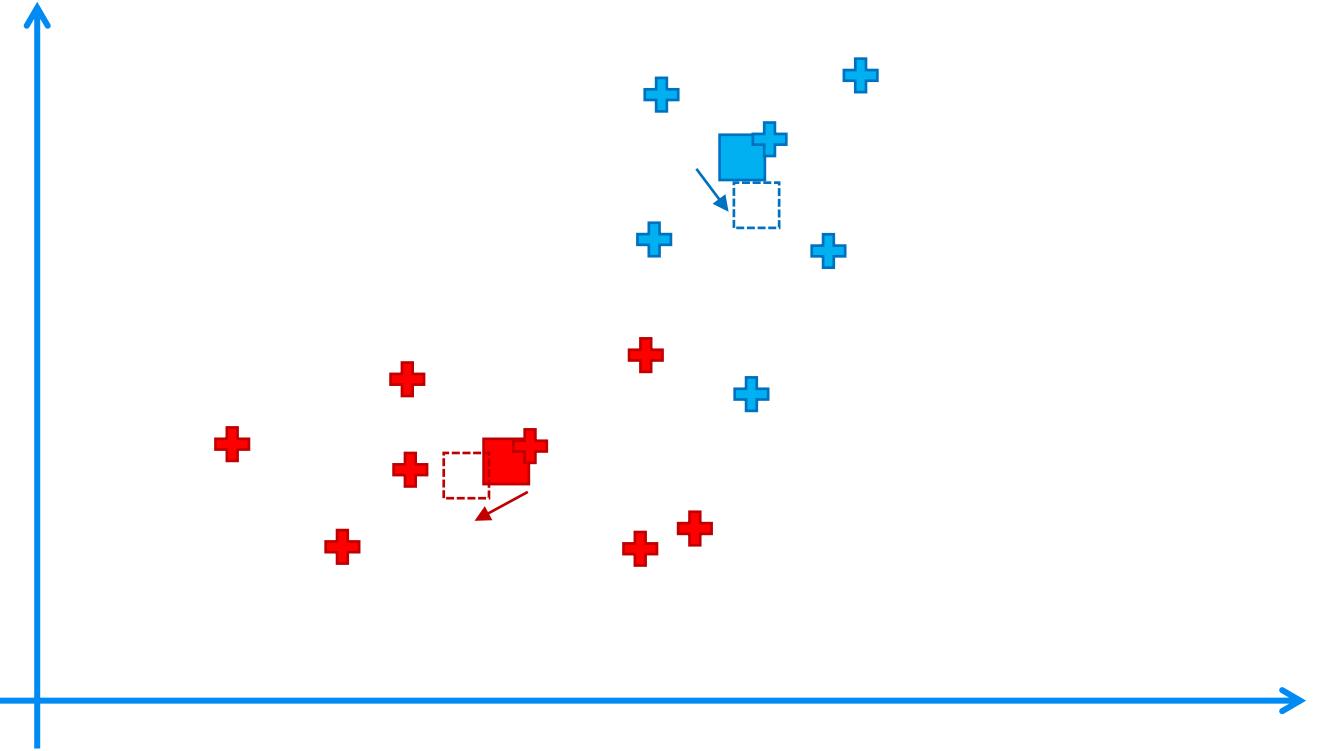
K-Means Clustering



K-Means Clustering

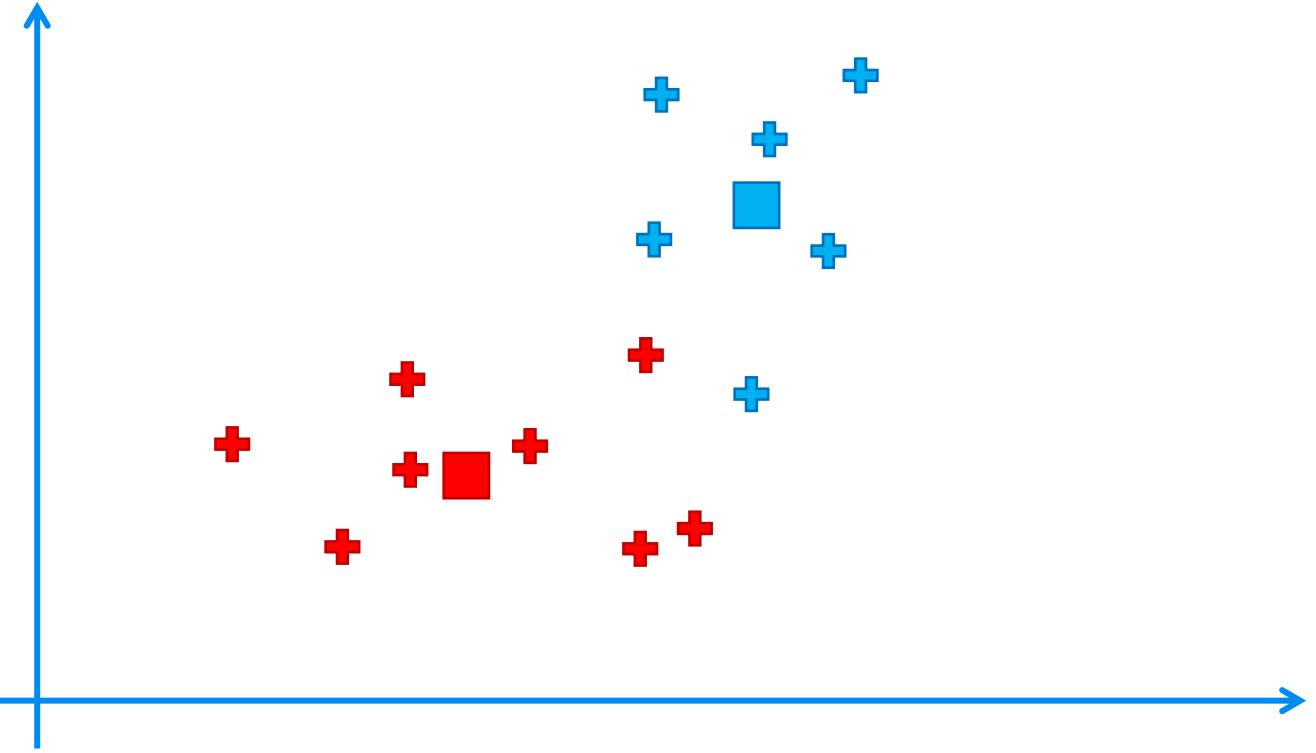


NOT FOR DISTRIBUTION © SUPERDATASCIENCE www.superdatascience.com



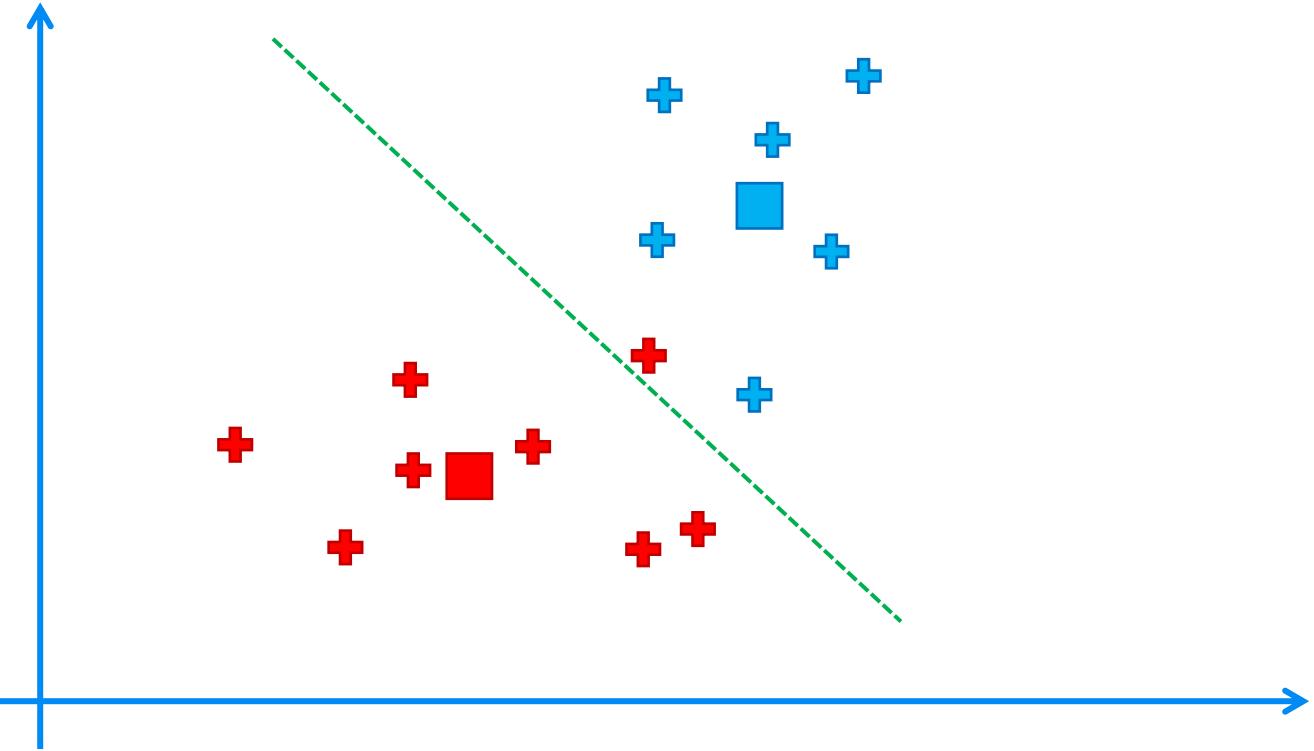


K-Means Clustering



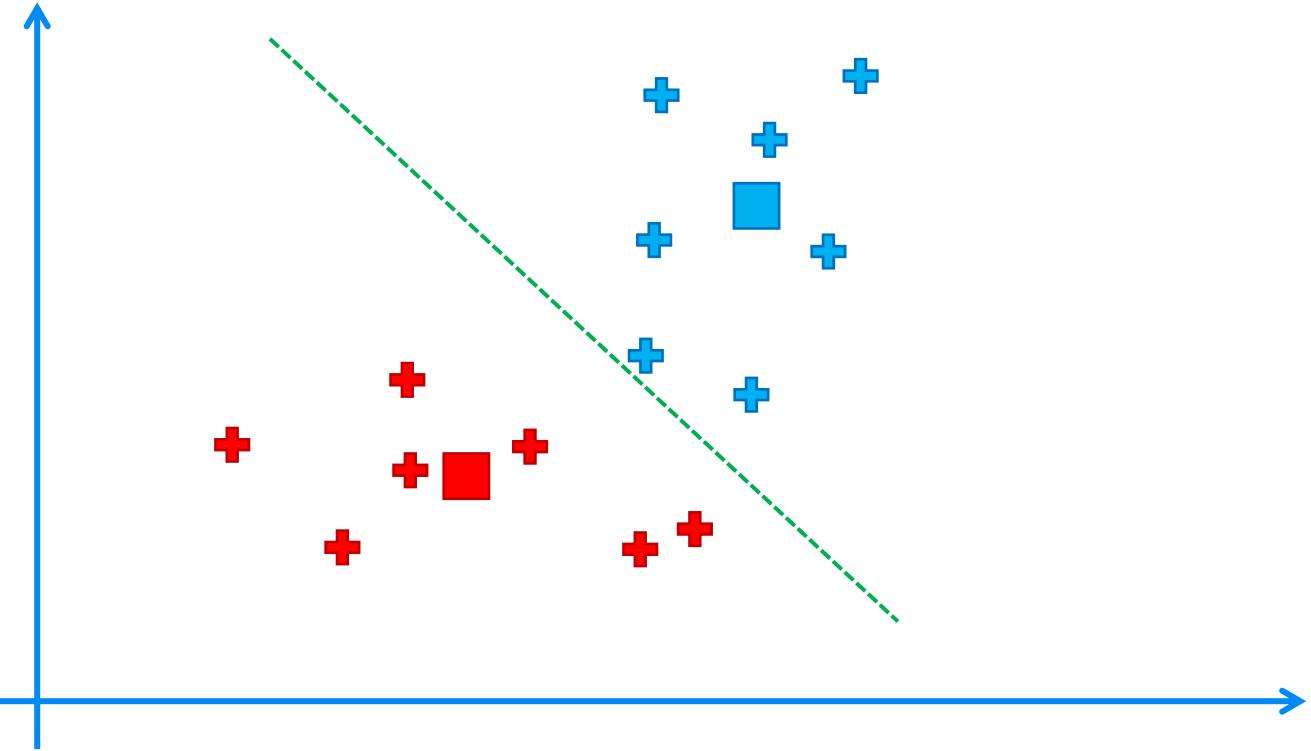


K-Means Clustering





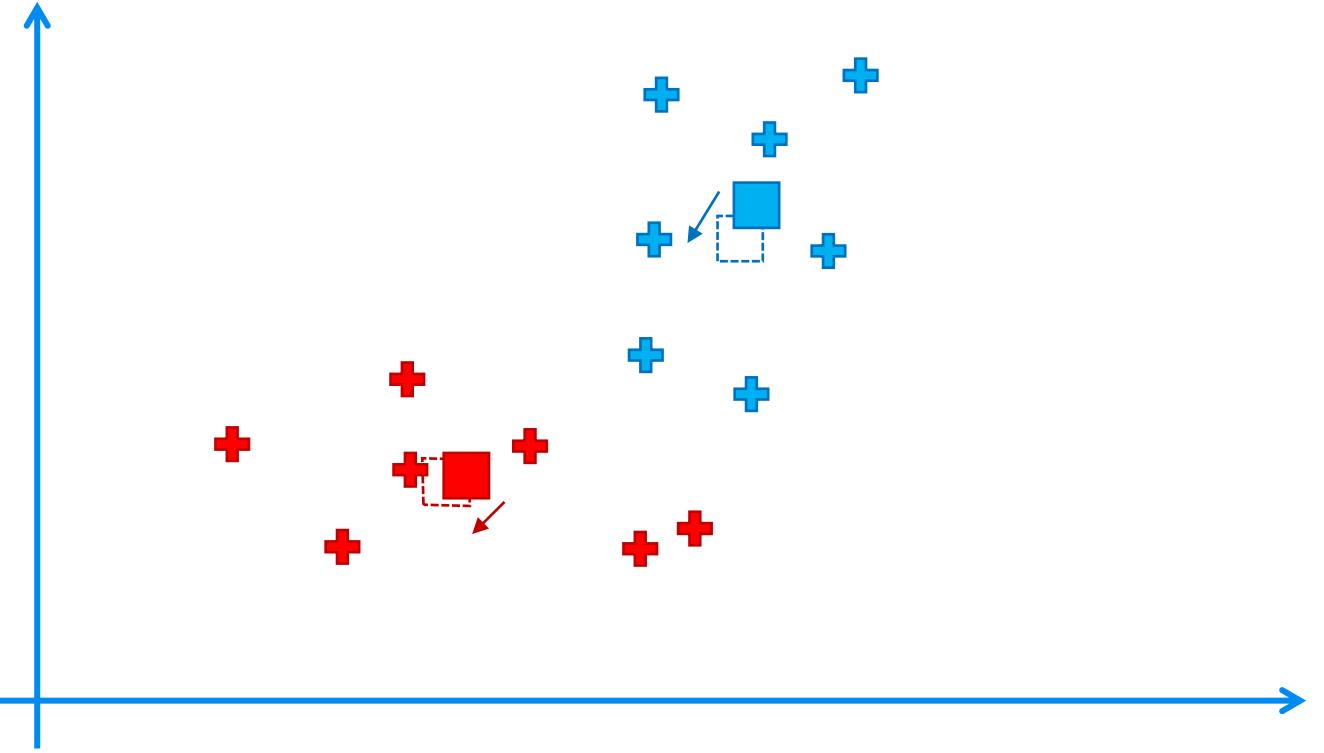
K-Means Clustering



K-Means Clustering

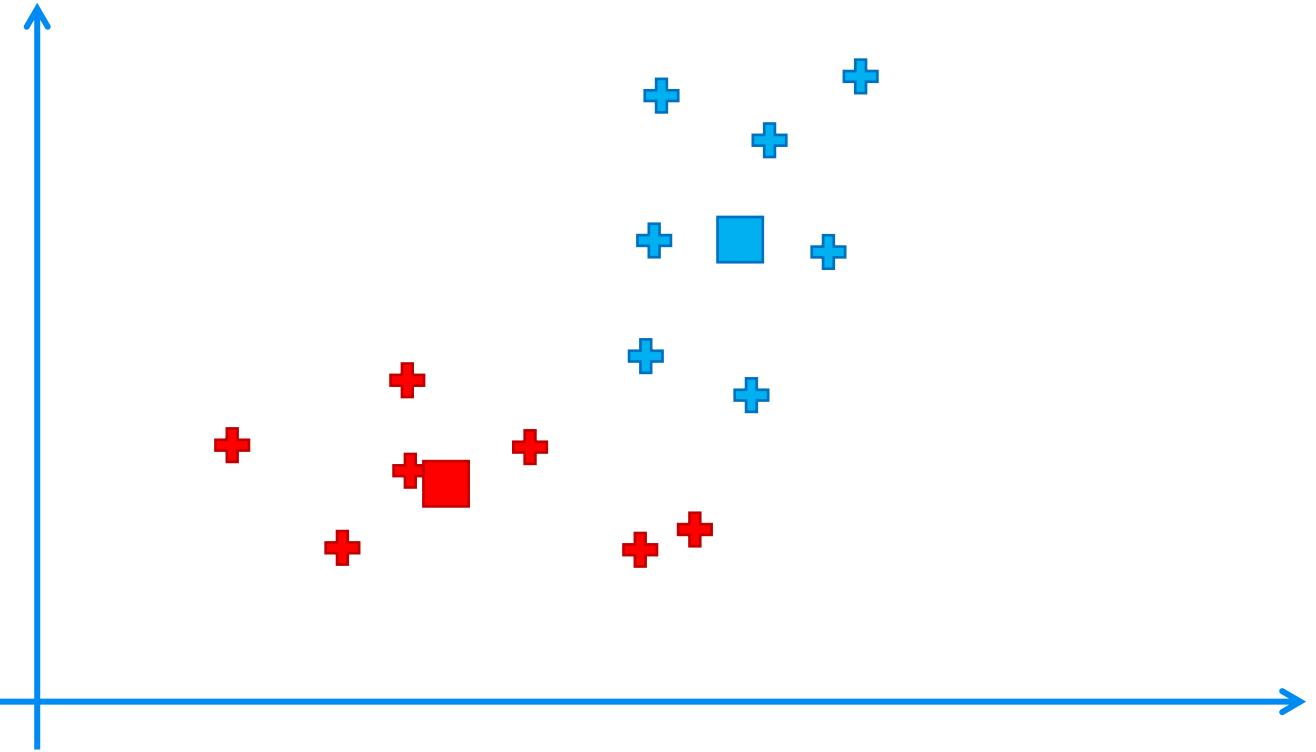


NOT FOR DISTRIBUTION © SUPERDATASCIENCE www.superdatascience.com



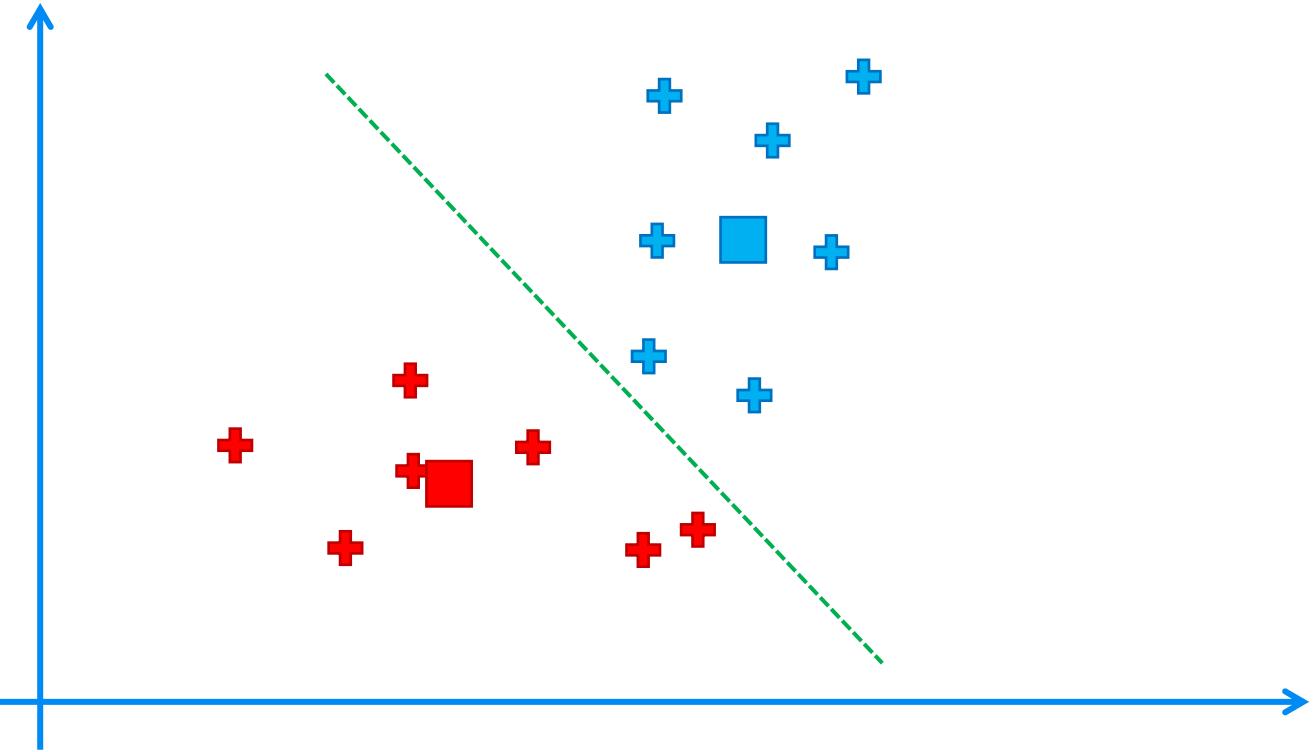


K-Means Clustering



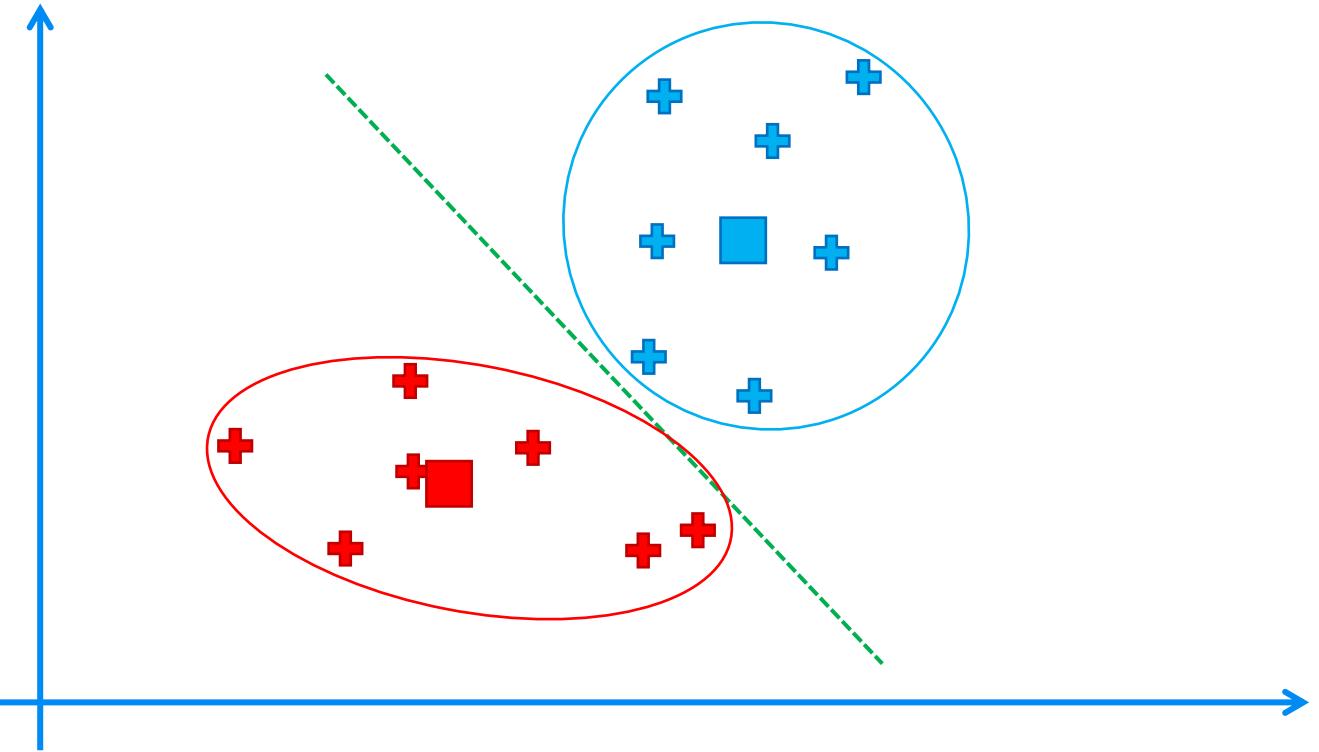


K-Means Clustering



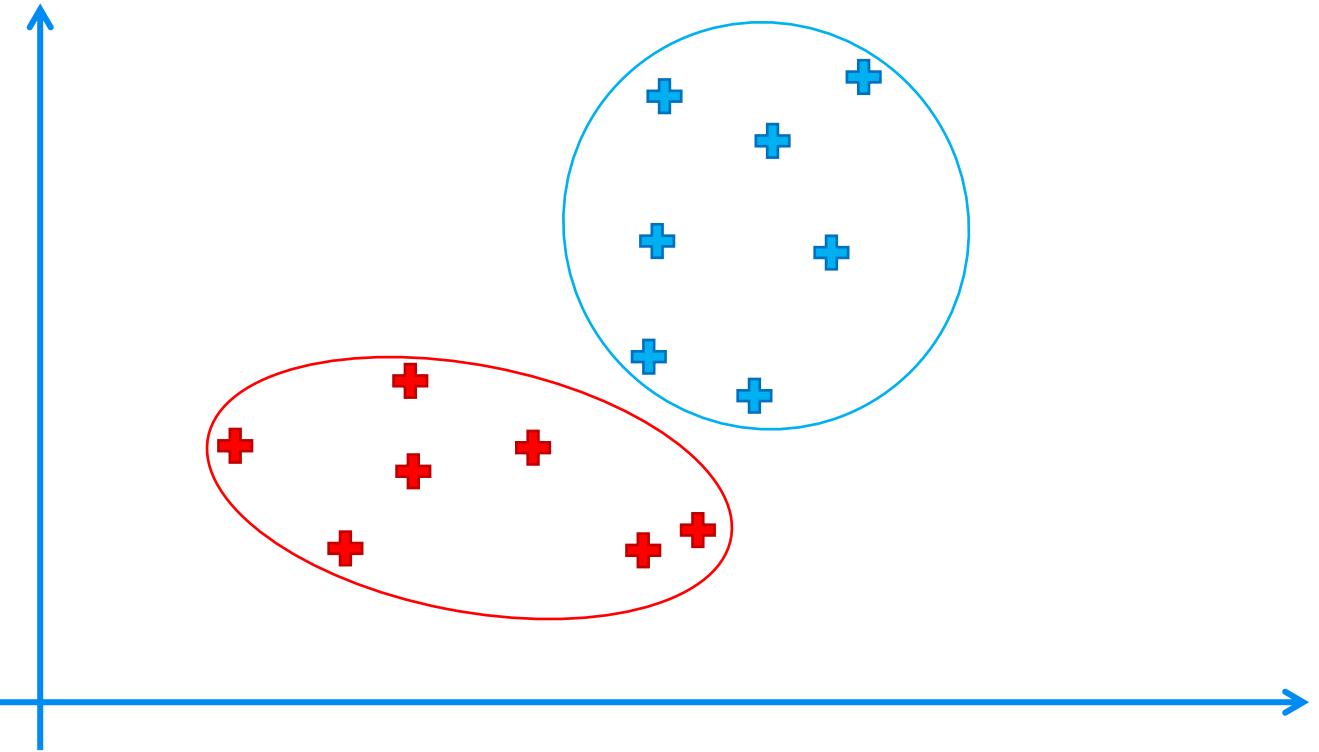


K-Means Clustering





K-Means Clustering

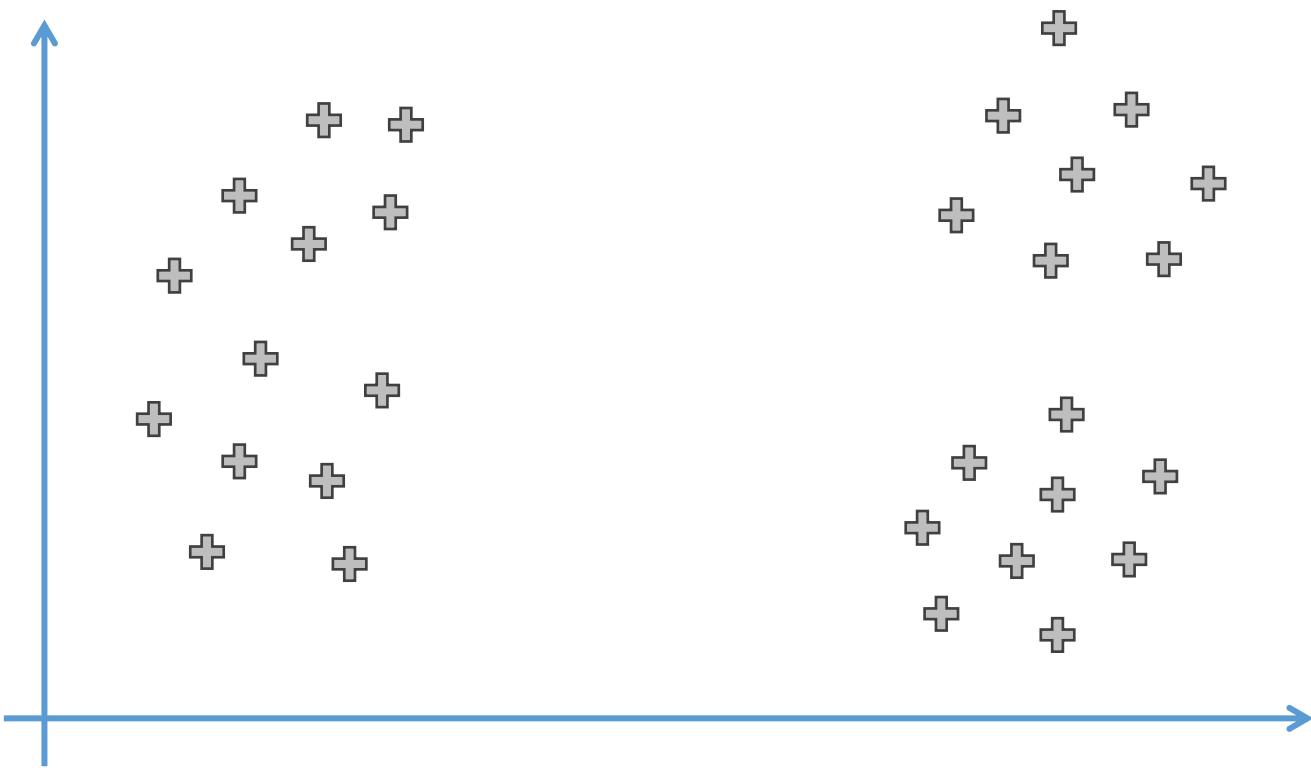


The Elbow Method





The Elbow Method





The Elbow Method

Within Cluster Sum of Squares:

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



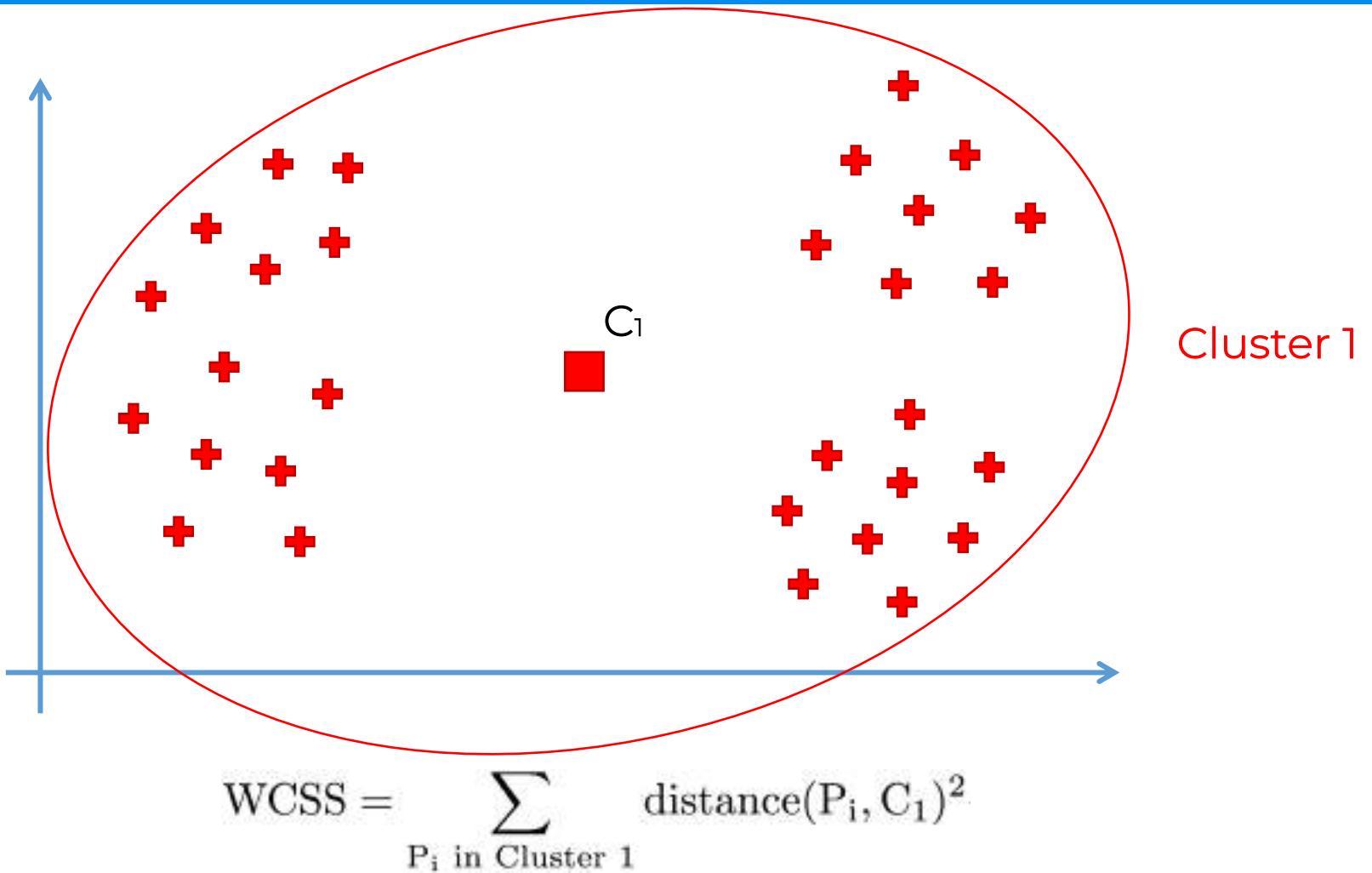


The Elbow Method



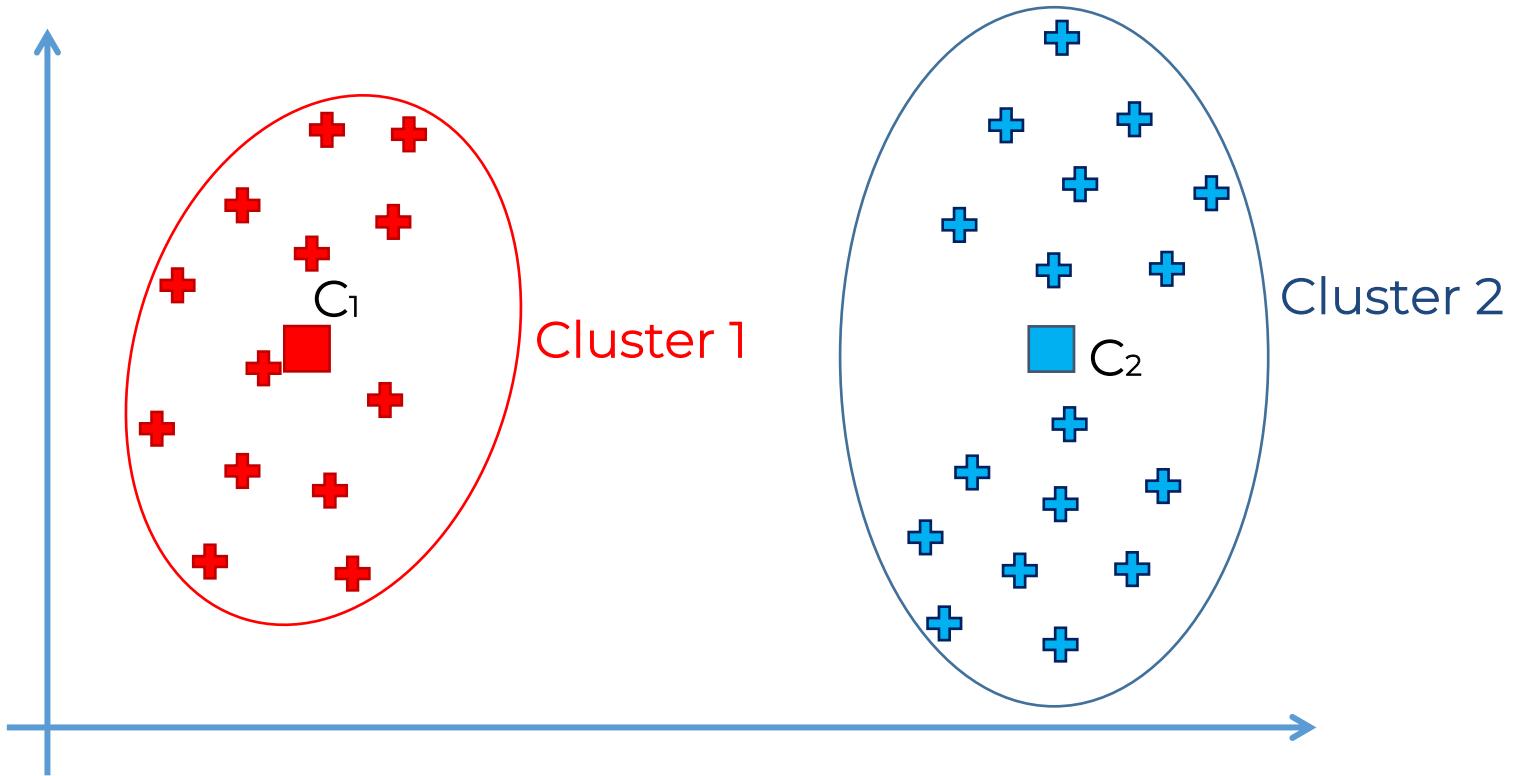


The Elbow Method





The Elbow Method

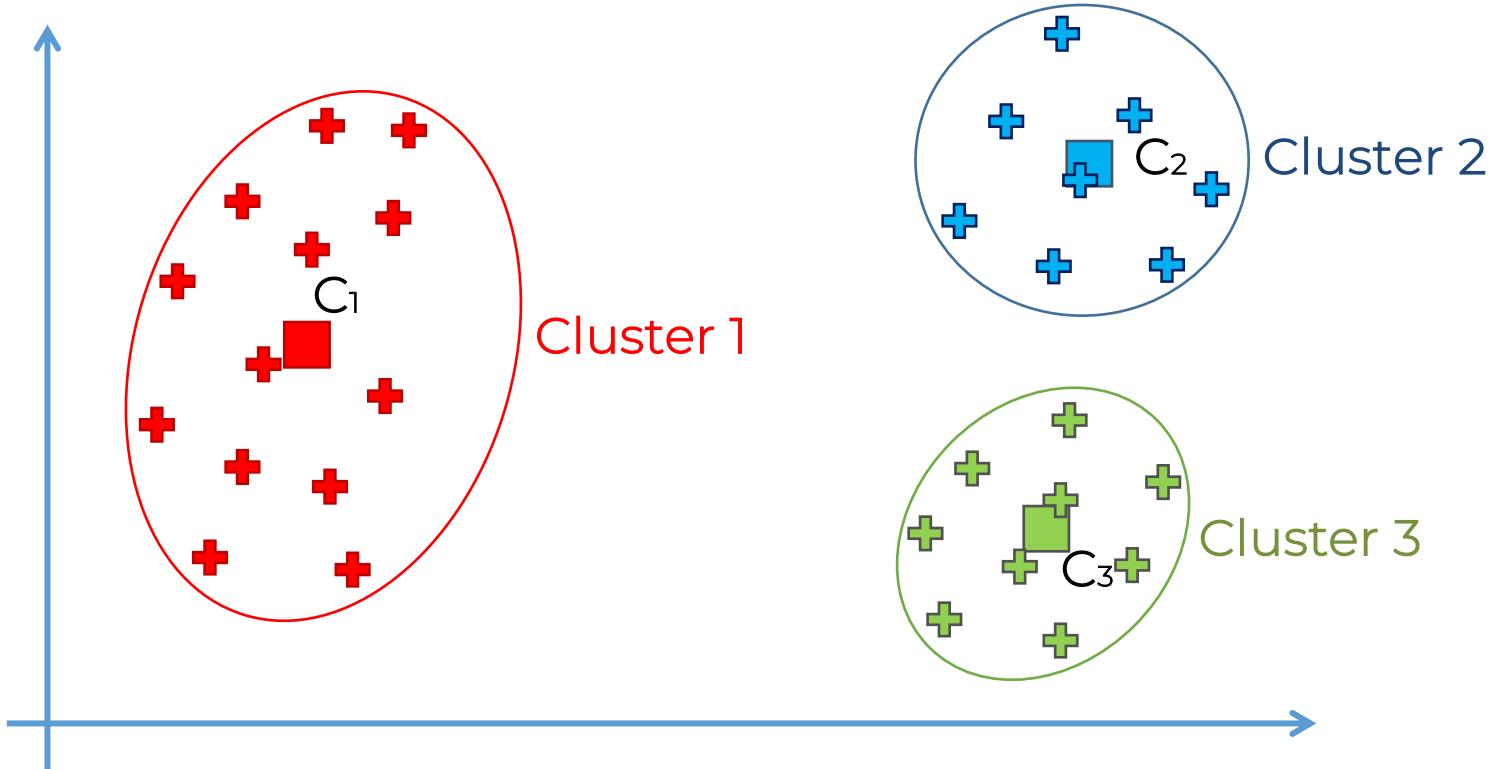


$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2$$





The Elbow Method



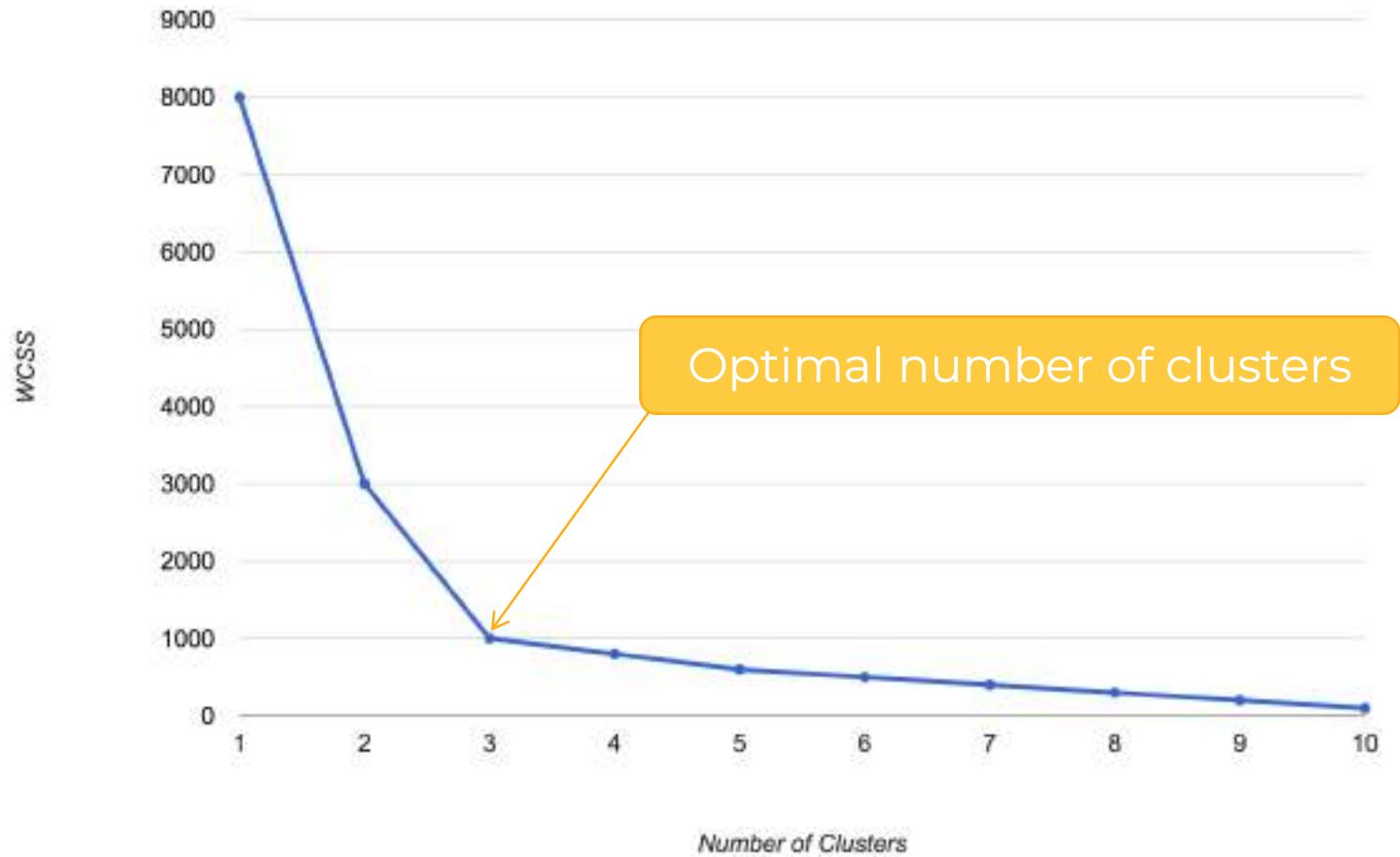
$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$





The Elbow Method

The Elbow Method

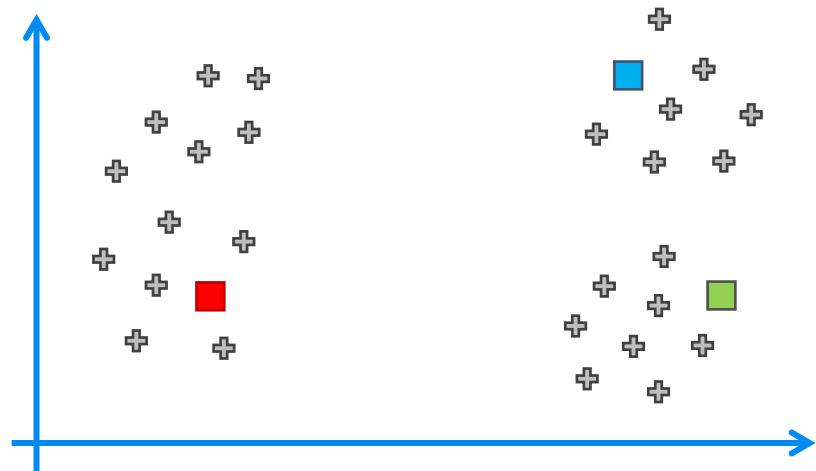


K-Means++

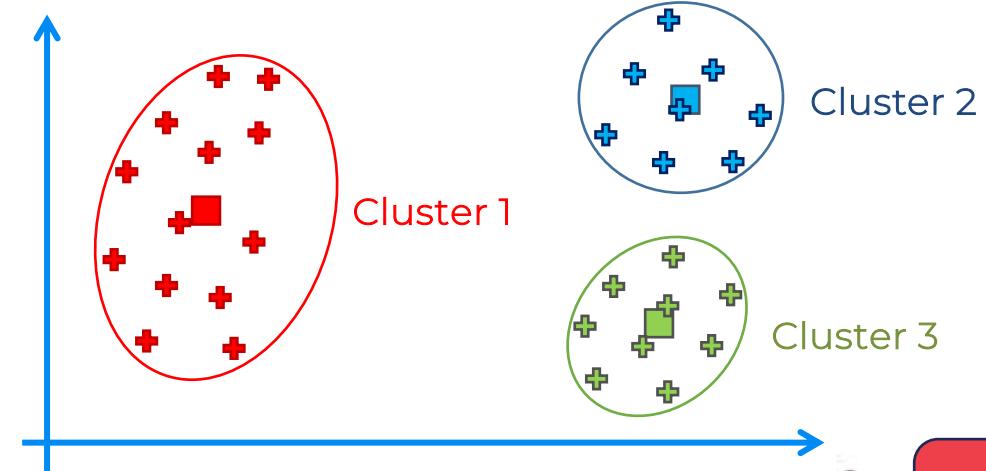




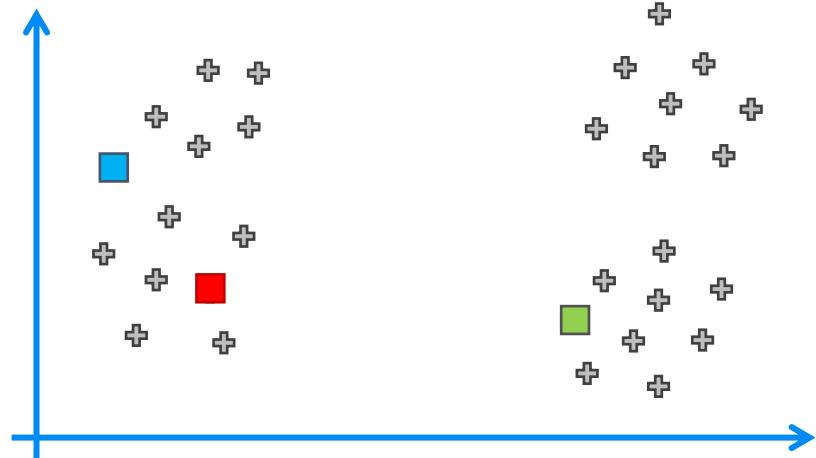
K-Means++



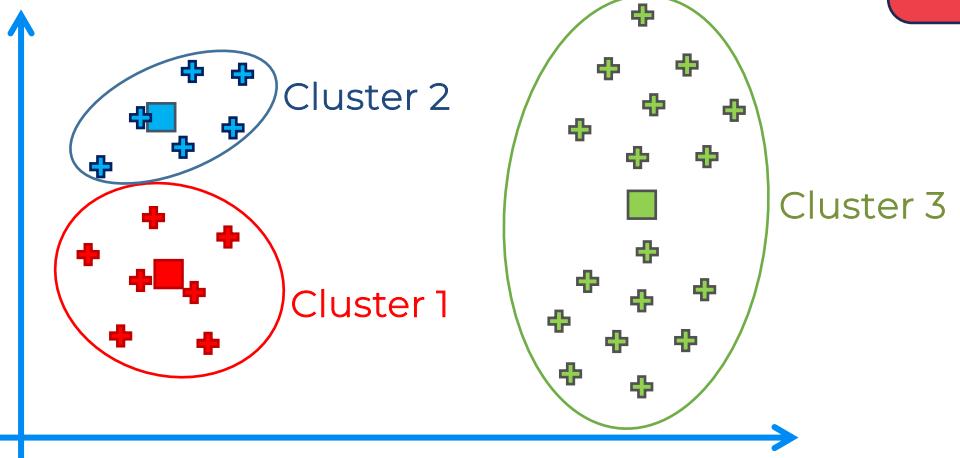
K-Means



Different results



K-Means



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

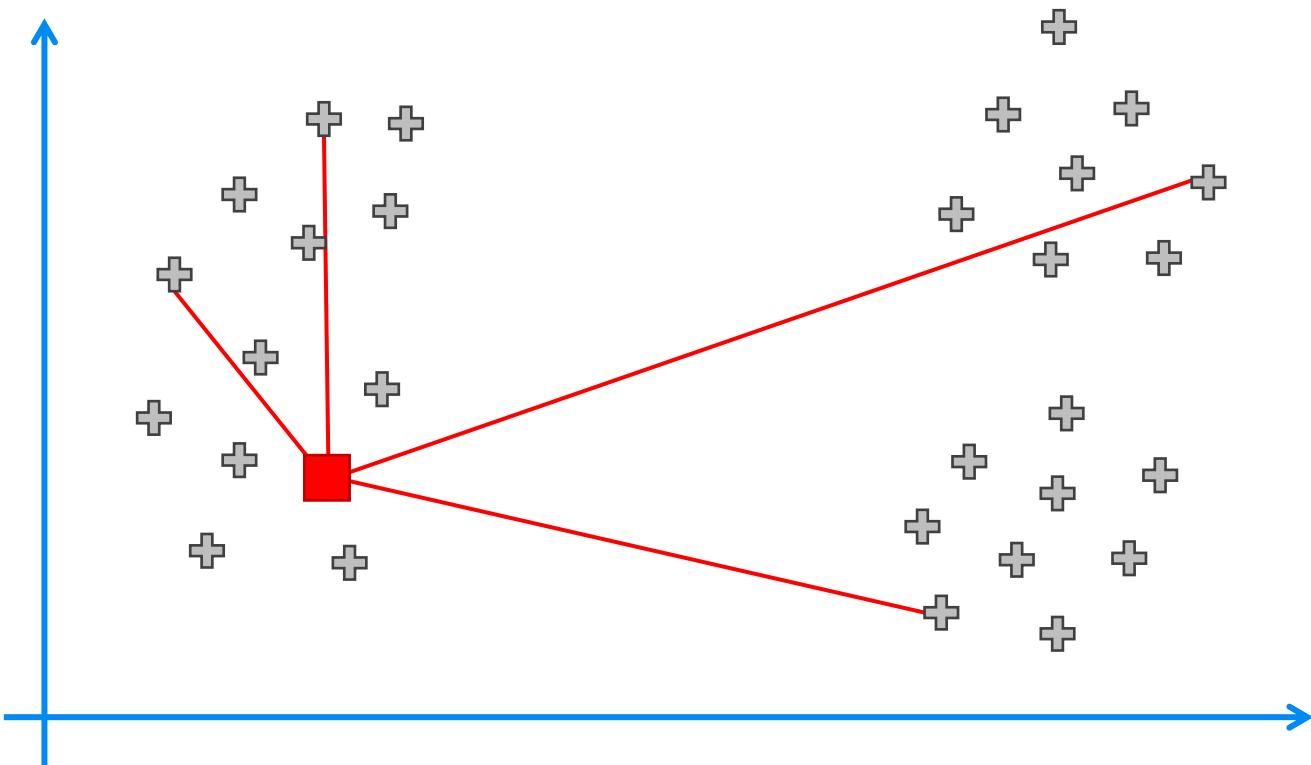
Step 4: Repeat Steps 2 and 3 until all k centroids have been selected

Step 5: Proceed with standard k-means clustering



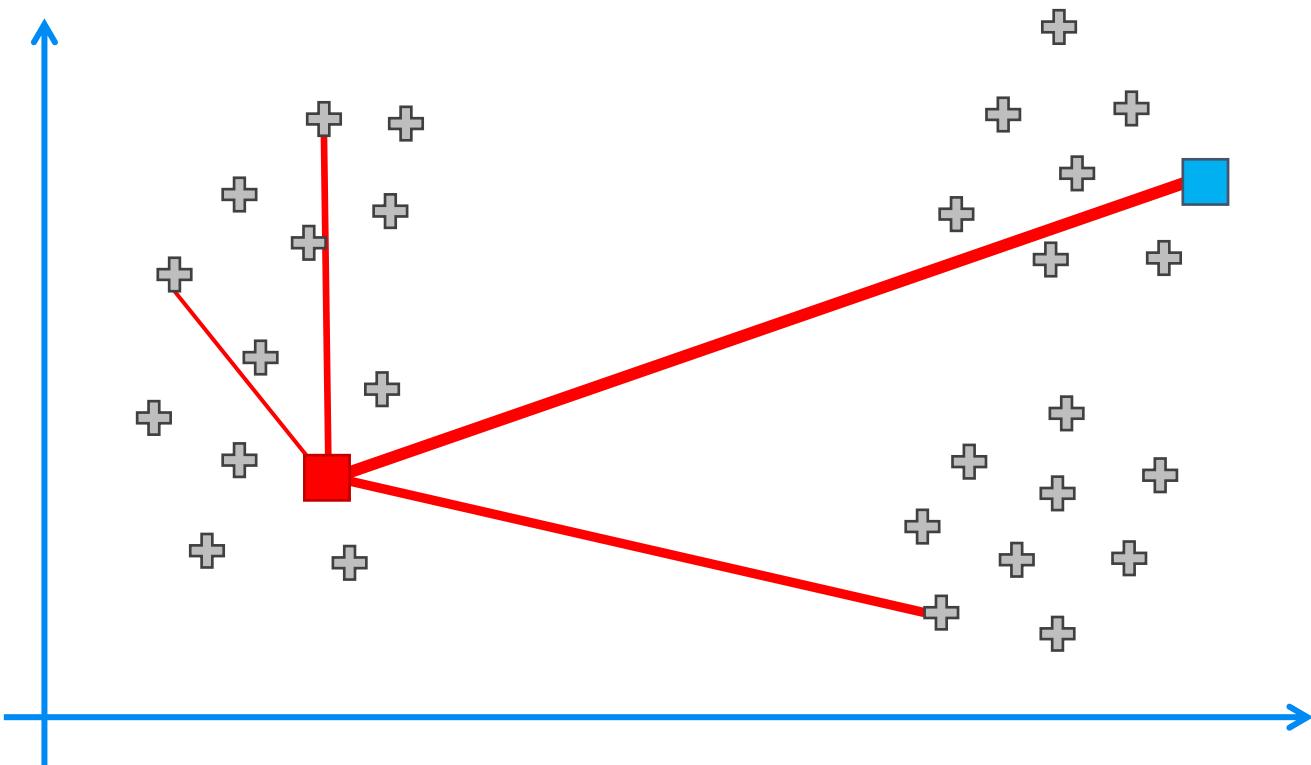


K-Means++



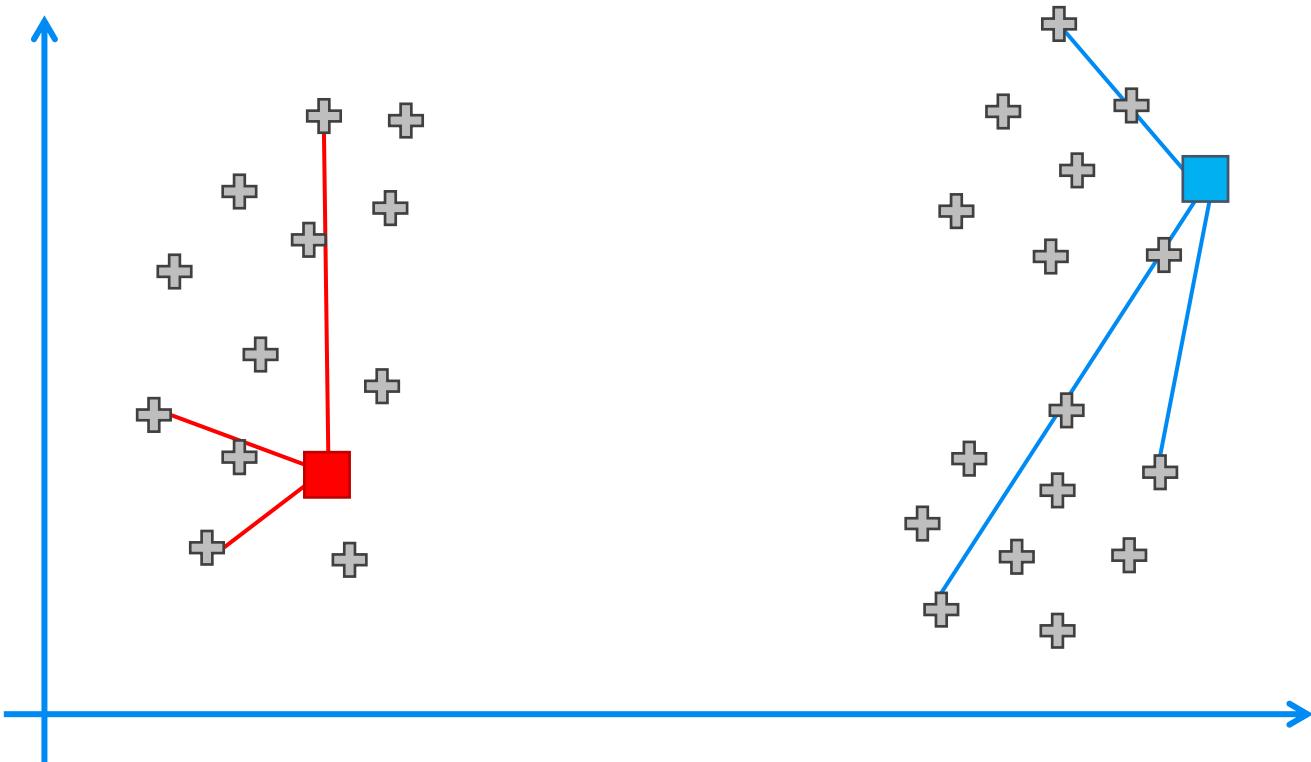


K-Means++



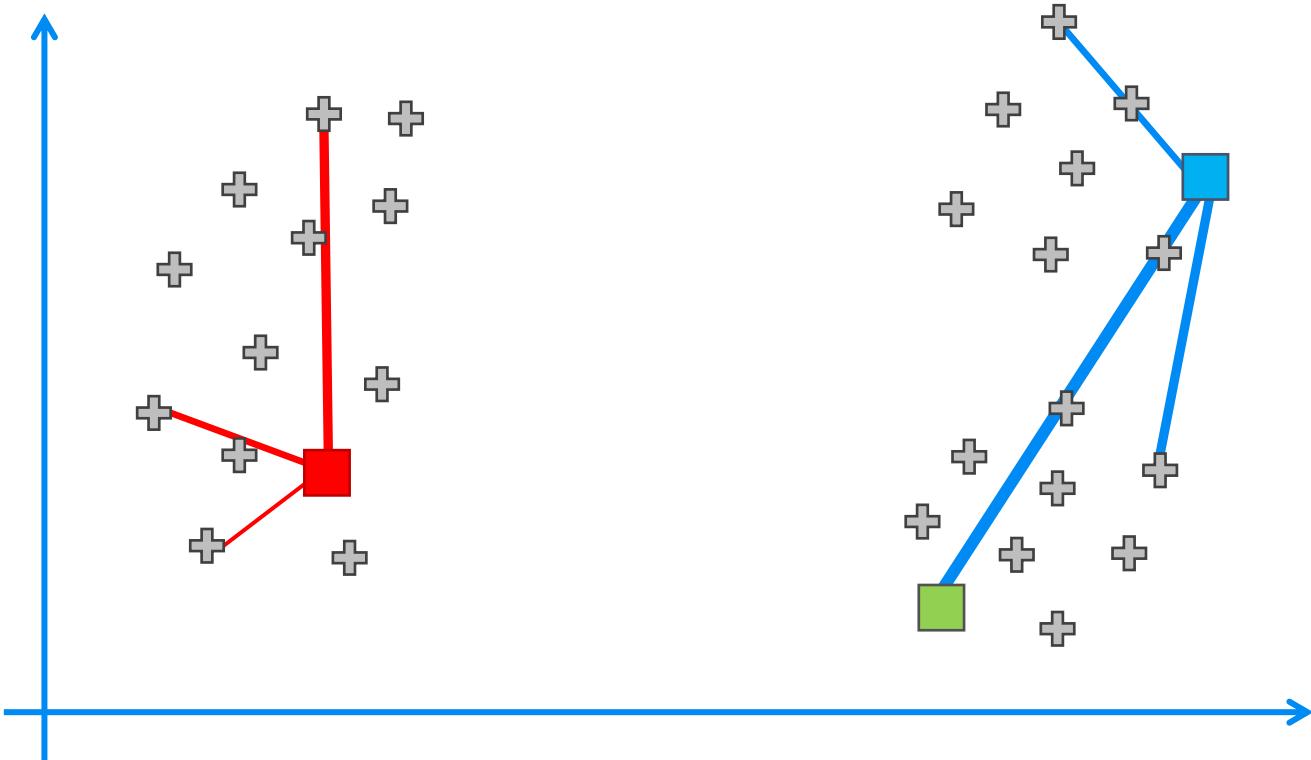


K-Means++



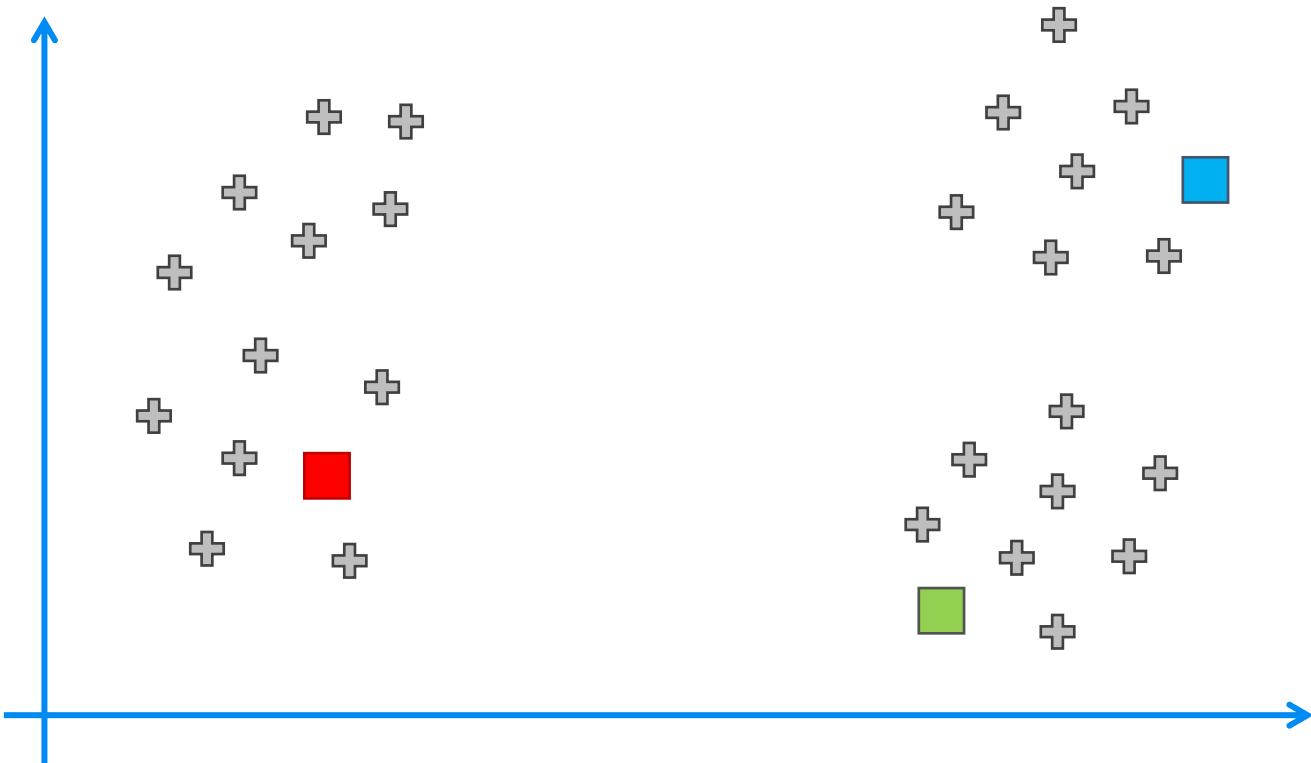


K-Means++

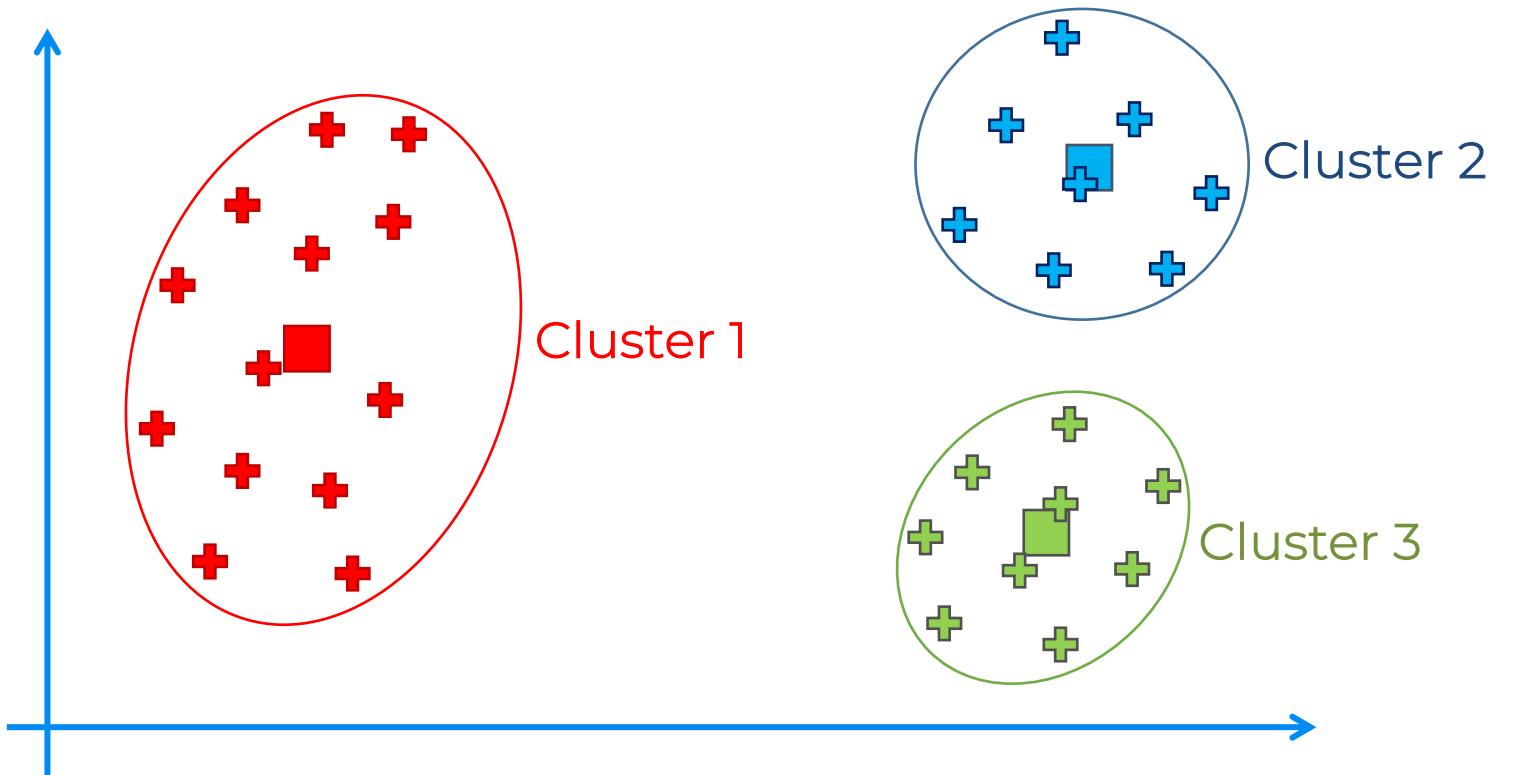




K-Means++

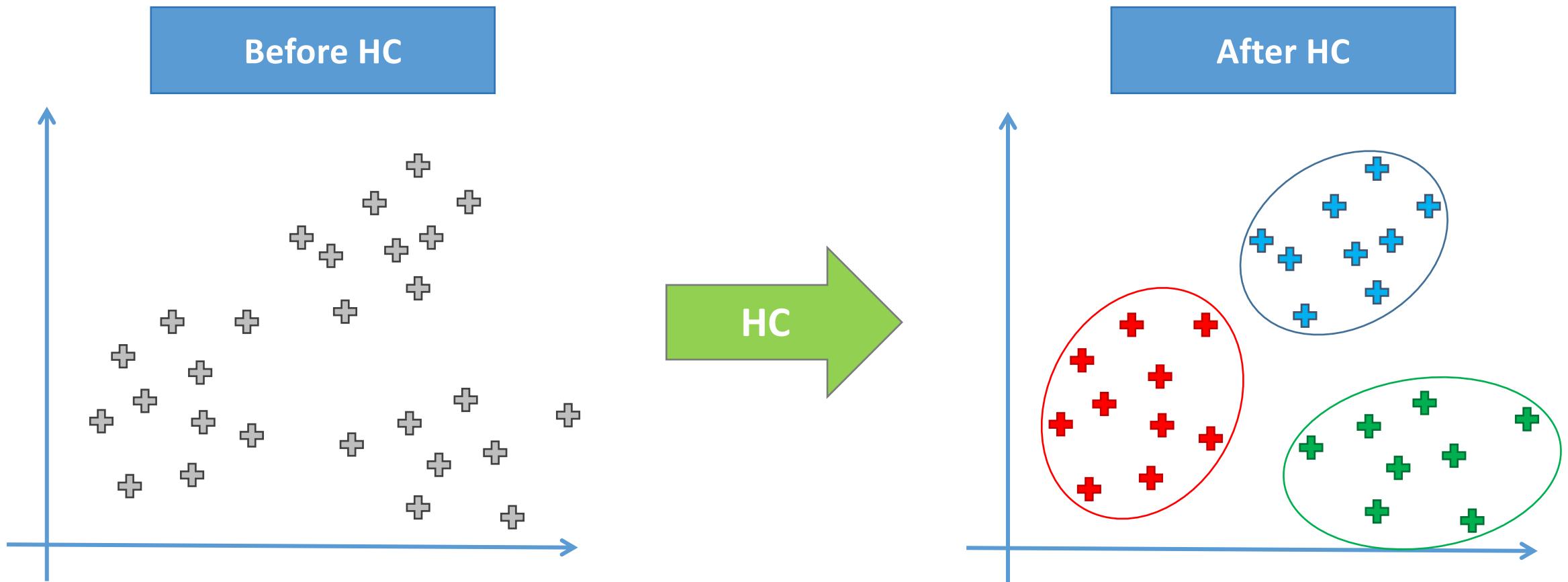


K-Means++



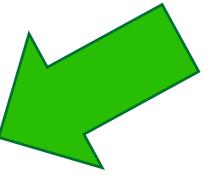
HC Intuition: Understanding HC

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

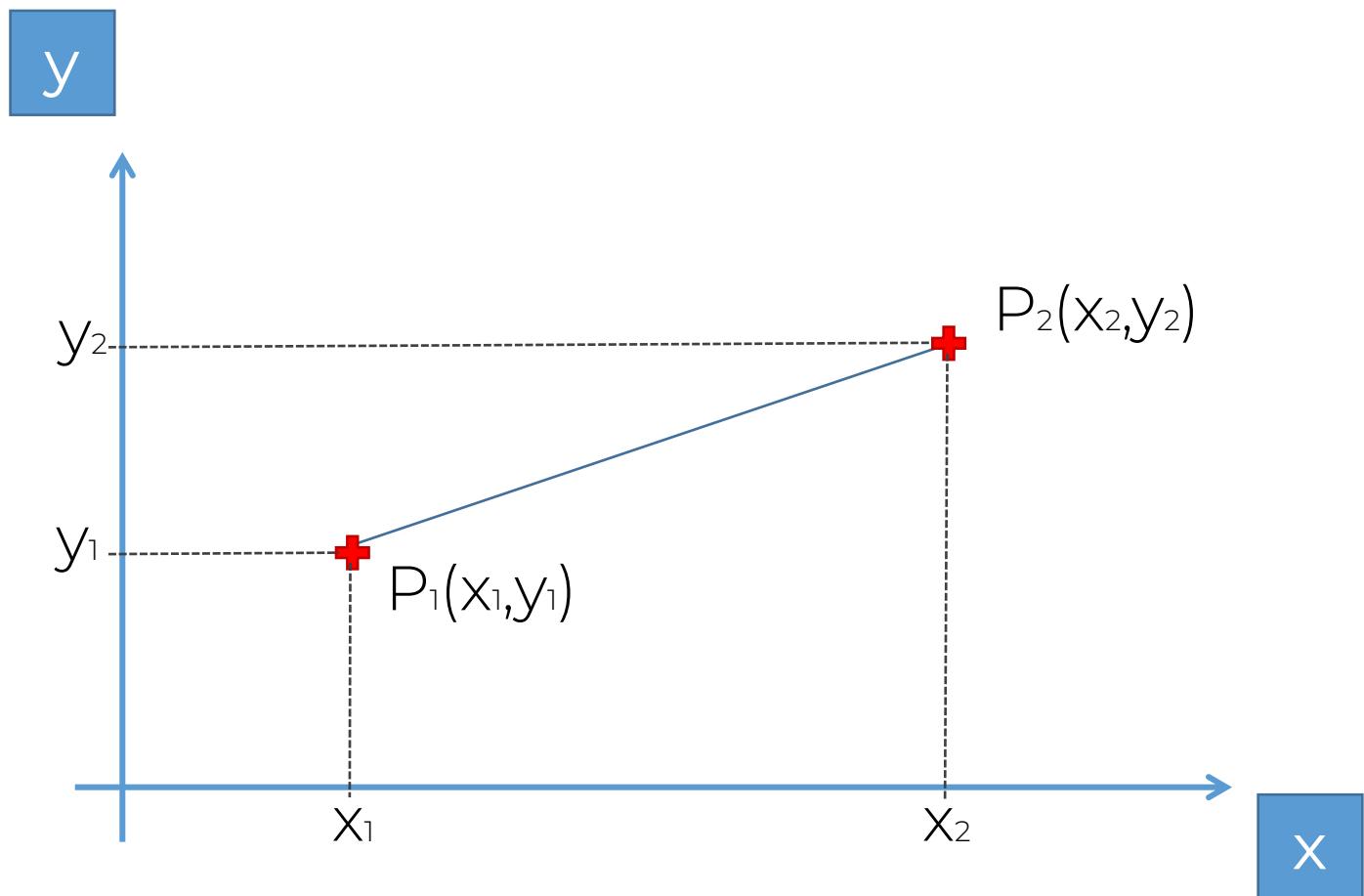


STEP 4: Repeat STEP 3 until there is only one cluster



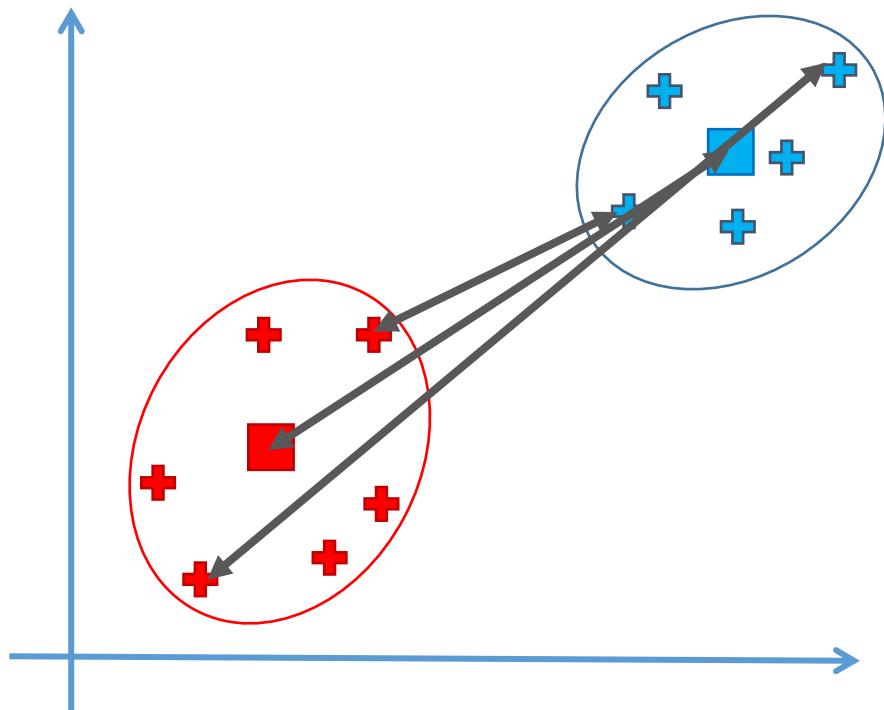
FIN

Euclidean Distance



Euclidean Distance between P_1 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



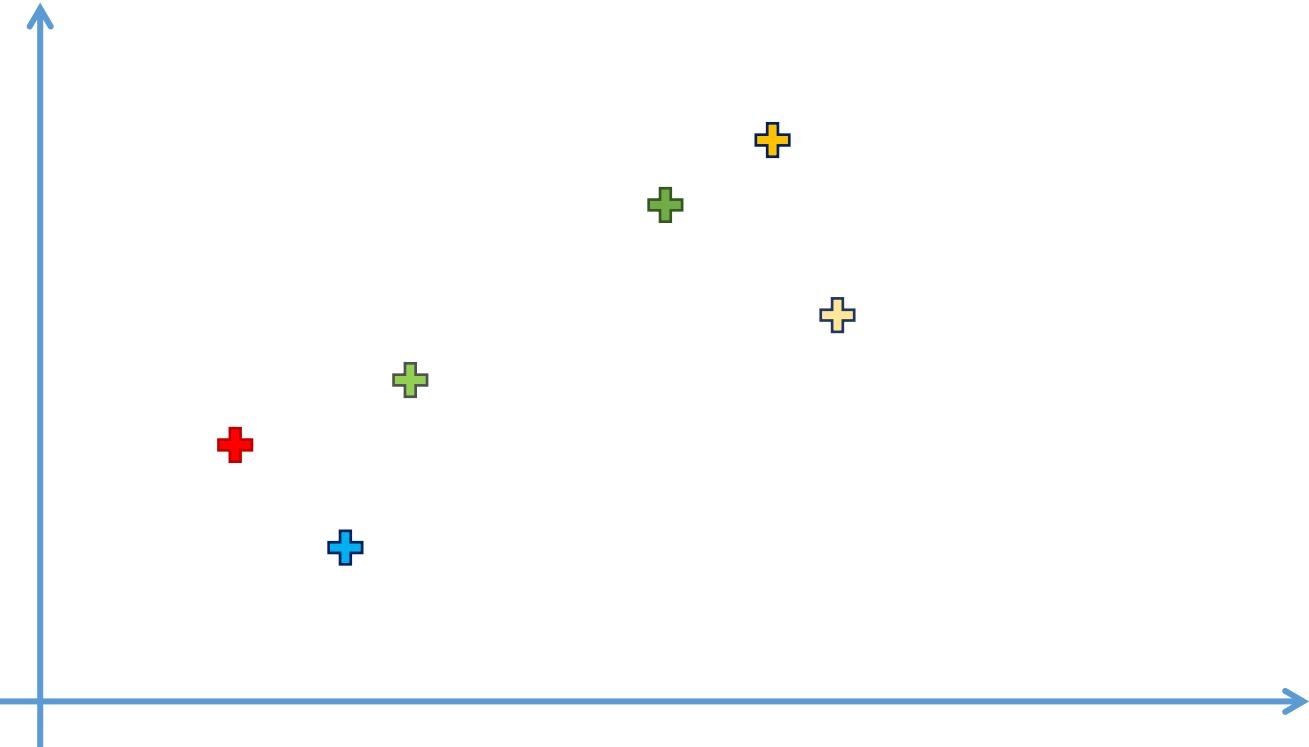
Agglomerative HC

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



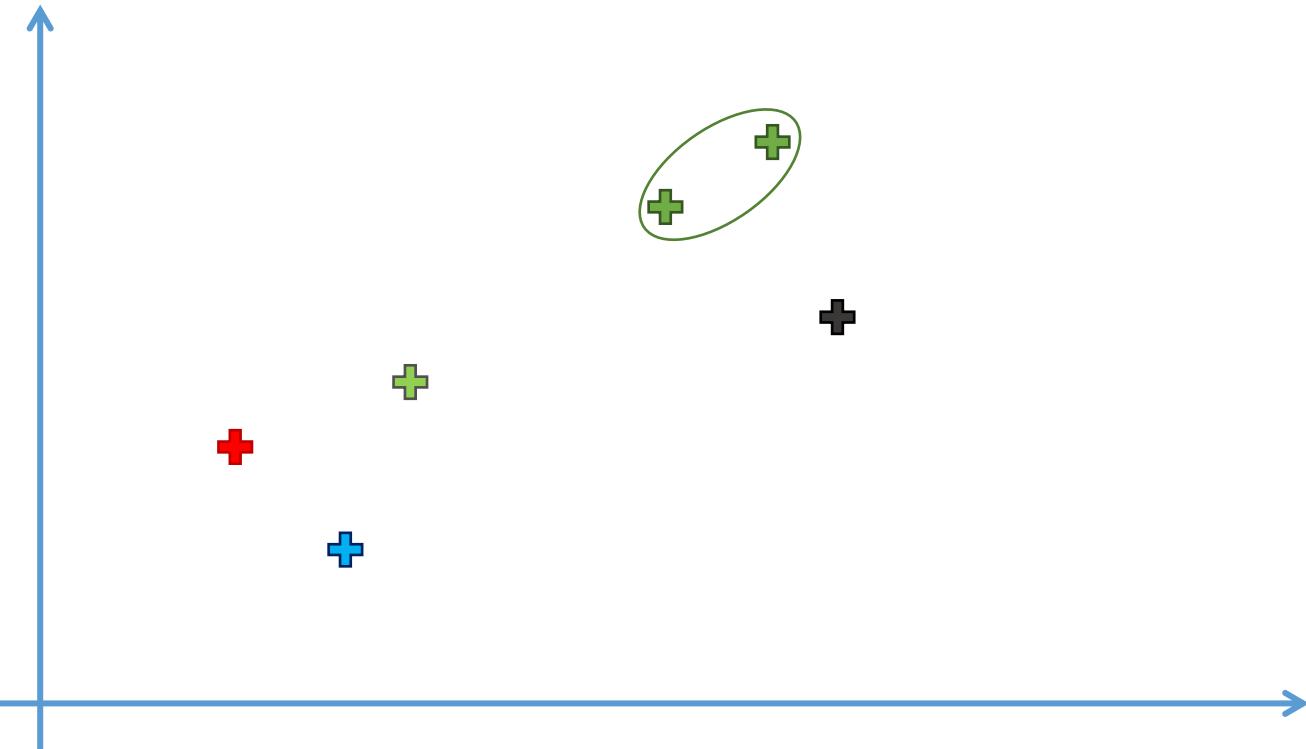
Agglomerative HC

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



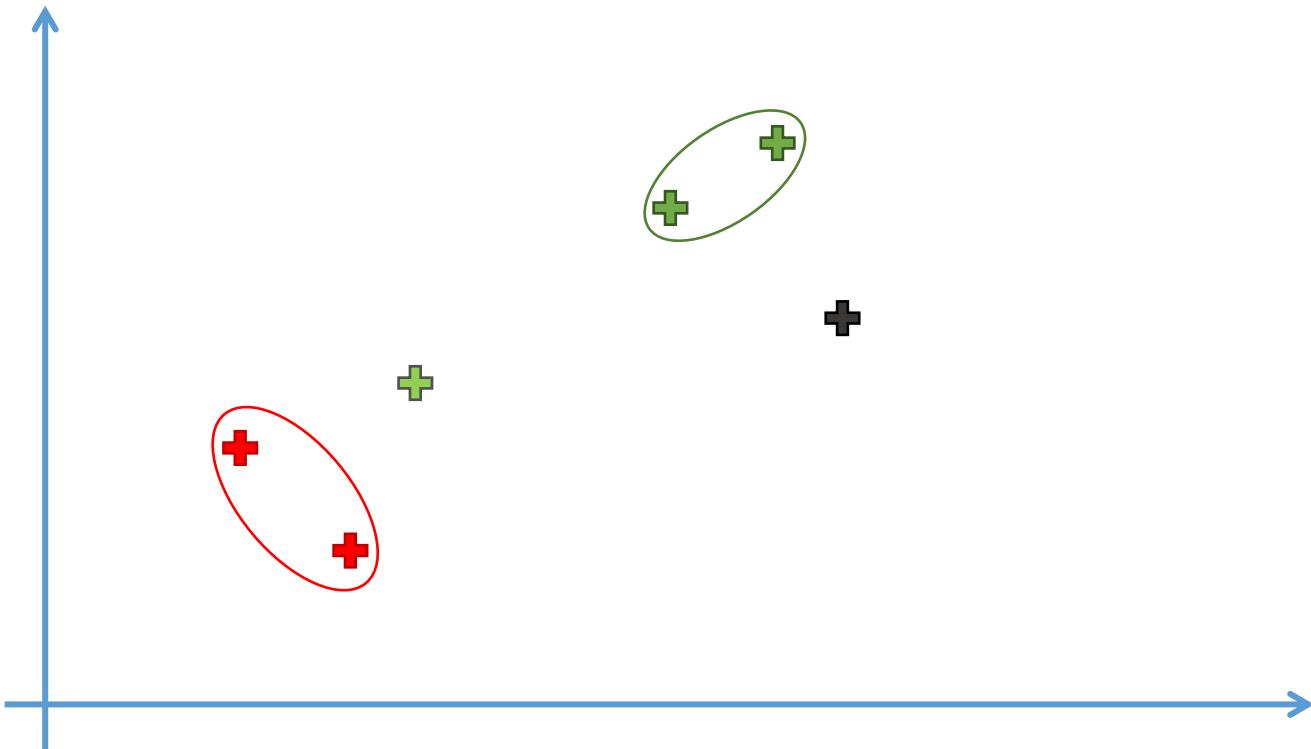
Agglomerative HC

STEP 2: Take the two closest data points and make them one cluster
→ That forms 5 clusters



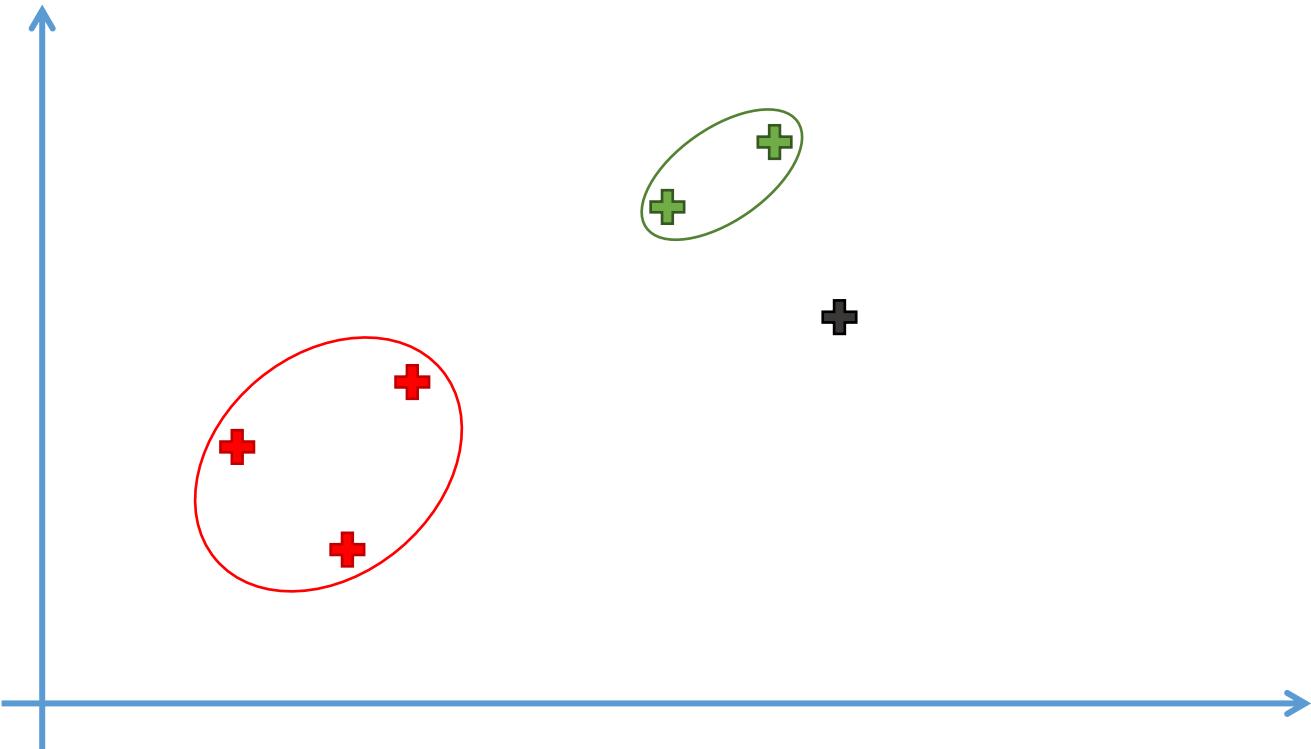
Agglomerative HC

STEP 3: Take the two closest clusters and make them one cluster
→ That forms 4 clusters



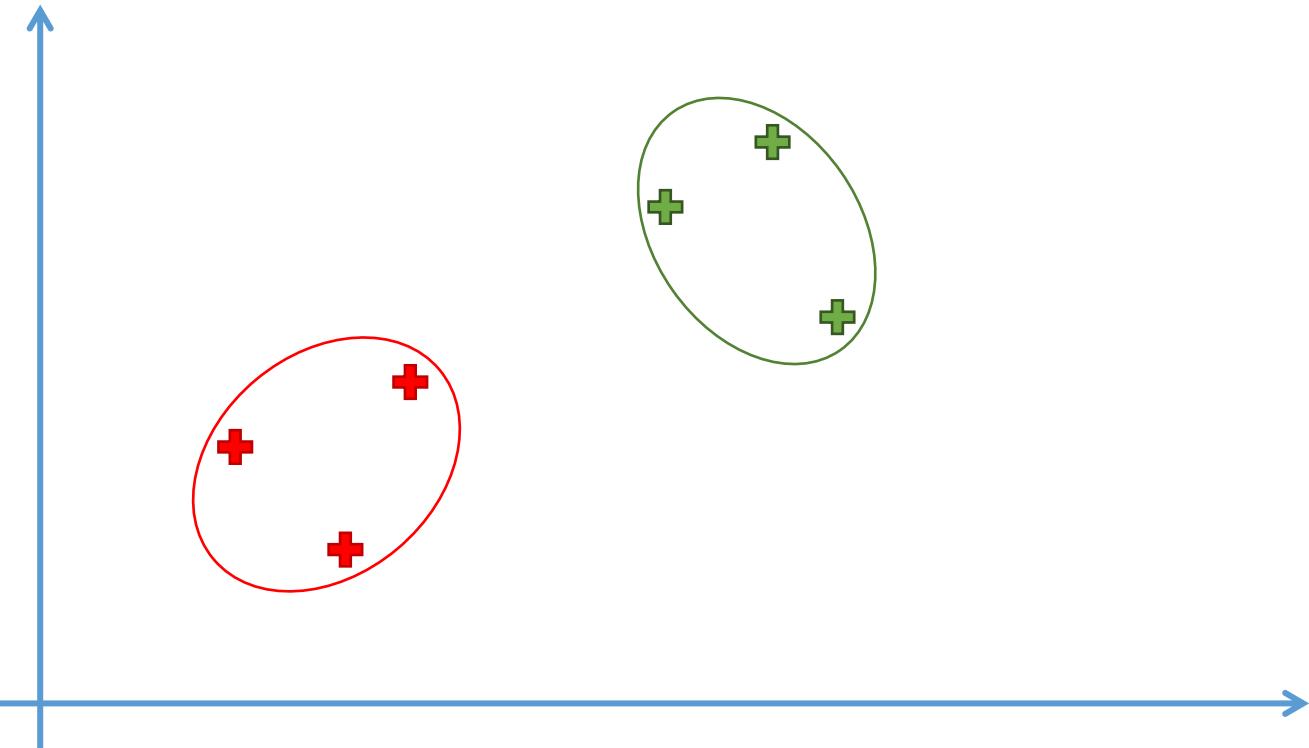
Agglomerative HC

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



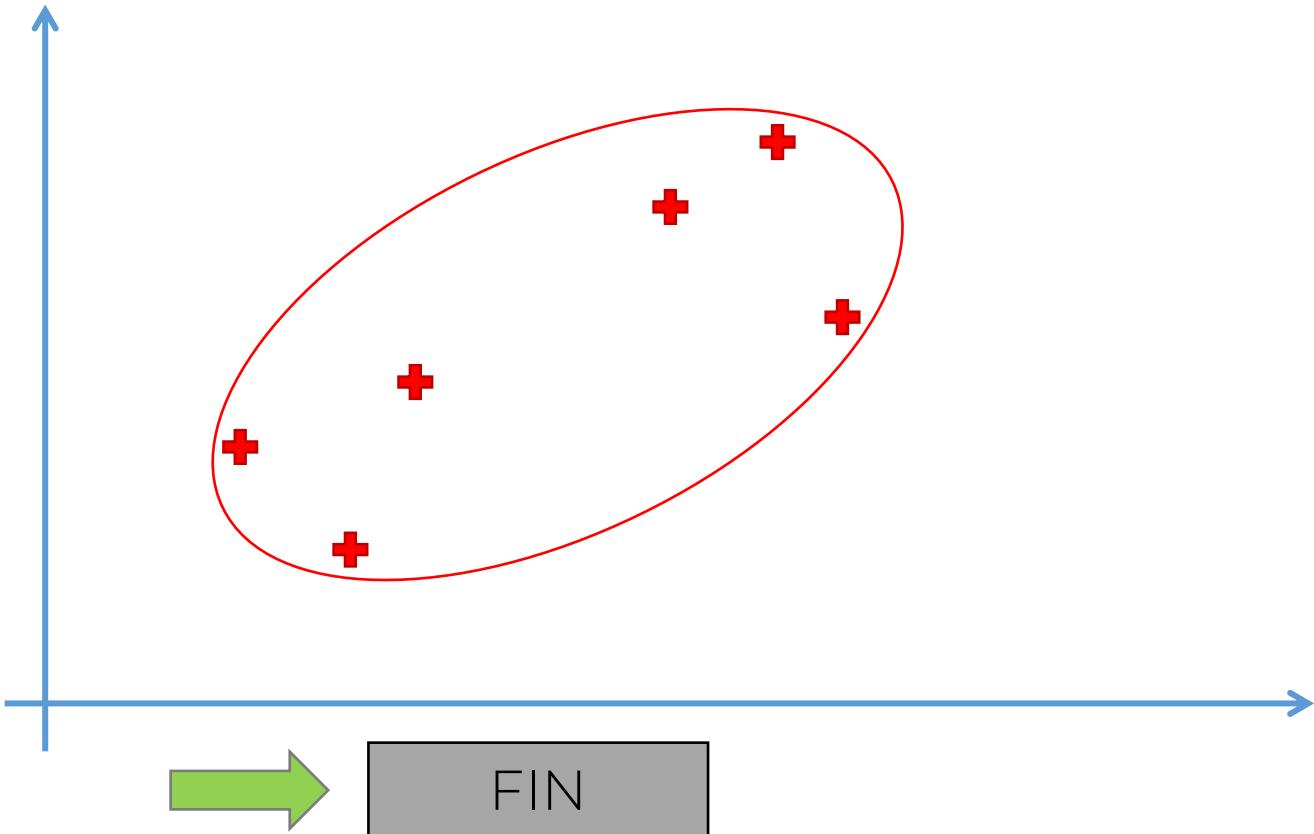
Agglomerative HC

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



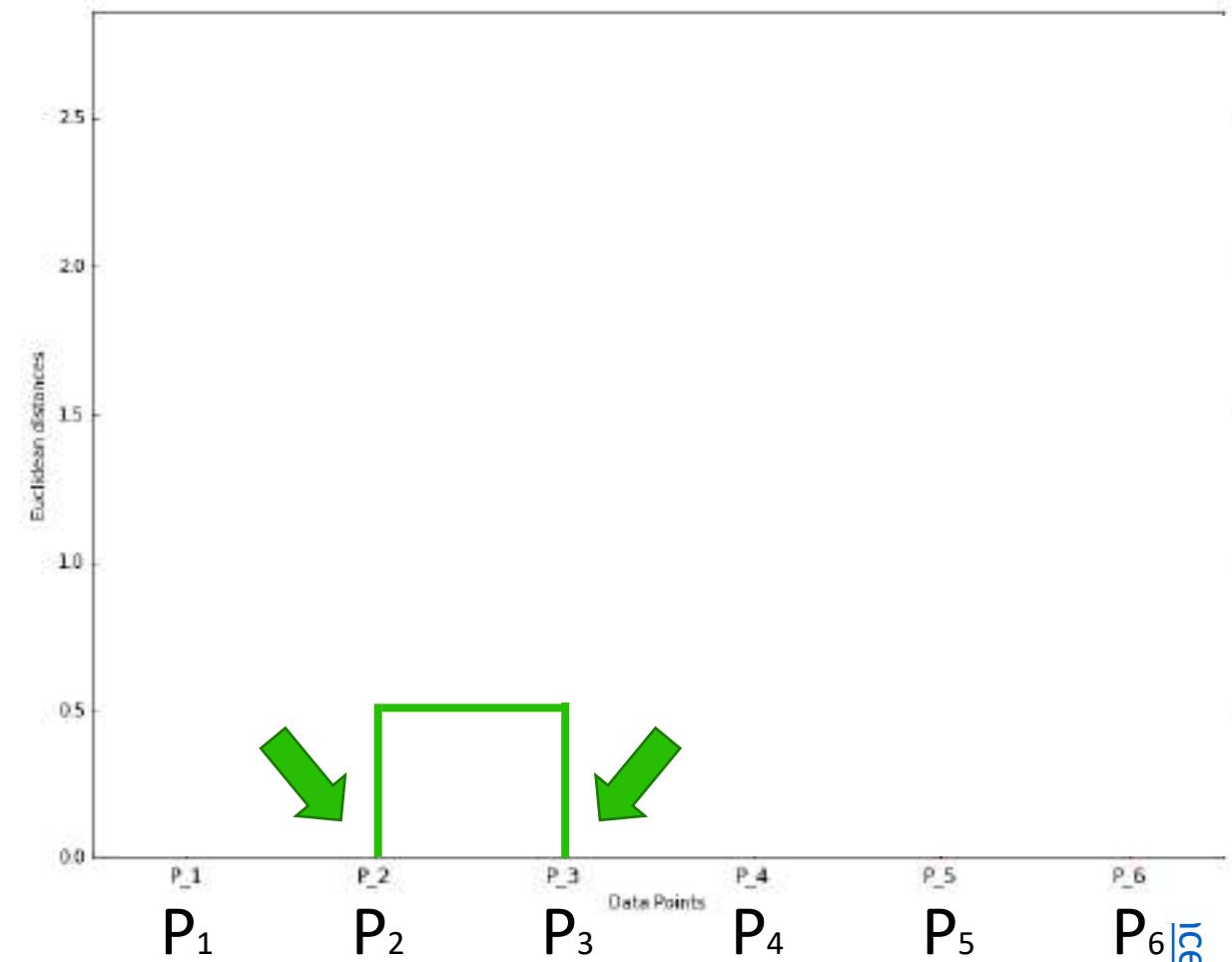
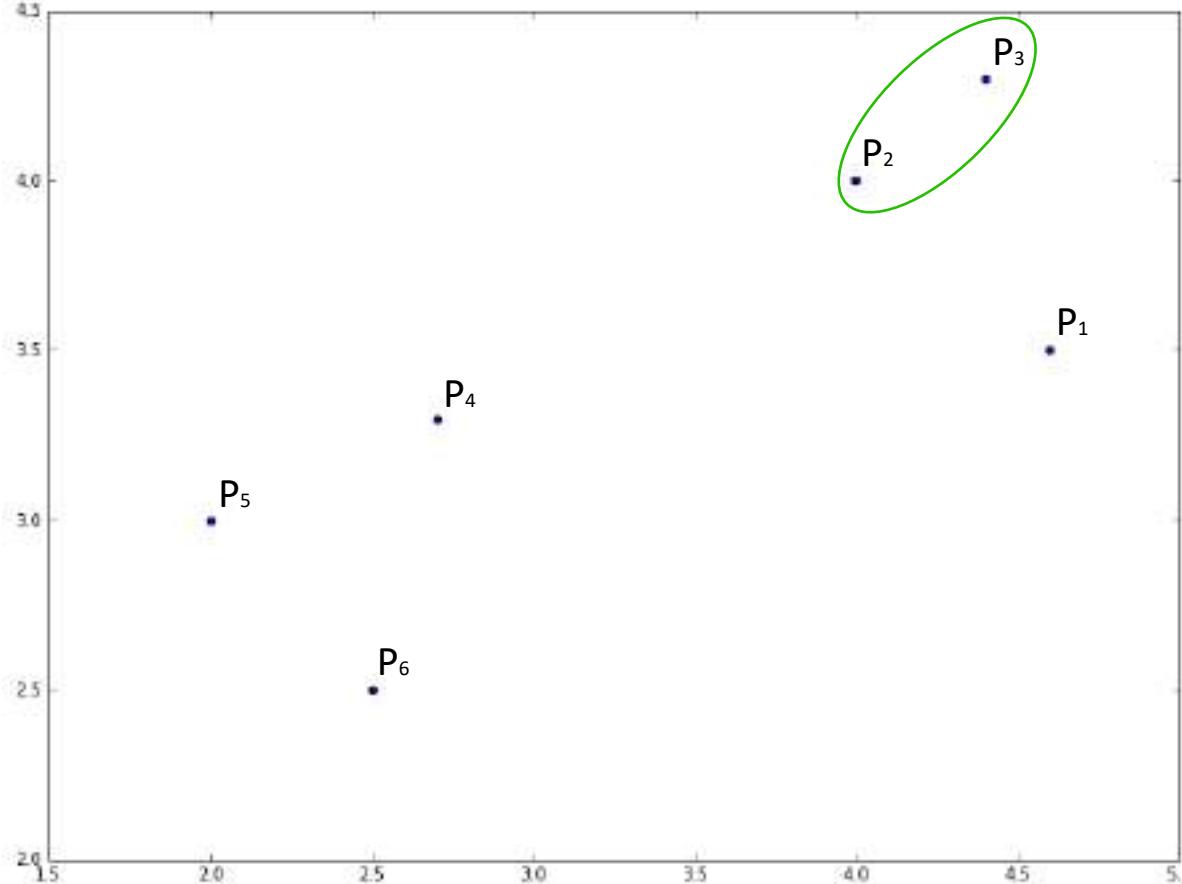
Agglomerative HC

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

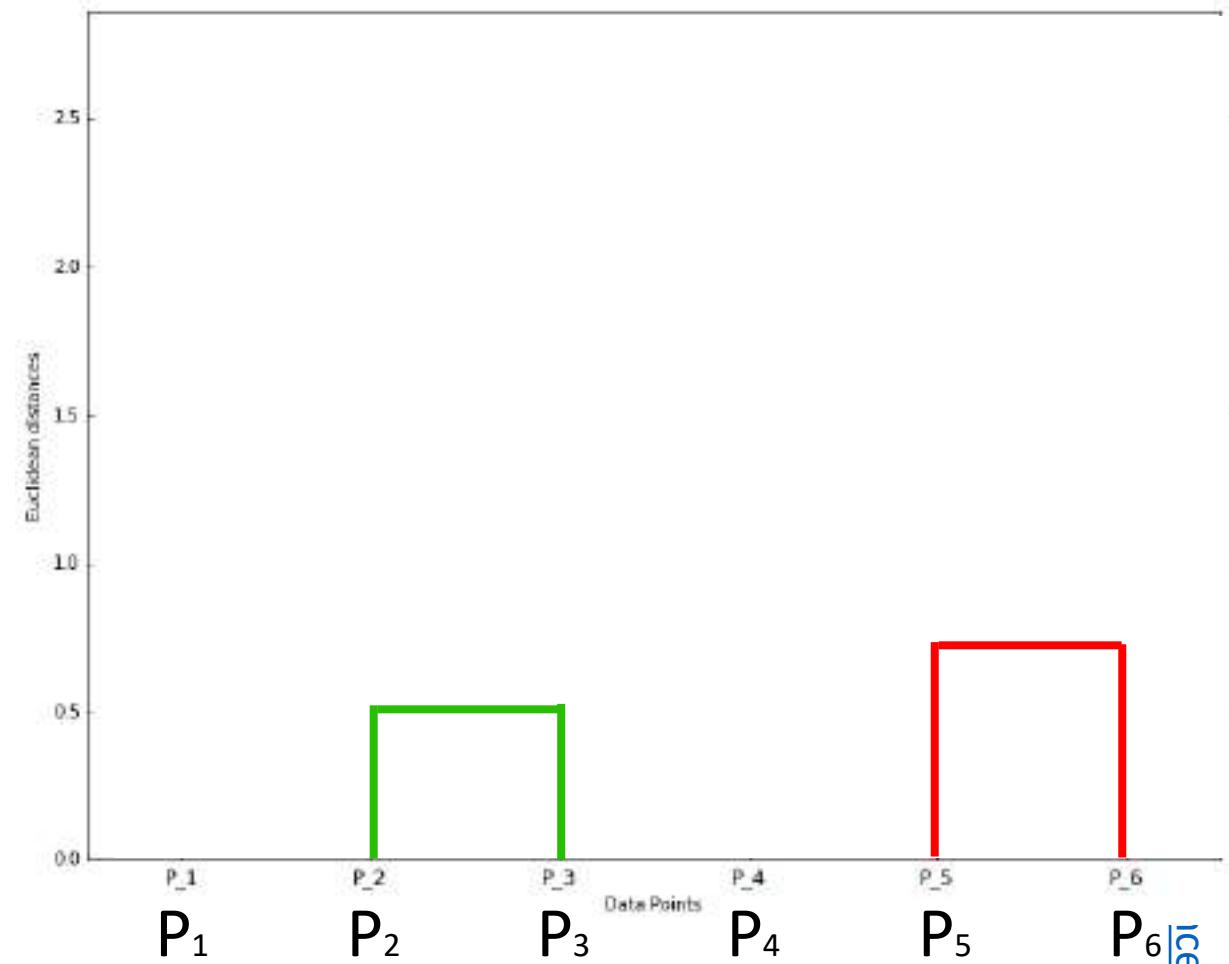
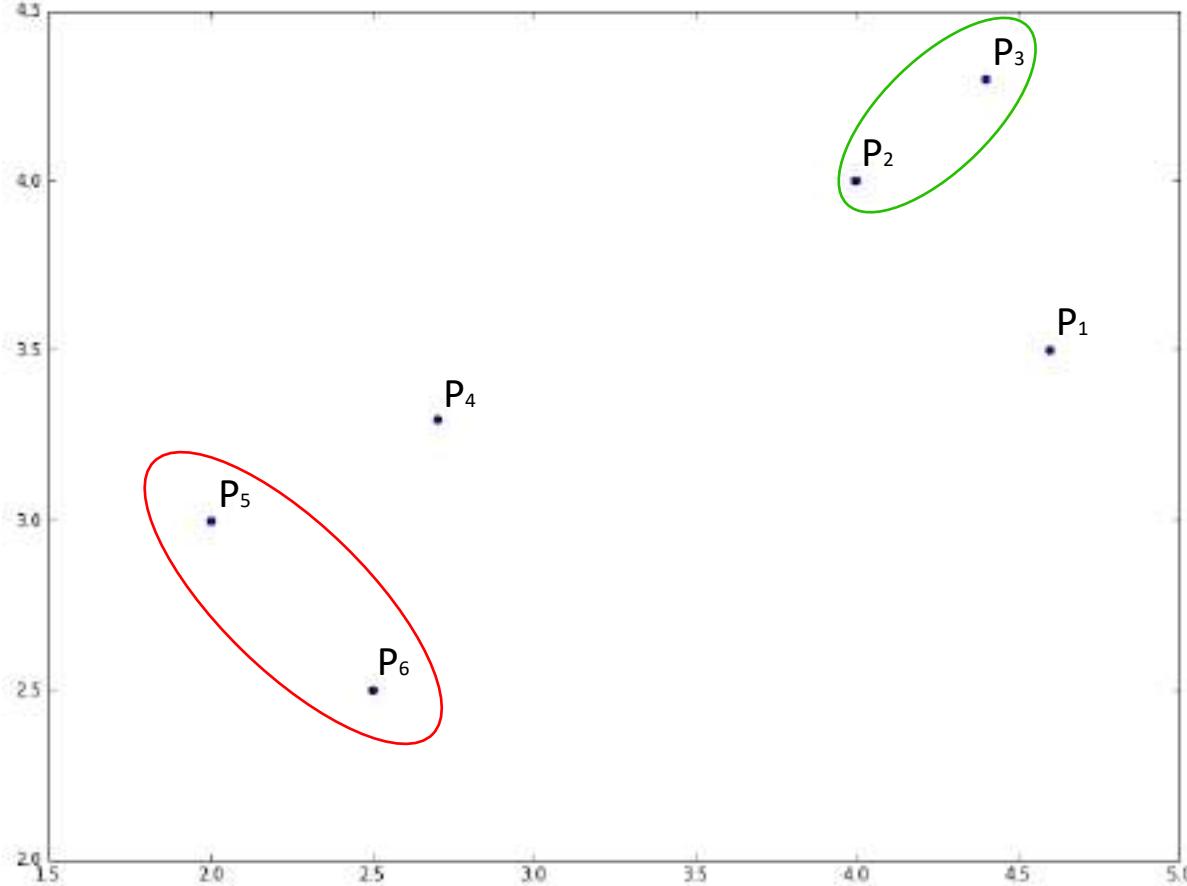


HC Intuition: How Do Dendograms Work?

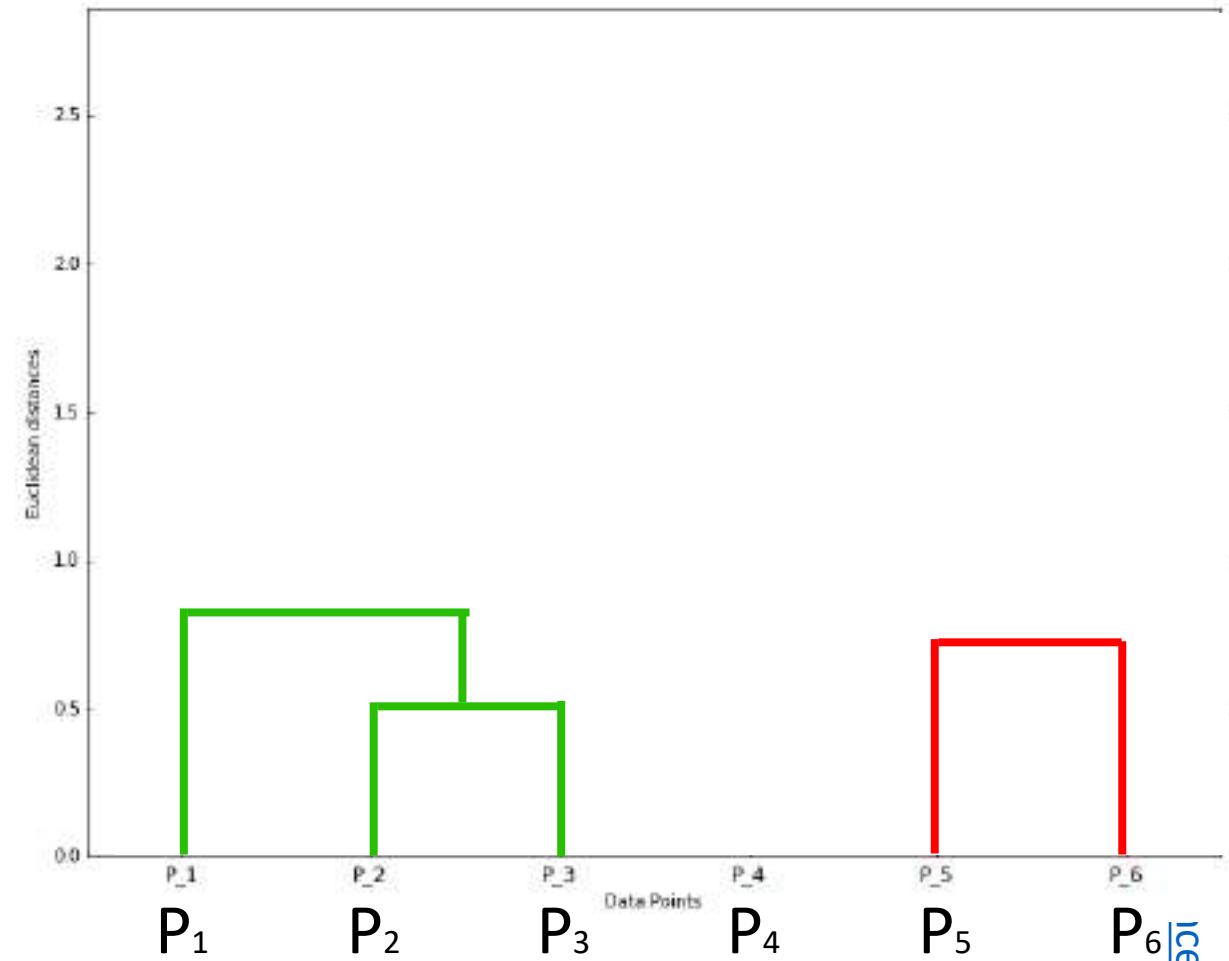
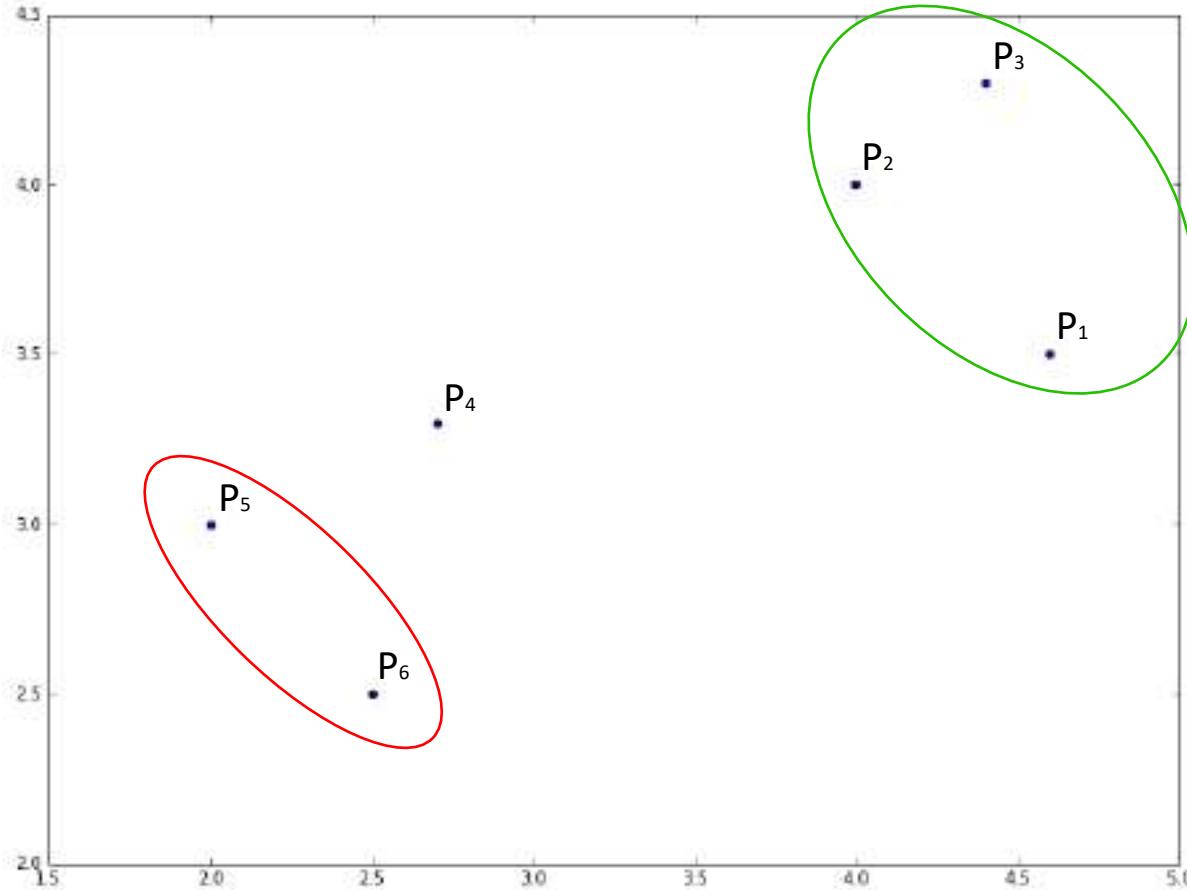
How Do Dendograms Work?



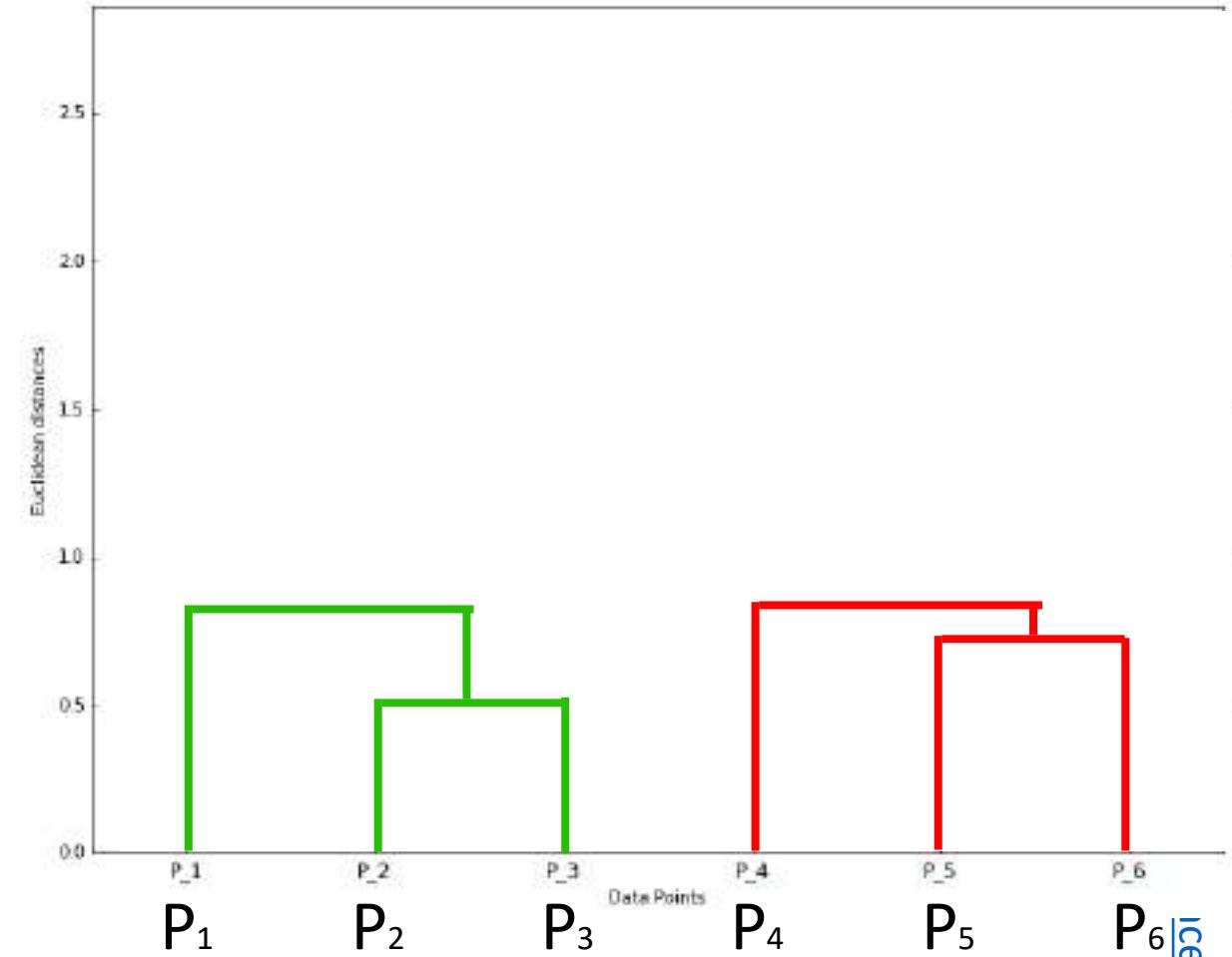
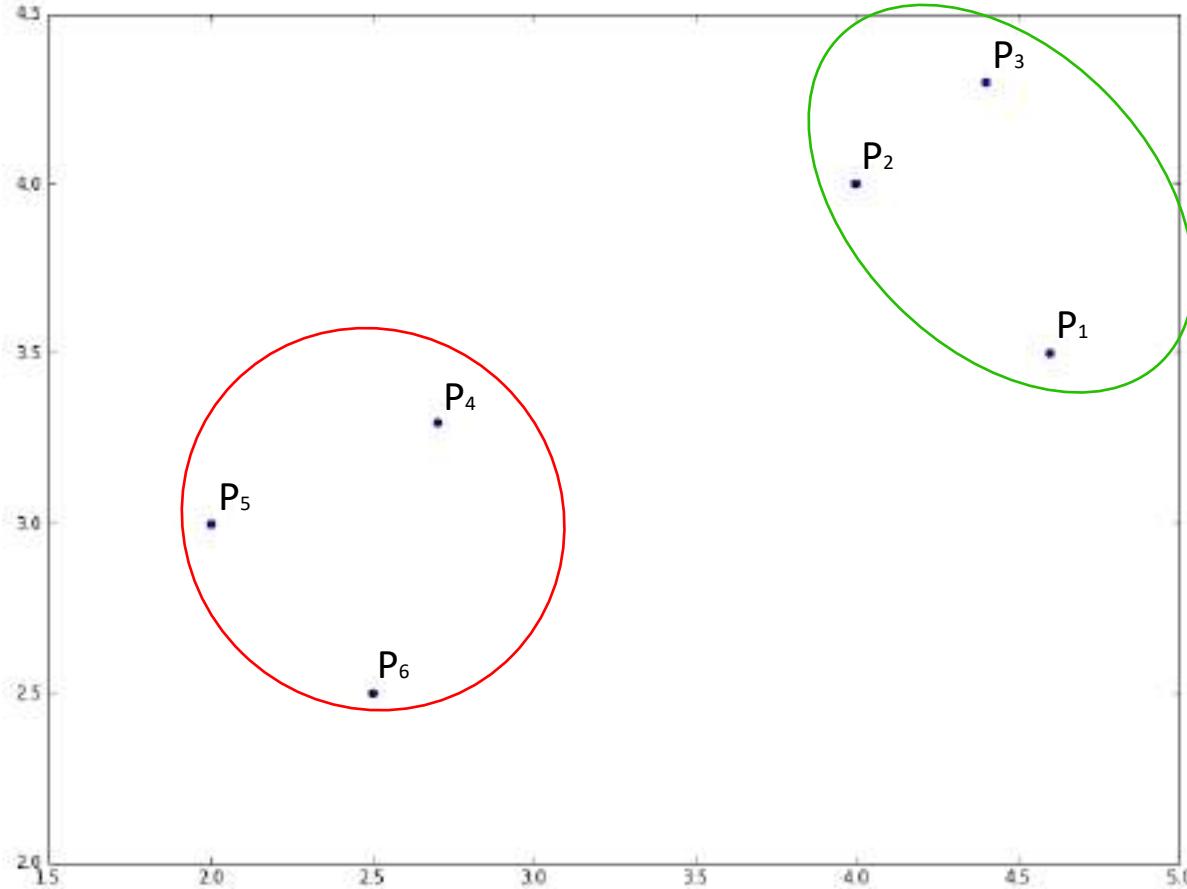
How Do Dendograms Work?



How Do Dendograms Work?

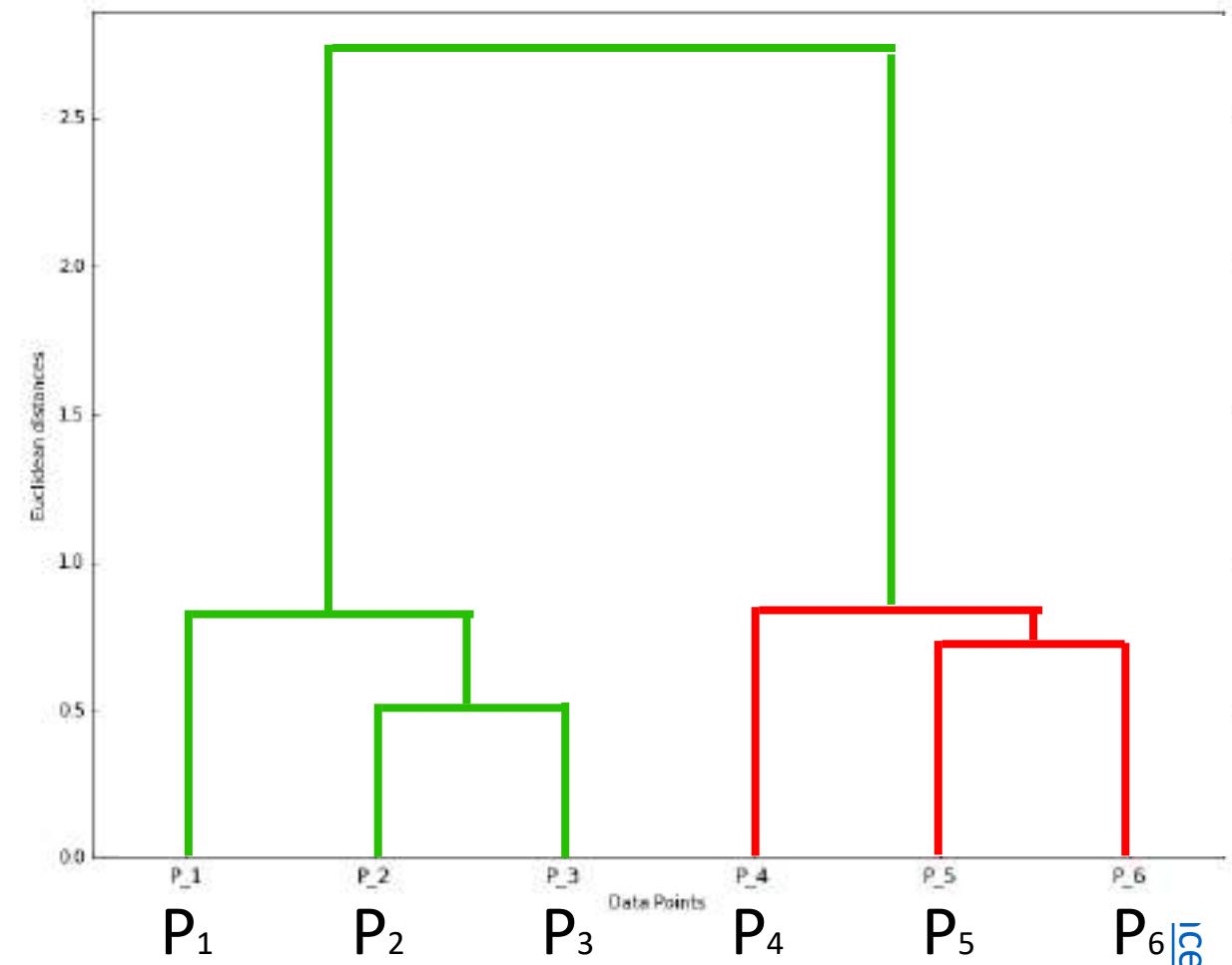
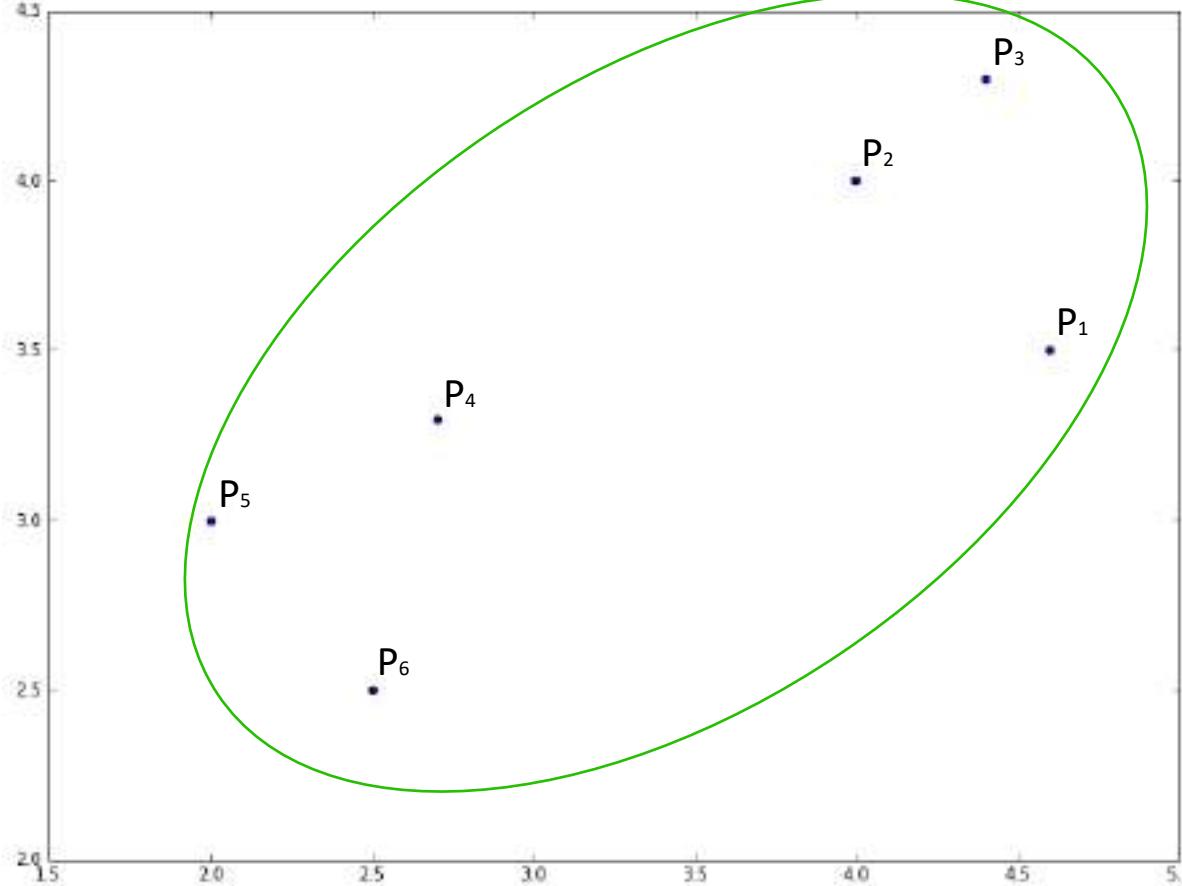


How Do Dendograms Work?

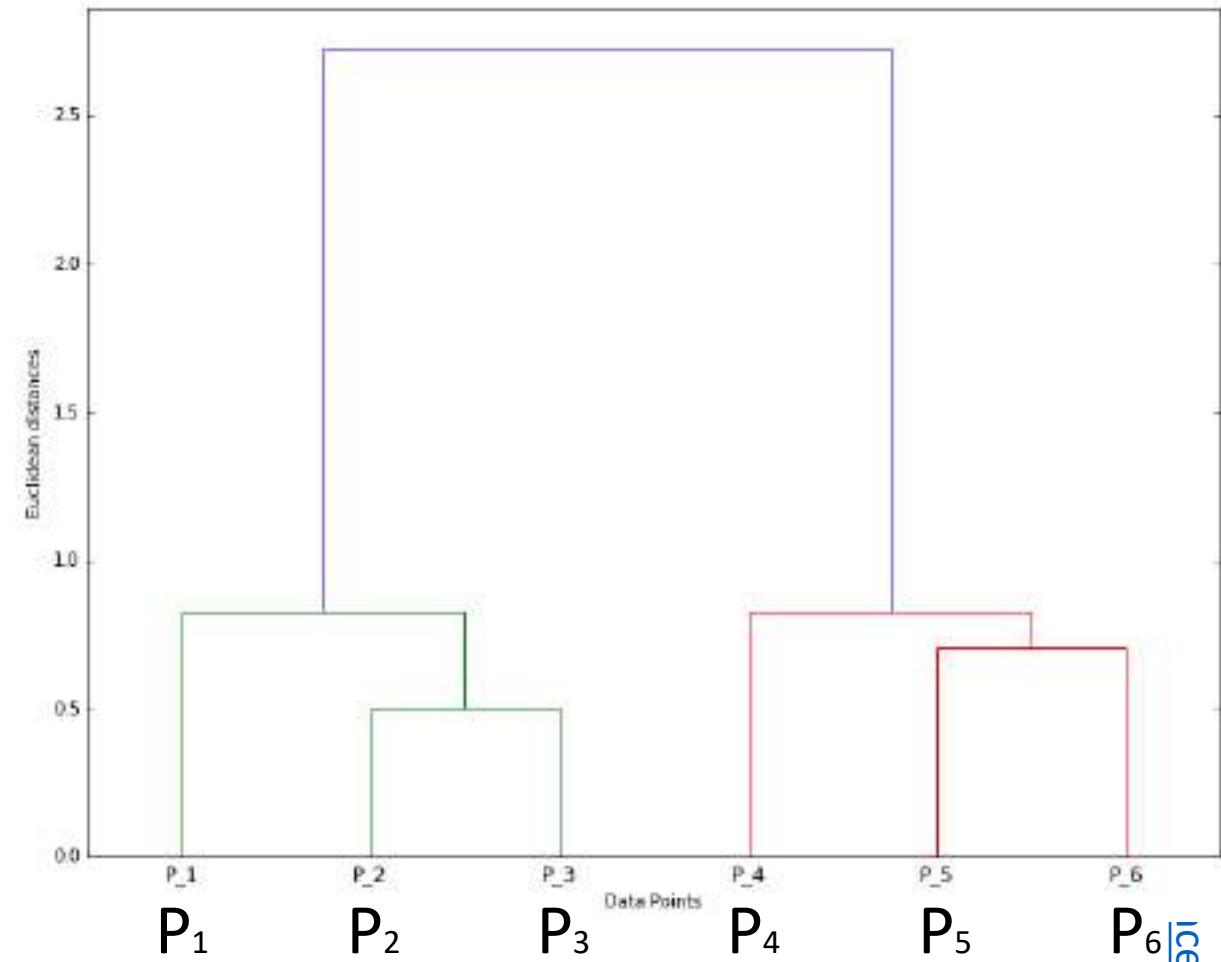
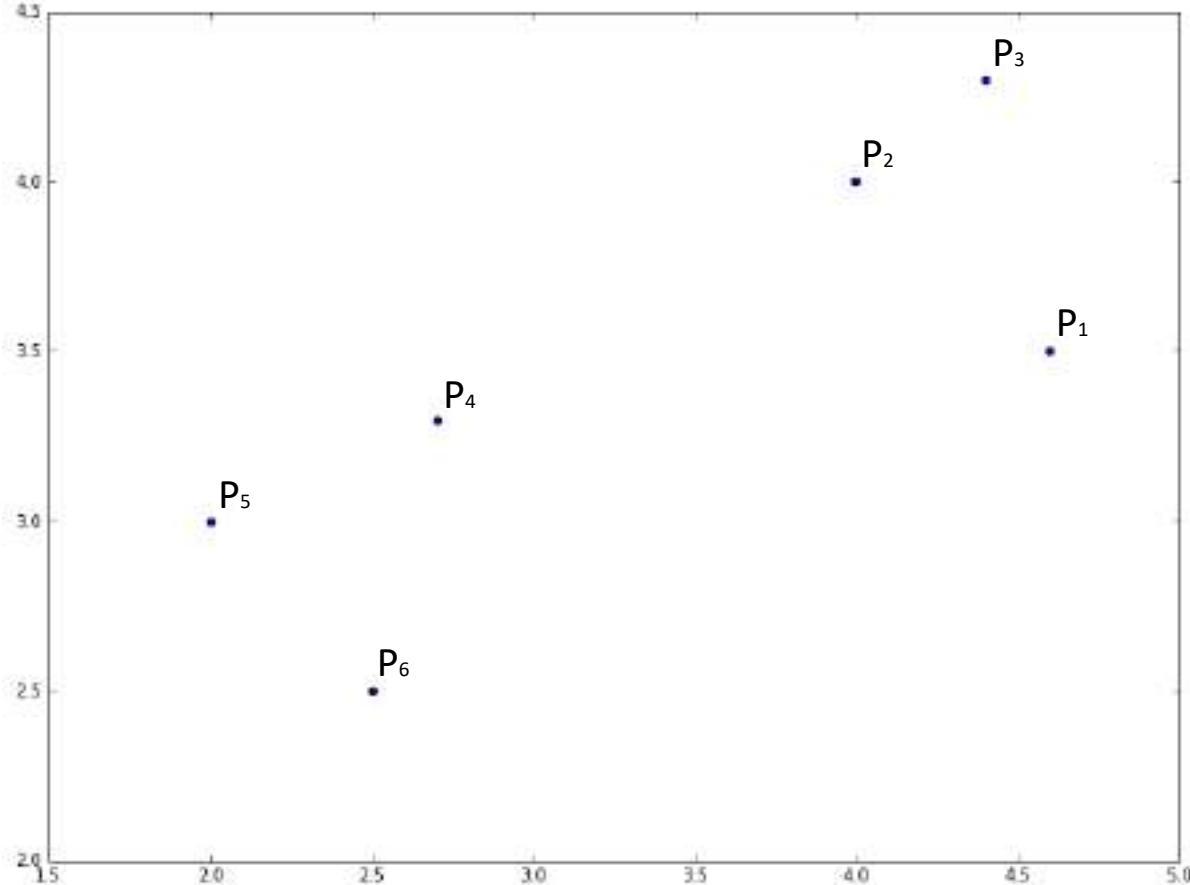


How Do Dendograms Work?

NOT FOR DISTRIBUTION

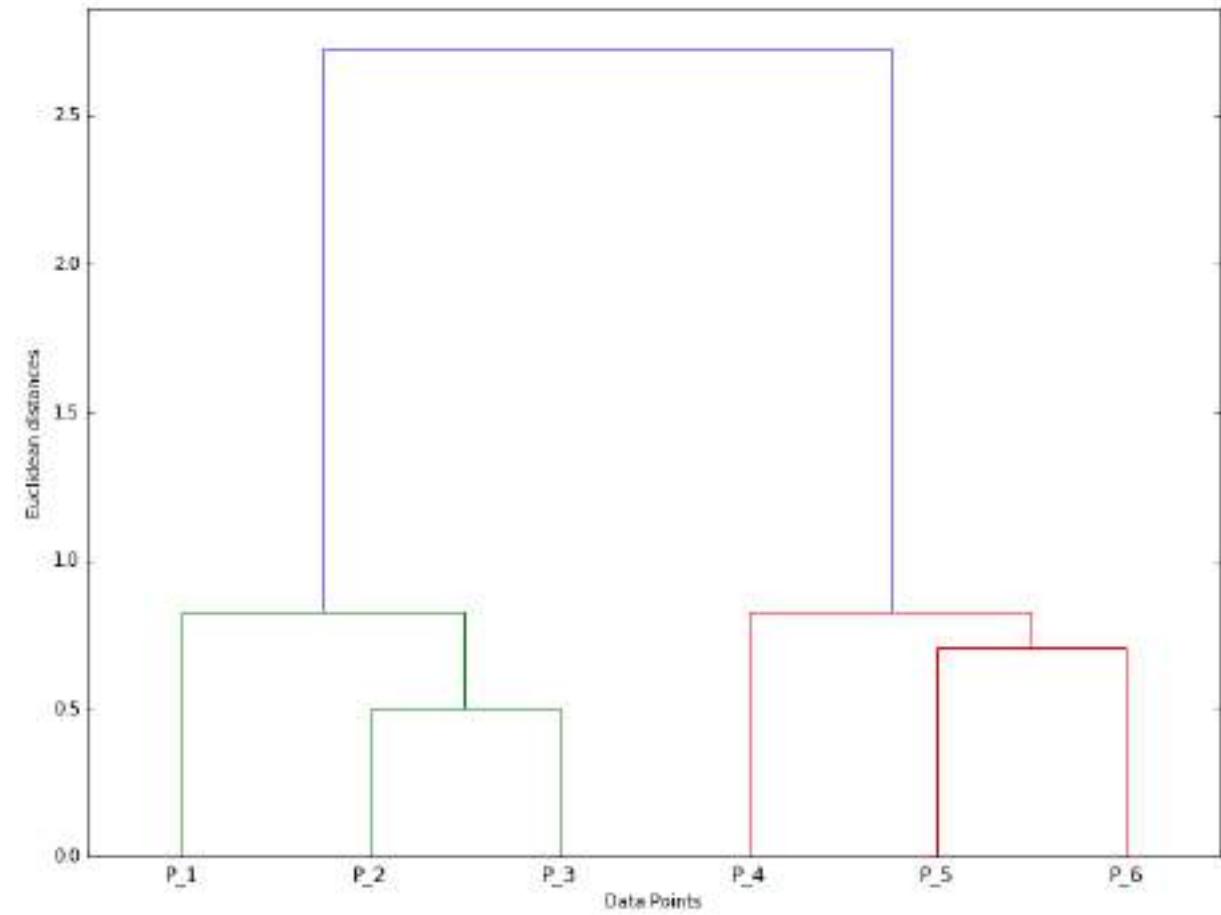
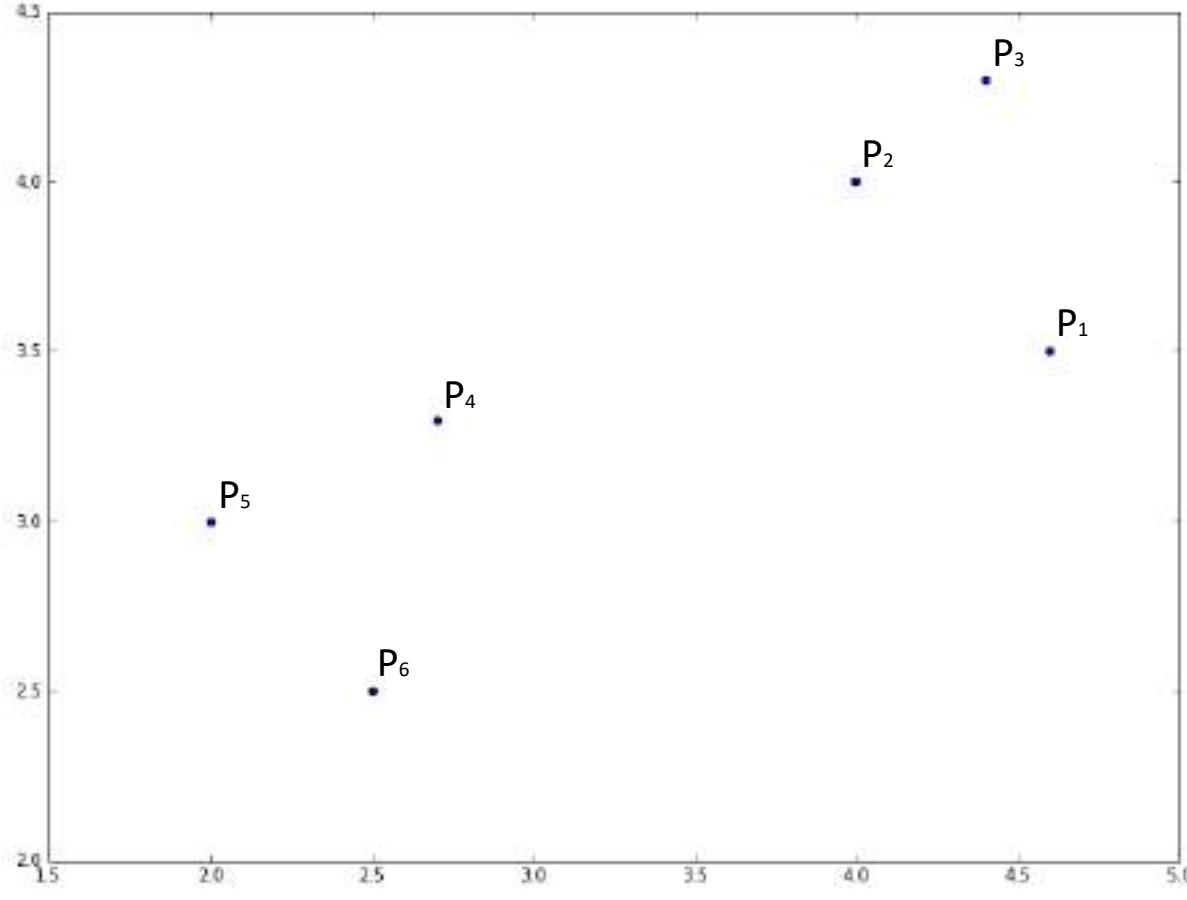


How Do Dendograms Work?

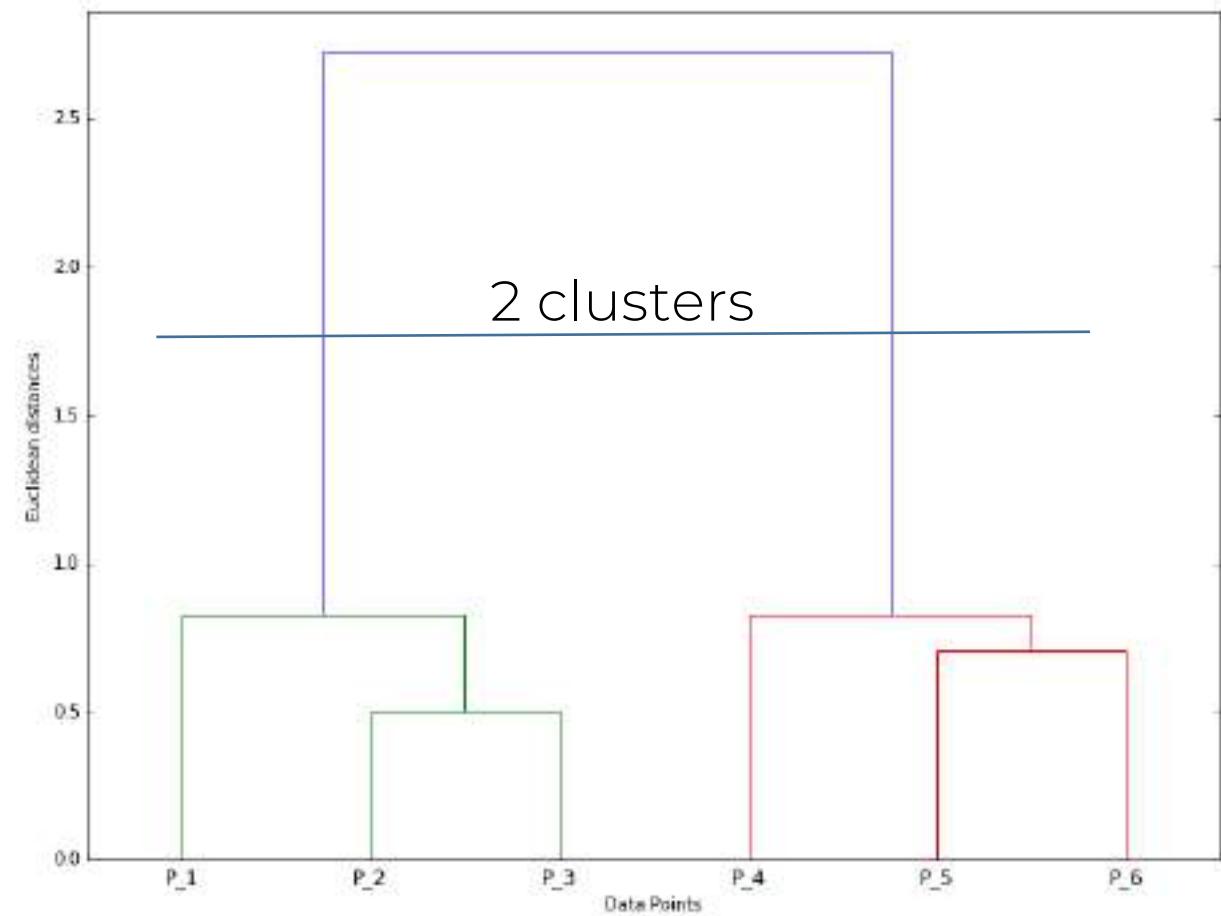
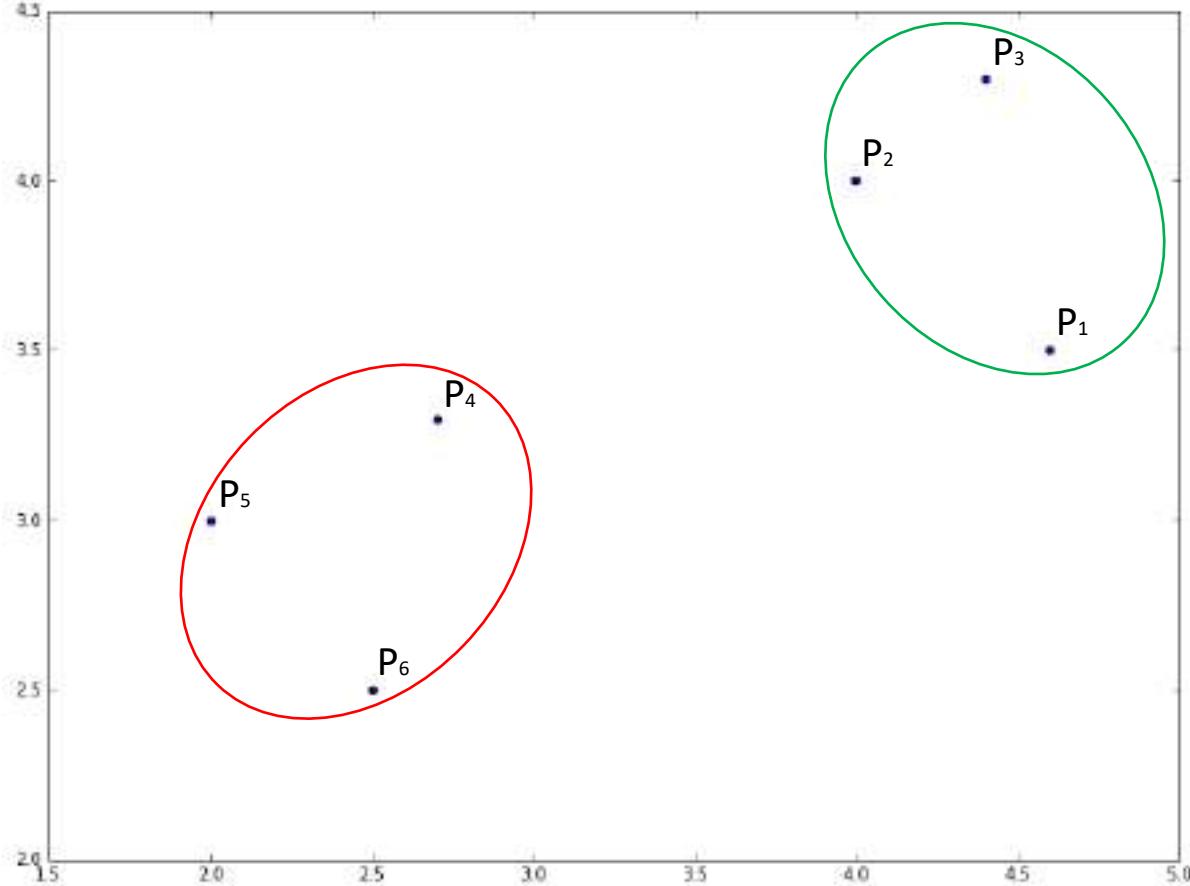


HC Intuition: Using Dendograms

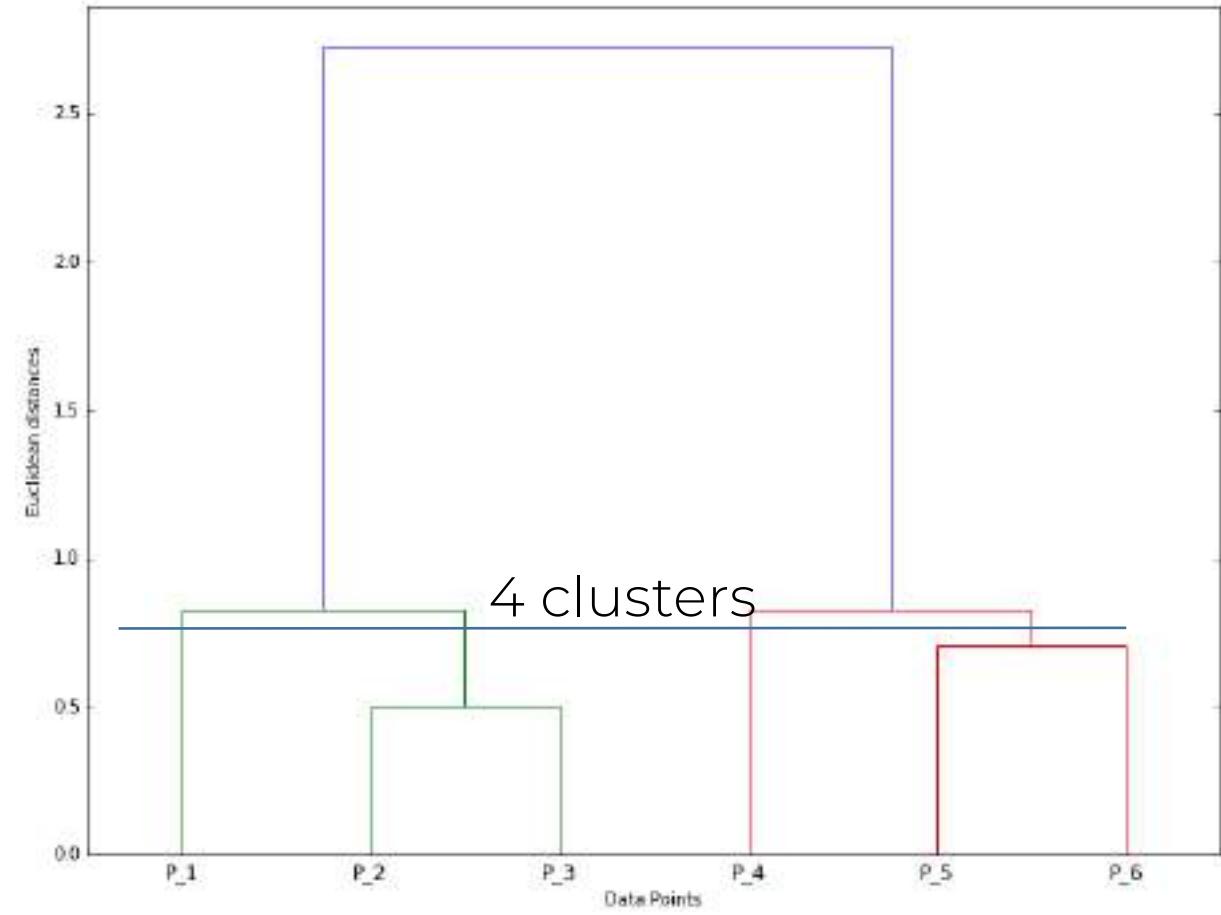
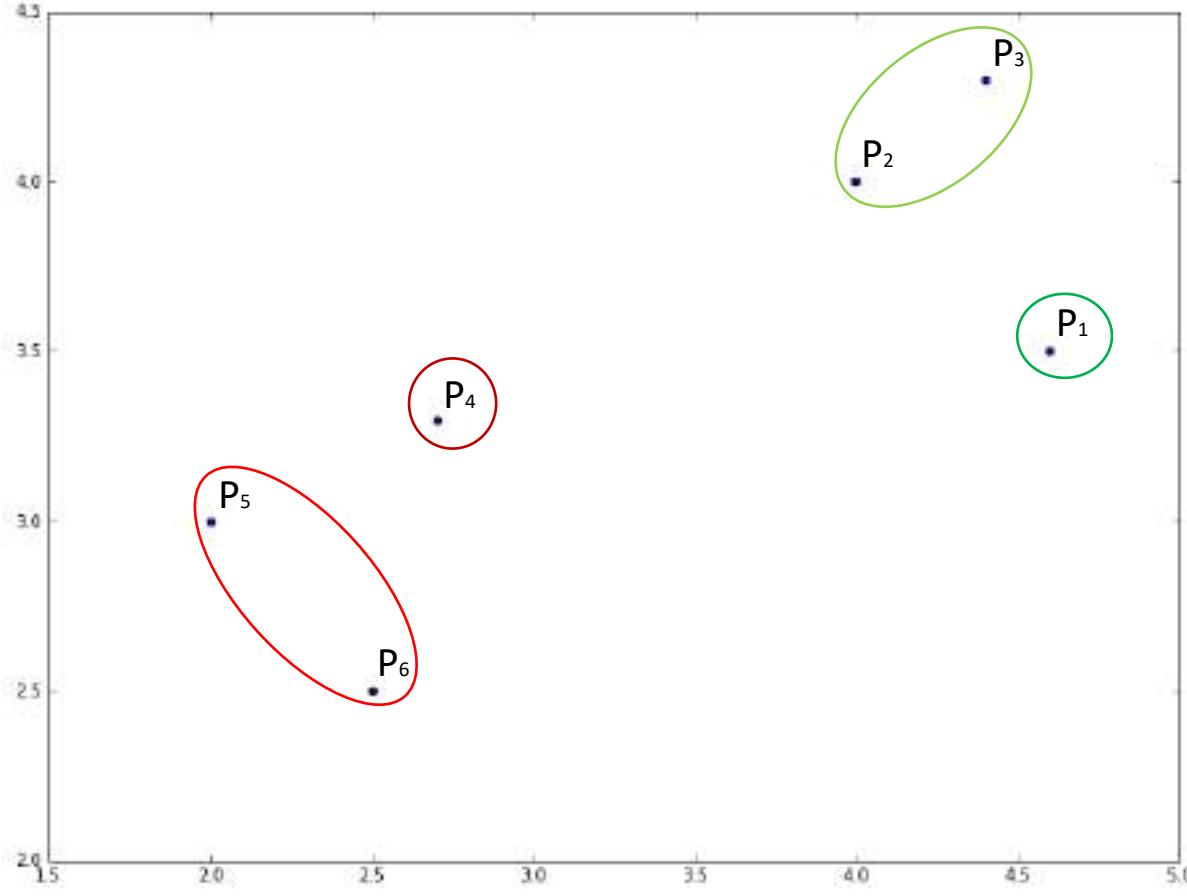
Dendrograms – Two Clusters



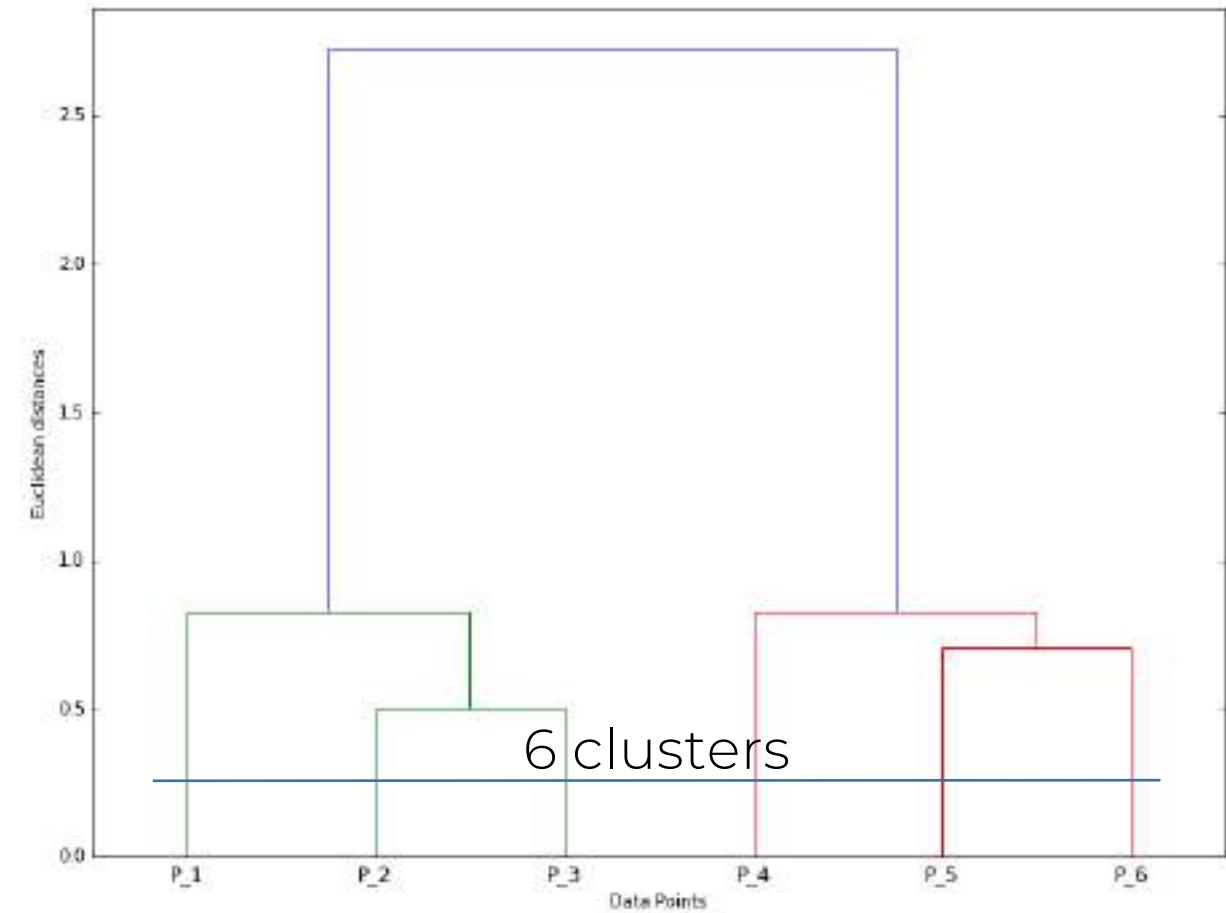
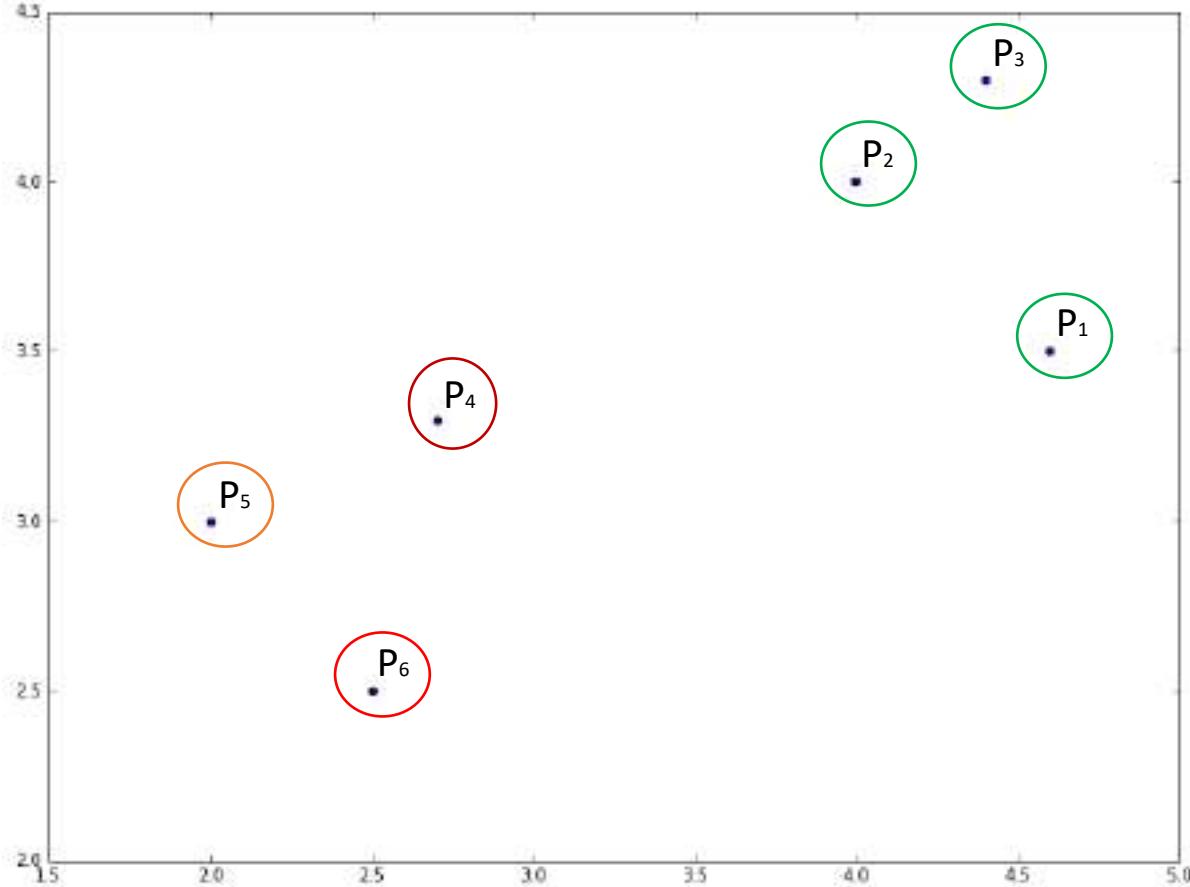
Dendrograms – Two Clusters



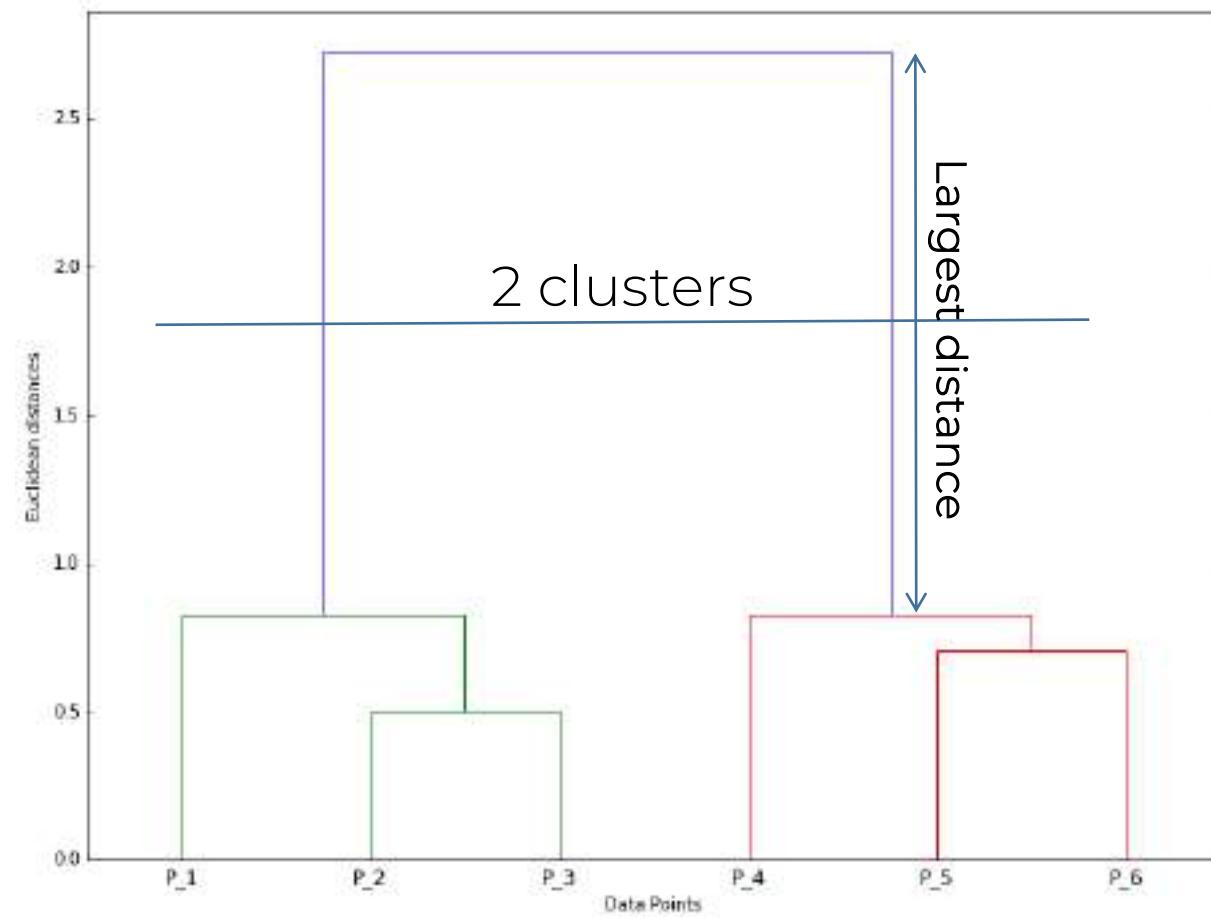
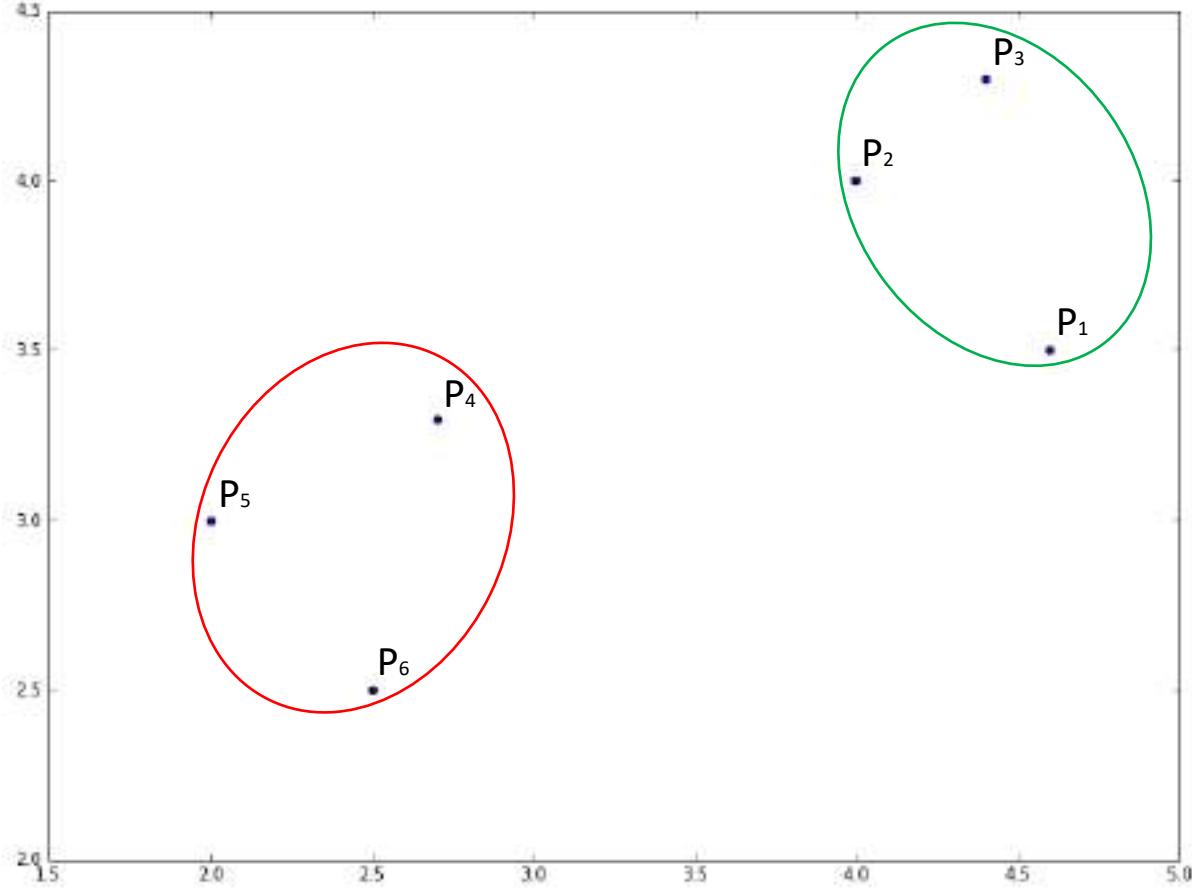
Dendrograms – Four Clusters



Dendrograms – Six Clusters

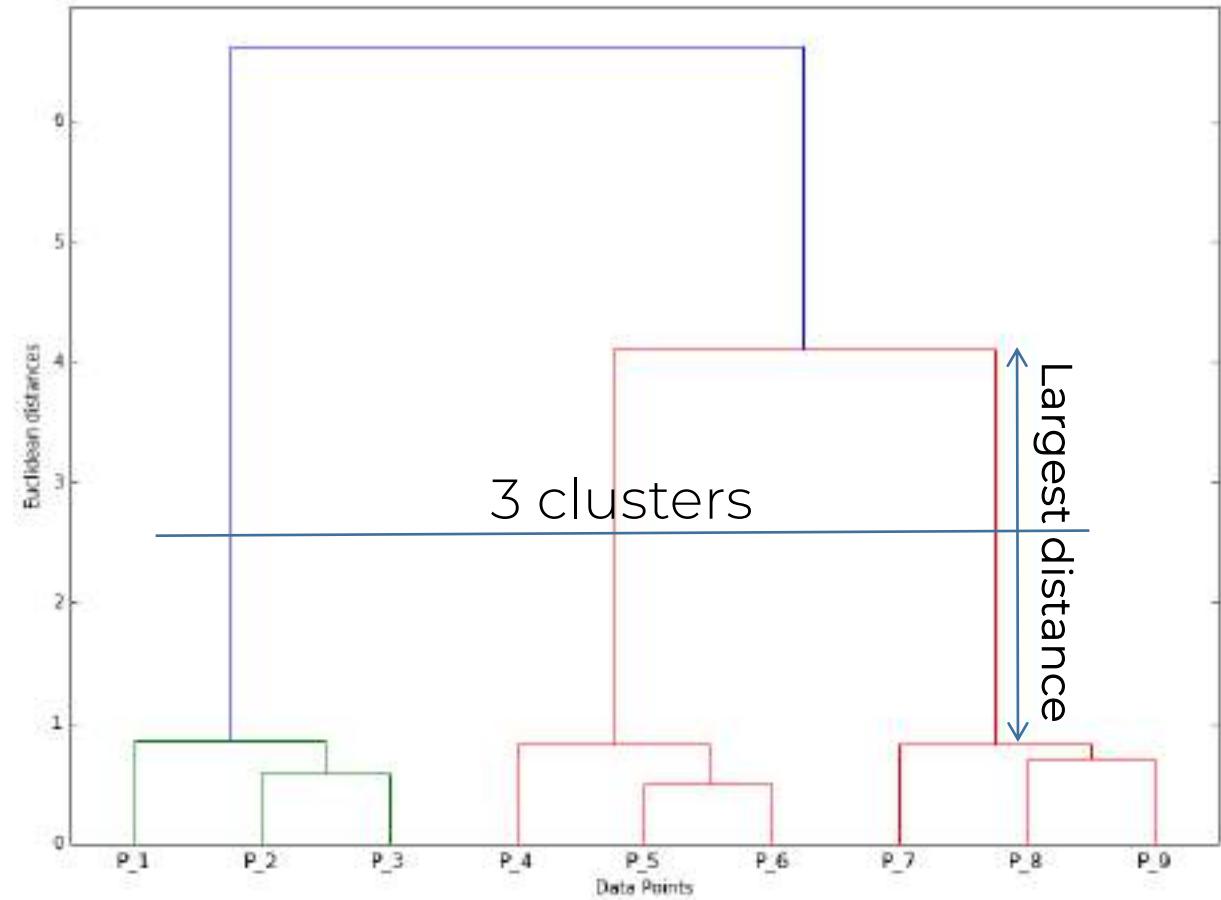
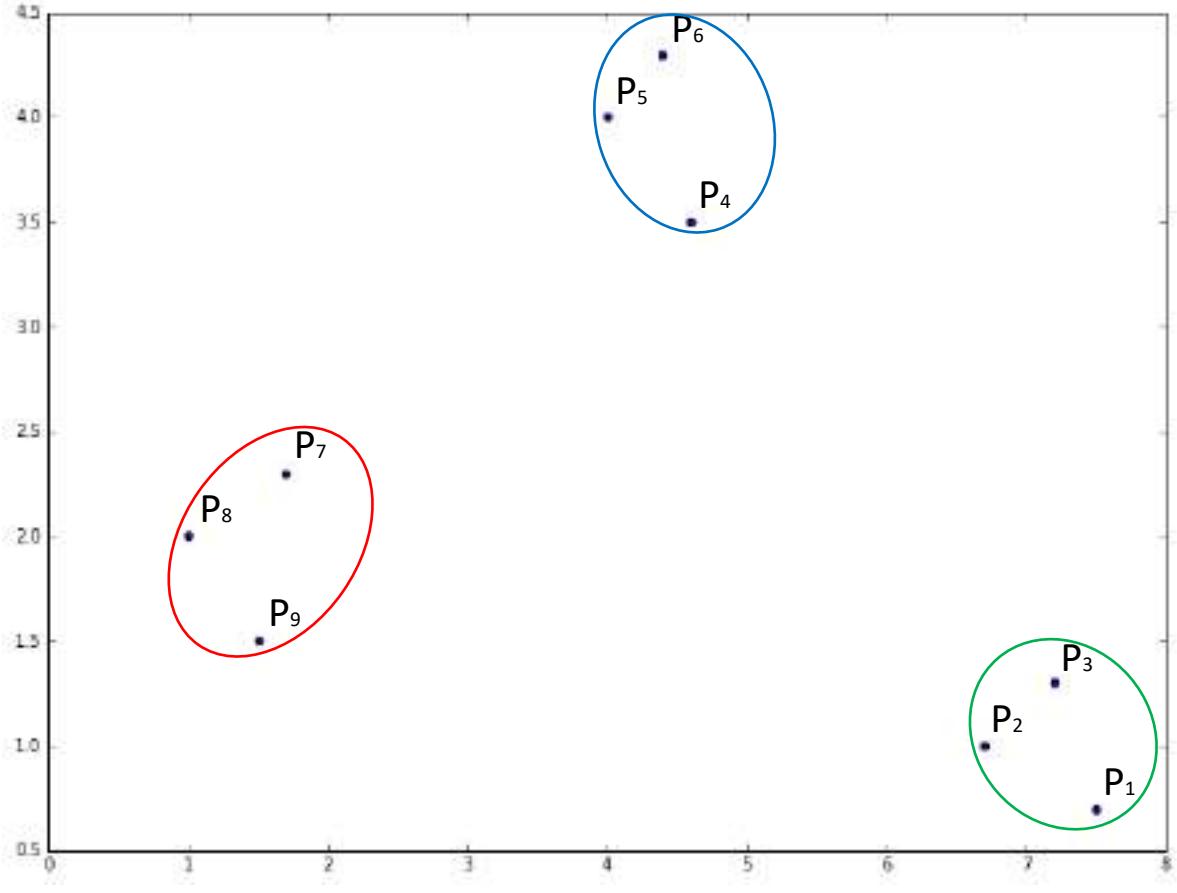


Dendograms – Optimal # of Clusters



Knowledge Test

Dendograms – Knowledge Test



Association Rule Learning

Apriori Intuition

ARL - What is it all about ?

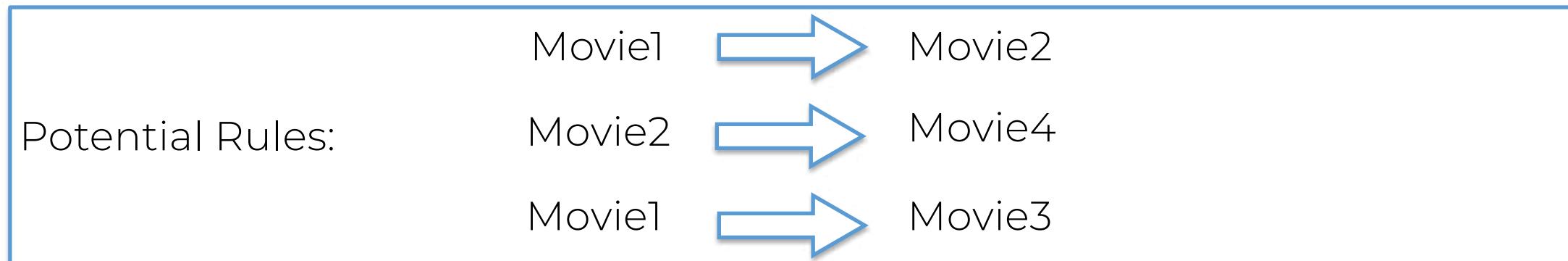


ARL - What is it all about ?

People who bought also bought ...

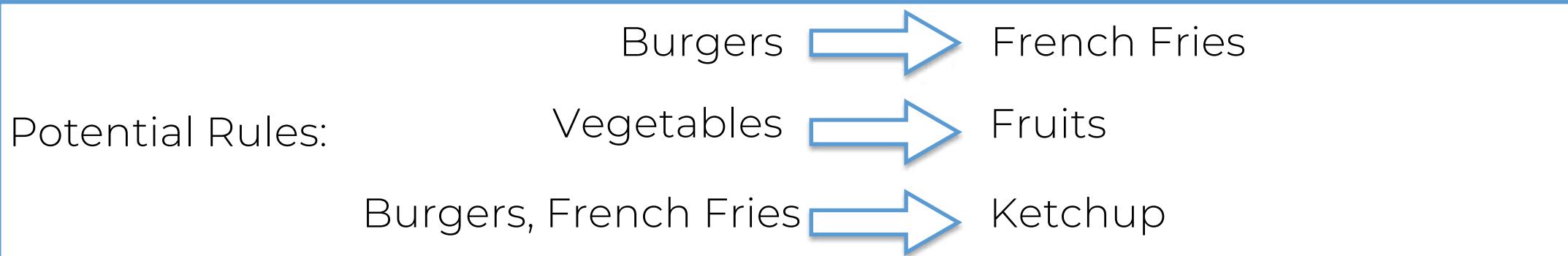
ARL- Movie Recommendation

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



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



Apriori - Support

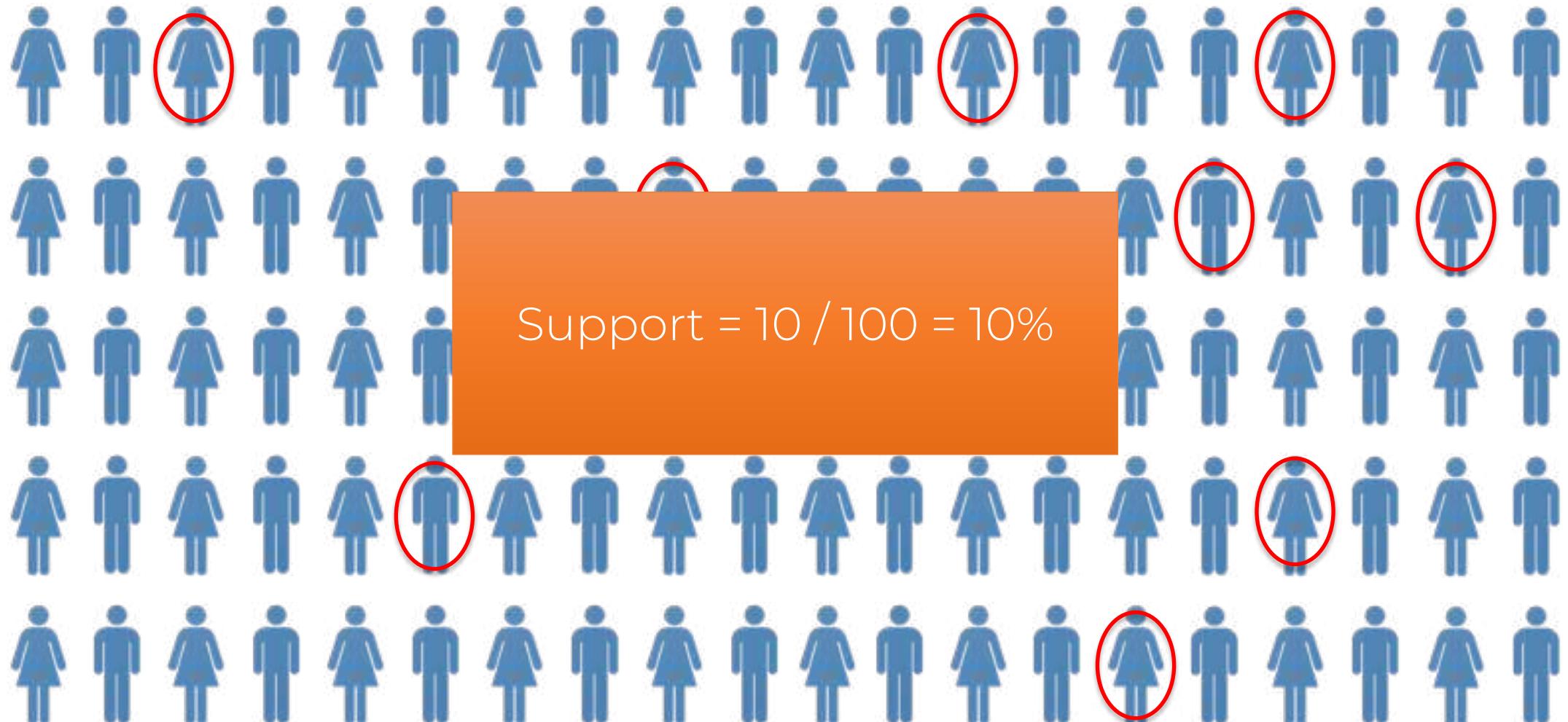
Movie Recommendation:

$$\text{support}(\mathbf{M}) = \frac{\# \text{ user watchlists containing } \mathbf{M}}{\# \text{ user watchlists}}$$

Market Basket Optimisation:

$$\text{support}(\mathbf{I}) = \frac{\# \text{ transactions containing } \mathbf{I}}{\# \text{ transactions}}$$

Apriori - Support



Apriori- Confidence

Movie Recommendation: $\text{confidence}(\mathbf{M}_1 \rightarrow \mathbf{M}_2) = \frac{\# \text{ user watchlists containing } \mathbf{M}_1 \text{ and } \mathbf{M}_2}{\# \text{ user watchlists containing } \mathbf{M}_1}$

Market Basket Optimisation: $\text{confidence}(\mathbf{l}_1 \rightarrow \mathbf{l}_2) = \frac{\# \text{ transactions containing } \mathbf{l}_1 \text{ and } \mathbf{l}_2}{\# \text{ transactions containing } \mathbf{l}_1}$

Apriori - Confidence



Apriori - Confidence



Apriori - Lift

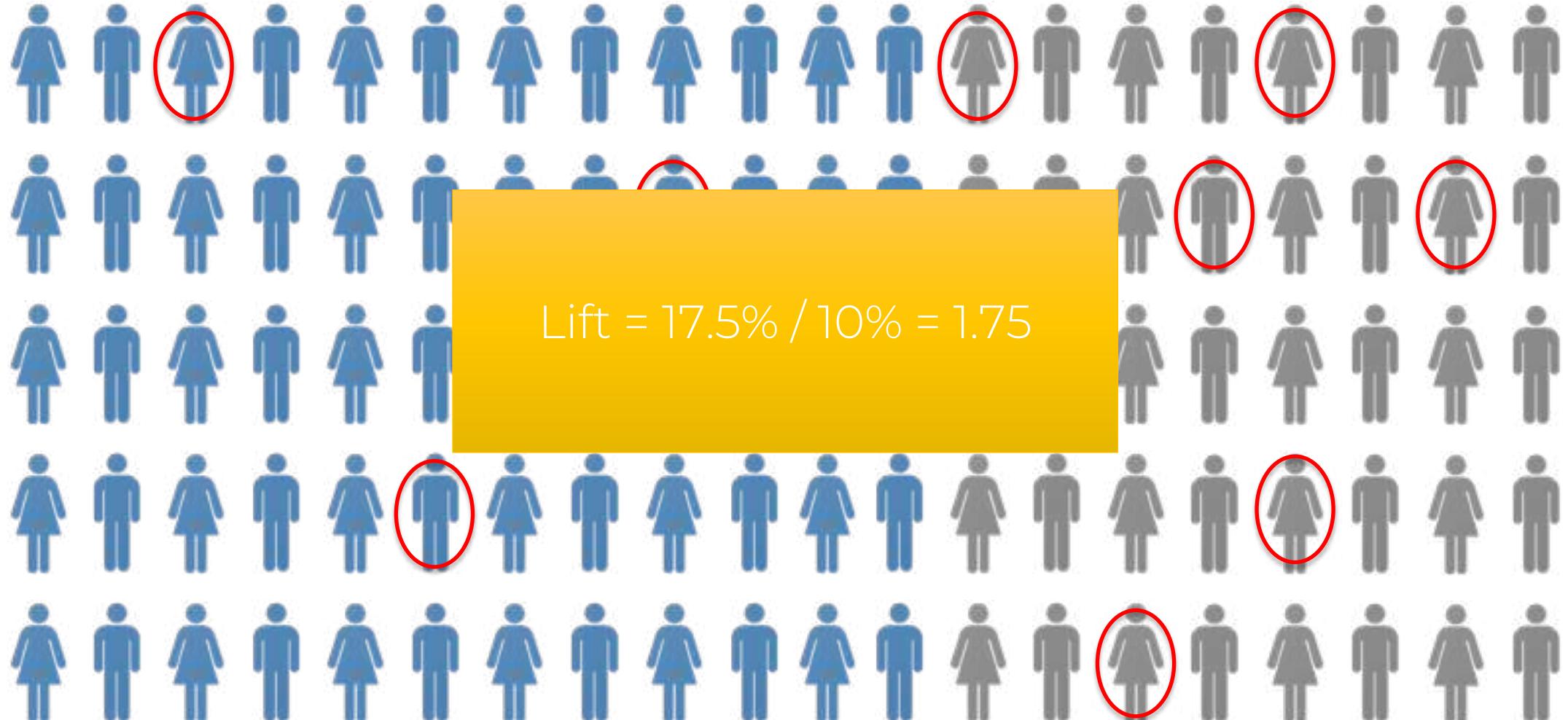
Movie Recommendation:

$$\text{lift}(\mathcal{M}_1 \rightarrow \mathcal{M}_2) = \frac{\text{confidence}(\mathcal{M}_1 \rightarrow \mathcal{M}_2)}{\text{support}(\mathcal{M}_2)}$$

Market Basket Optimisation:

$$\text{lift}(I_1 \rightarrow I_2) = \frac{\text{confidence}(I_1 \rightarrow I_2)}{\text{support}(I_2)}$$

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

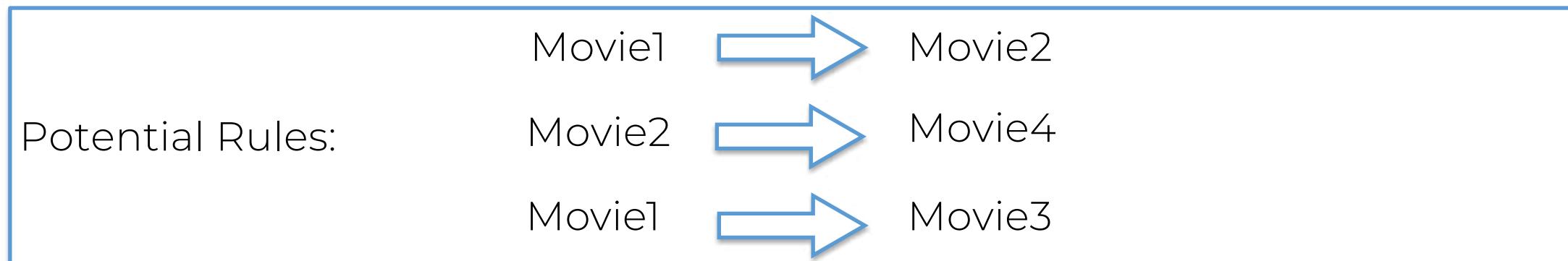
Eclat Intuition

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



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

Eclat - Support

Movie Recommendation:

$$\text{support}(\mathbf{M}) = \frac{\# \text{ user watchlists containing } \mathbf{M}}{\# \text{ user watchlists}}$$

Market Basket Optimisation:

$$\text{support}(\mathbf{I}) = \frac{\# \text{ transactions containing } \mathbf{I}}{\# \text{ transactions}}$$

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

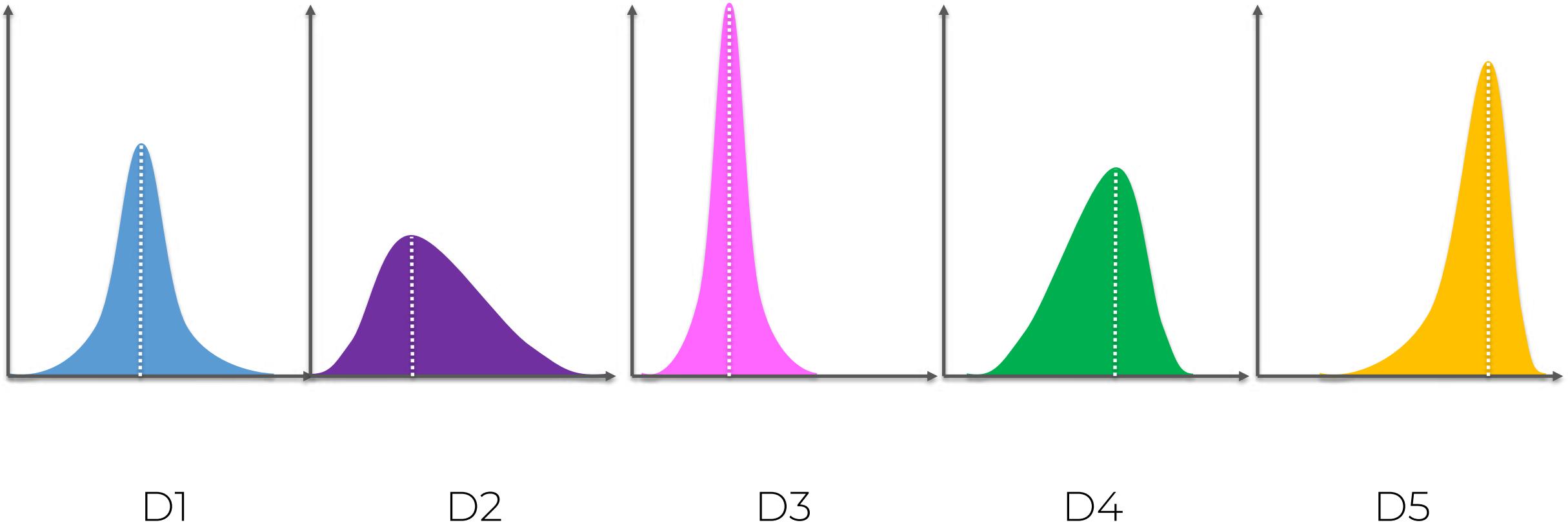
D2

D3

D4

D5

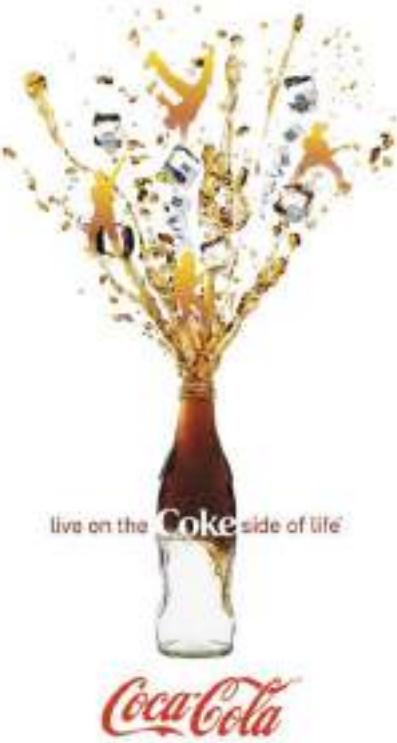
The Multi-Armed Bandit Problem



The Multi-Armed Bandit Problem



D1



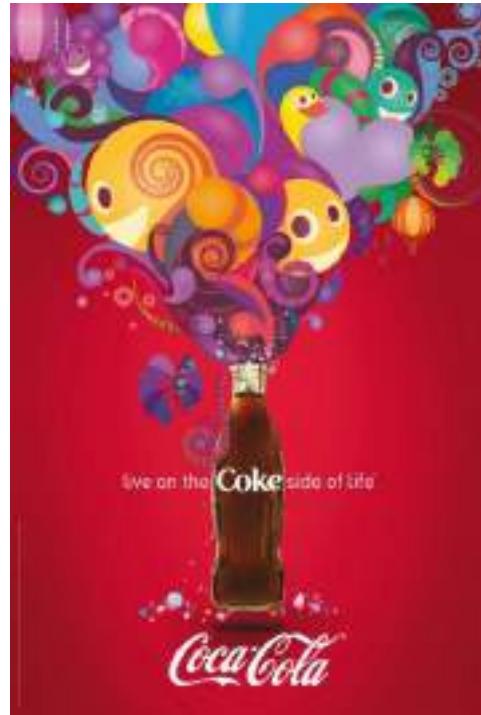
D2



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

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

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

Upper Confidence Bound Algorithm



D1

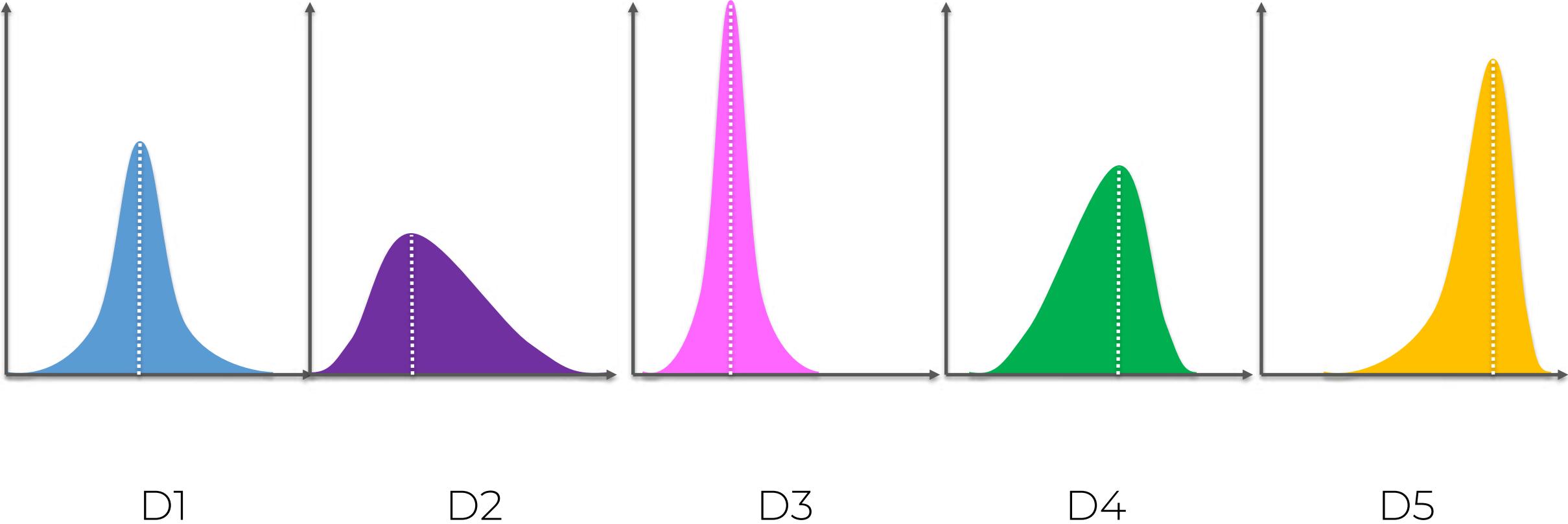
D2

D3

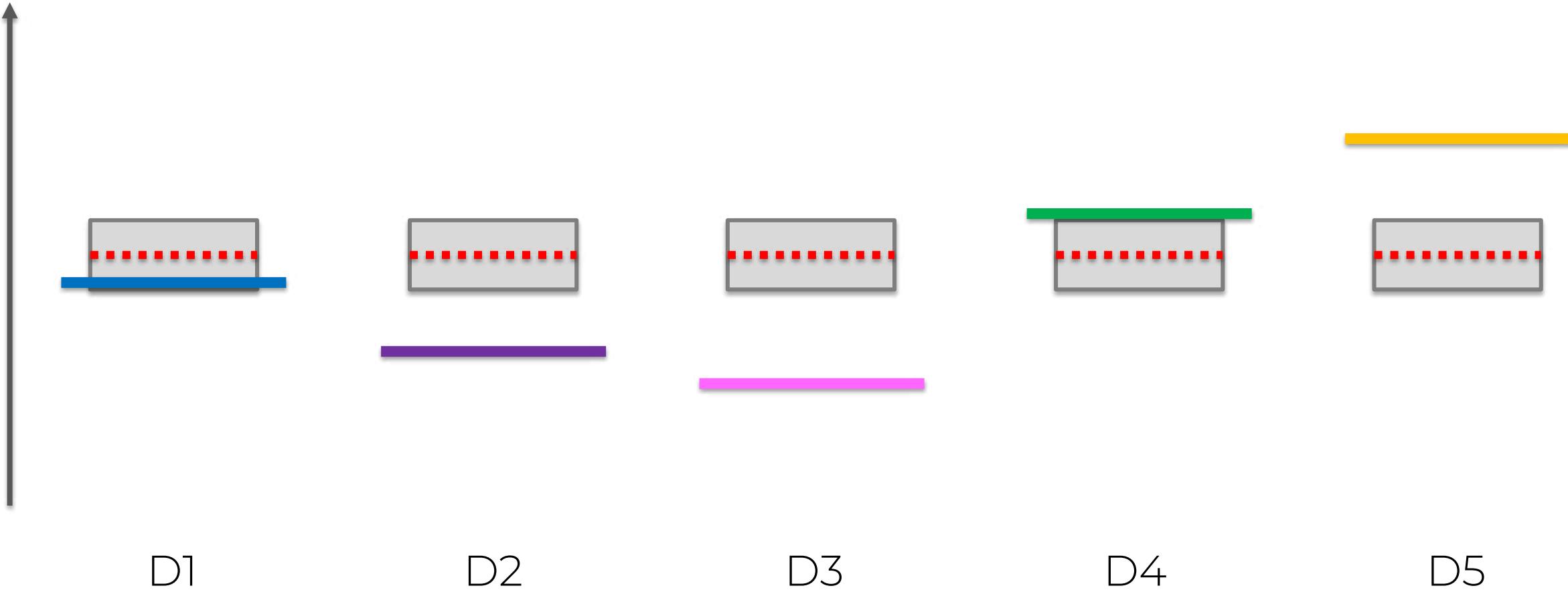
D4

D5

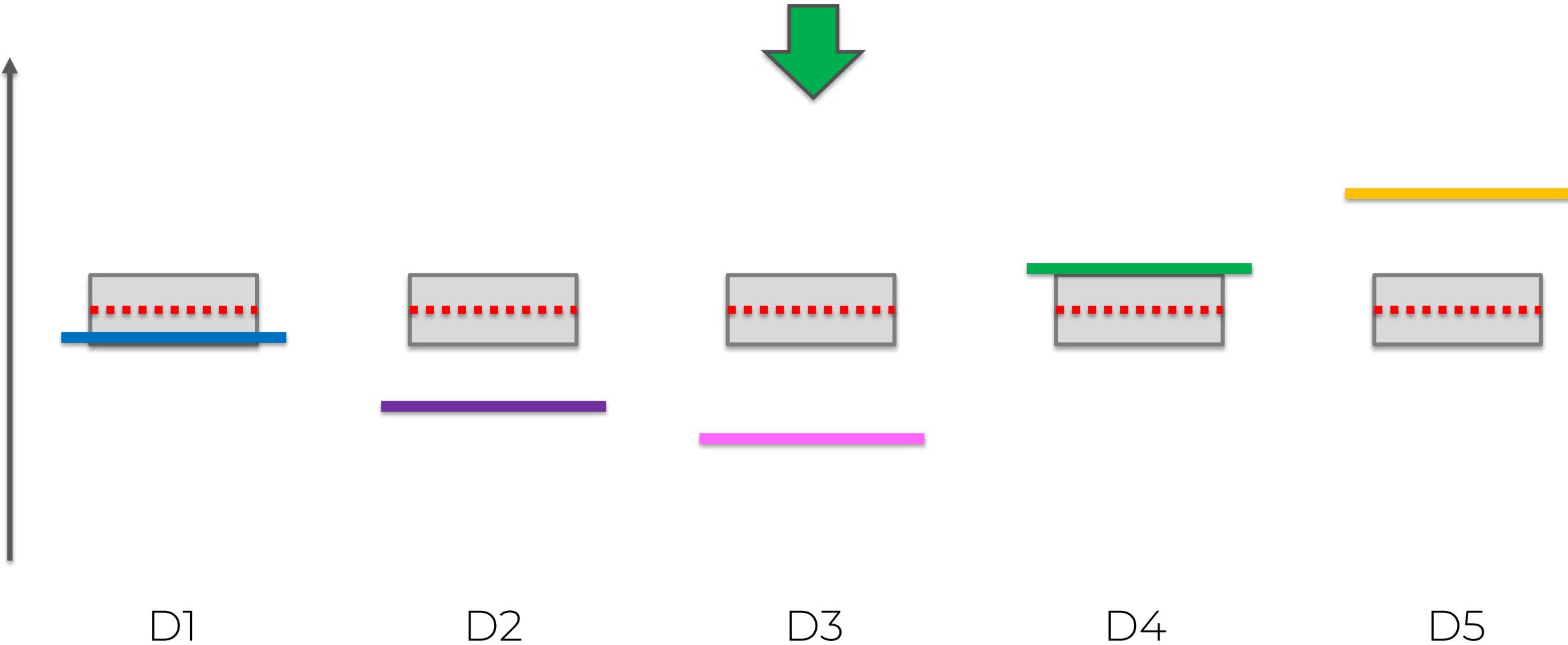
Upper Confidence Bound Algorithm



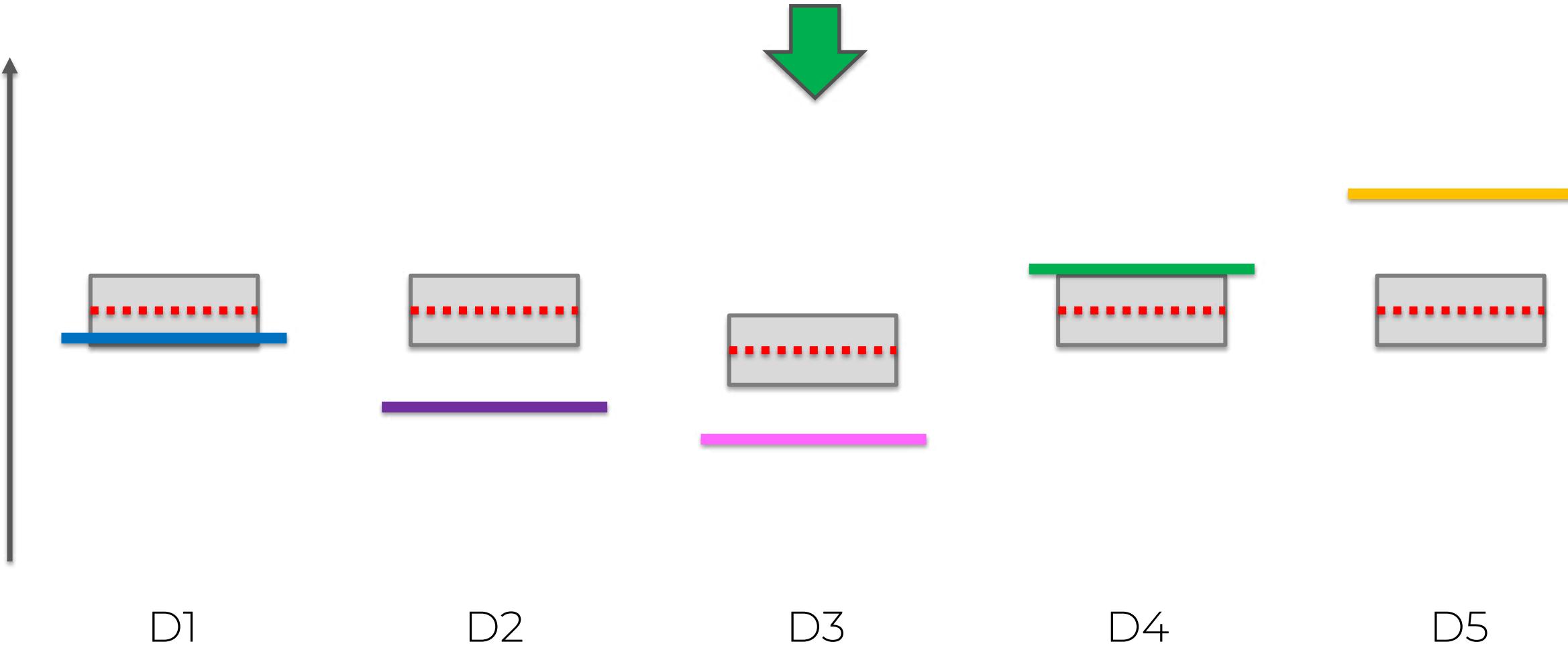
Upper Confidence Bound Algorithm



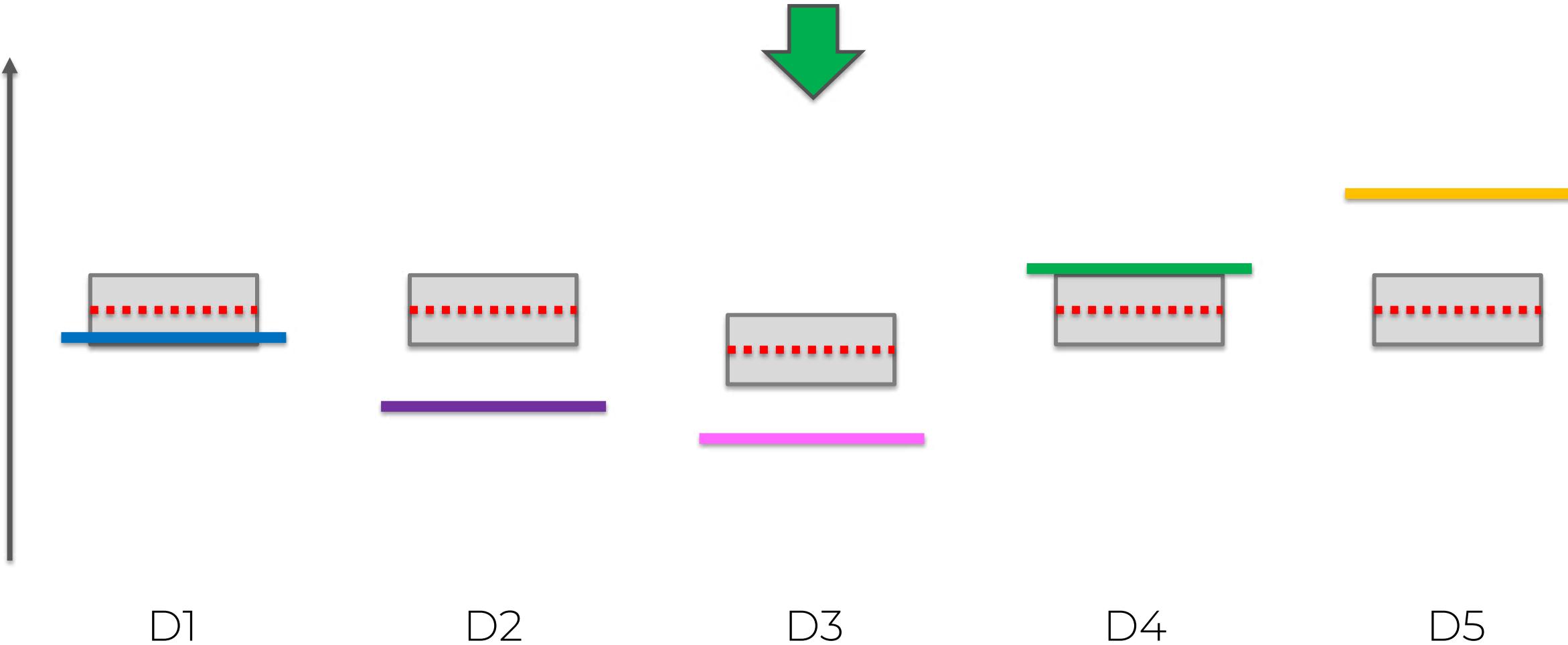
Upper Confidence Bound Algorithm



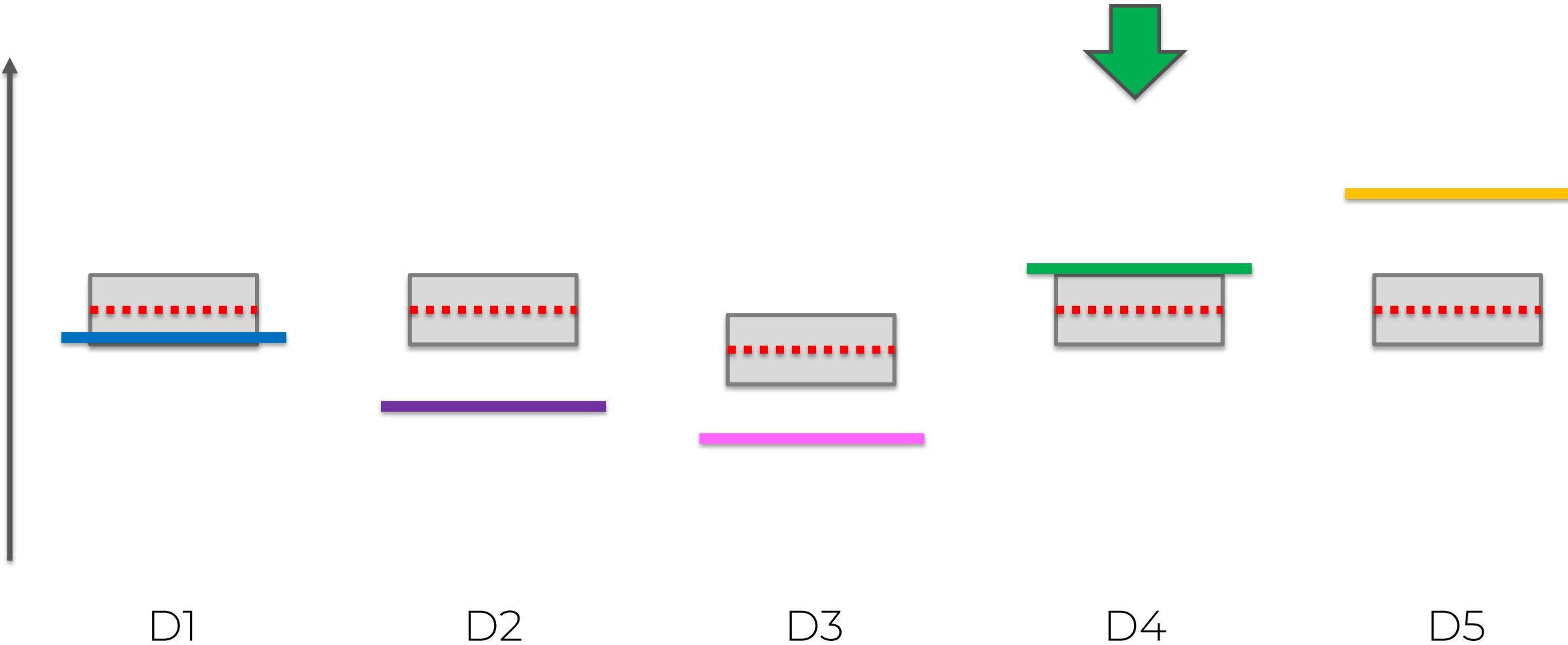
Upper Confidence Bound Algorithm



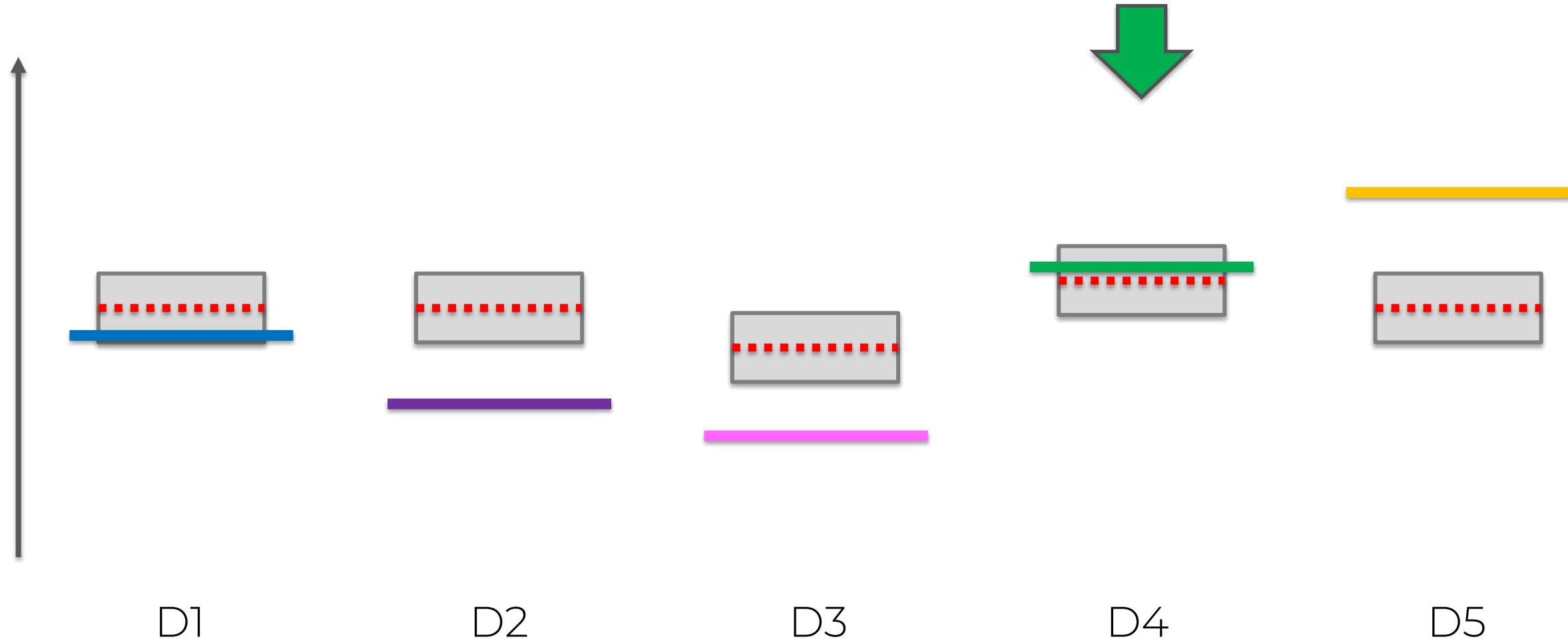
Upper Confidence Bound Algorithm



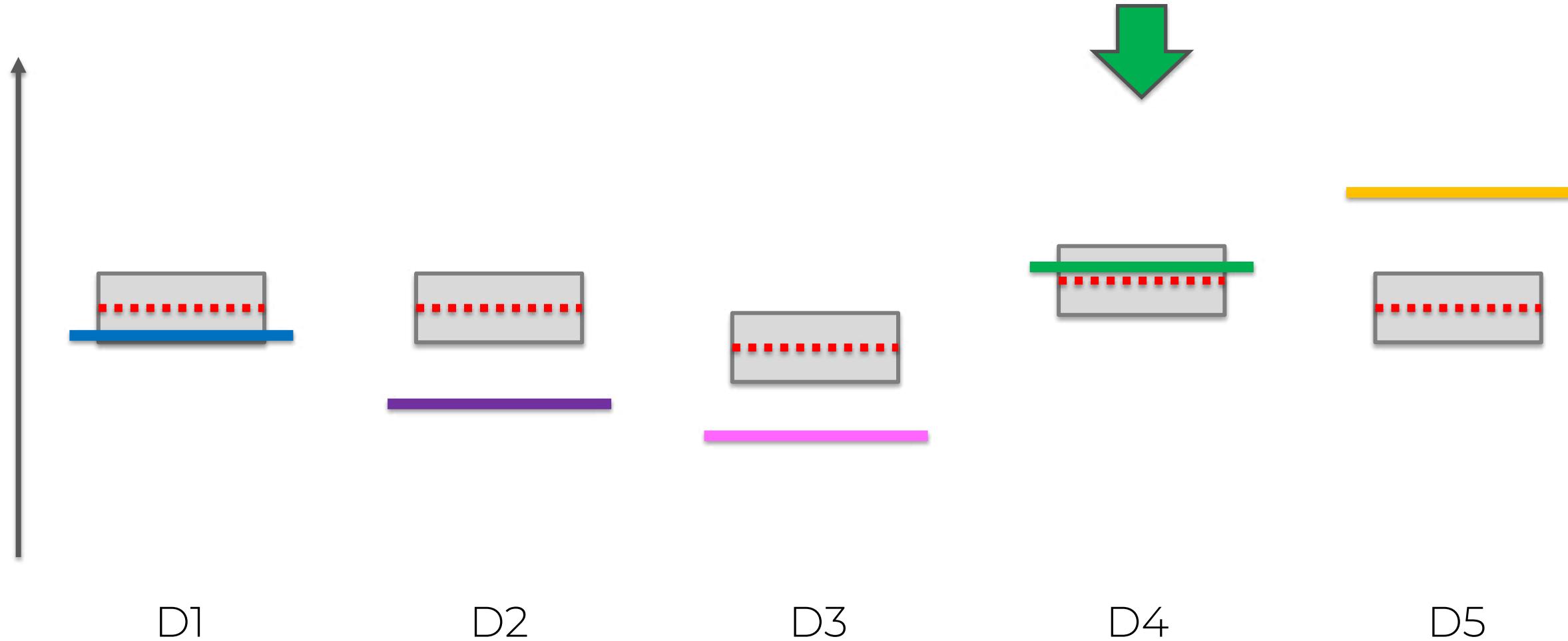
Upper Confidence Bound Algorithm



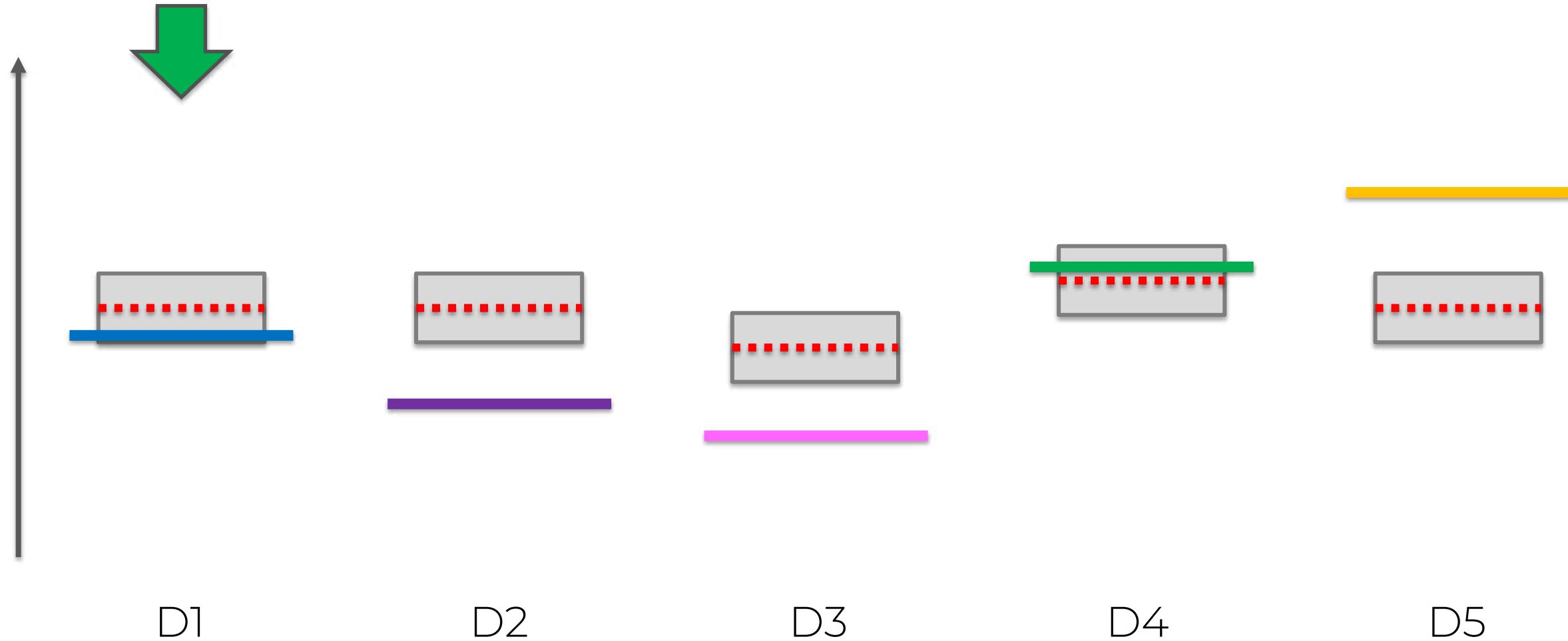
Upper Confidence Bound Algorithm



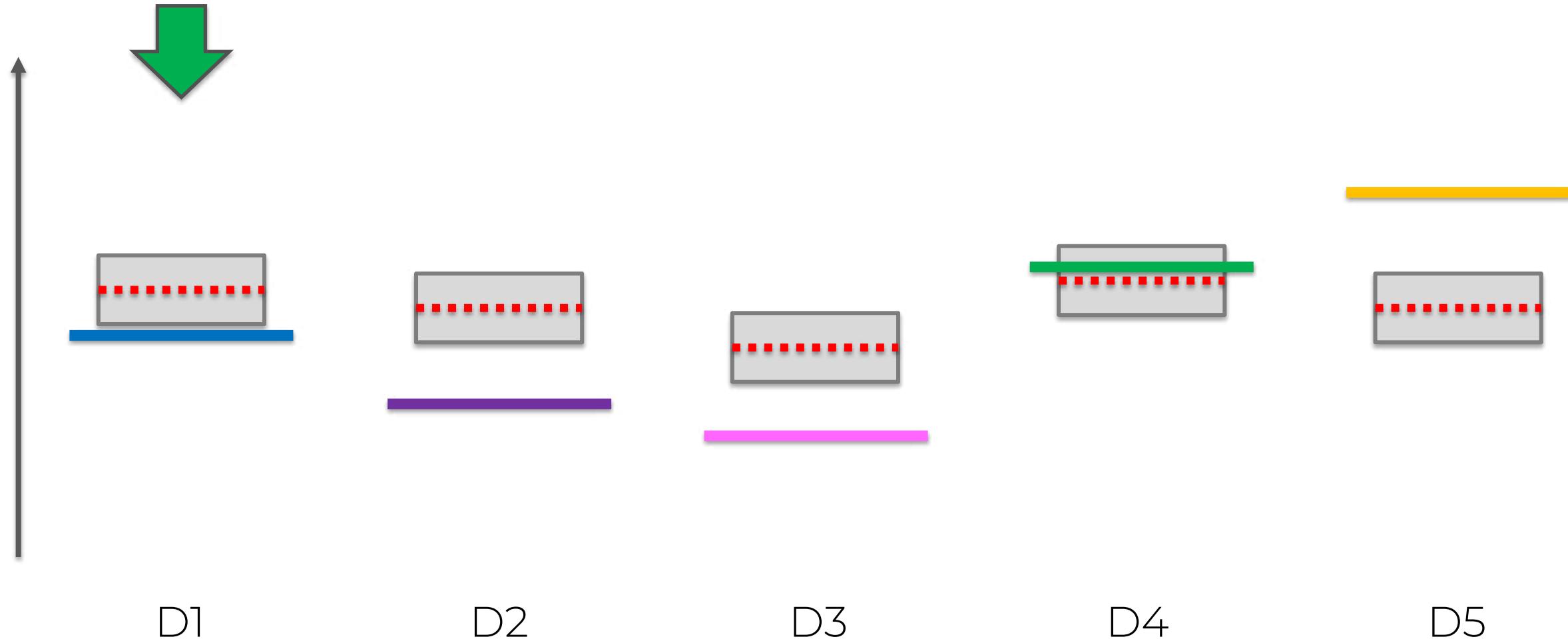
Upper Confidence Bound Algorithm



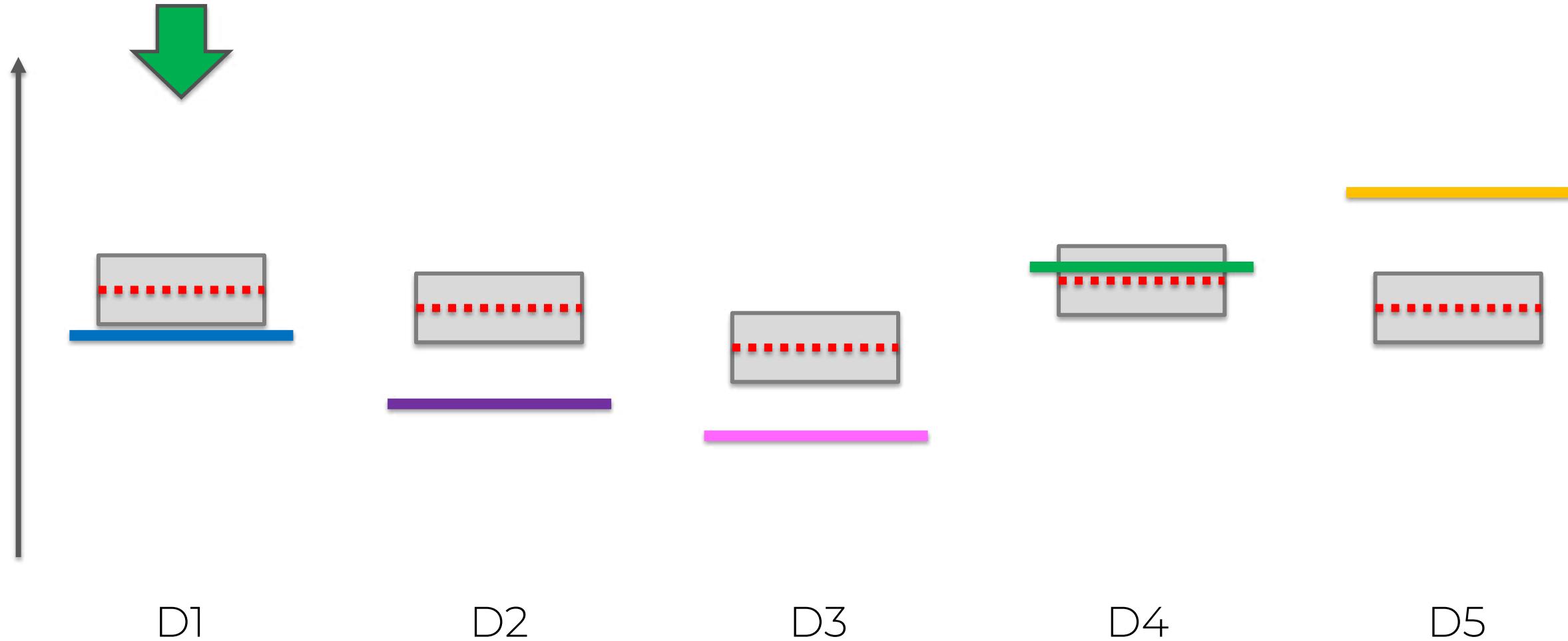
Upper Confidence Bound Algorithm



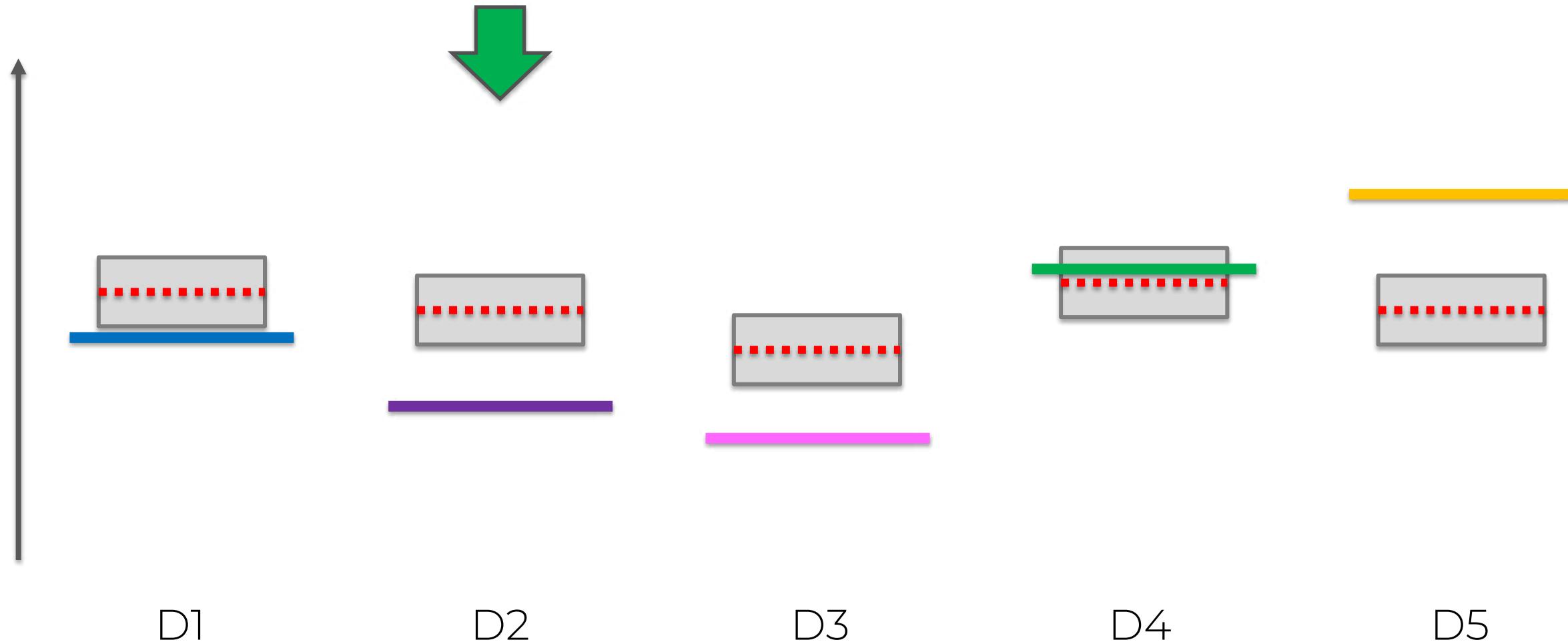
Upper Confidence Bound Algorithm



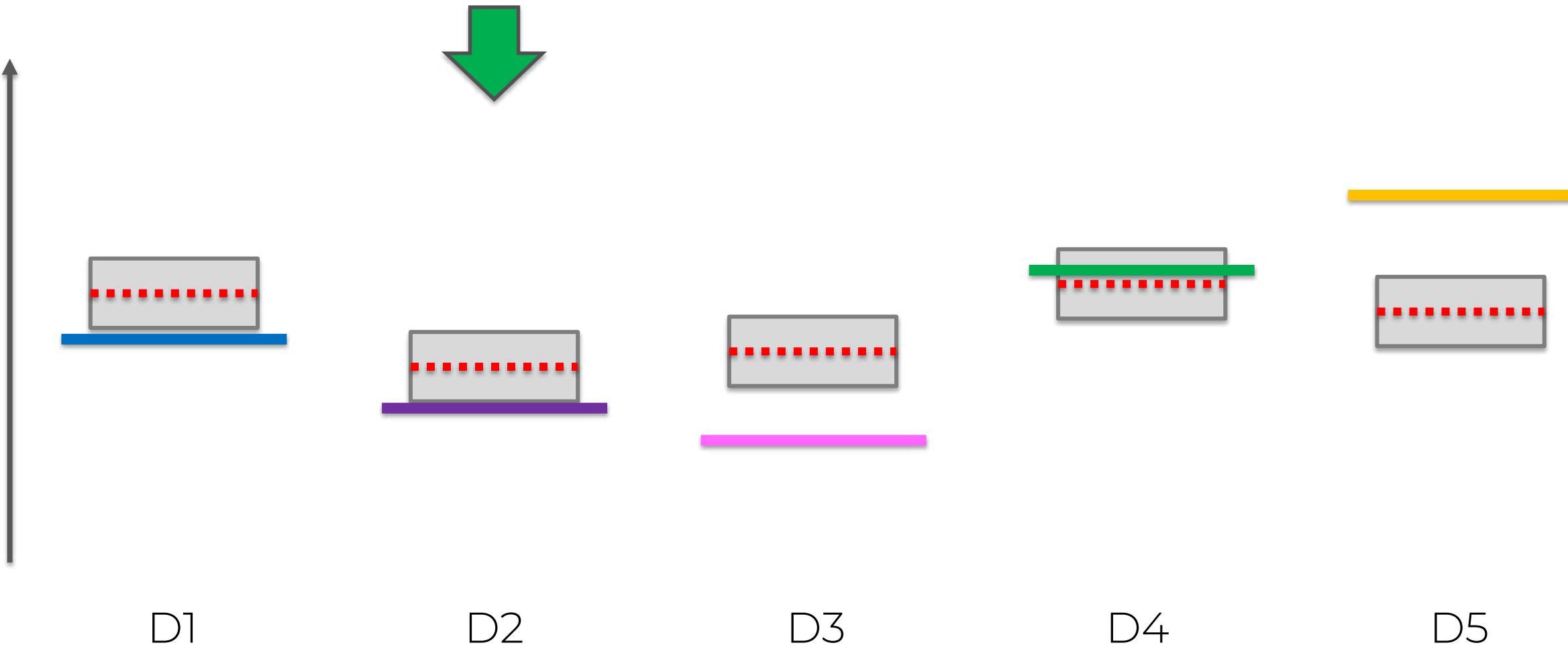
Upper Confidence Bound Algorithm



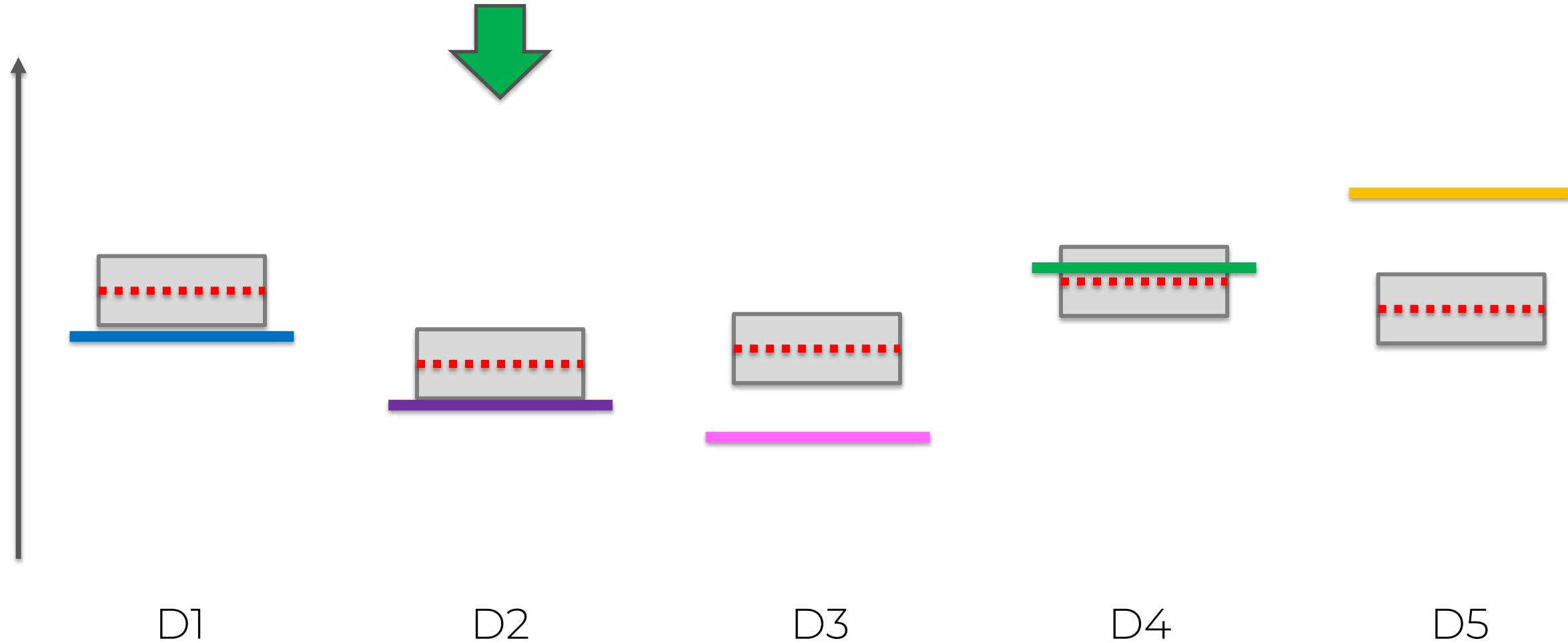
Upper Confidence Bound Algorithm



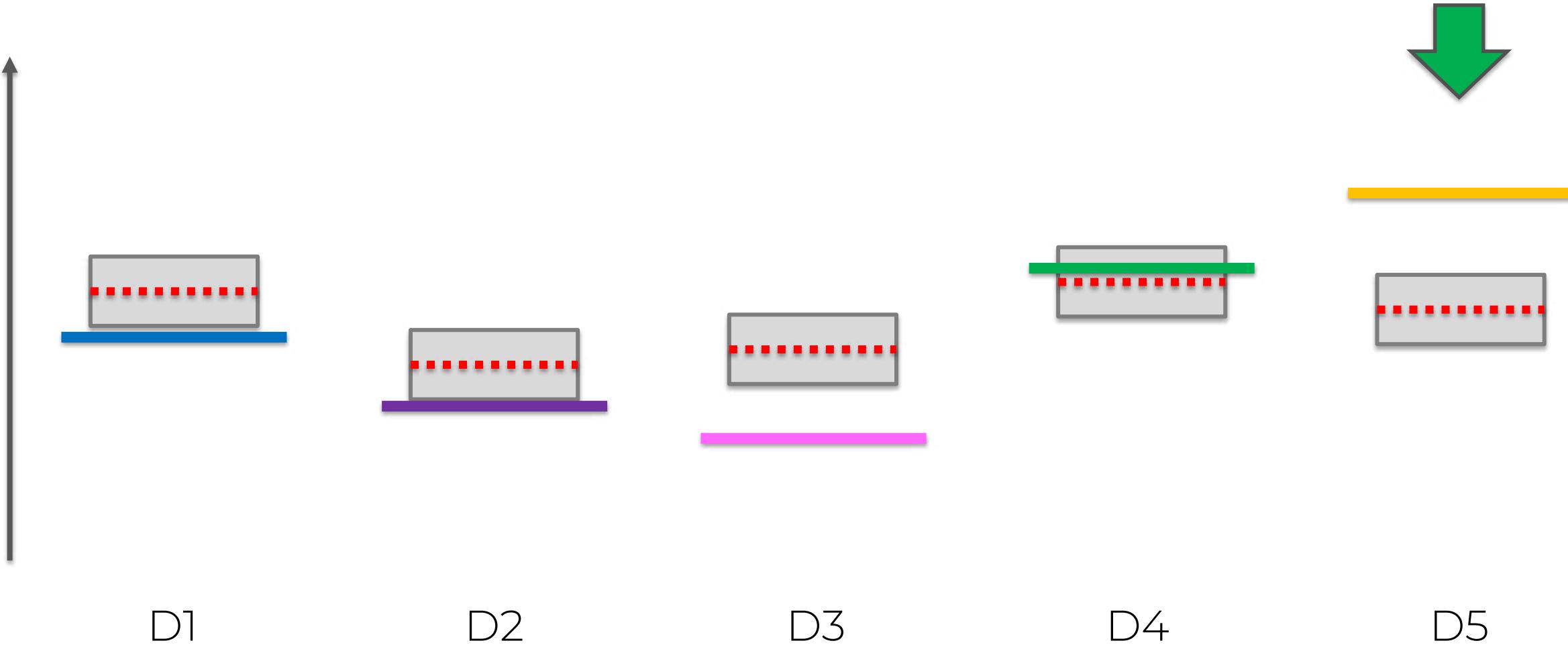
Upper Confidence Bound Algorithm



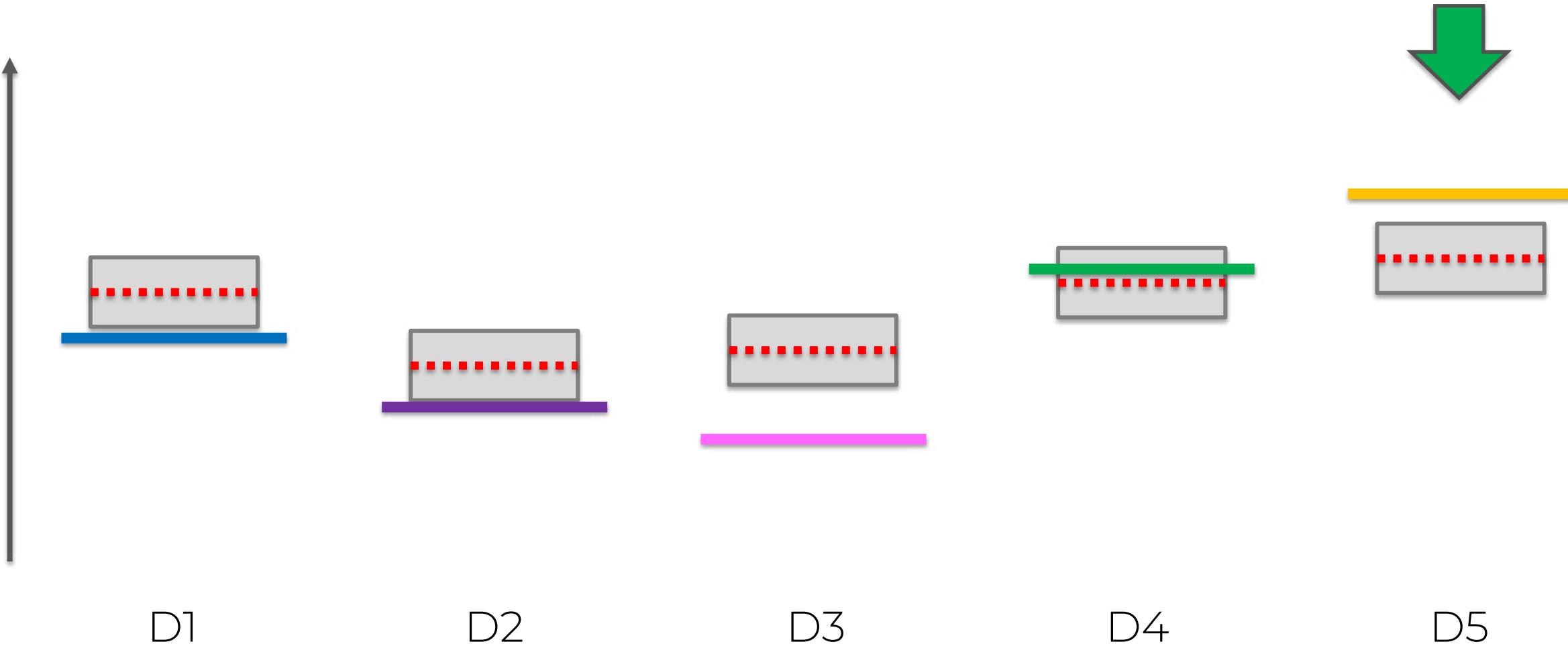
Upper Confidence Bound Algorithm



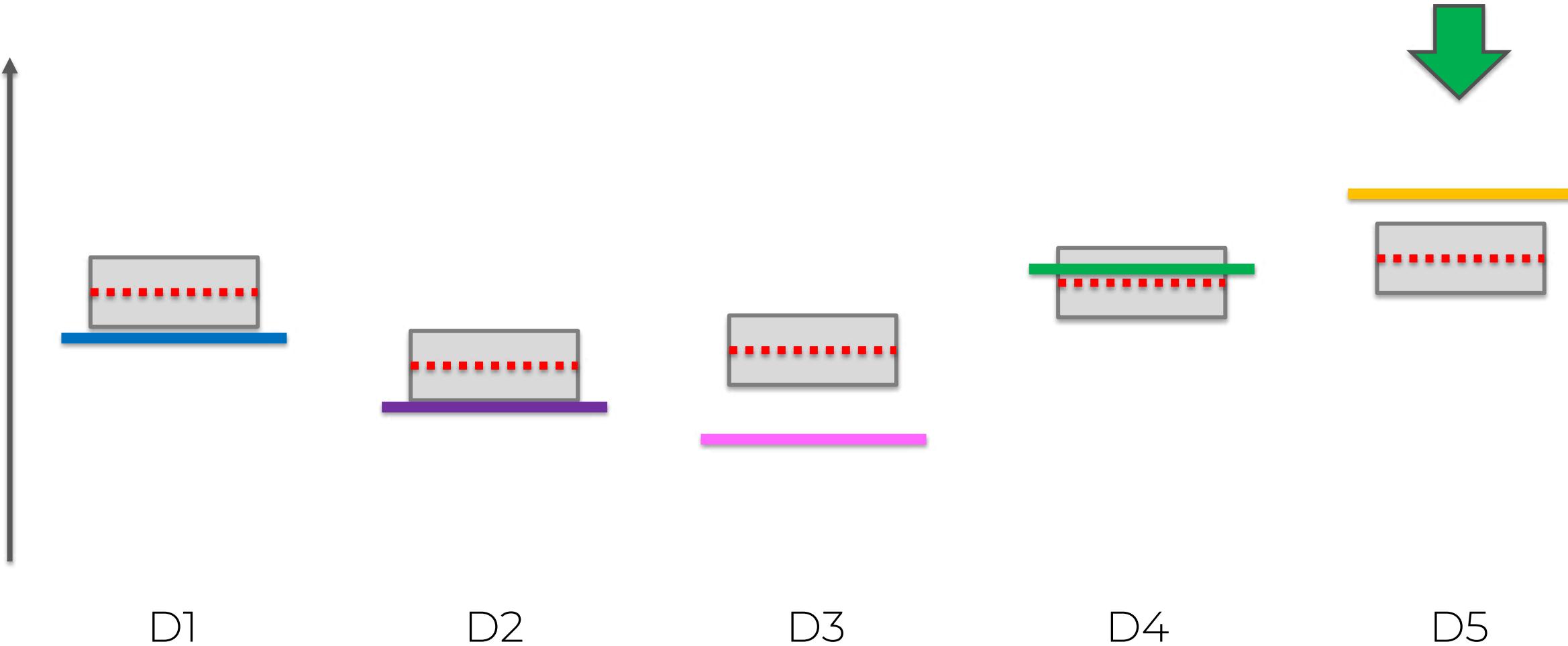
Upper Confidence Bound Algorithm



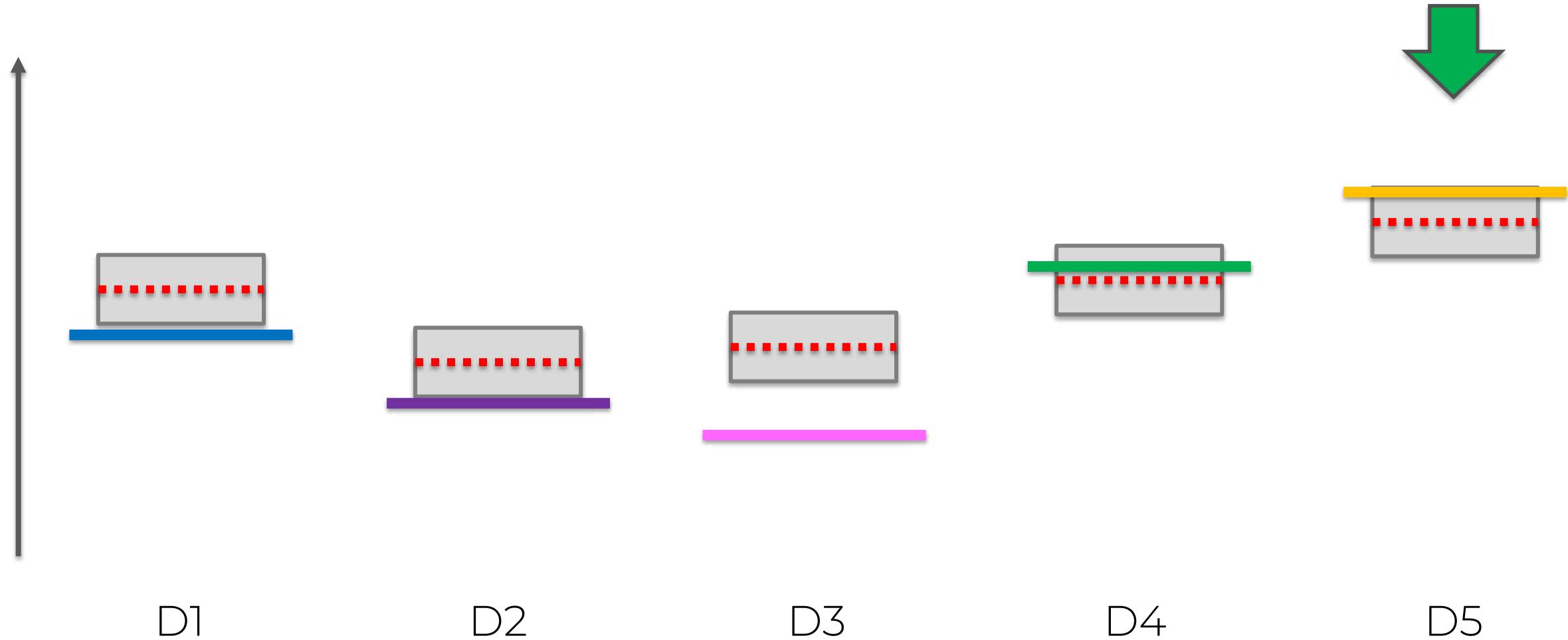
Upper Confidence Bound Algorithm



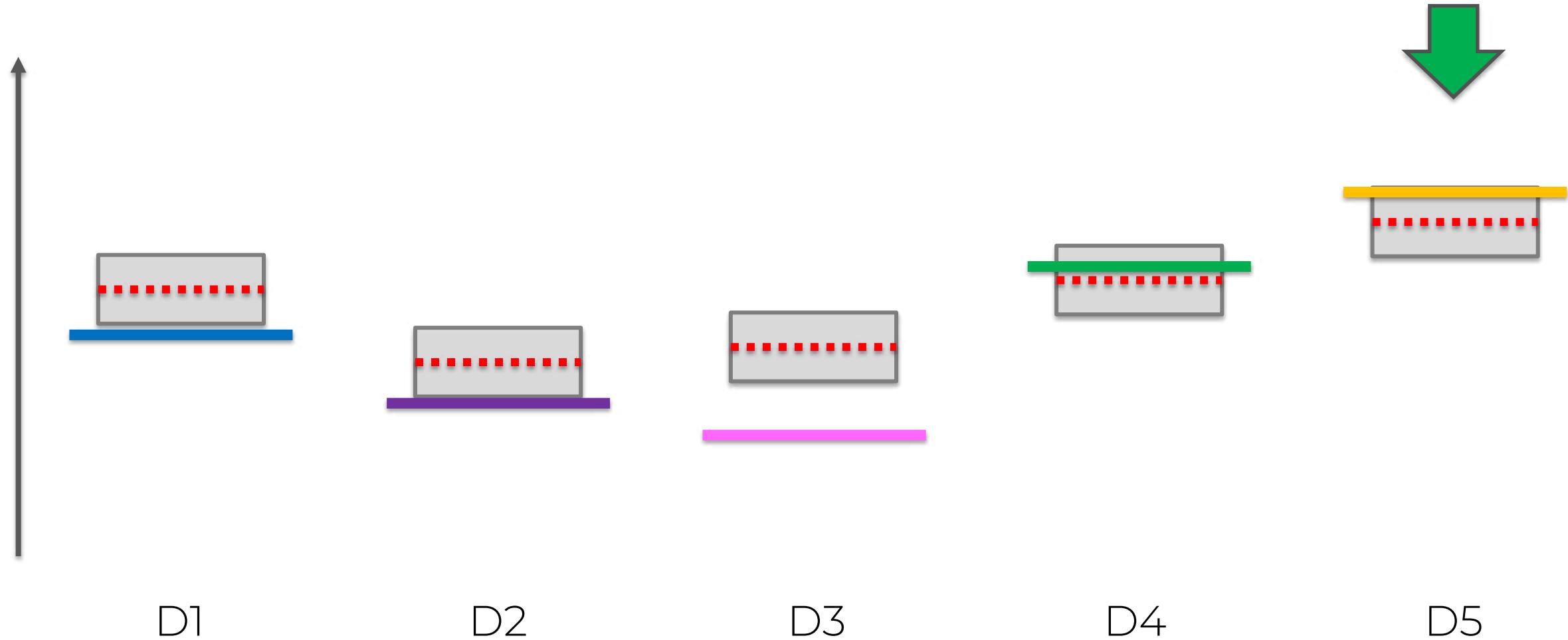
Upper Confidence Bound Algorithm



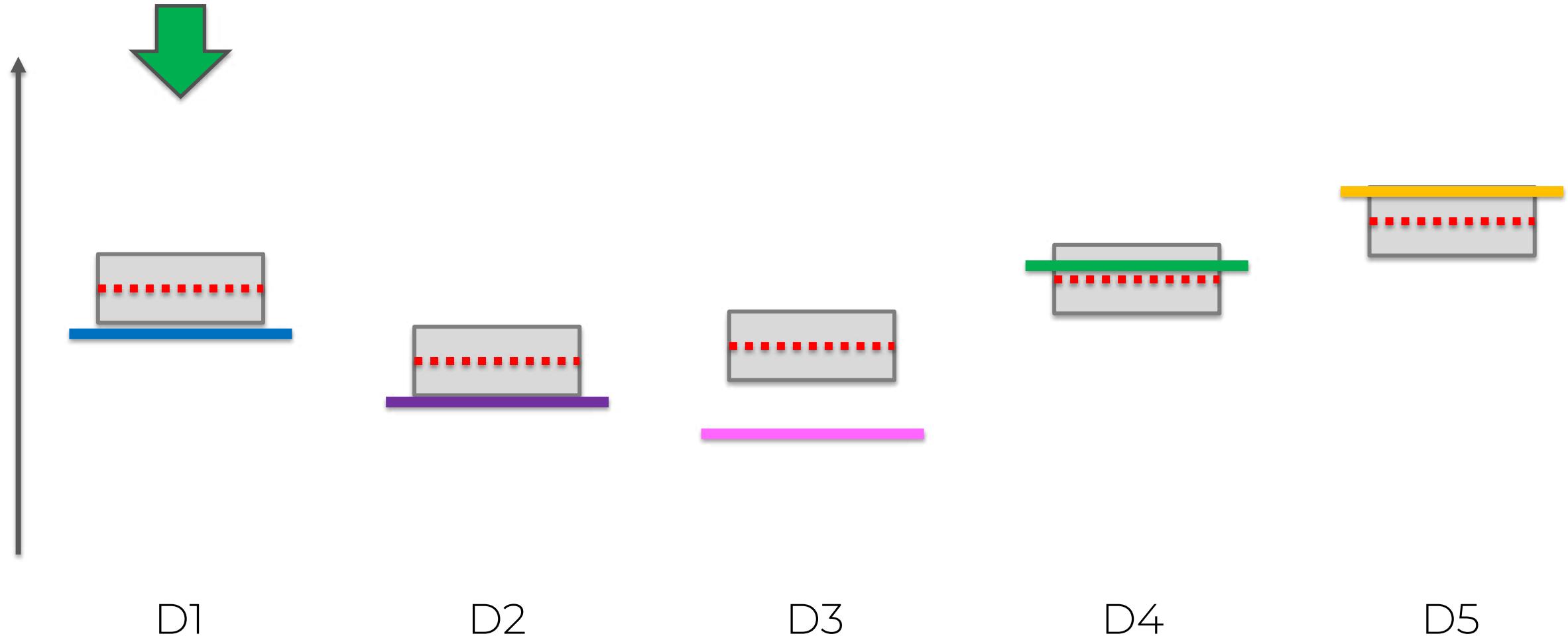
Upper Confidence Bound Algorithm



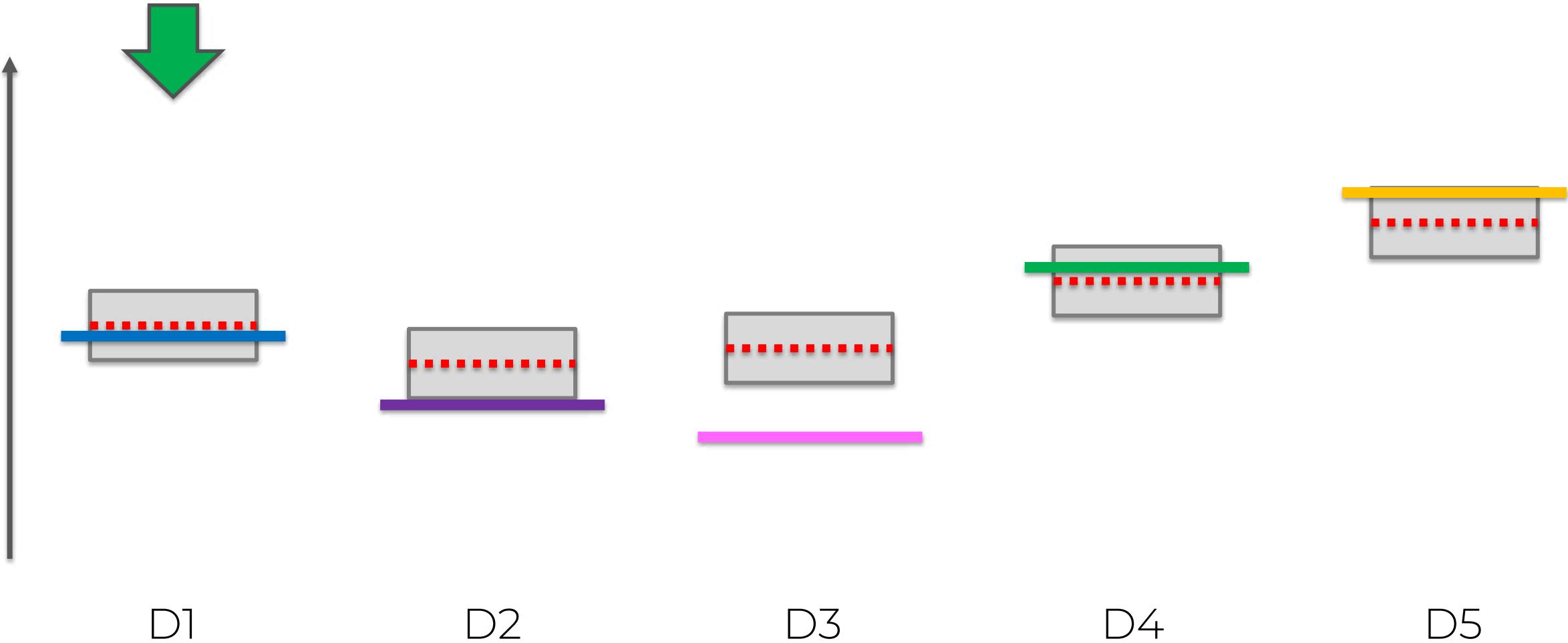
Upper Confidence Bound Algorithm



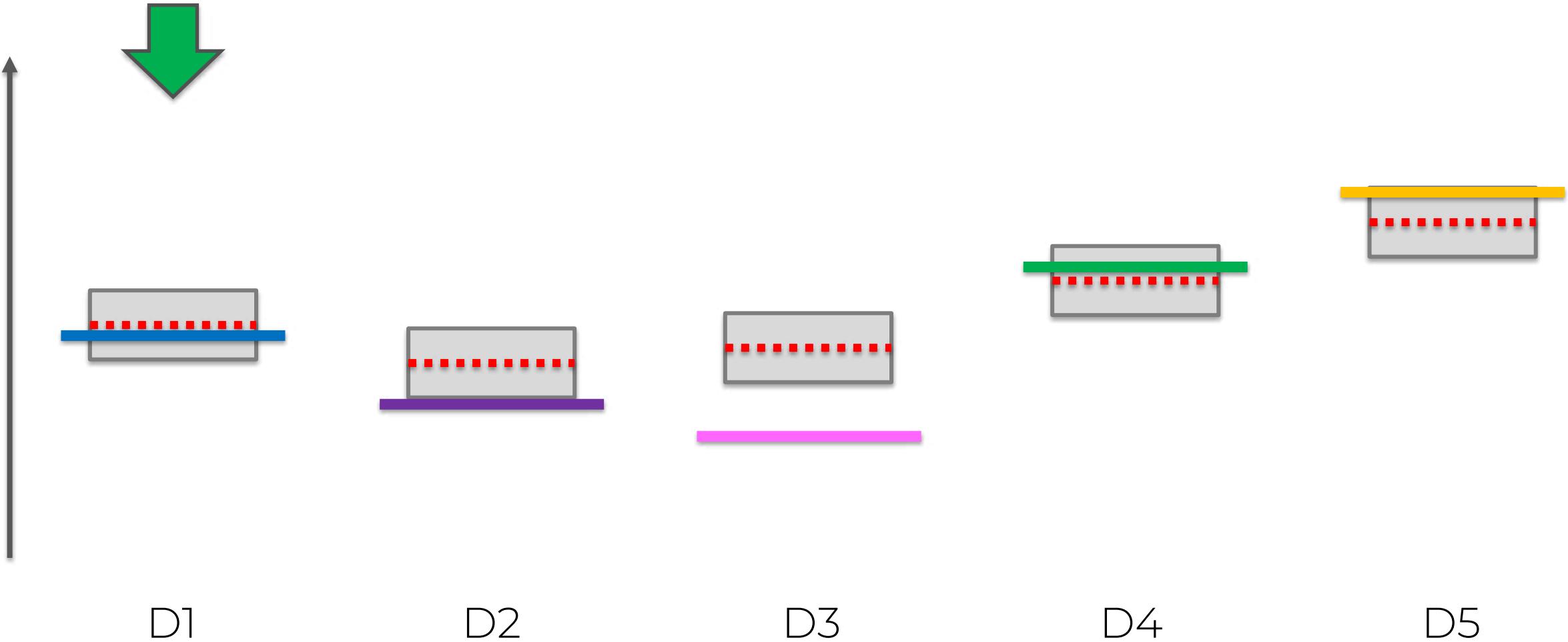
Upper Confidence Bound Algorithm



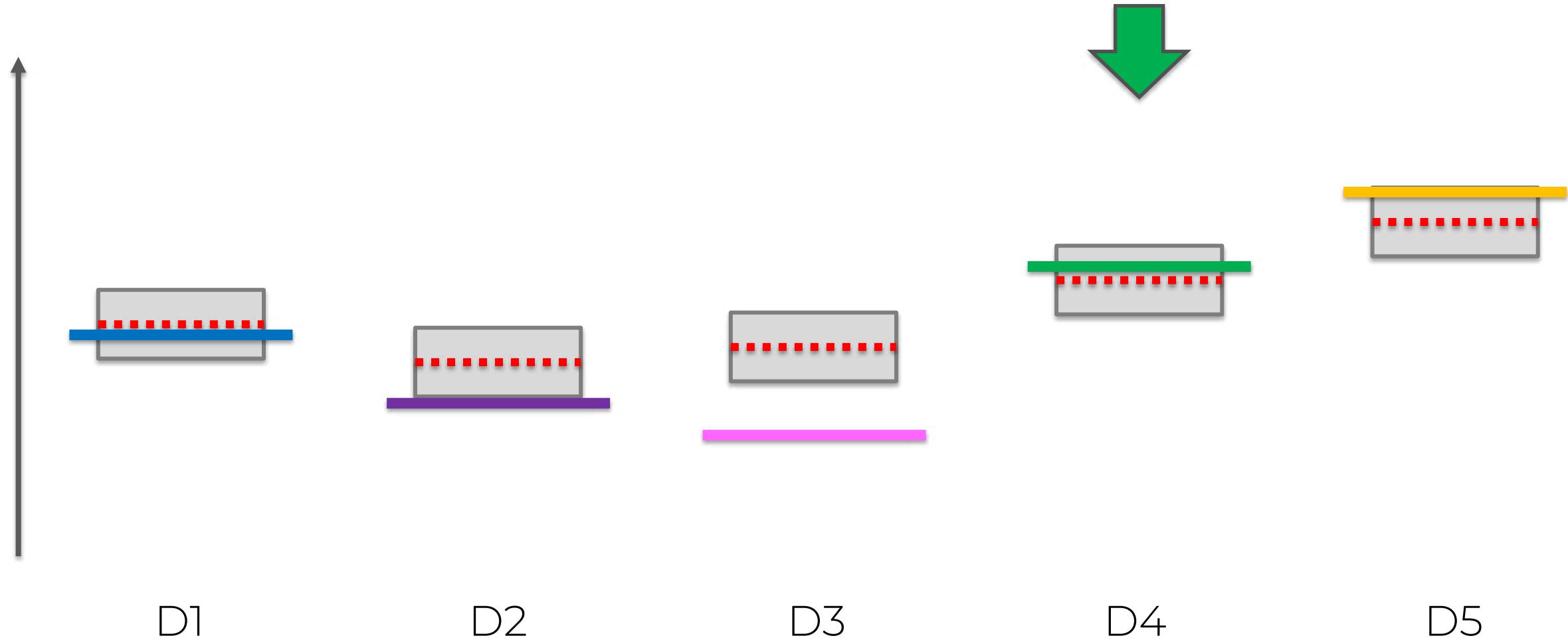
Upper Confidence Bound Algorithm



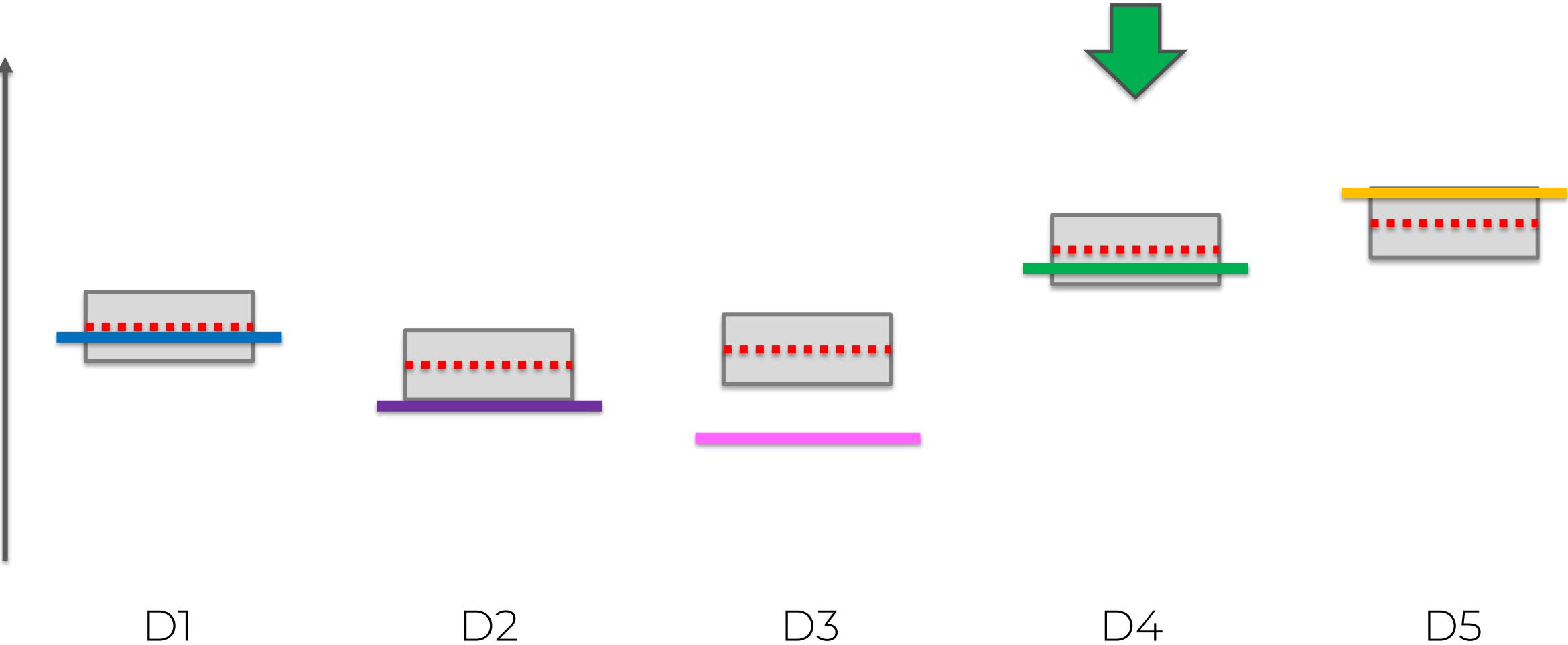
Upper Confidence Bound Algorithm



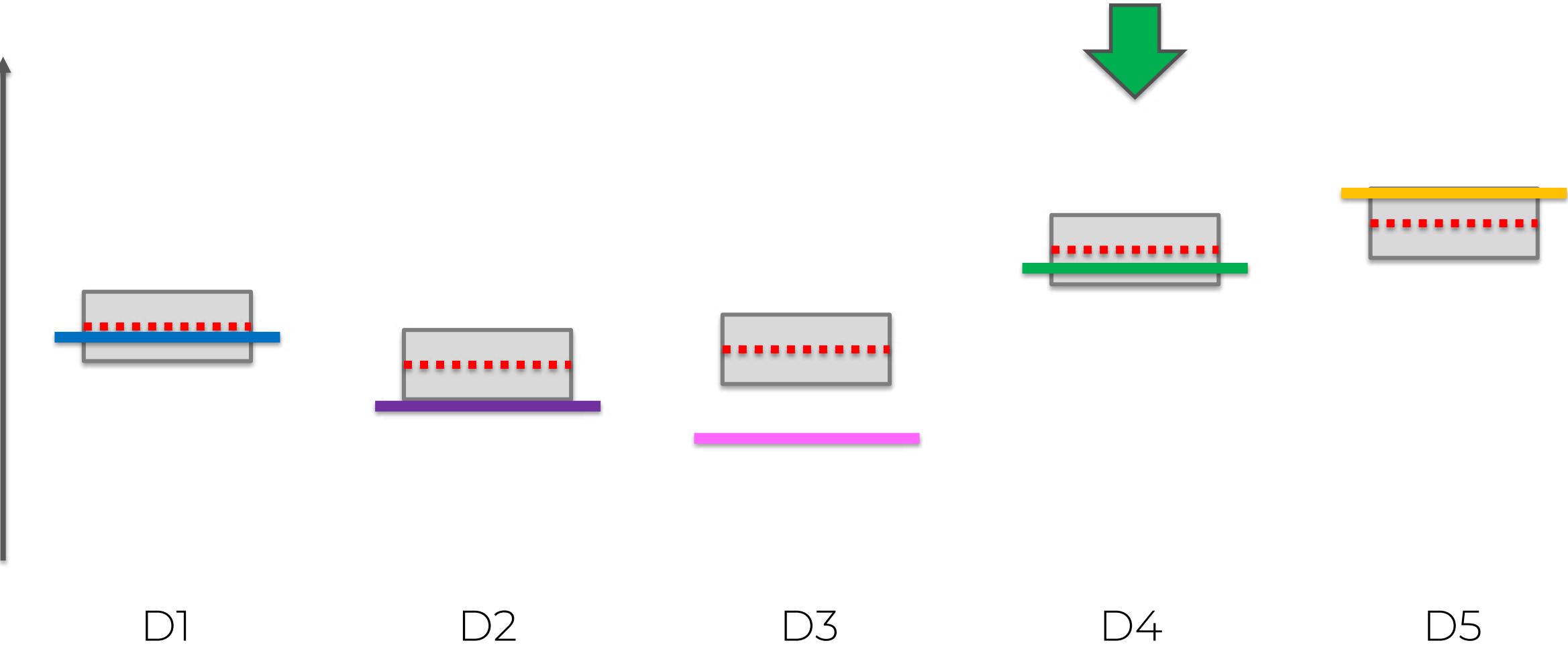
Upper Confidence Bound Algorithm



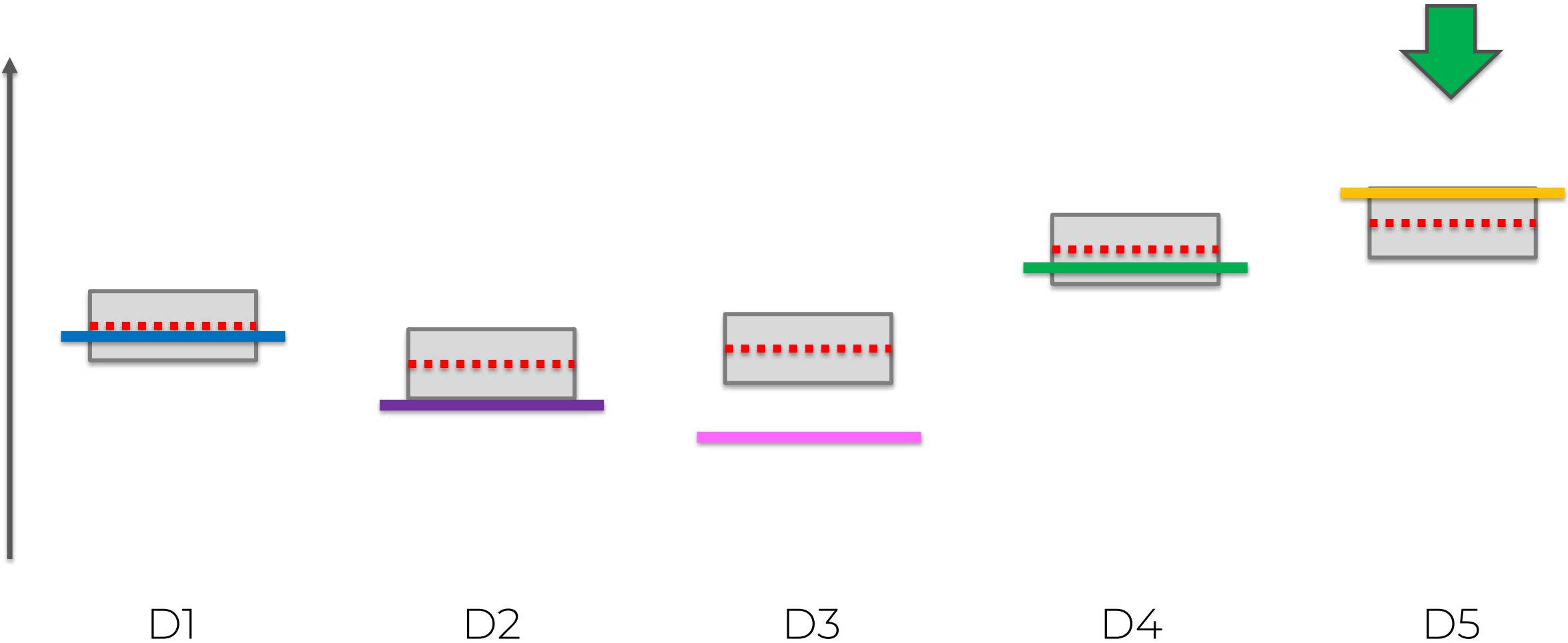
Upper Confidence Bound Algorithm



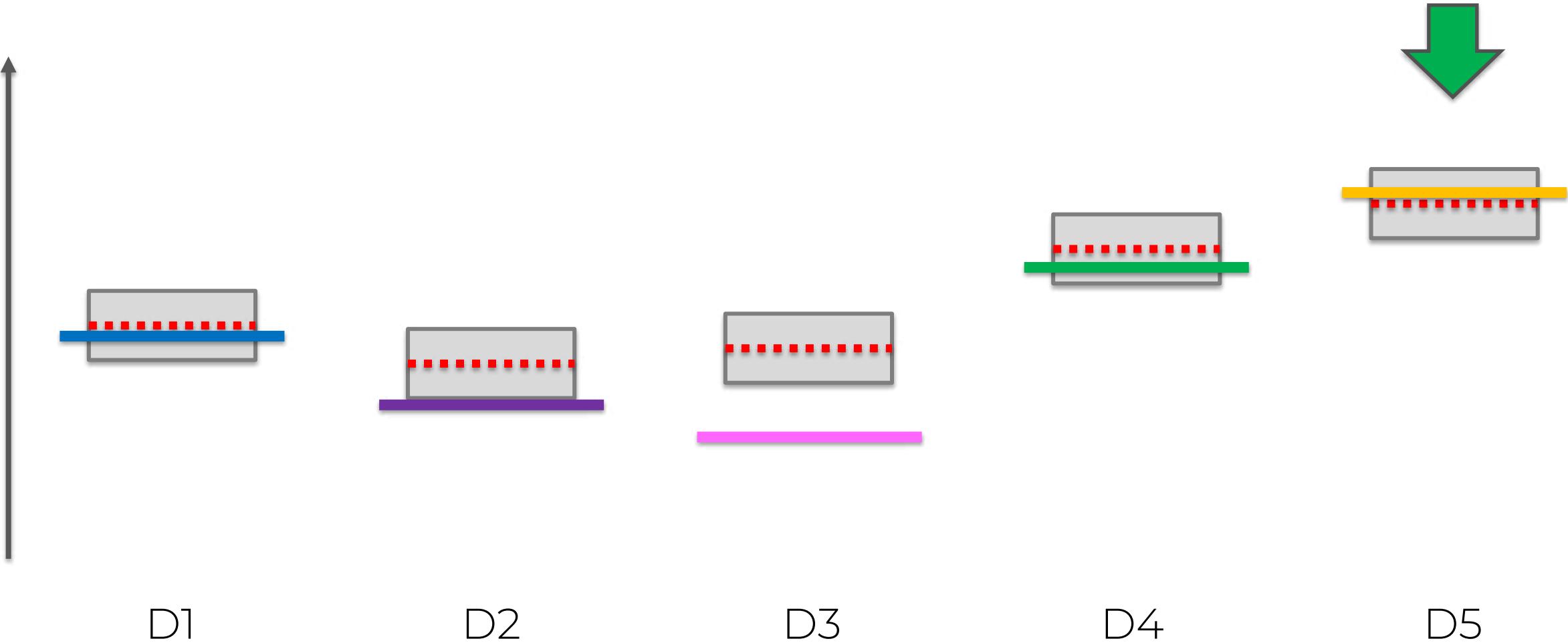
Upper Confidence Bound Algorithm



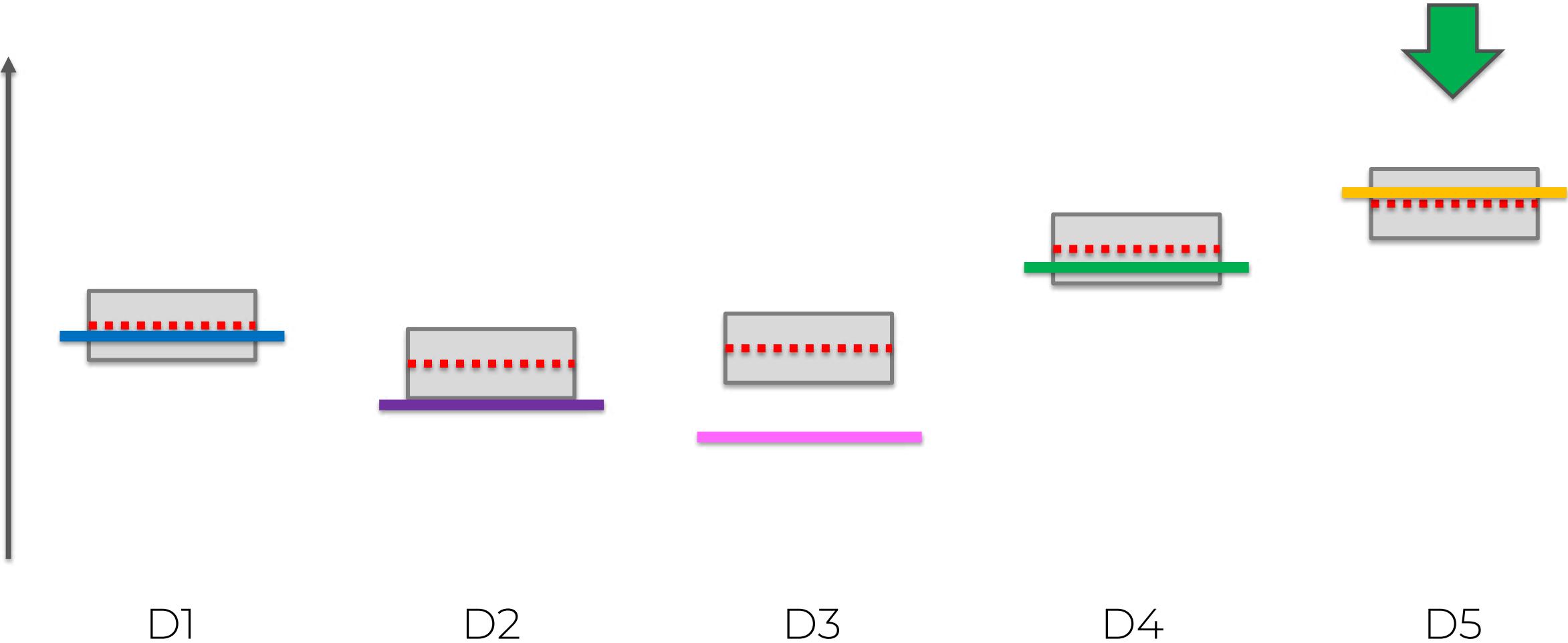
Upper Confidence Bound Algorithm



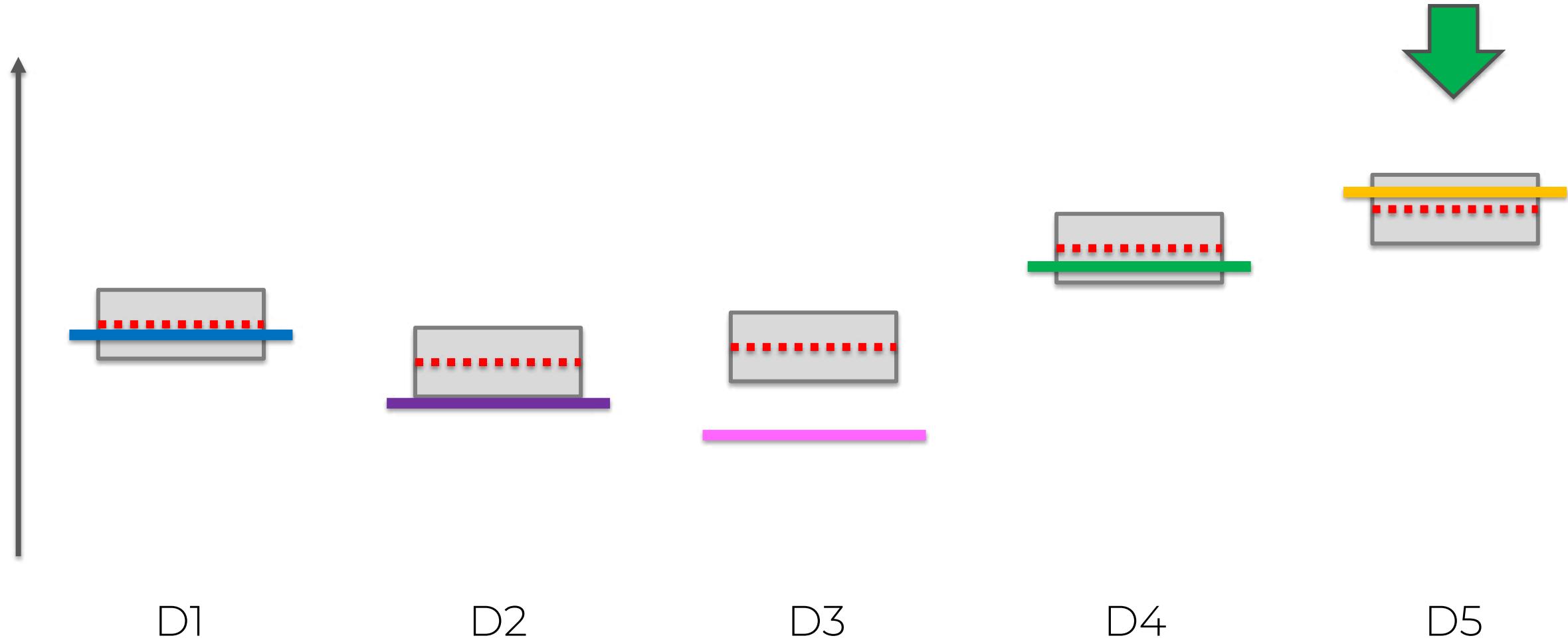
Upper Confidence Bound Algorithm



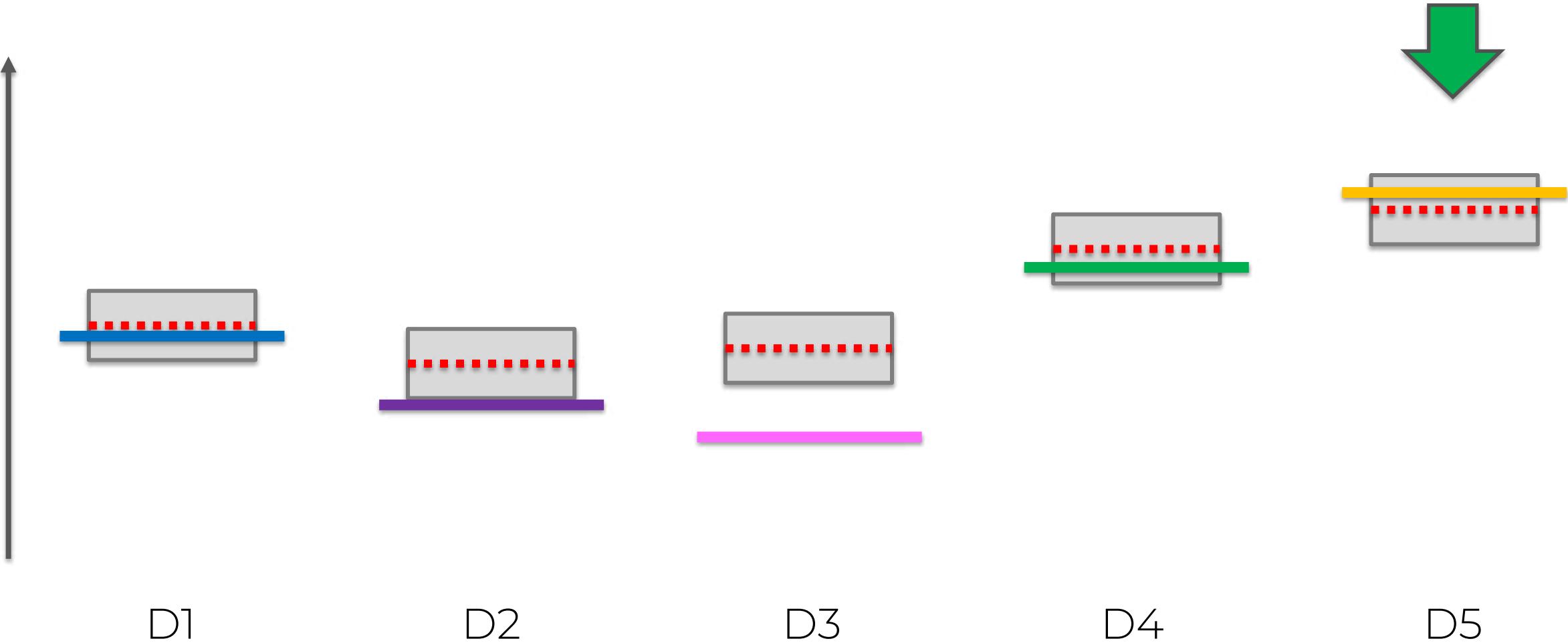
Upper Confidence Bound Algorithm



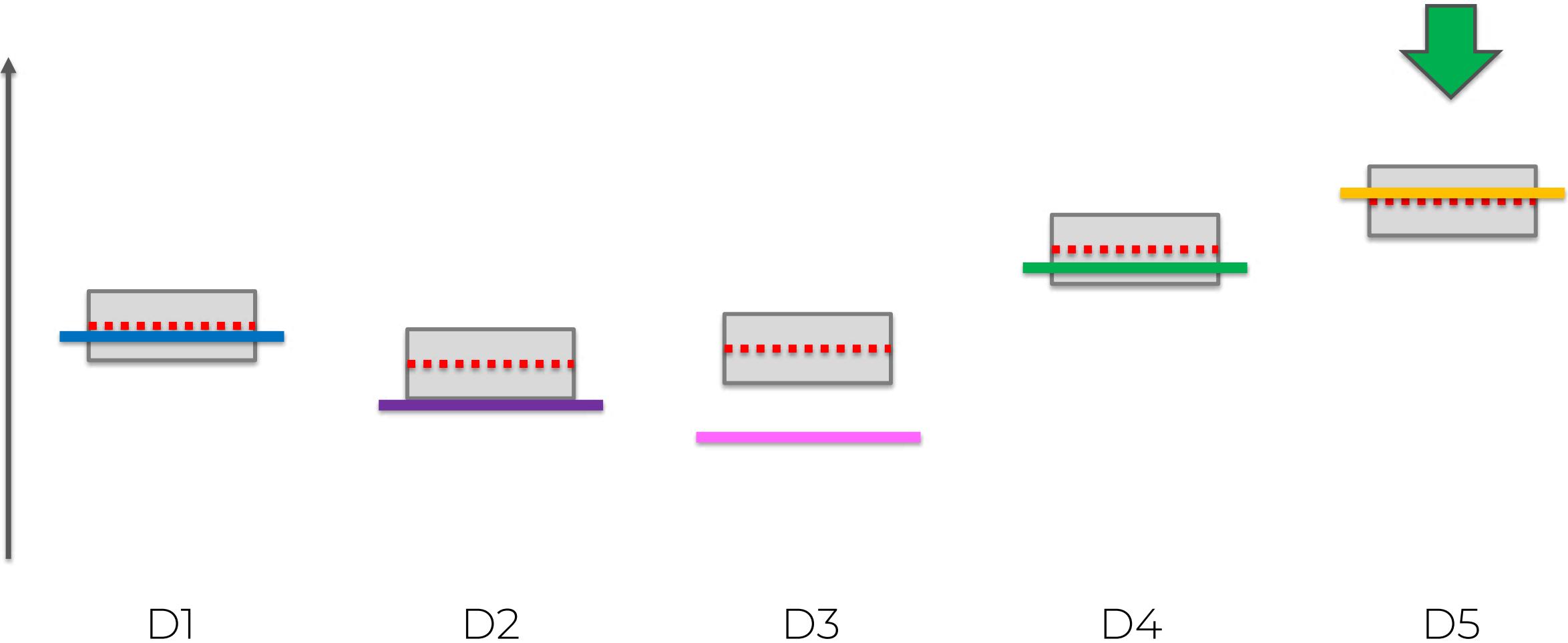
Upper Confidence Bound Algorithm



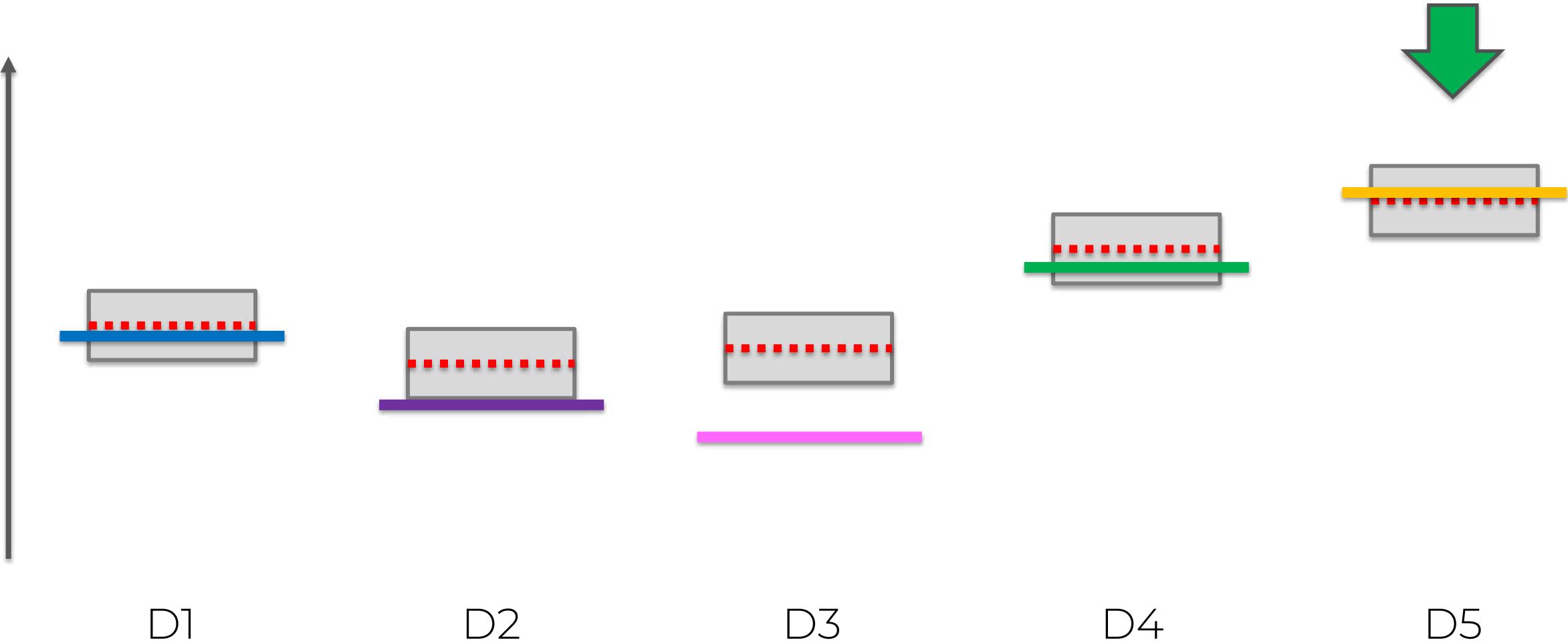
Upper Confidence Bound Algorithm



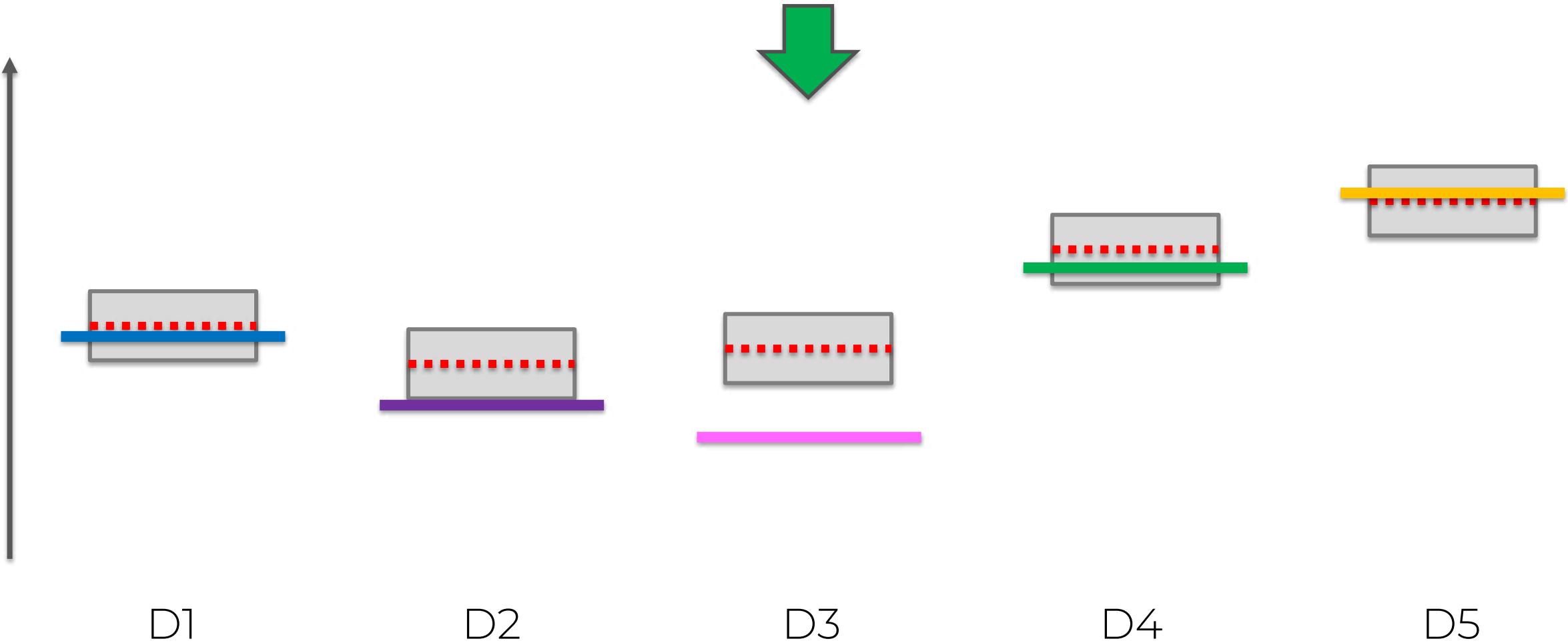
Upper Confidence Bound Algorithm



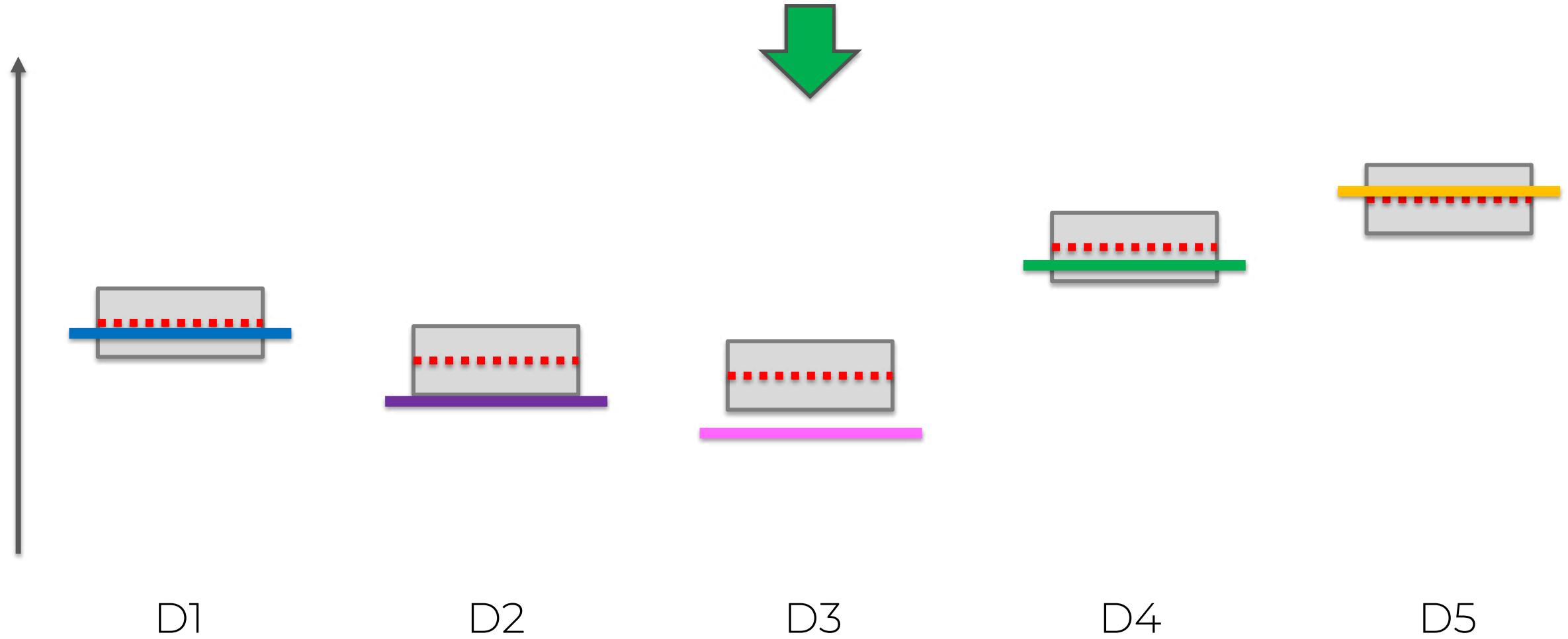
Upper Confidence Bound Algorithm



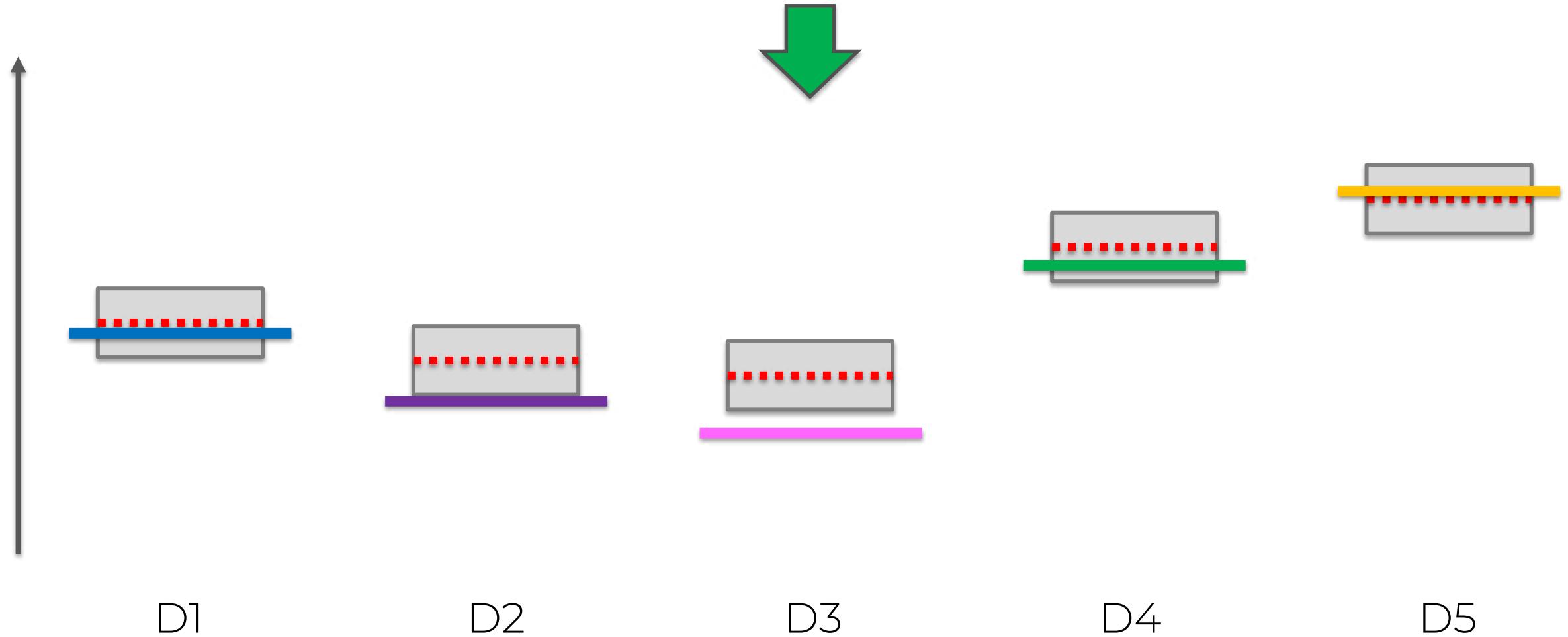
Upper Confidence Bound Algorithm



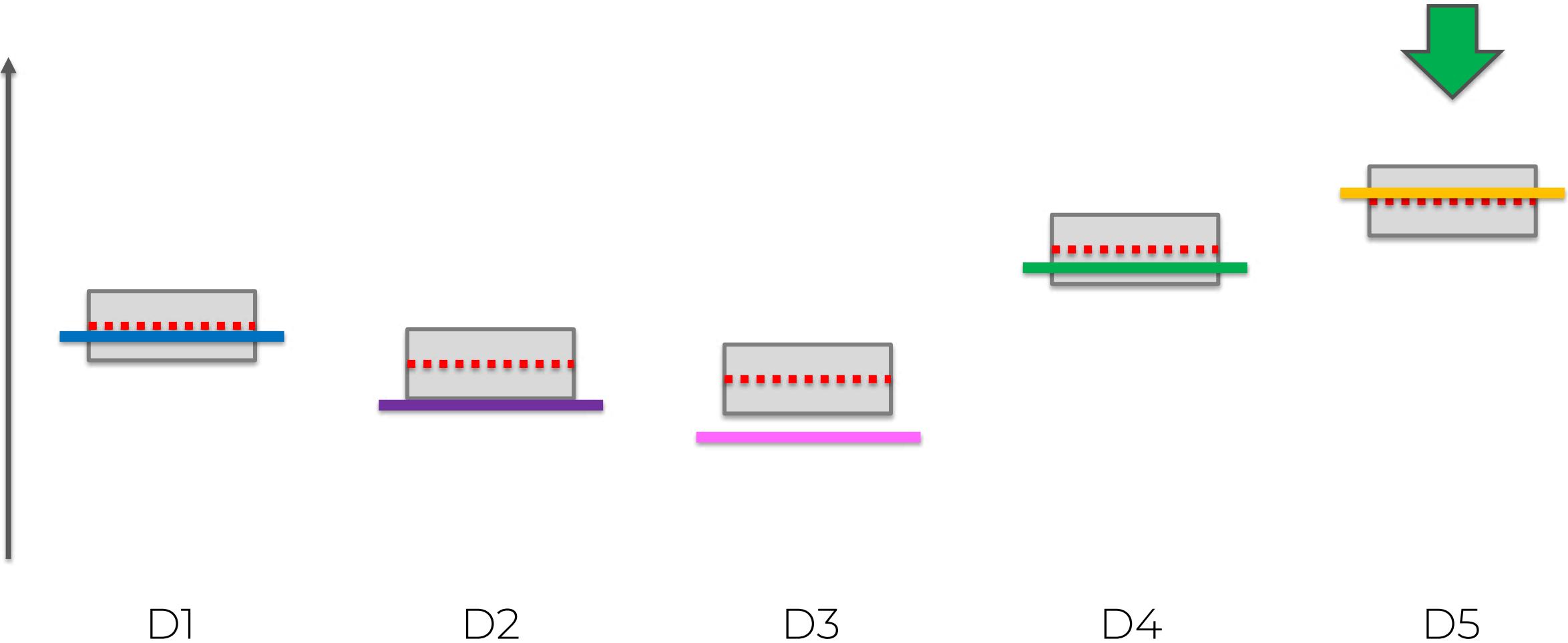
Upper Confidence Bound Algorithm



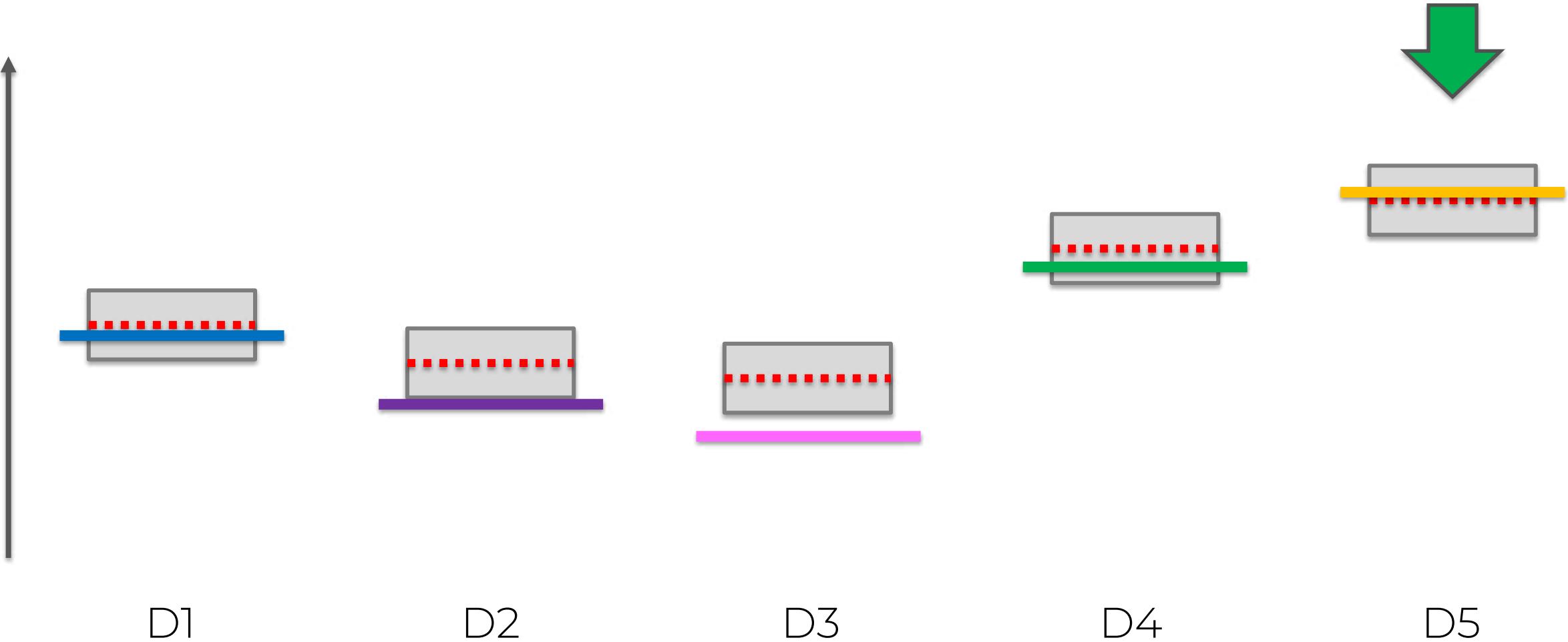
Upper Confidence Bound Algorithm



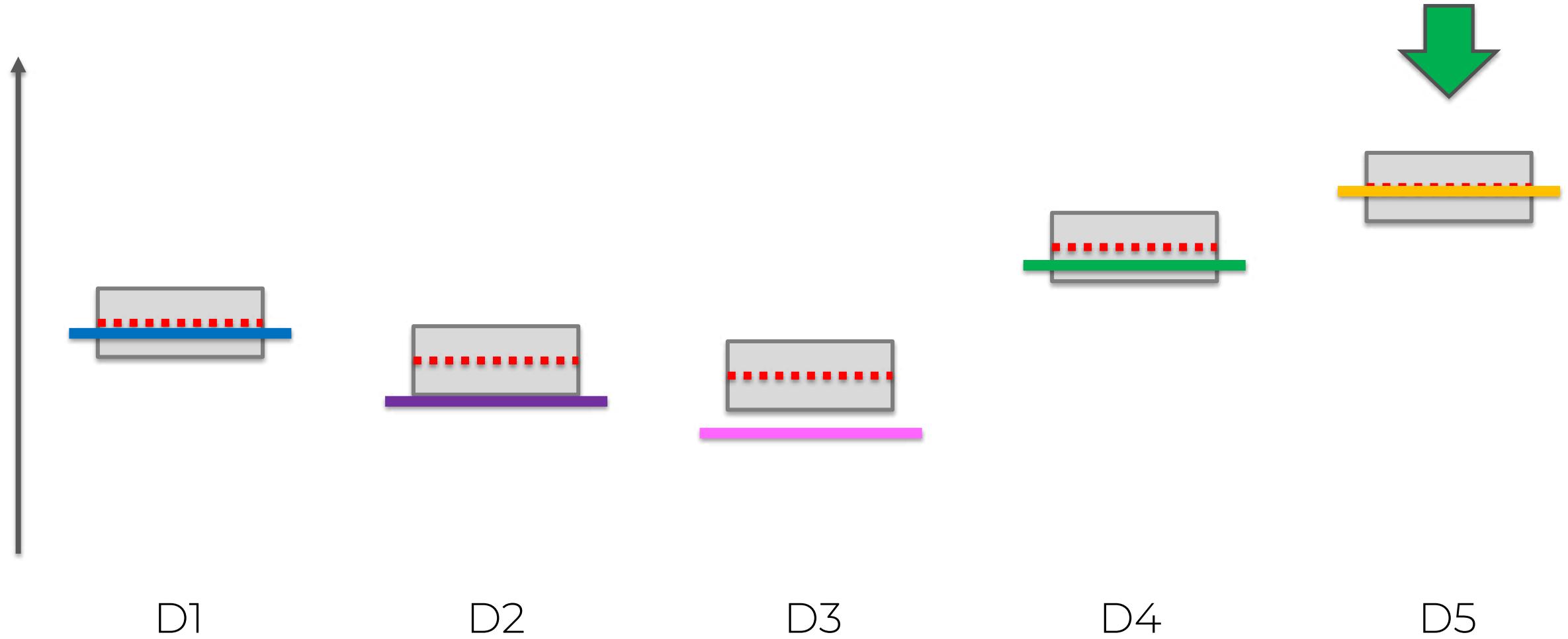
Upper Confidence Bound Algorithm



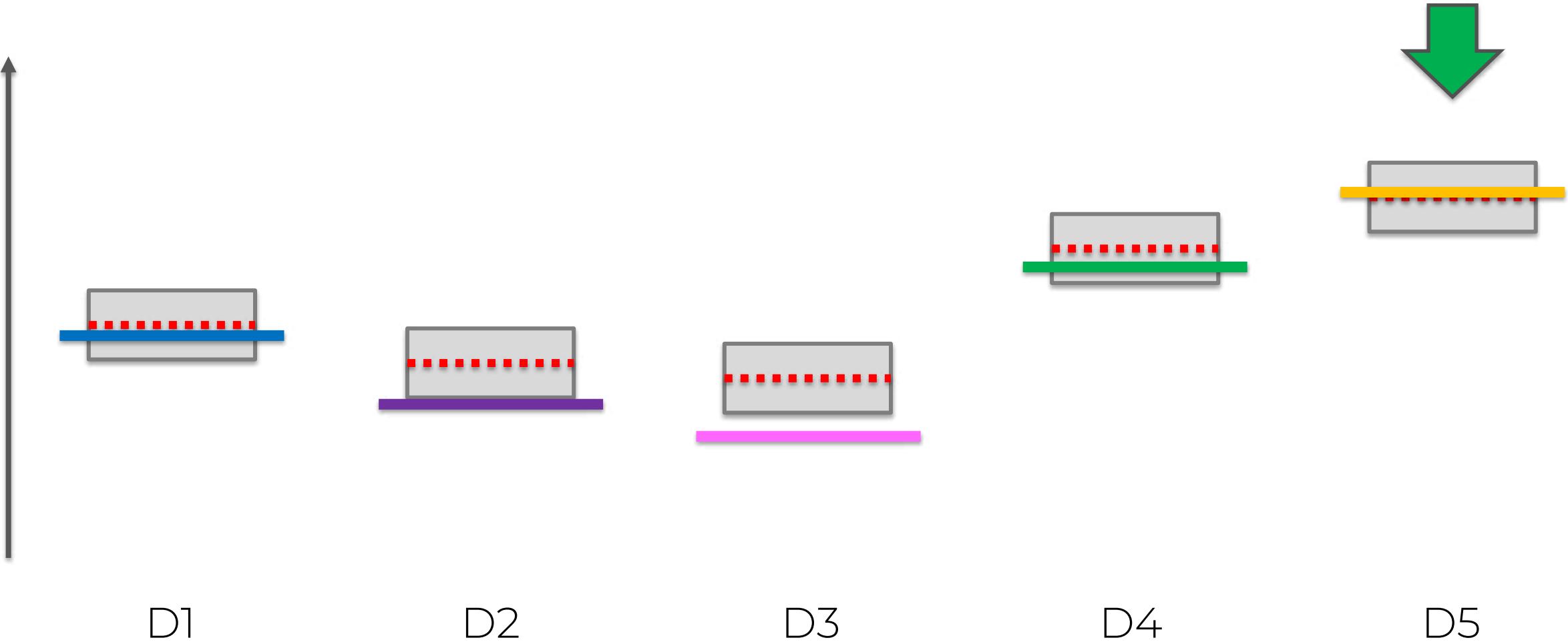
Upper Confidence Bound Algorithm



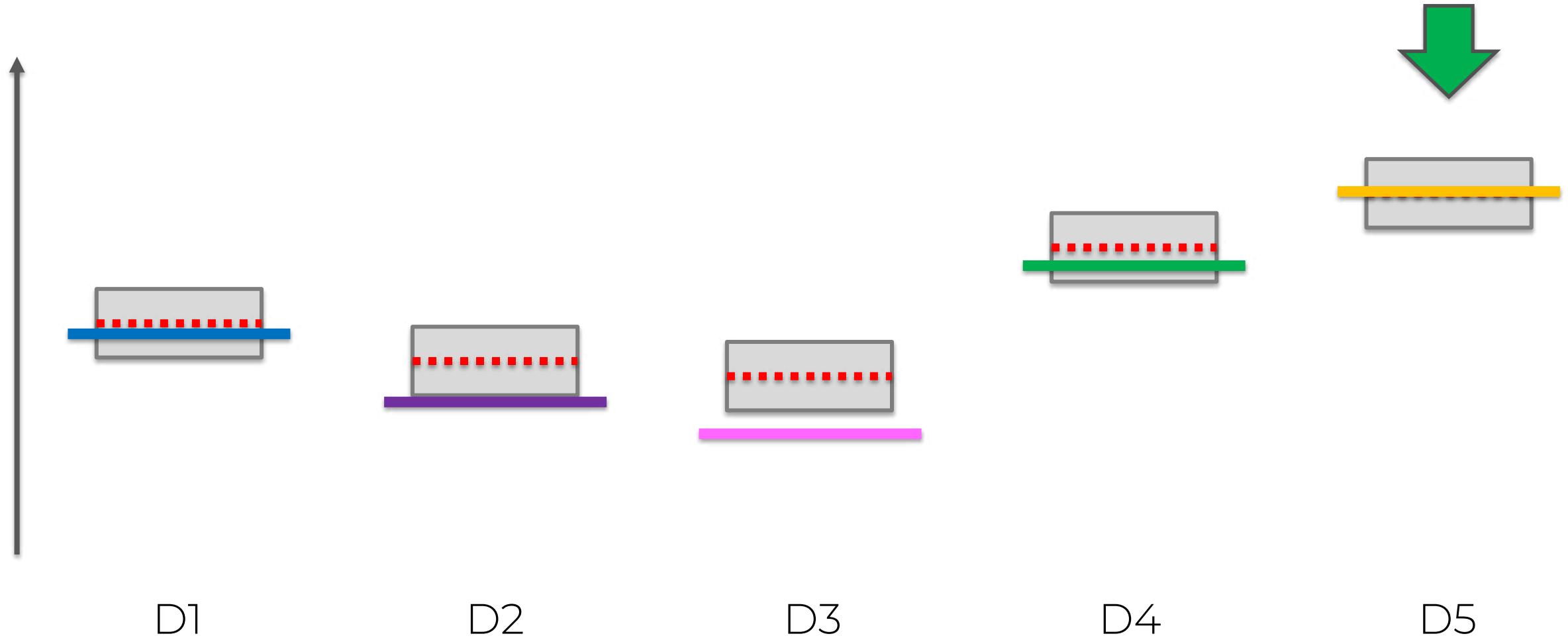
Upper Confidence Bound Algorithm



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 \mathbf{y} from Bernoulli distribution $p(\mathbf{y}|\theta_i) \sim \mathcal{B}(\theta_i)$.
- 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 $p(\theta_i|\mathbf{y})$, 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 :

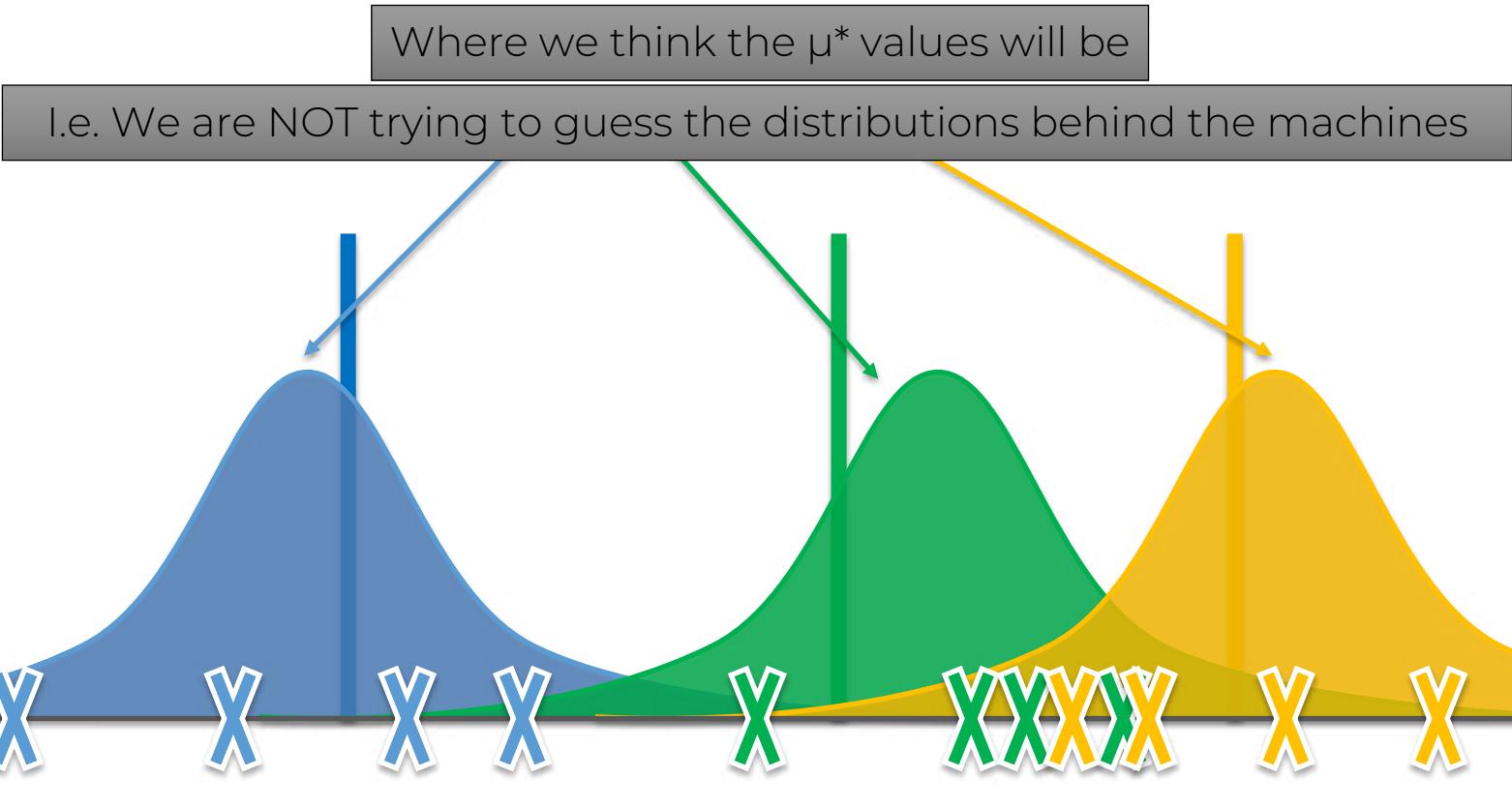
- $N_i^1(n)$ - the number of times the ad i got reward 1 up to round n ,
- $N_i^0(n)$ - the number of times the ad i got reward 0 up to round n .

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

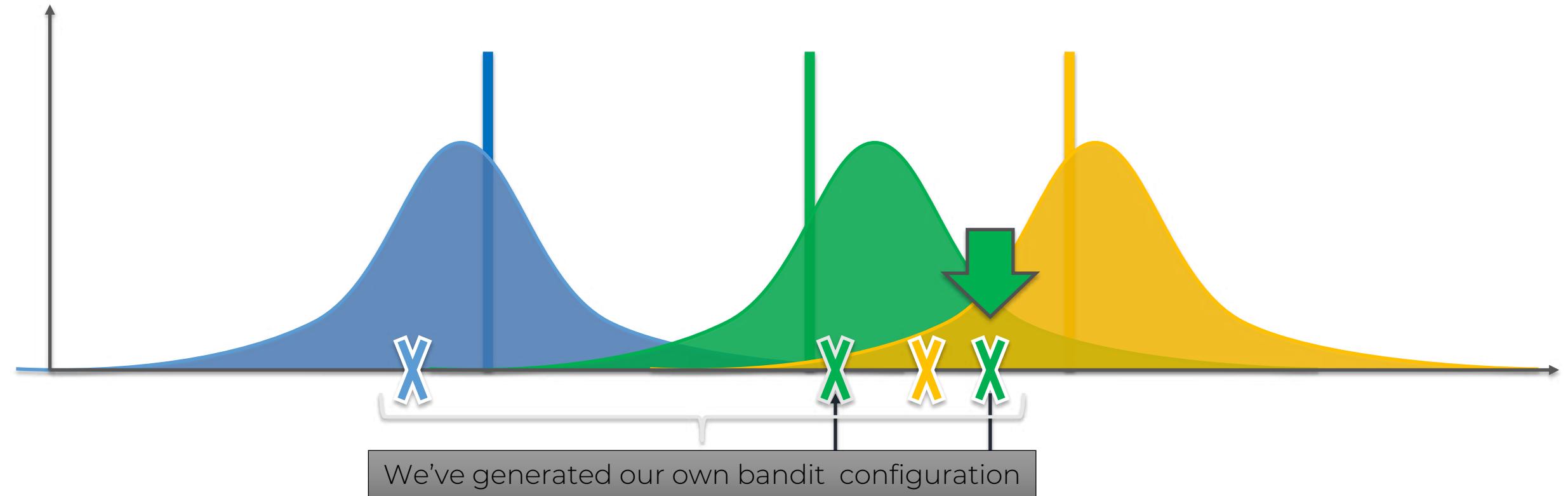
$$\theta_i(n) = \beta(N_i^1(n) + 1, N_i^0(n) + 1)$$

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

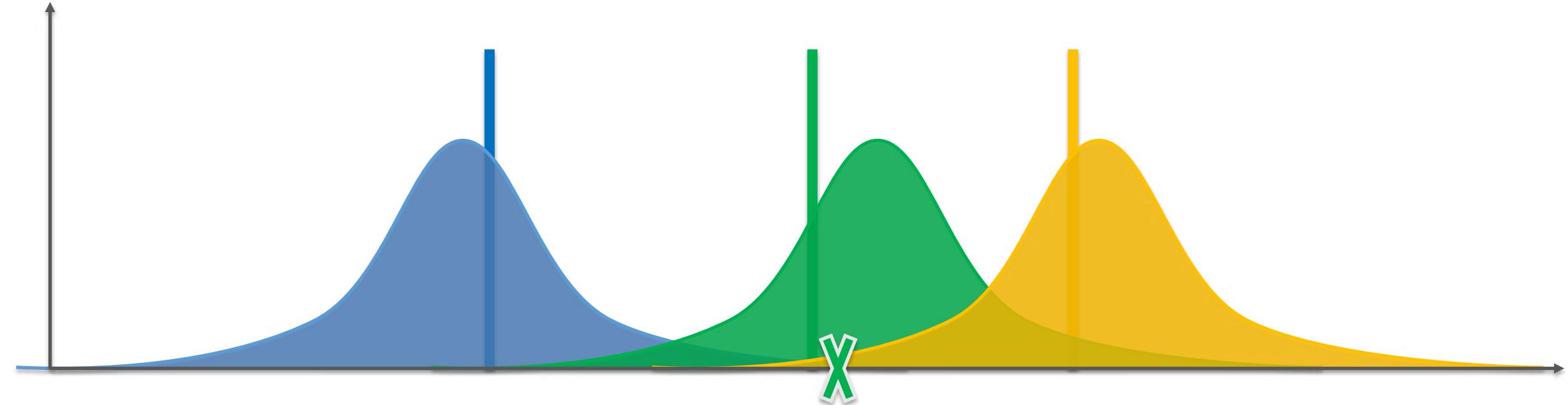
Thompson Sampling Algorithm



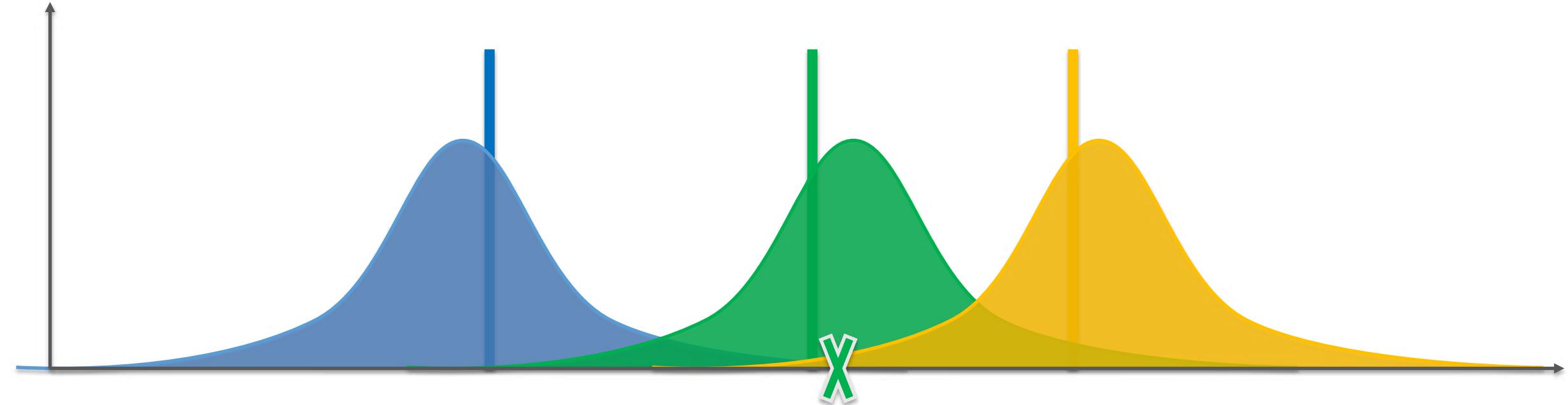
Thompson Sampling Algorithm



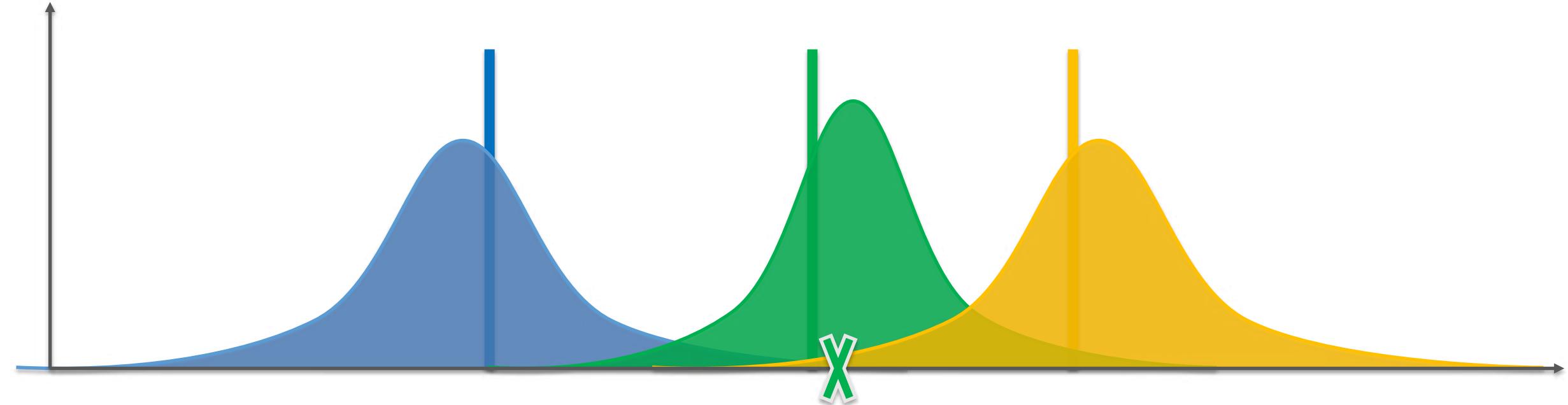
Thompson Sampling Algorithm



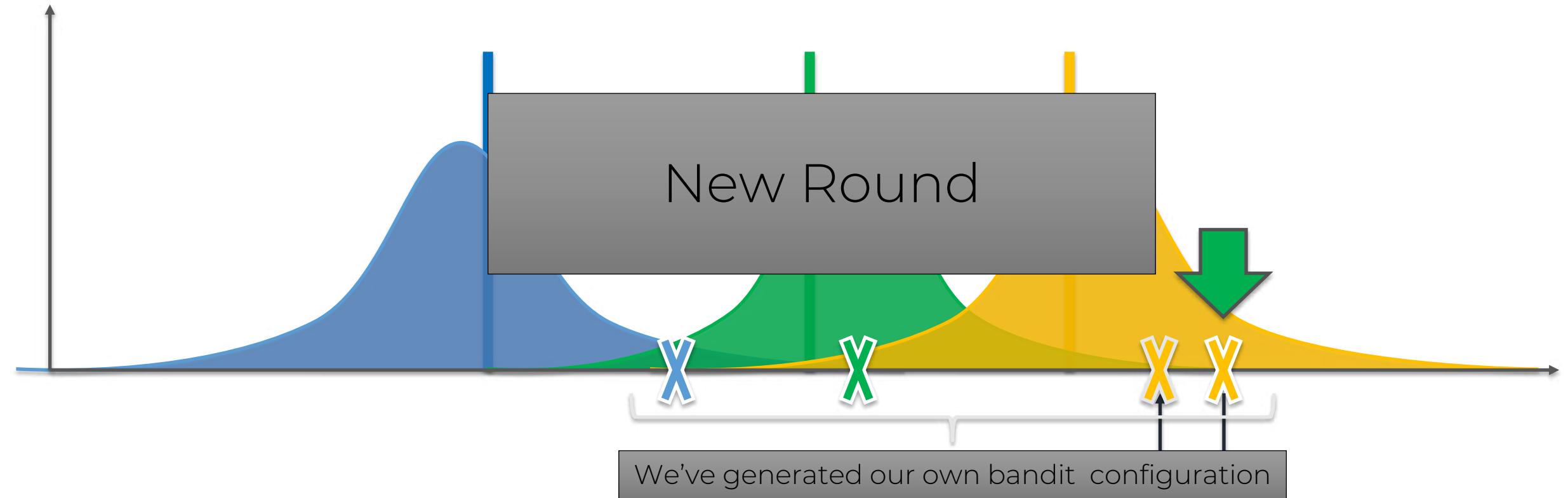
Thompson Sampling Algorithm



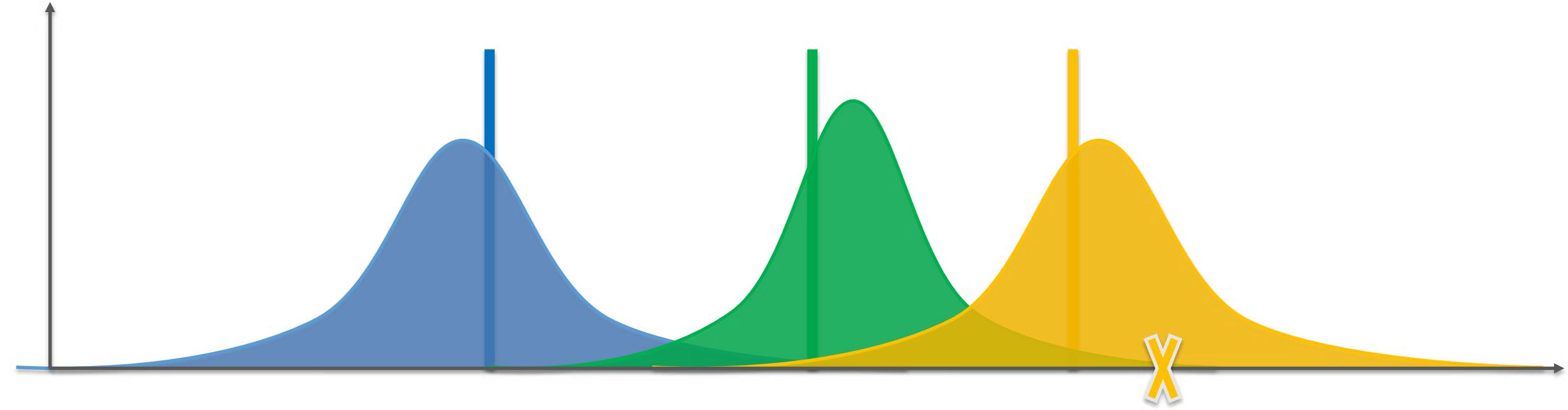
Thompson Sampling Algorithm



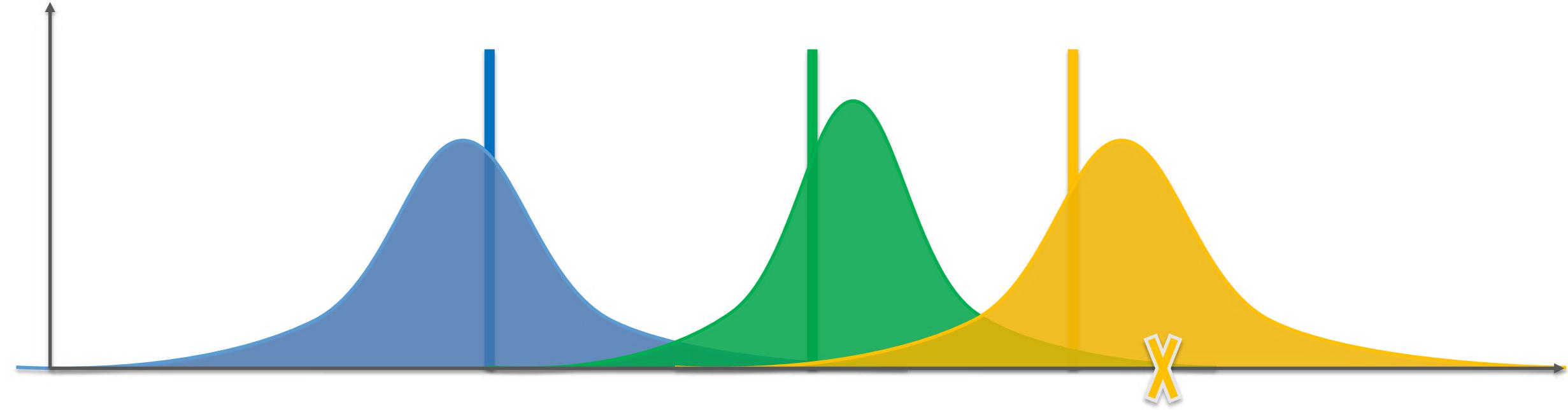
Thompson Sampling Algorithm



Thompson Sampling Algorithm



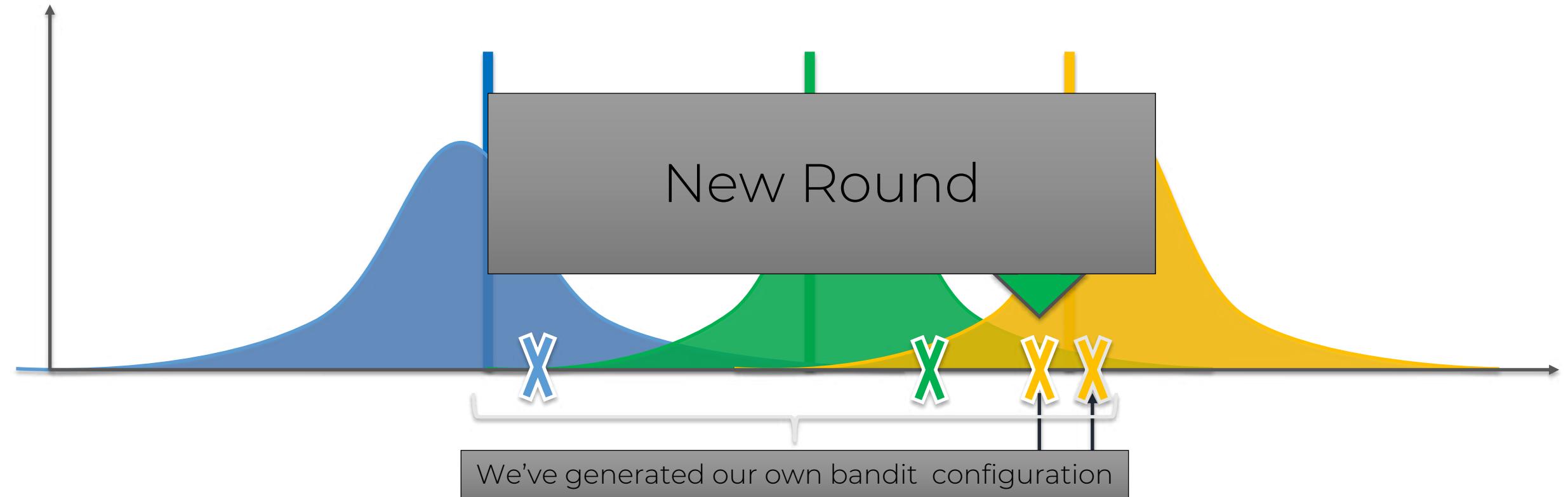
Thompson Sampling Algorithm



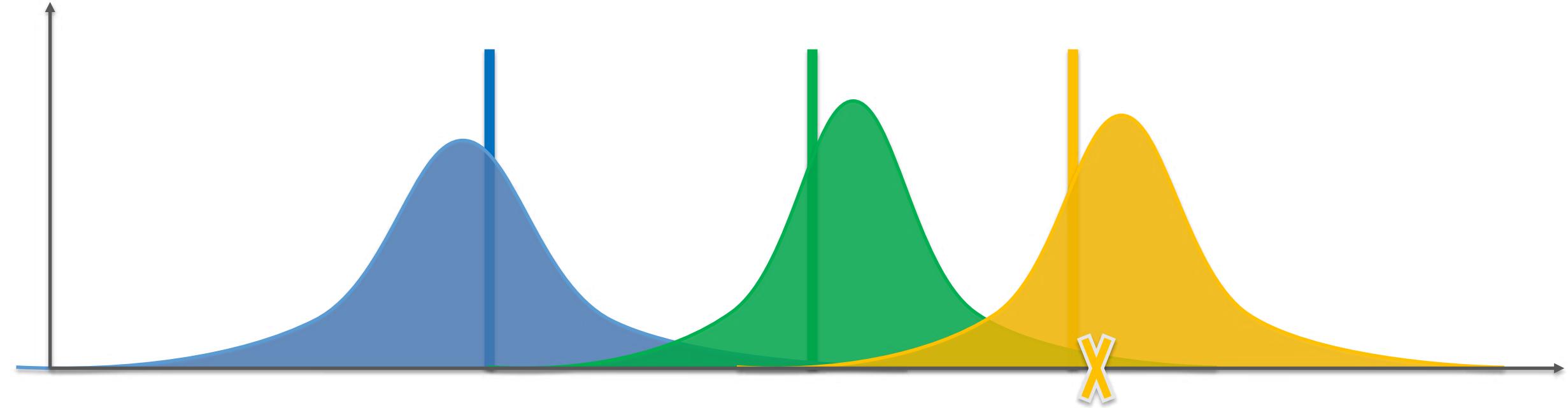
Thompson Sampling Algorithm



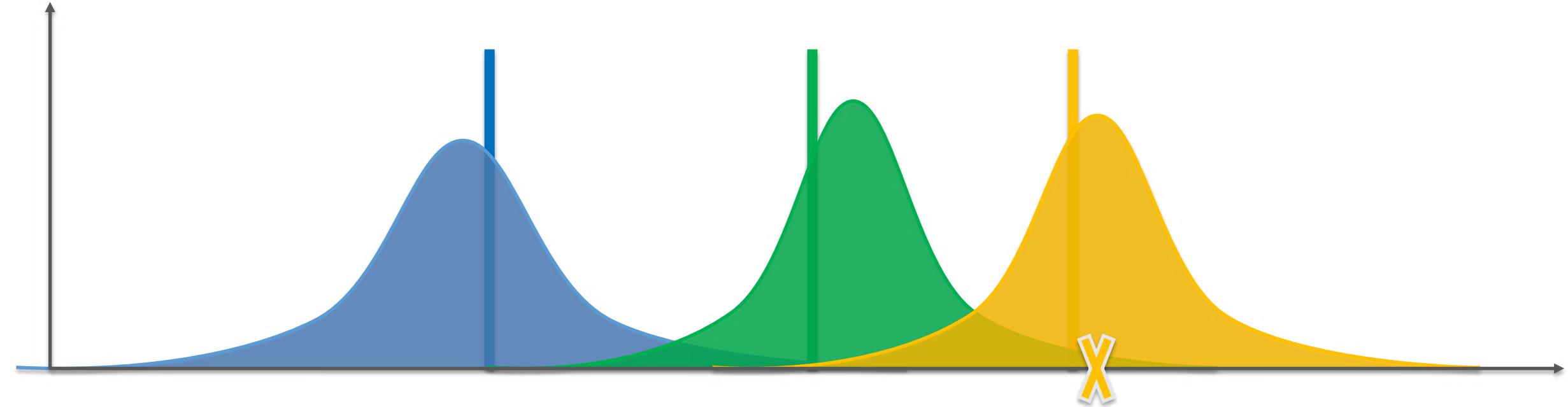
Thompson Sampling Algorithm



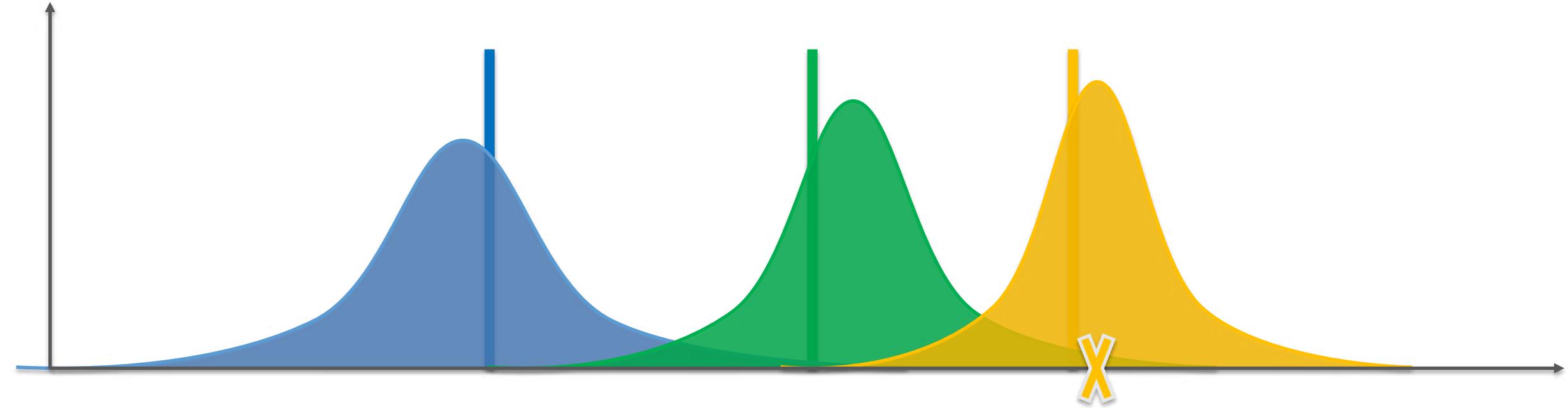
Thompson Sampling Algorithm



Thompson Sampling Algorithm



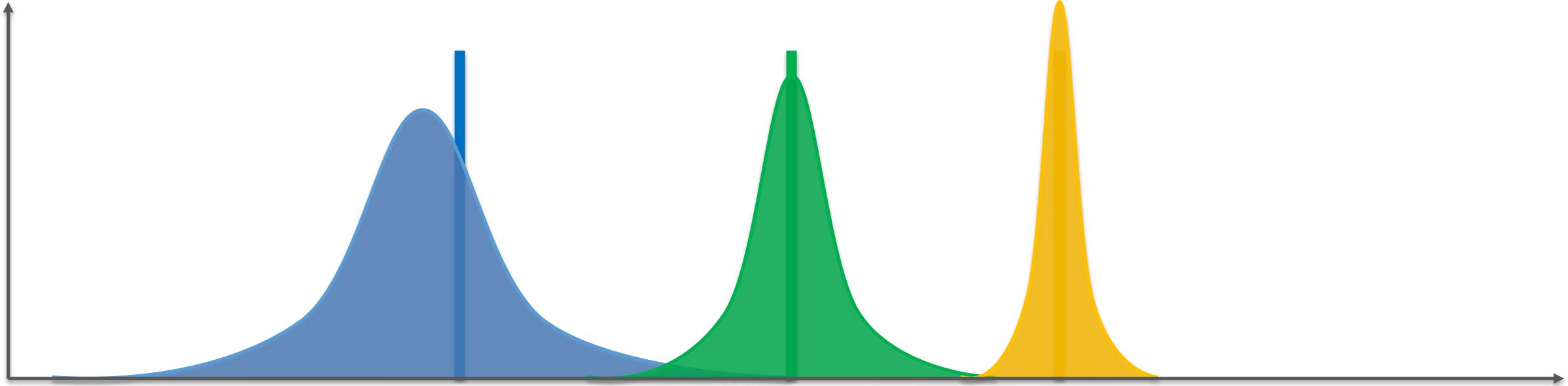
Thompson Sampling Algorithm



Thompson Sampling Algorithm

And so on...

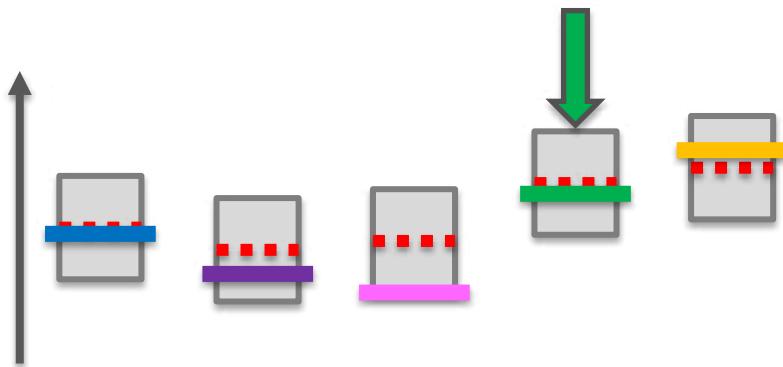
Thompson Sampling Algorithm



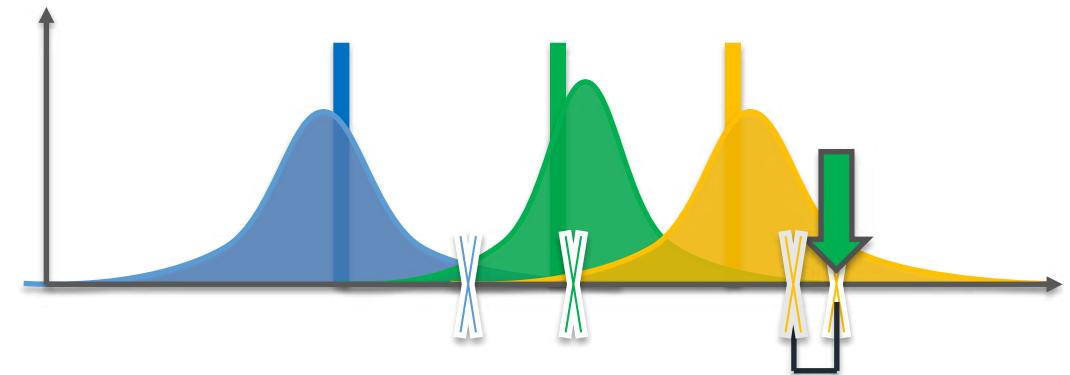
UCB vs Thompson Sampling

Thompson Sampling Algorithm

UCB



Thompson Sampling



- Deterministic
- Requires update at every round

- Probabilistic
- Can accommodate delayed feedback
- Better empirical evidence

Natural Language Processing (NLP)

Plan of Attack

Plan of Attack

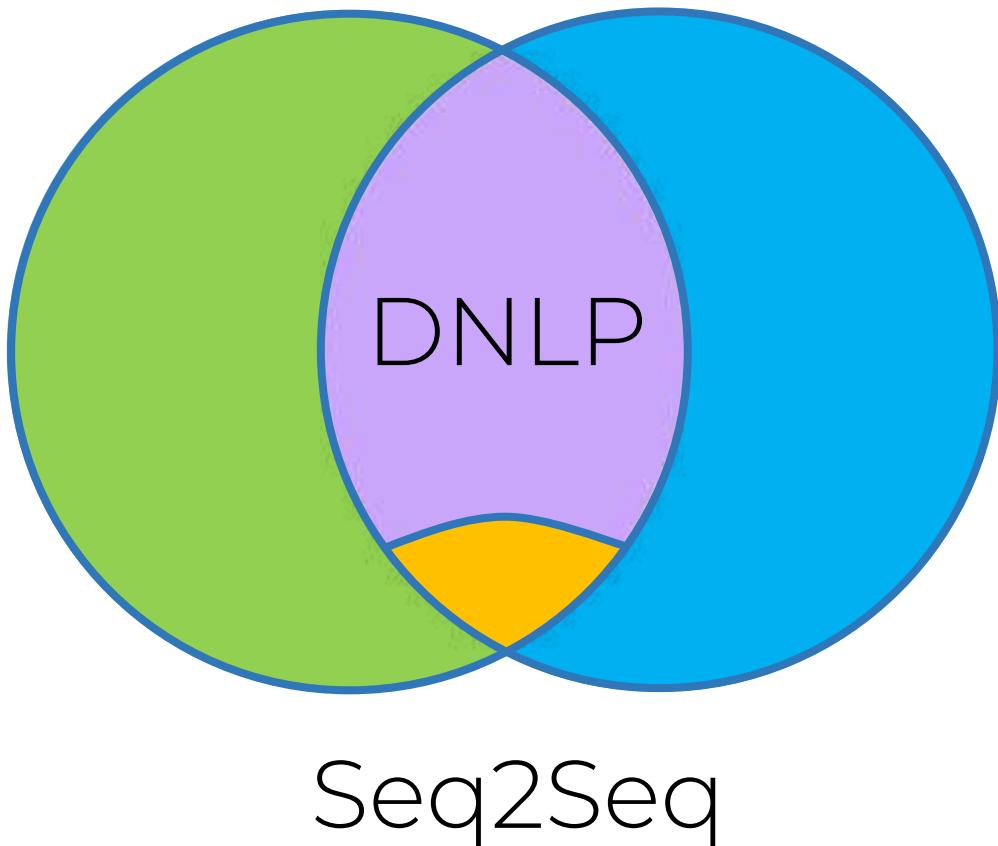
Here's what we will learn:

- 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

Natural
Language
Processing

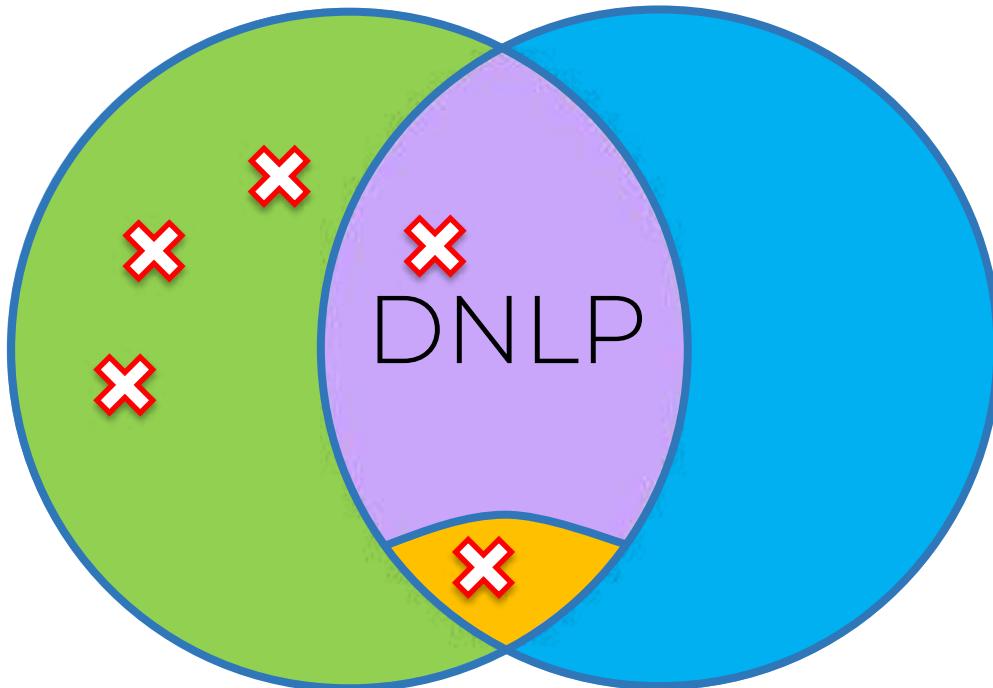


Deep
Learning

Classical vs Deep Learning Models

Classical vs Deep Learning Models

Natural
Language
Processing



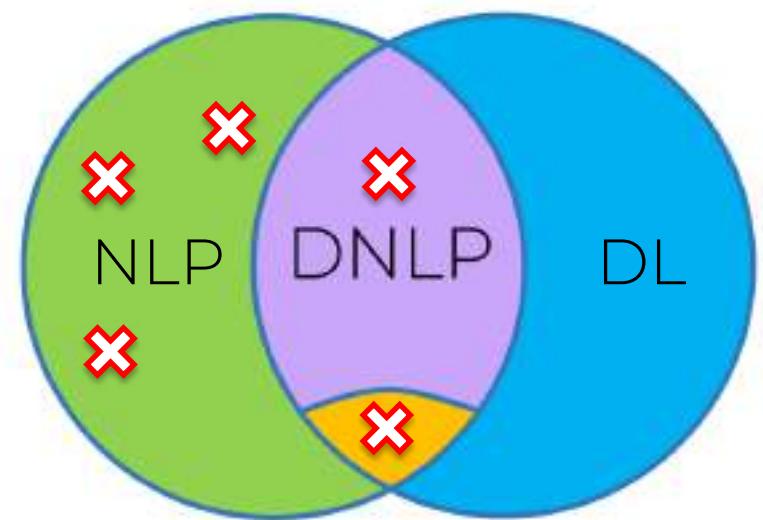
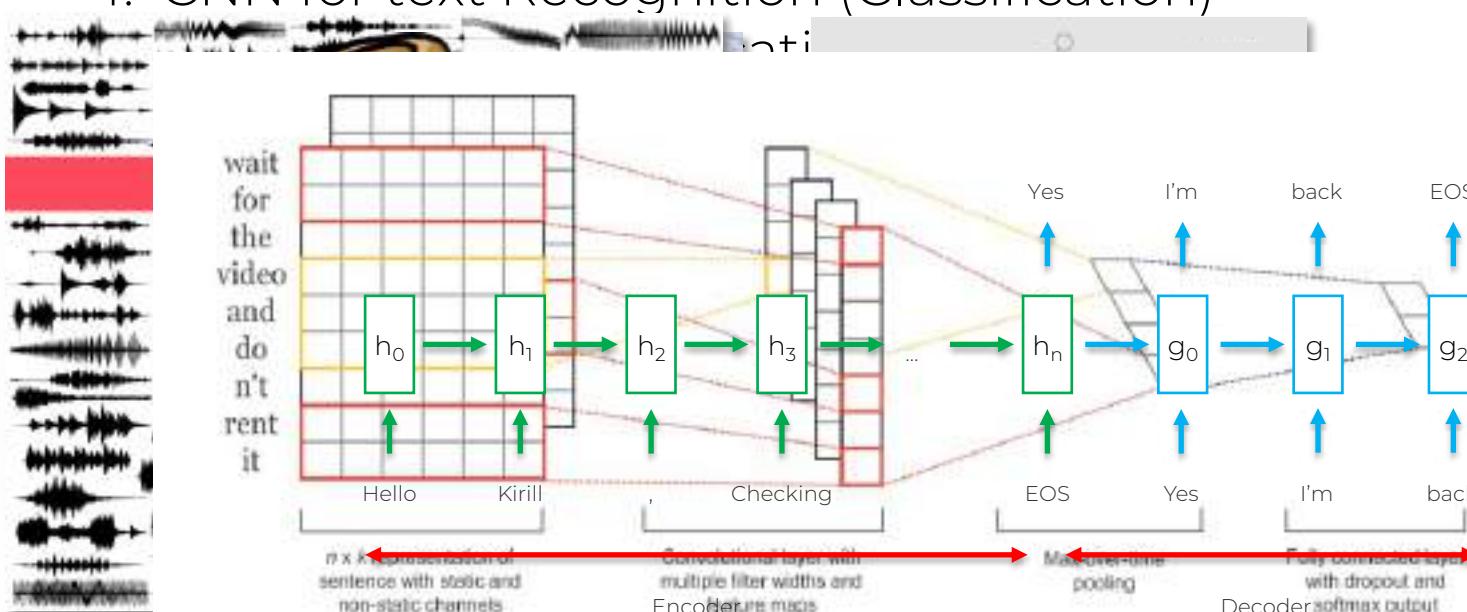
Seq2Seq

Deep
Learning

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)

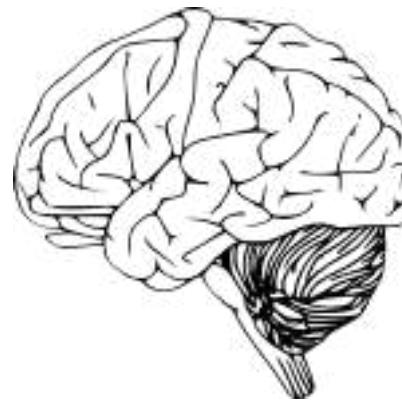
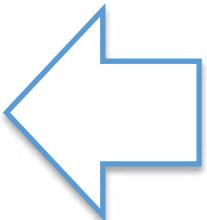
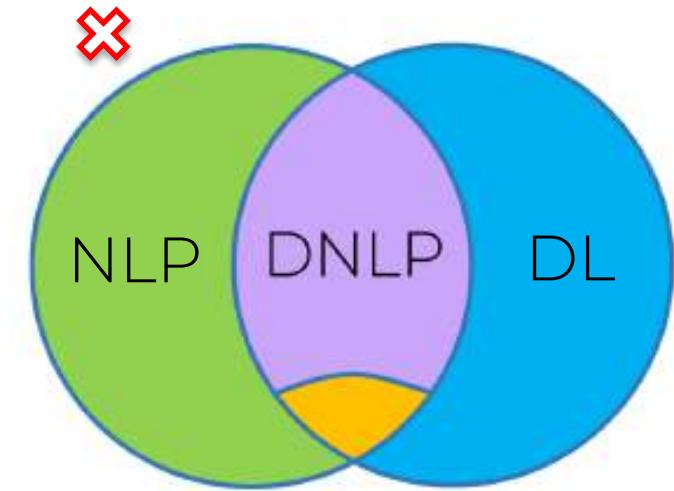
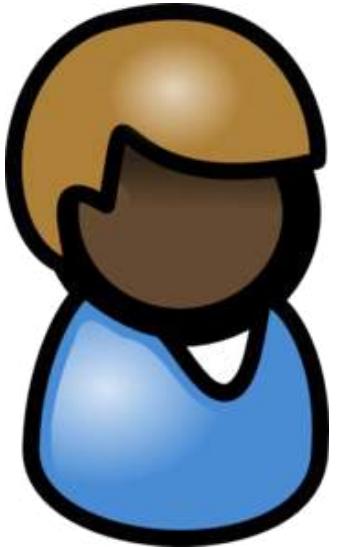


Seq2Seq

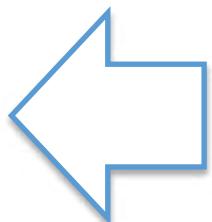
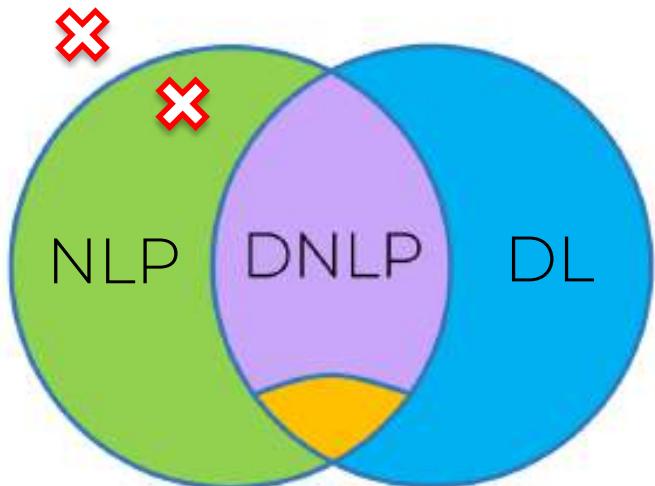
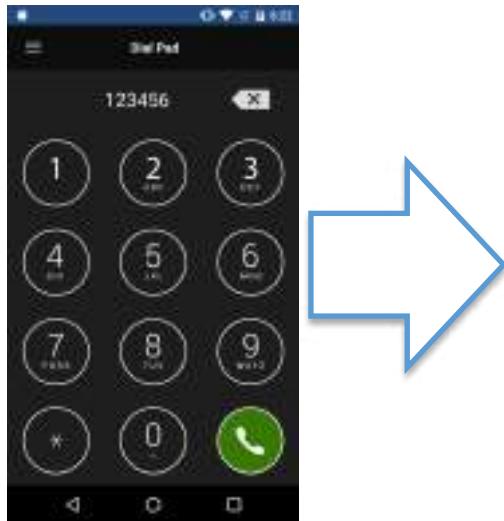
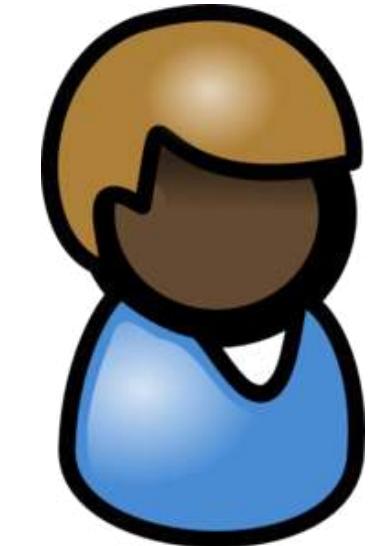
Image Source: www.wildml.com

End-to-end Deep Learning Models

End-to-end Deep Learning Models



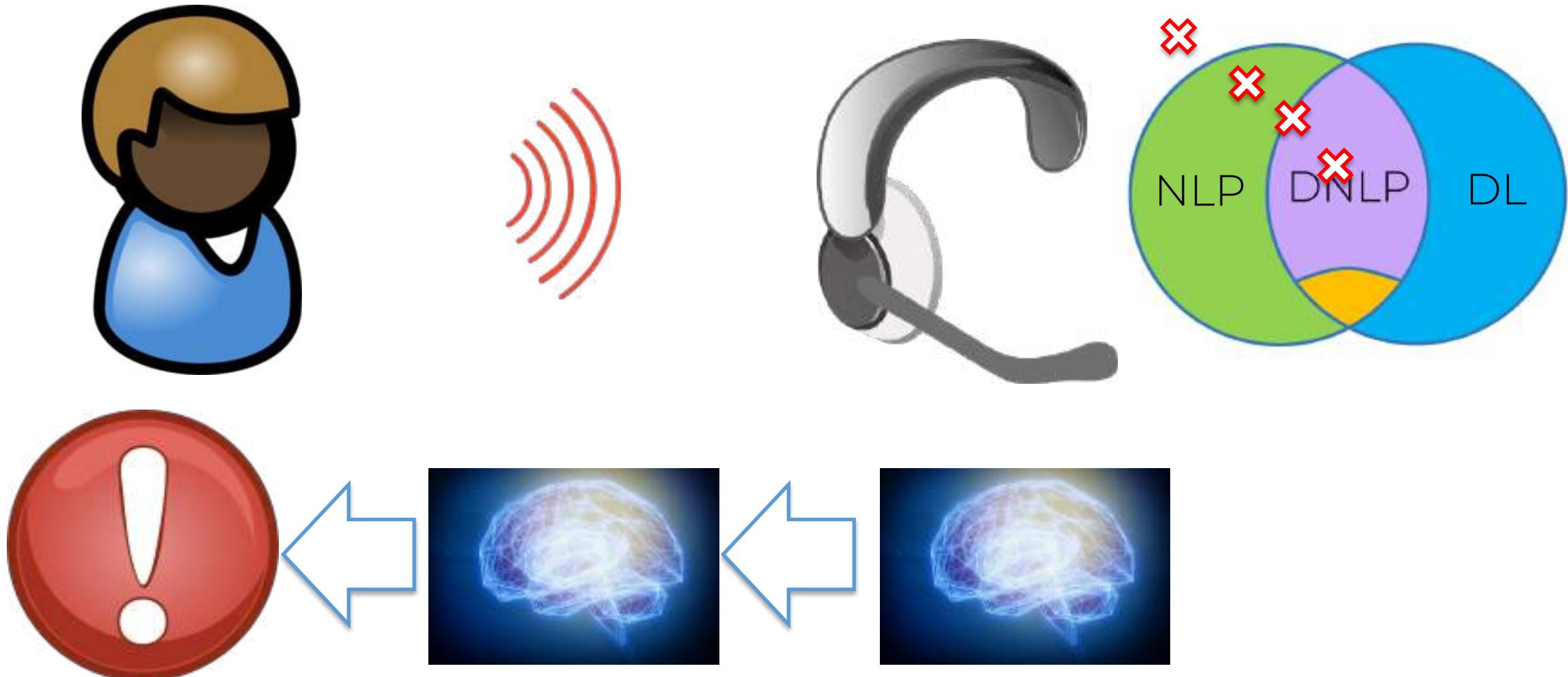
End-to-end Deep Learning Models



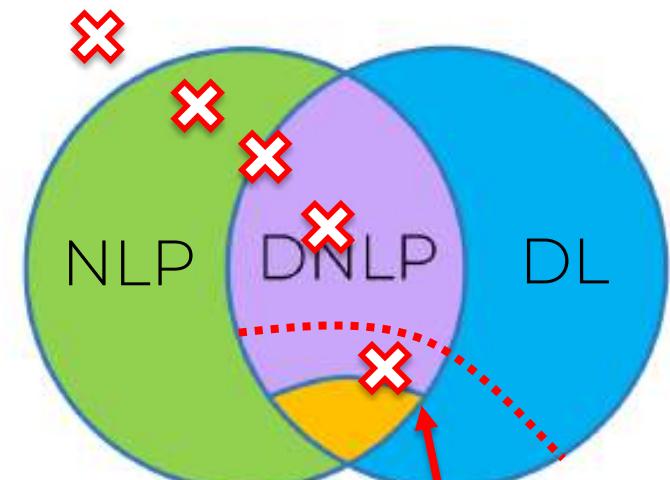
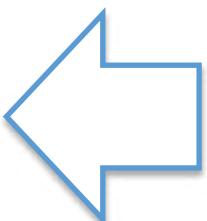
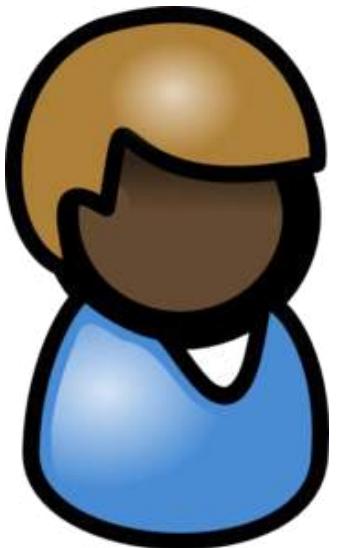
End-to-end Deep Learning Models



End-to-end Deep Learning Models



End-to-end Deep Learning Models



End-to-end
Deep Learning
Models

Bag-Of-Words



Catch up? [Inbox](#) [Business](#)



Kirill Eremenko

to me

6:18 pm ...

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

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://www.esl-dictionaries.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.english-test.net/vocabulary/words-in-the-english-language/)

Bag-Of-Words

A horizontal blue double-headed arrow, consisting of a thick blue line with arrows at both ends, spanning the width of the page.

SOS

EOS

Special Words

Bag-Of-Words

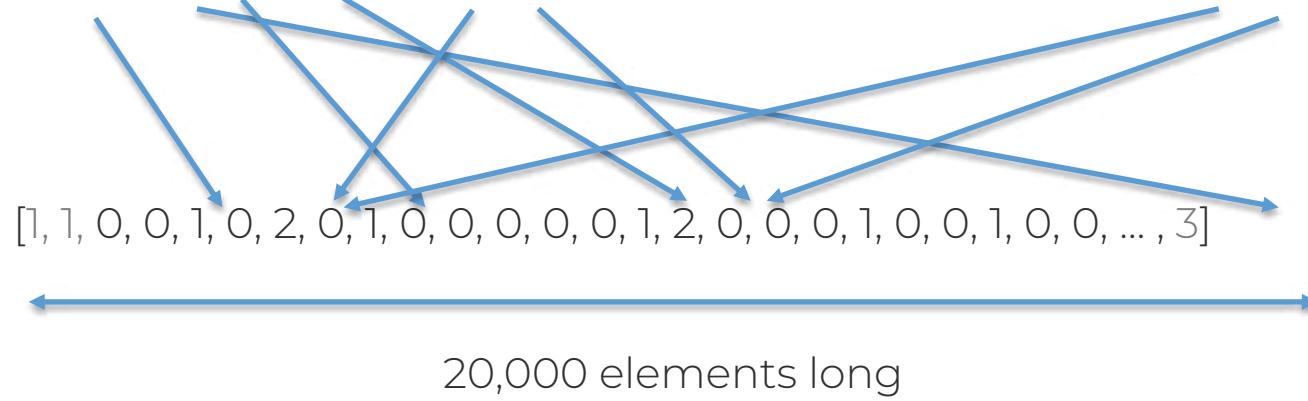
Hello Kirill, Checking if you are back to Oz. Let me know if you are around ... Cheers, V

← →

20,000 elements long

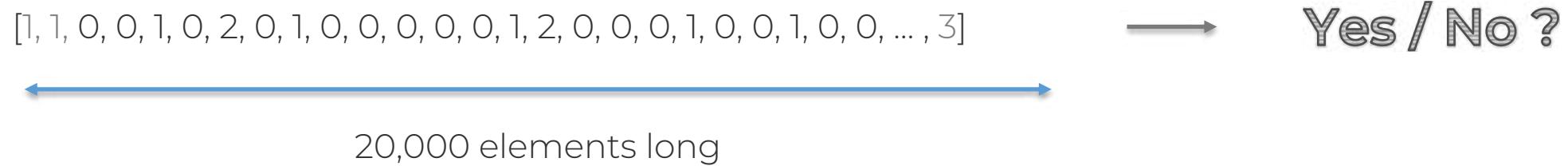
Bag-Of-Words

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



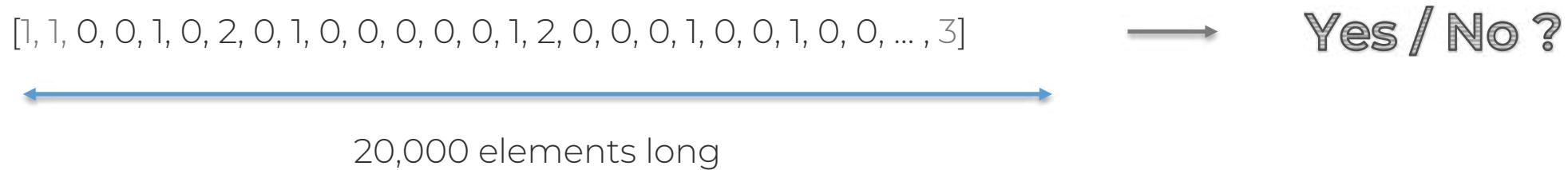
Bag-Of-Words

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



Bag-Of-Words

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



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?



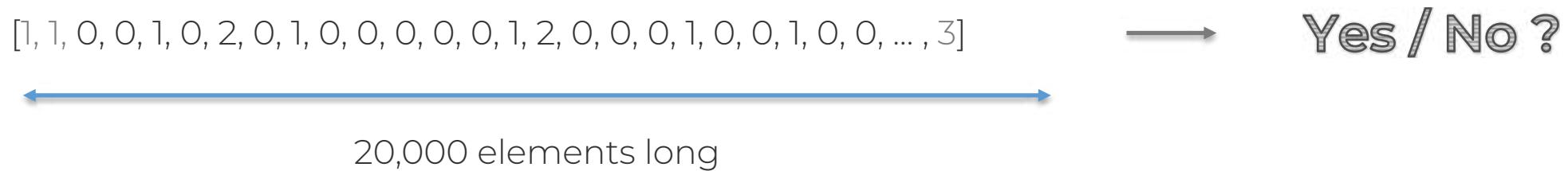
Yes

...

...

Bag-Of-Words

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

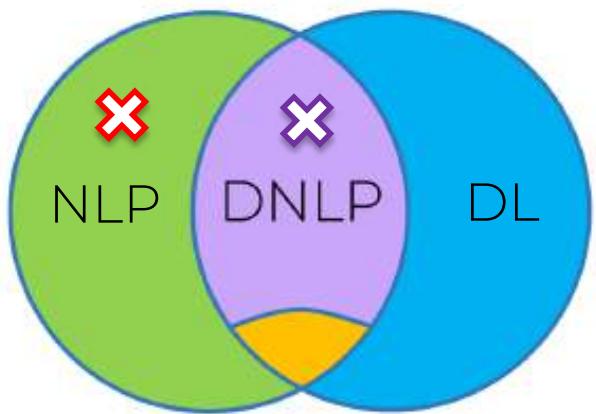


Training Data:

[1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, ..., 2]	→	No
[1, 1, 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, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, ..., 1]	→	Yes
[1, 1, 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, 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, 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, 1, 0, 0, 0]
[1, 1, 0, 0, 0]
[1, 1, 0, 0, 0]
[1, 1, 0, 0, 0]

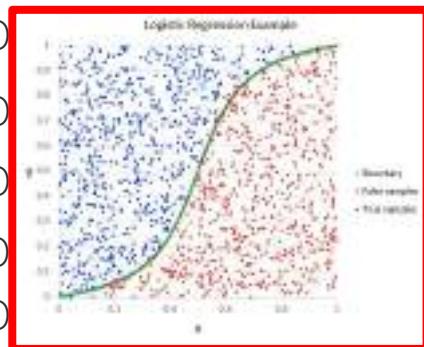
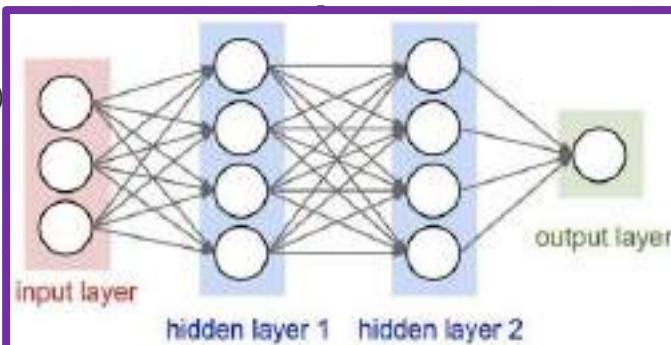


Image Source: www.helloacm.com



No
Yes
Yes
No
Yes
...



Artificial Neural Network Intuition

A Clip From The Today Show, 1994

What is Deep Learning?

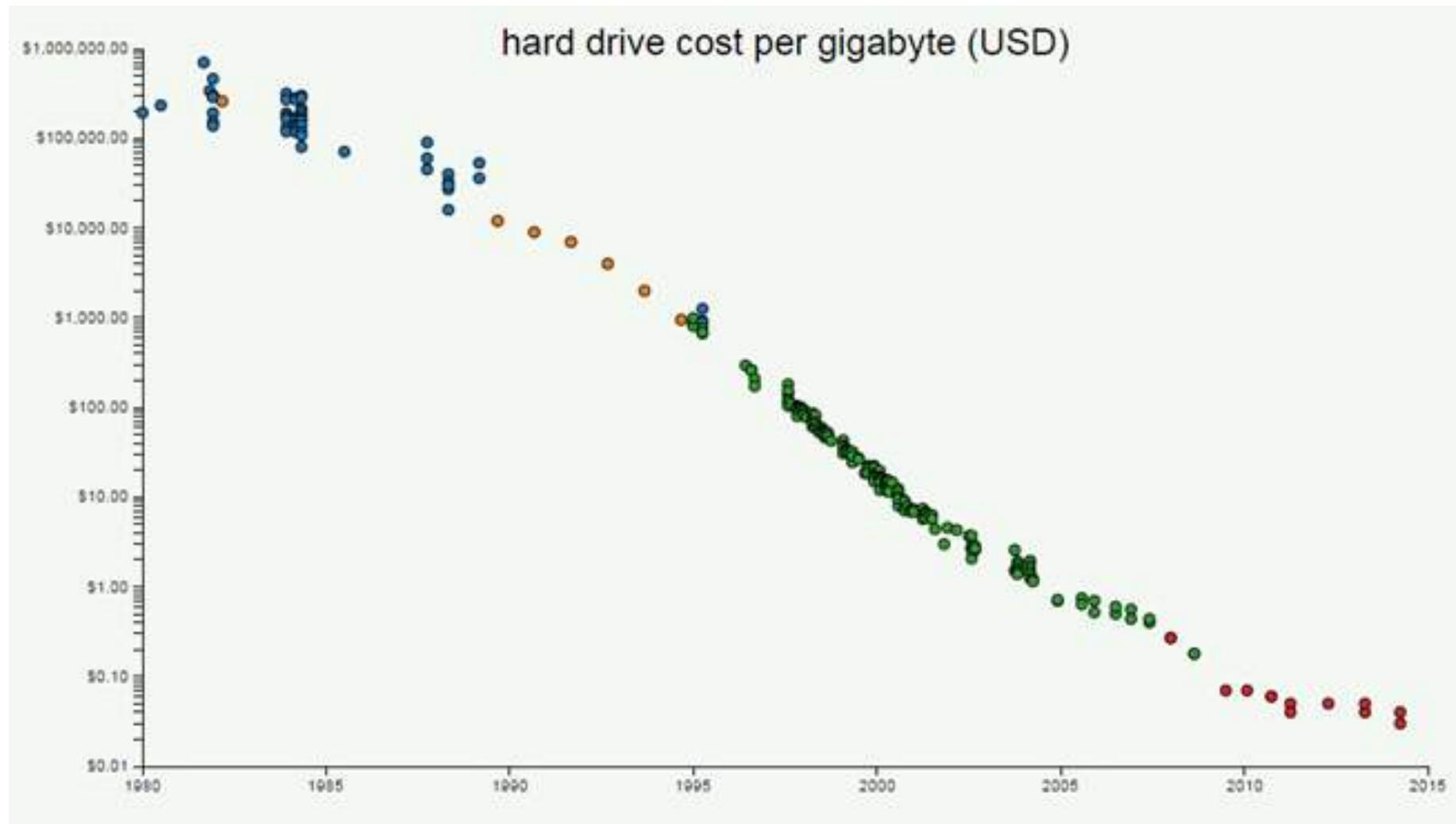
What is Deep Learning?



What is Deep Learning?



What is Deep Learning?



Source: mkomo.com

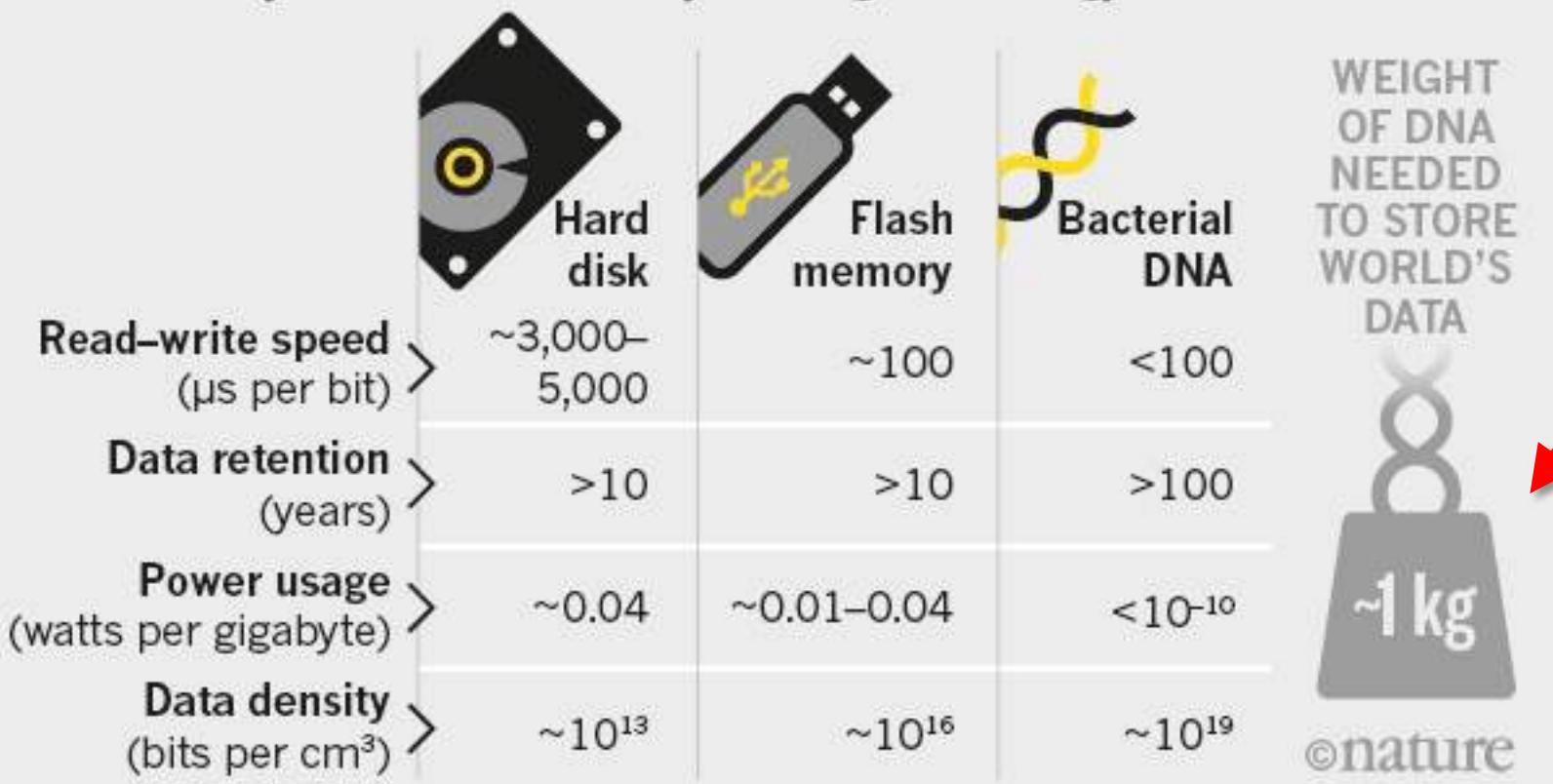
What is Deep Learning?



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.



Source: nature.com

What is Deep Learning?

1 The accelerating pace of change ...

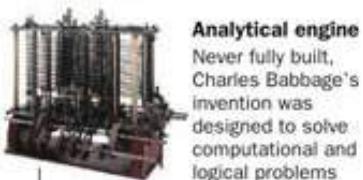


2 ... and exponential growth in computing power ...

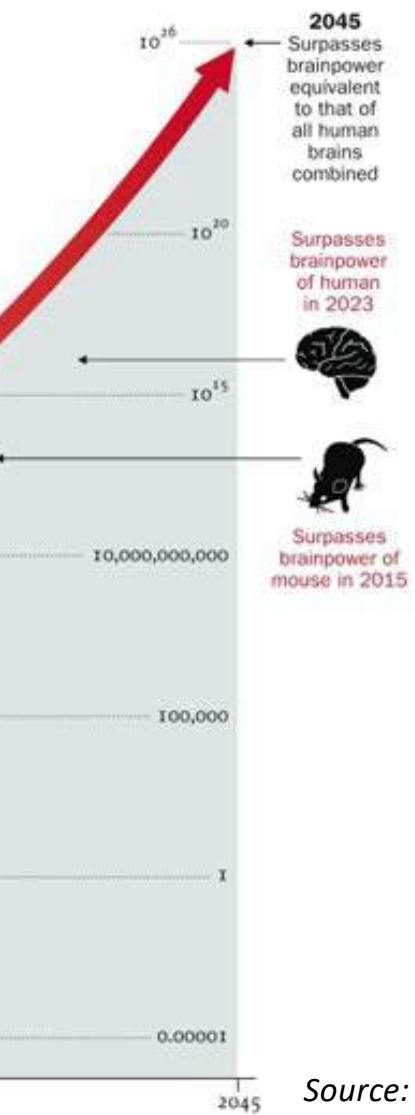
Computer technology, shown here climbing dramatically by powers of 10, is now progressing more each hour than it did in its entire first 90 years

COMPUTER RANKINGS

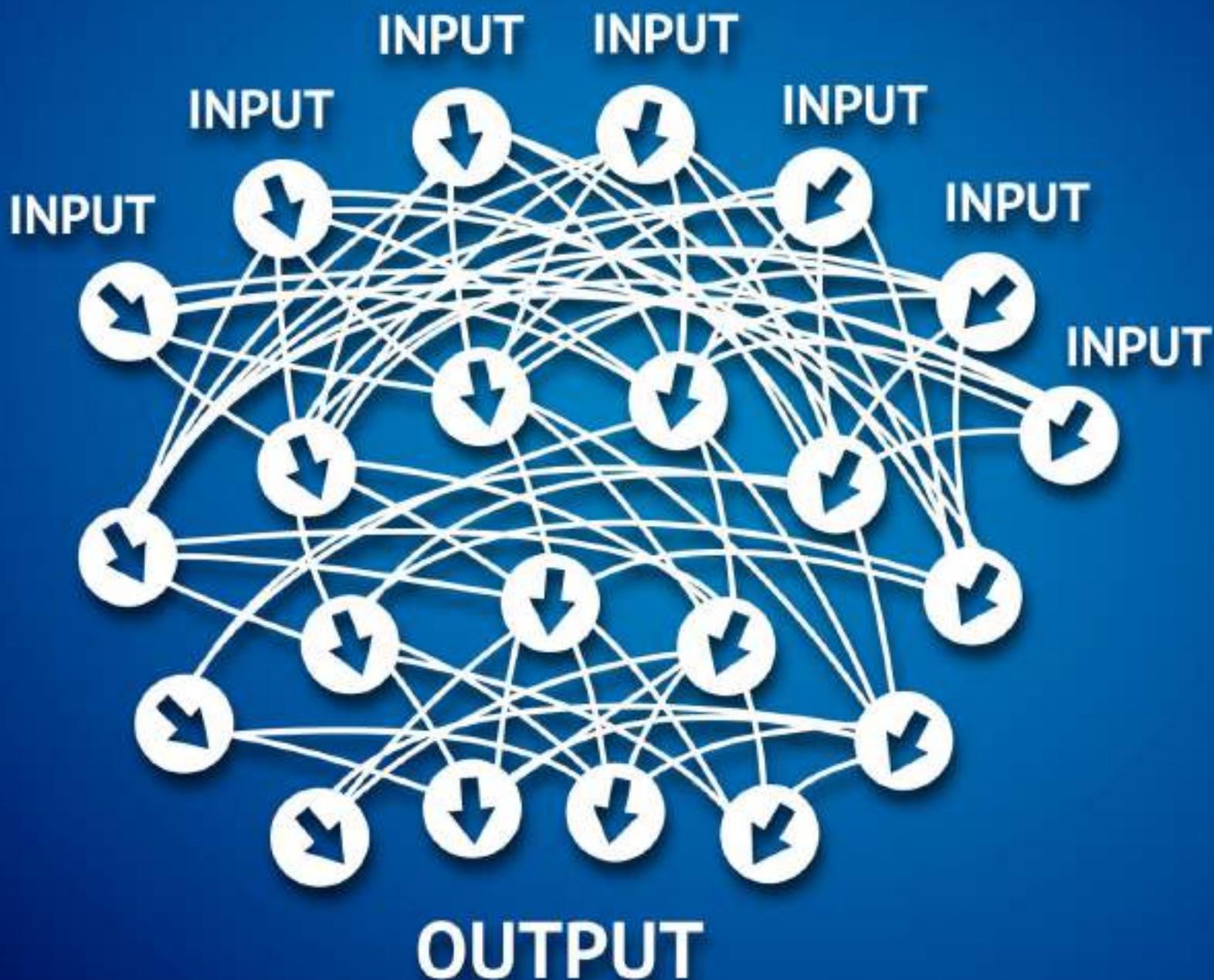
By calculations per second per \$1,000



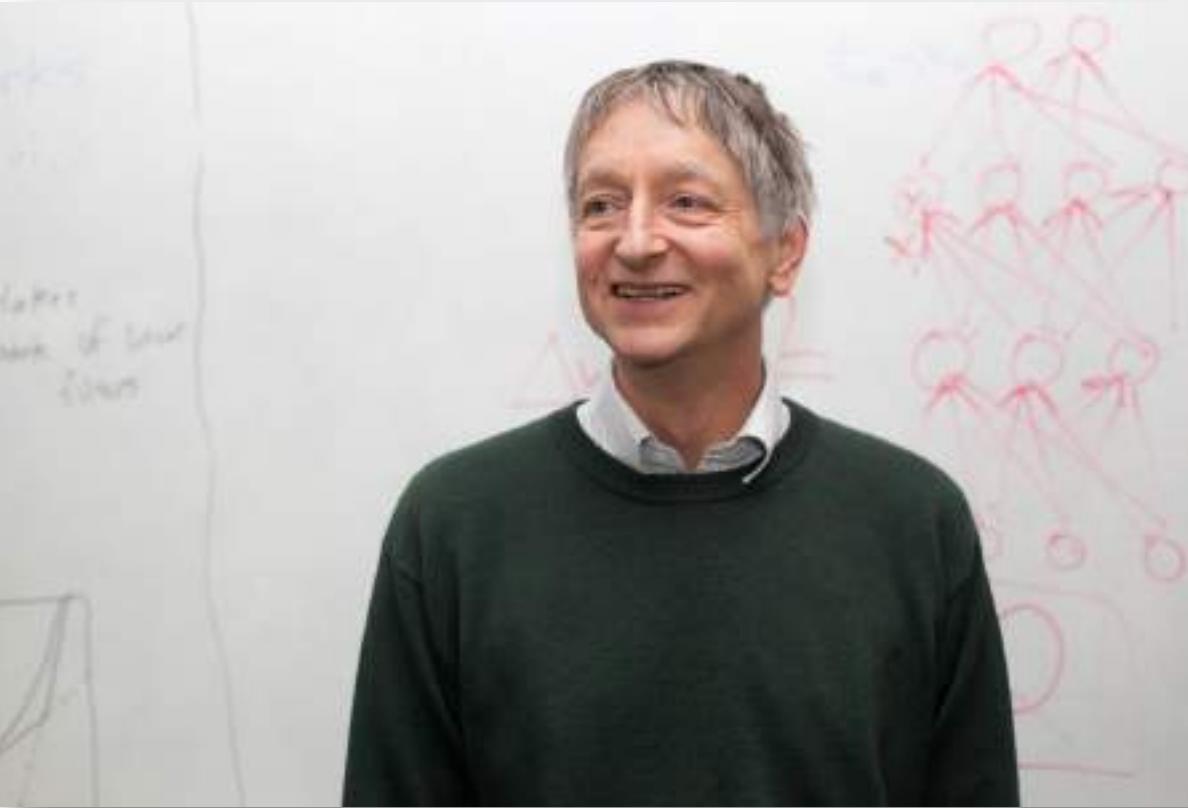
3 ... will lead to the Singularity



Source: Time Magazine



What is Deep Learning?



Geoffrey Hinton

What is Deep Learning?

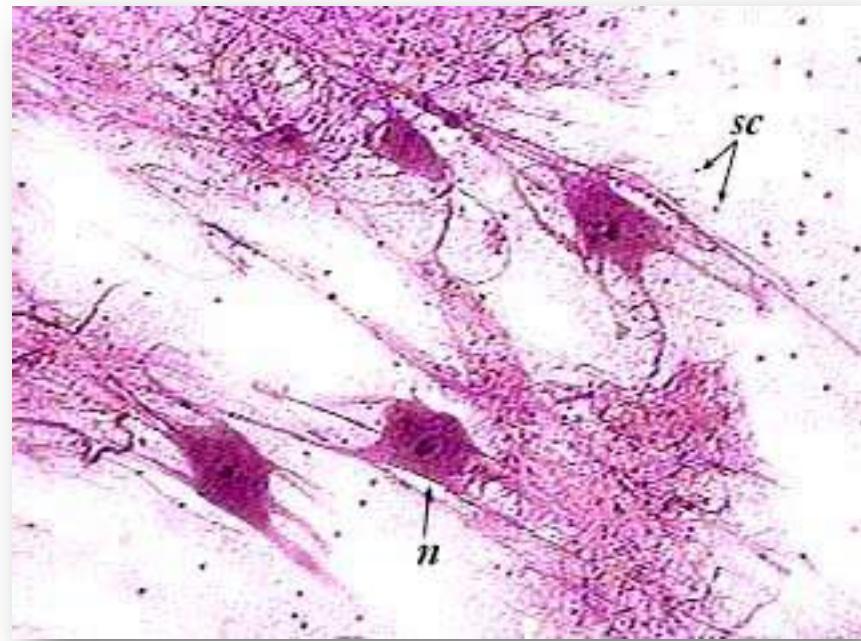
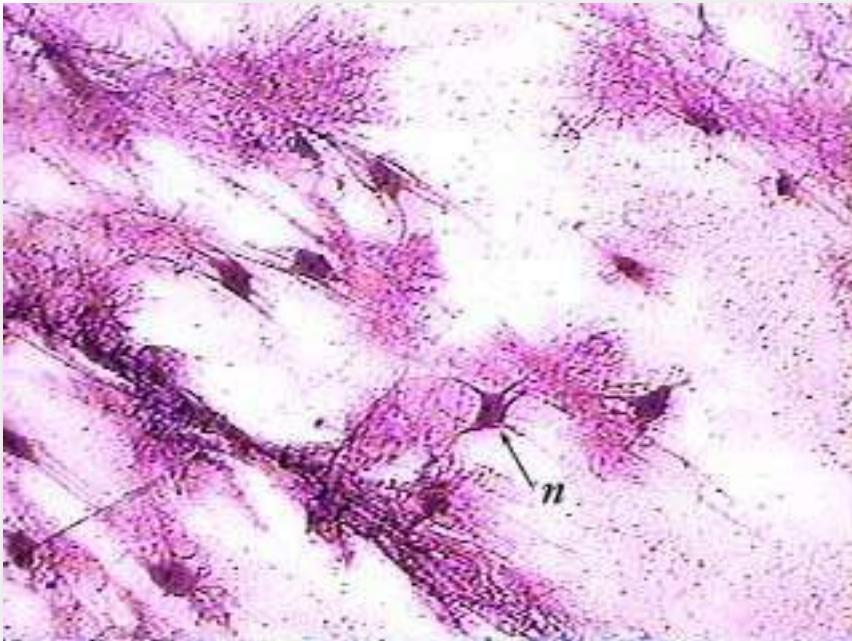
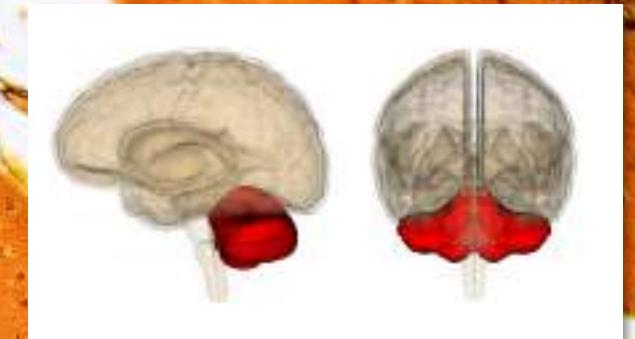
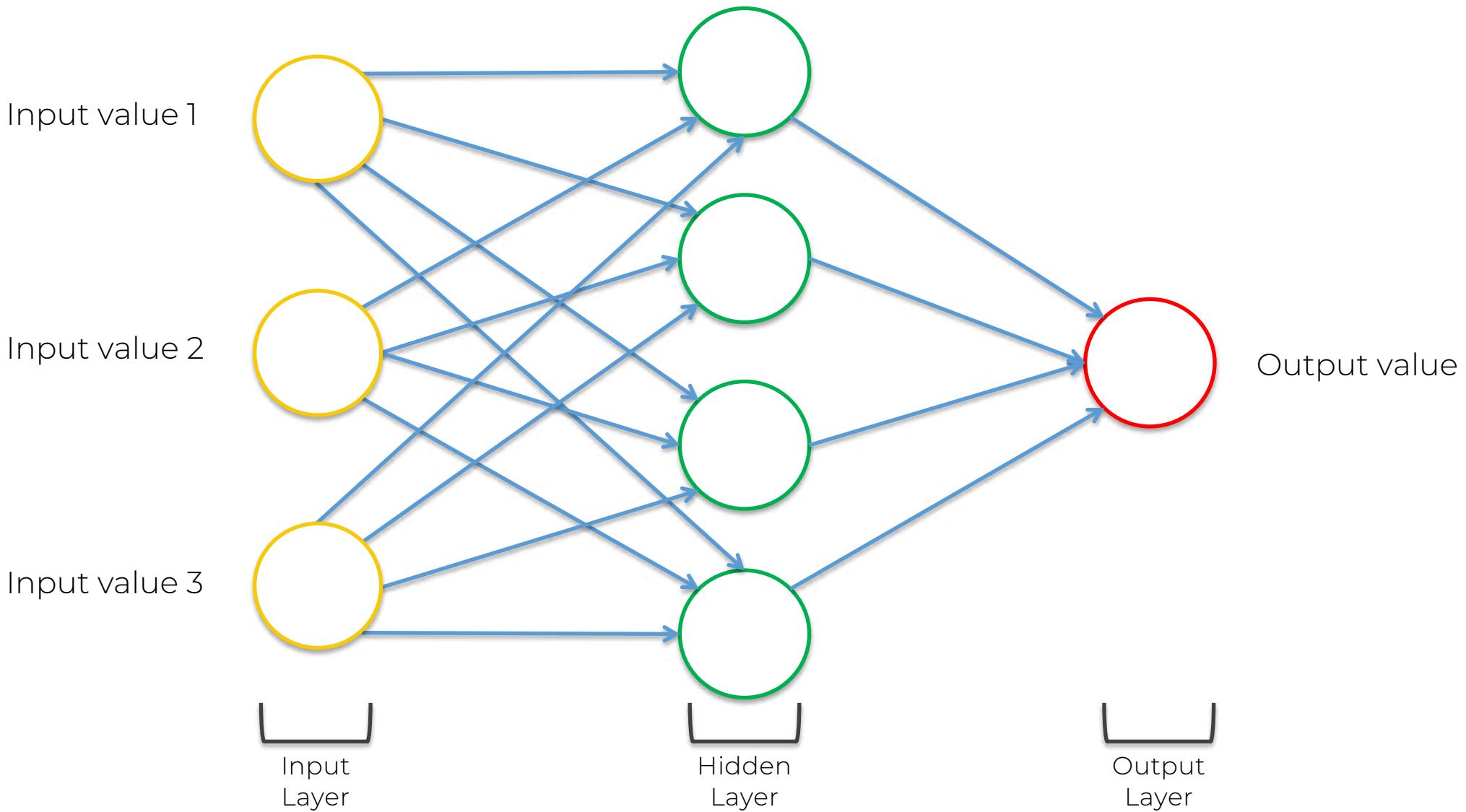


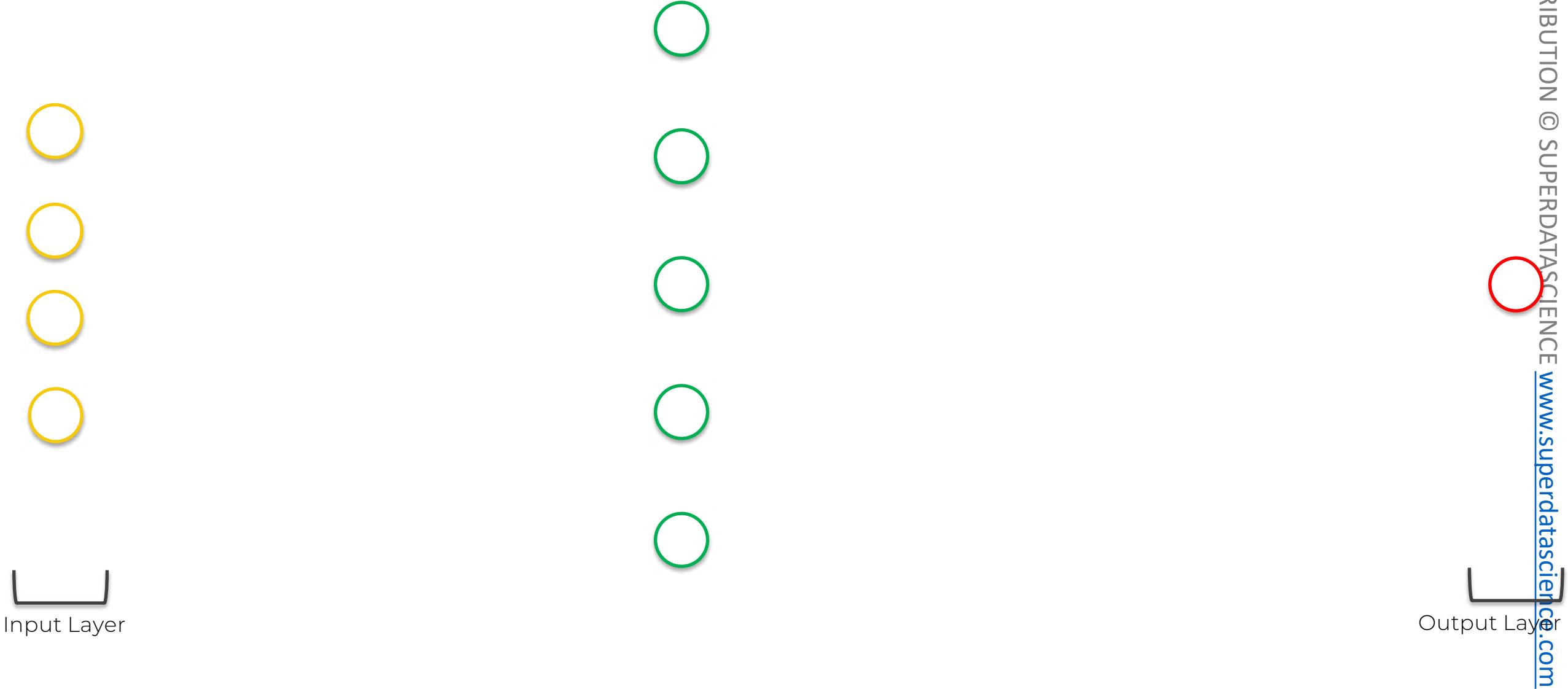
Image Source: www.austincc.edu



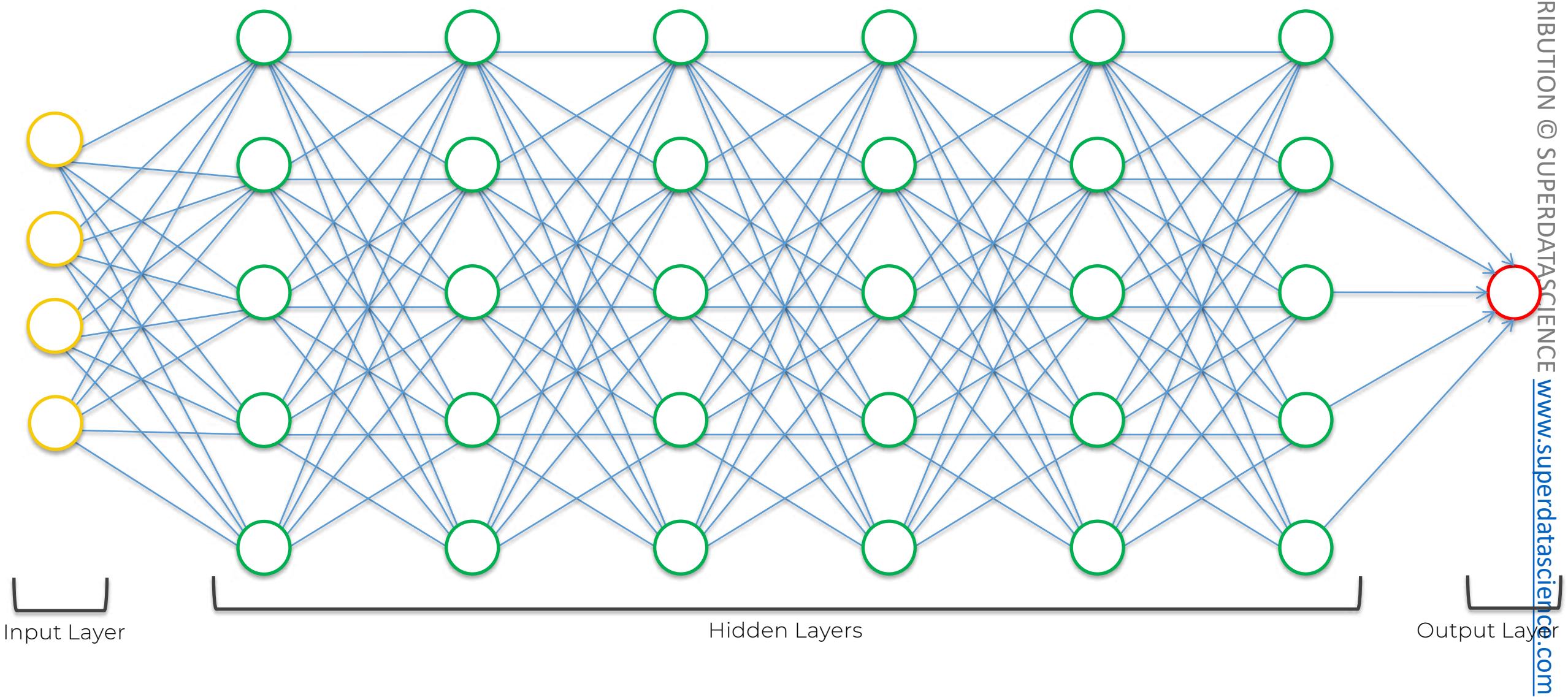
What is Deep Learning?



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

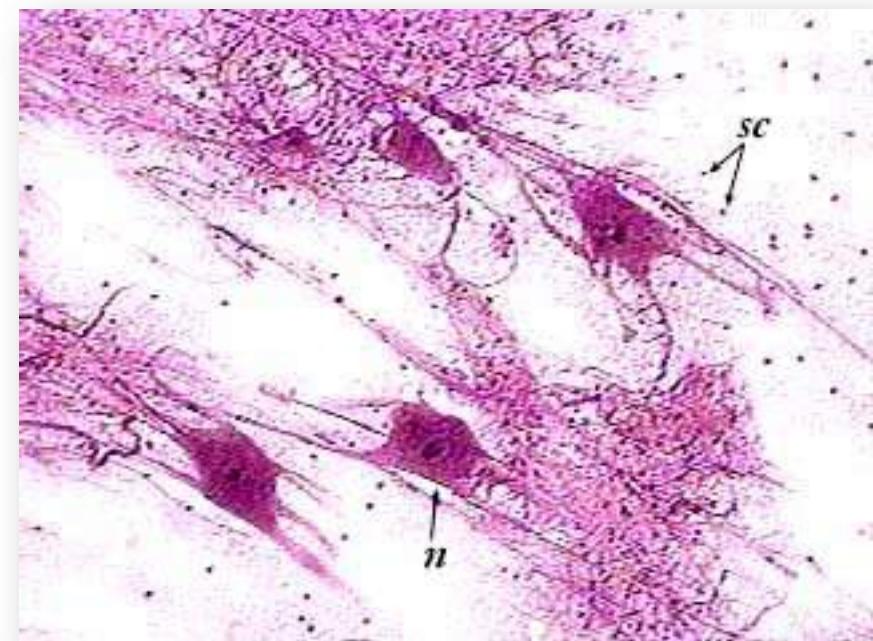
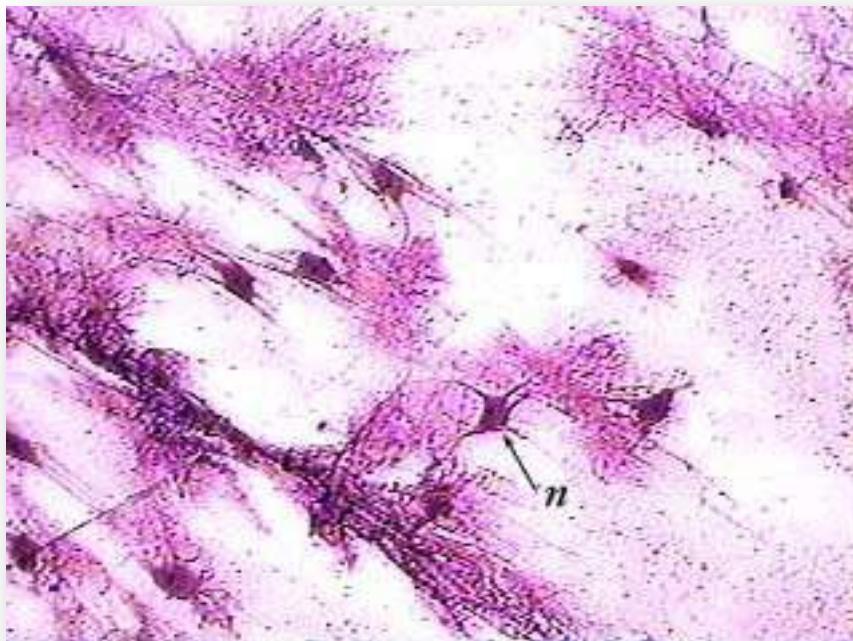


Image Source: www.austincc.edu

The Neuron

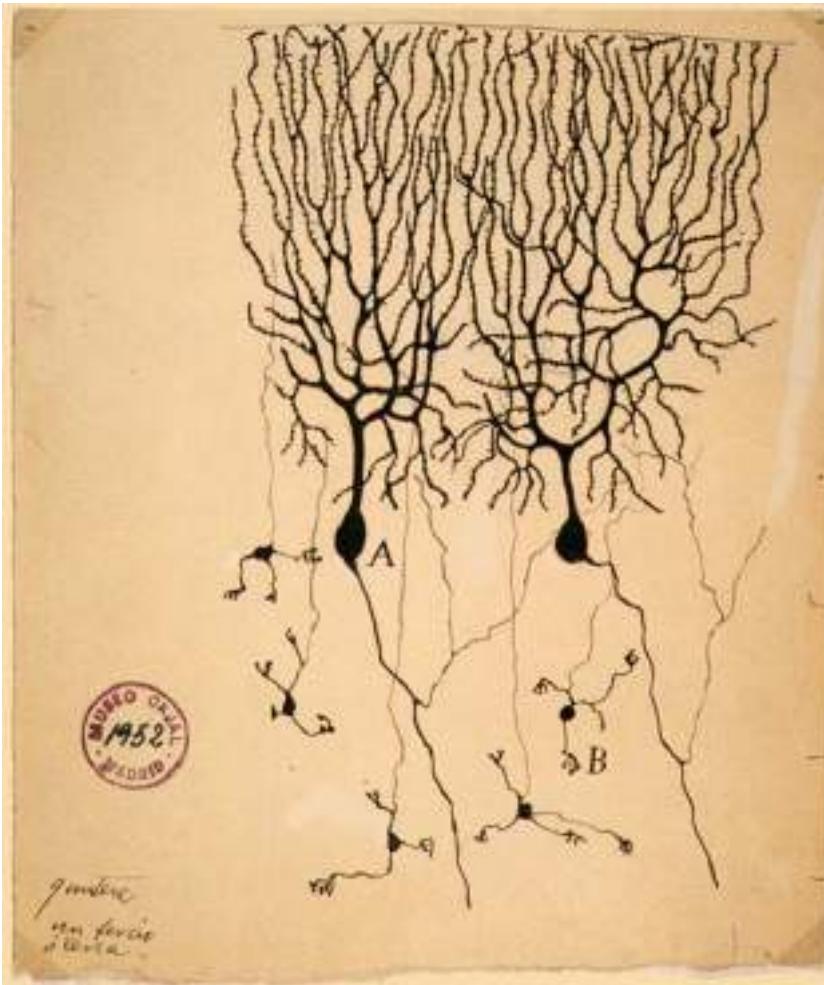


Image Source: Wikipedia

The Neuron

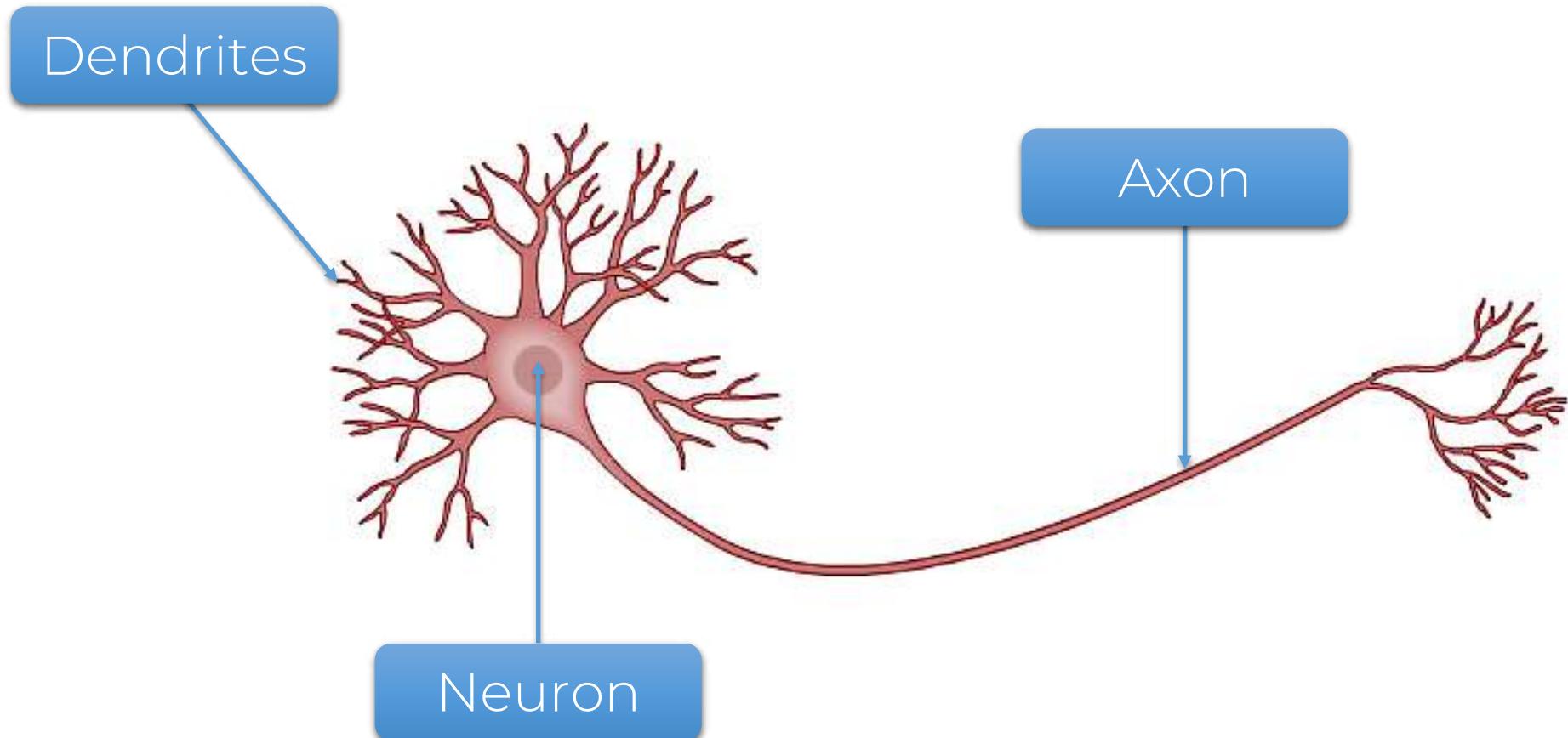


Image Source: Wikipedia

The Neuron

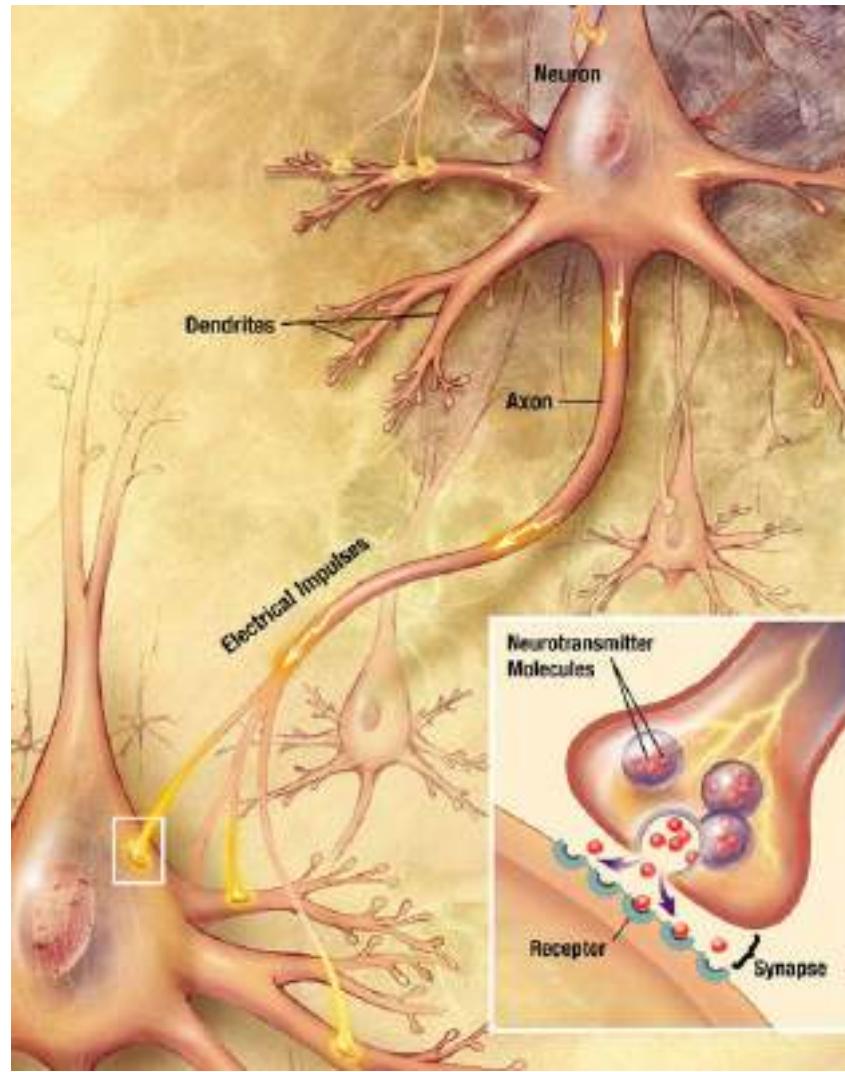
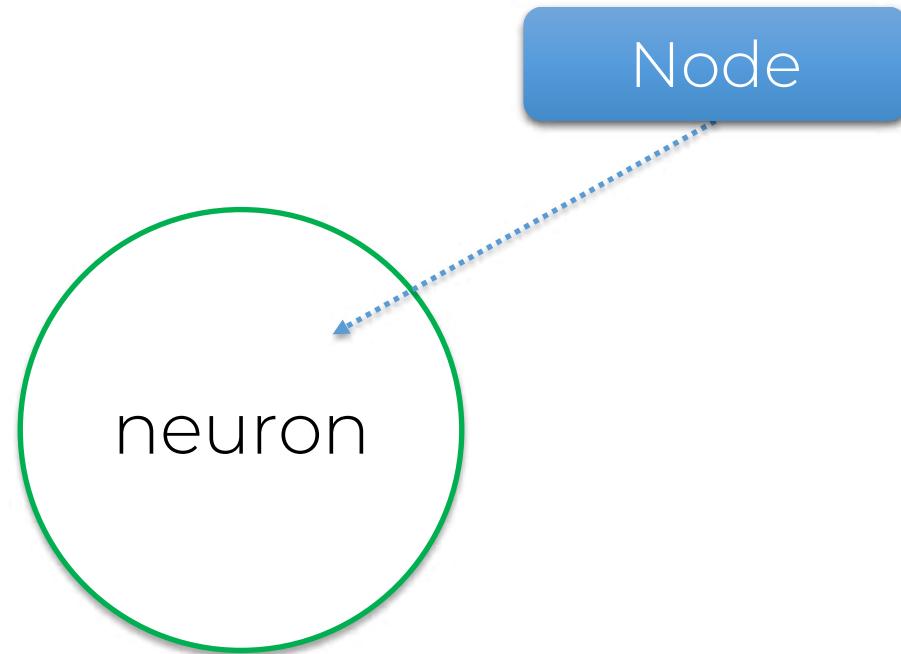
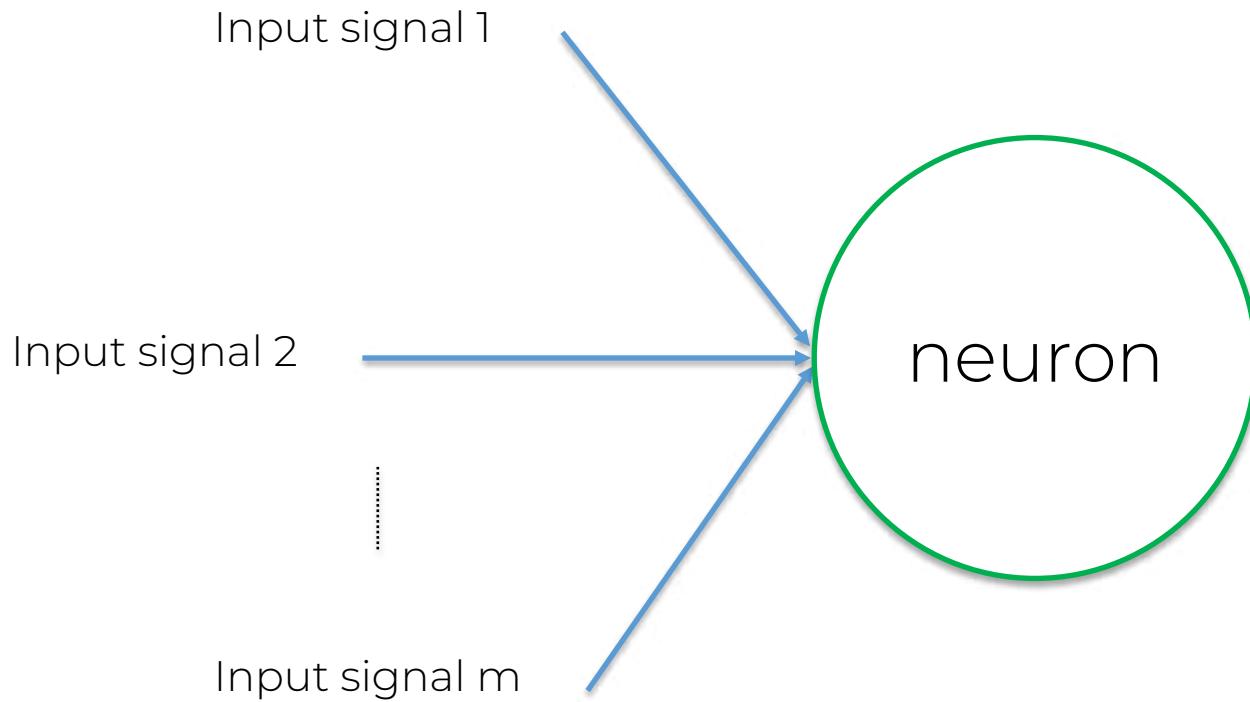


Image Source: Wikipedia

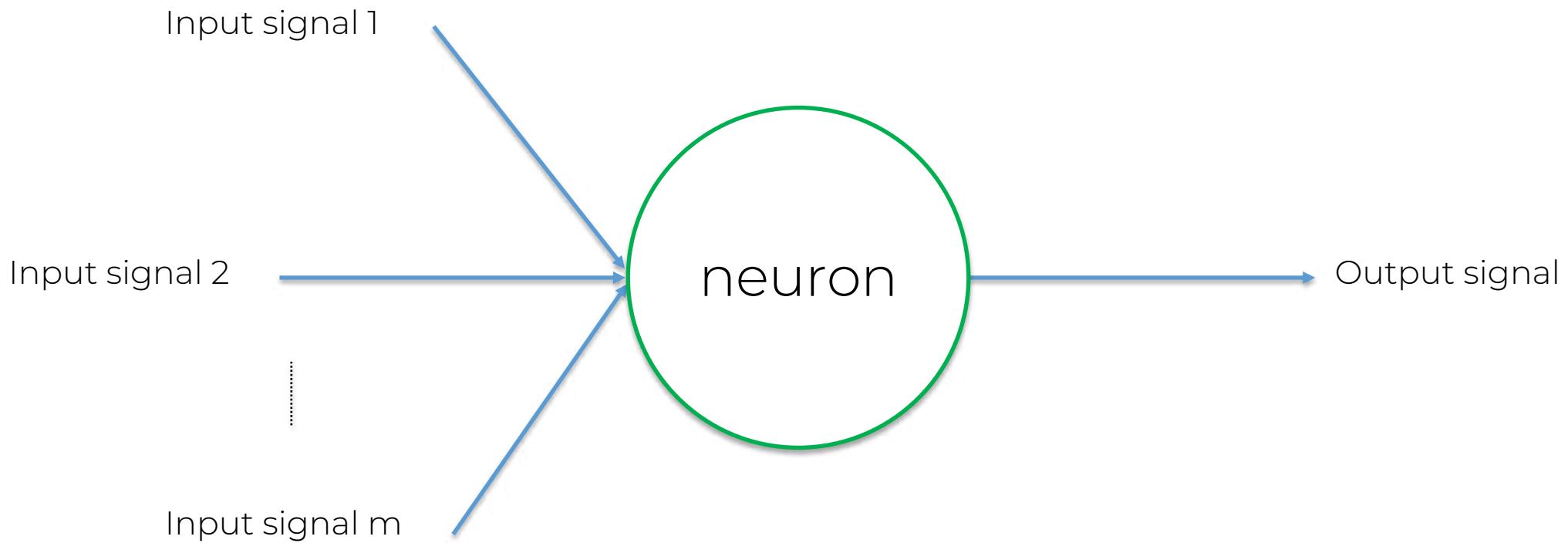
The Neuron



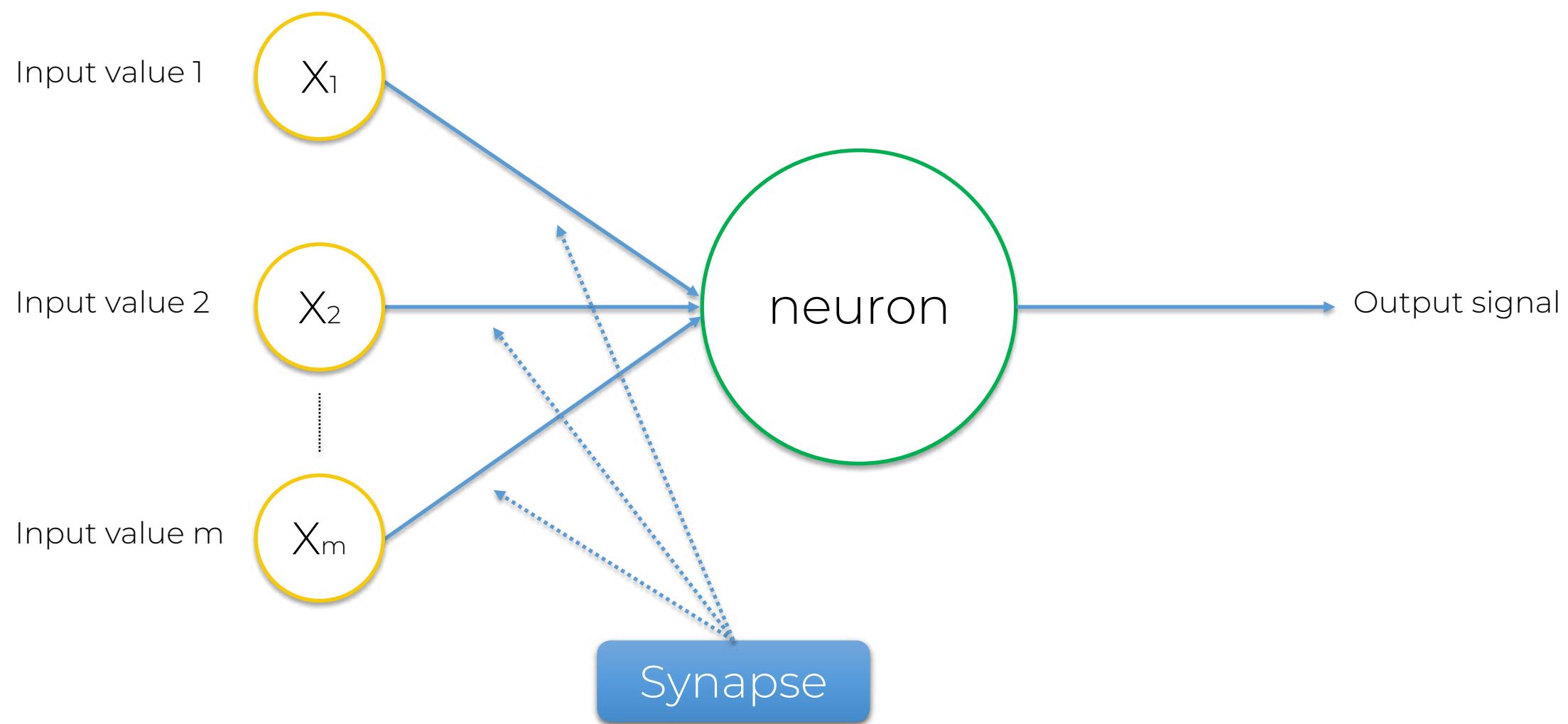
The Neuron



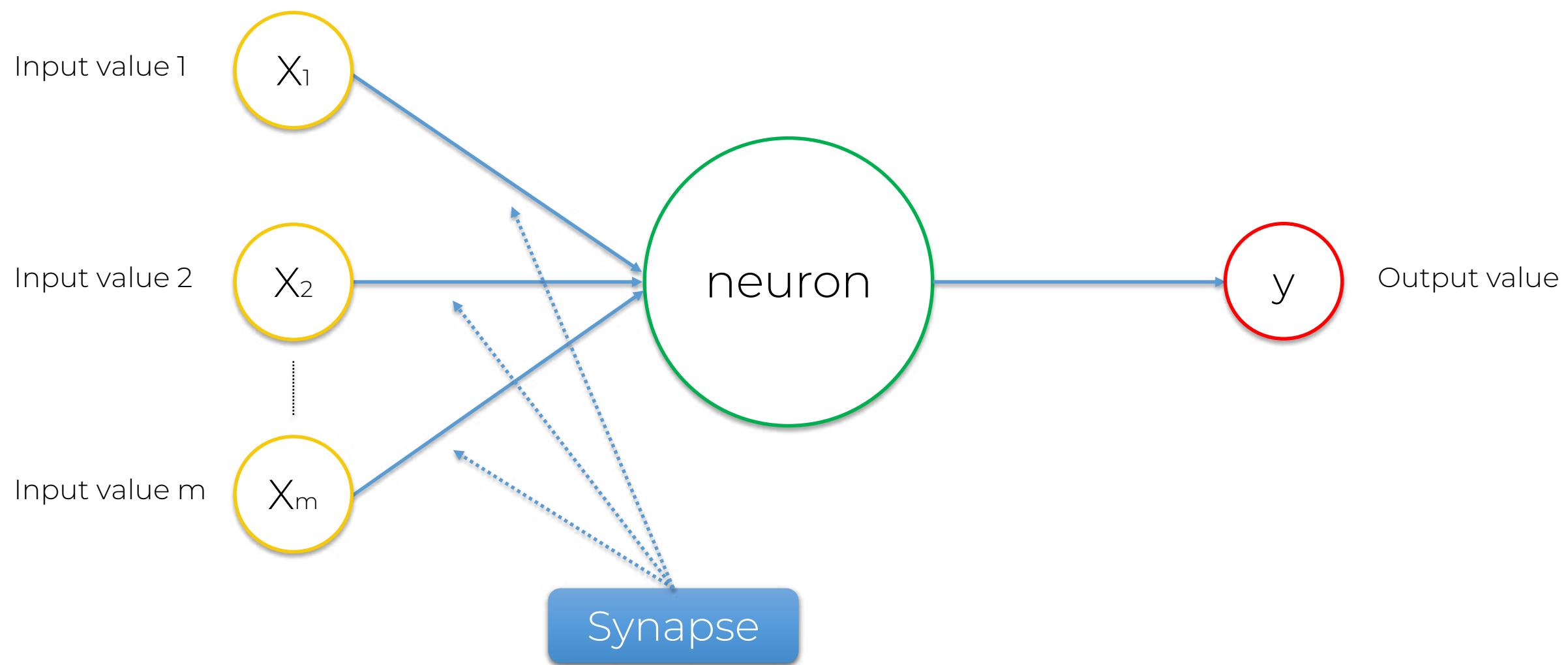
The Neuron



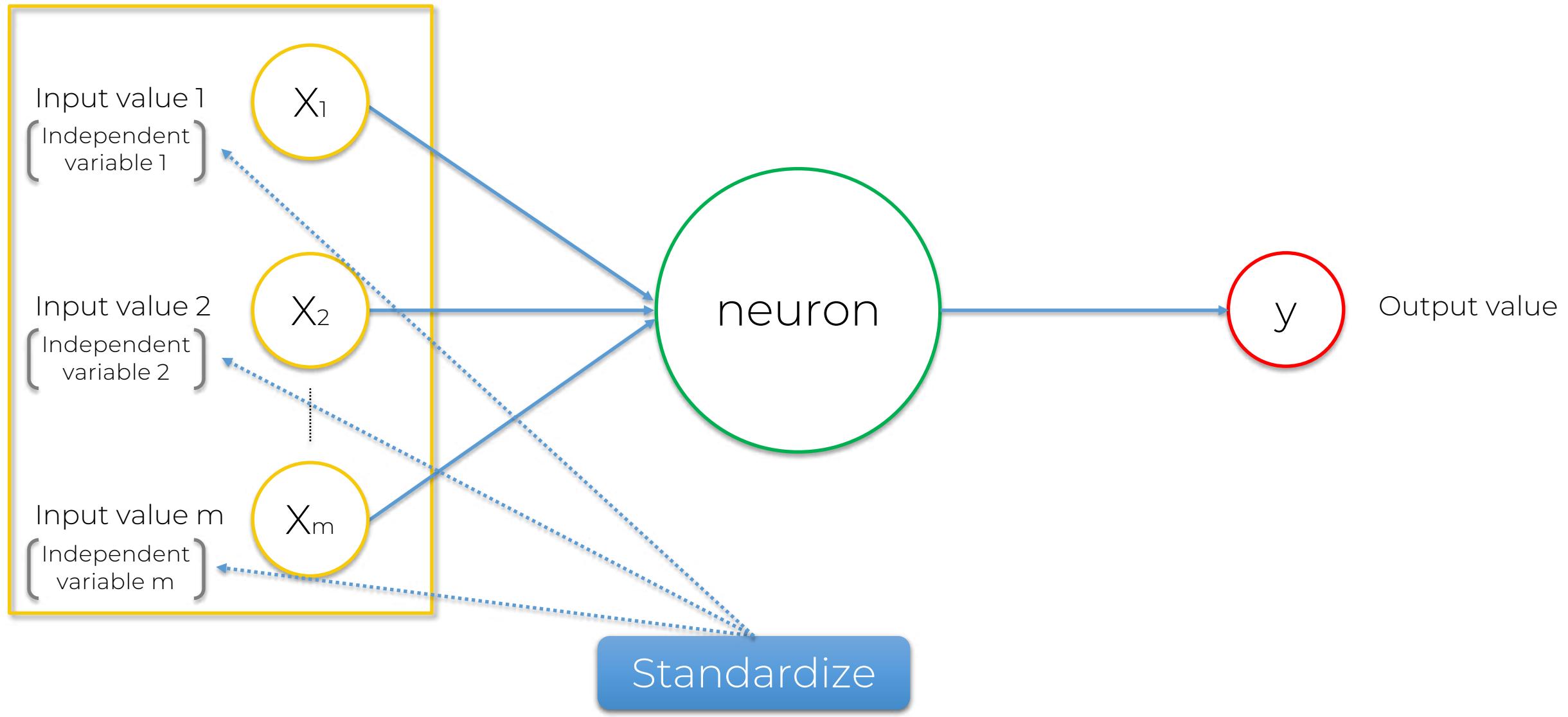
The Neuron



The Neuron



The Neuron



The Neuron

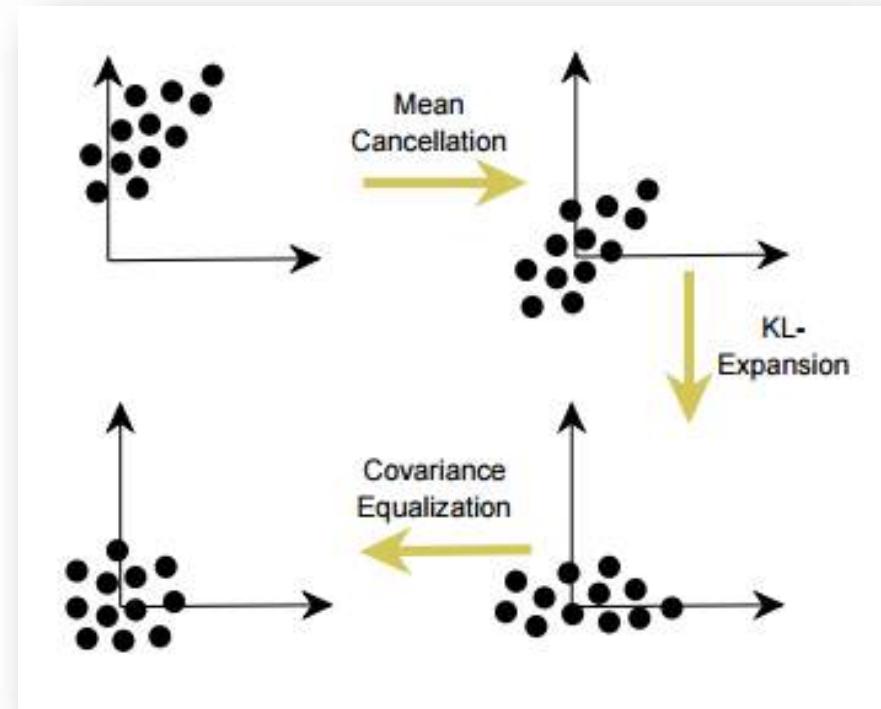
Additional Reading:

Efficient BackProp

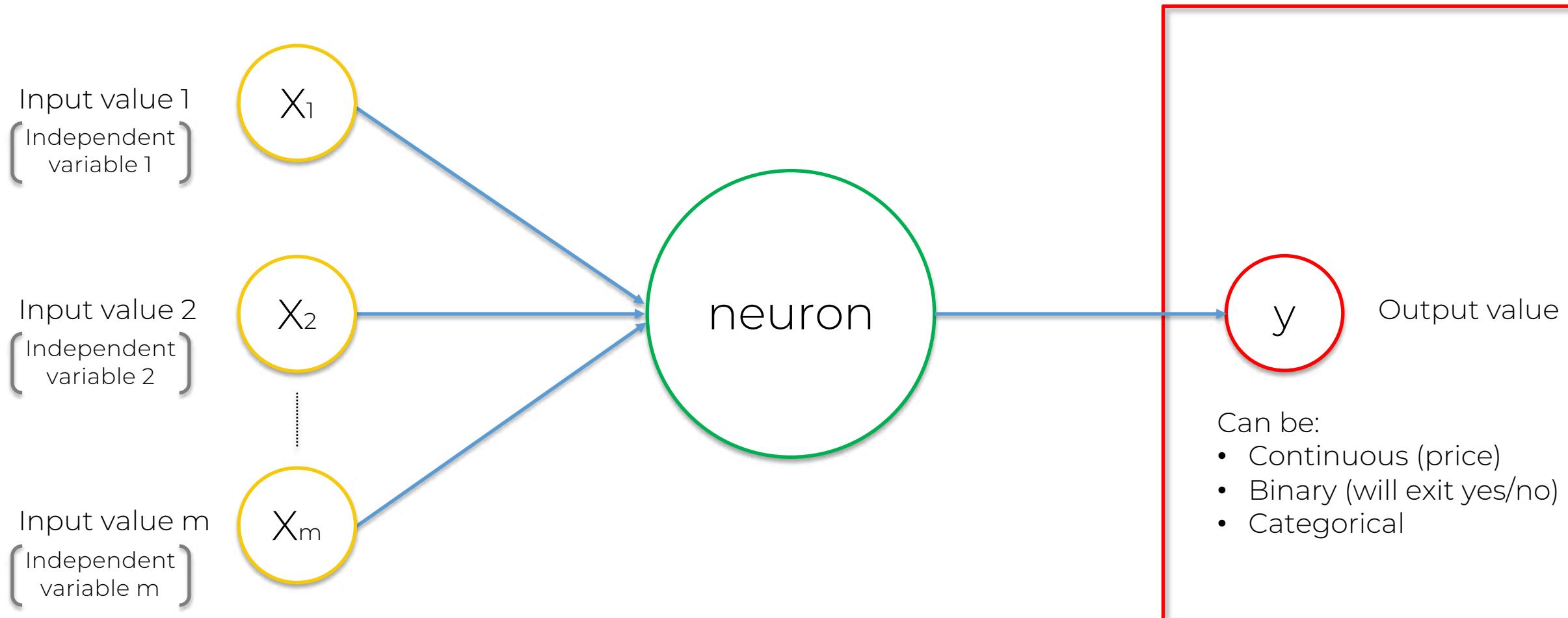
By Yann LeCun et al. (1998)

Link:

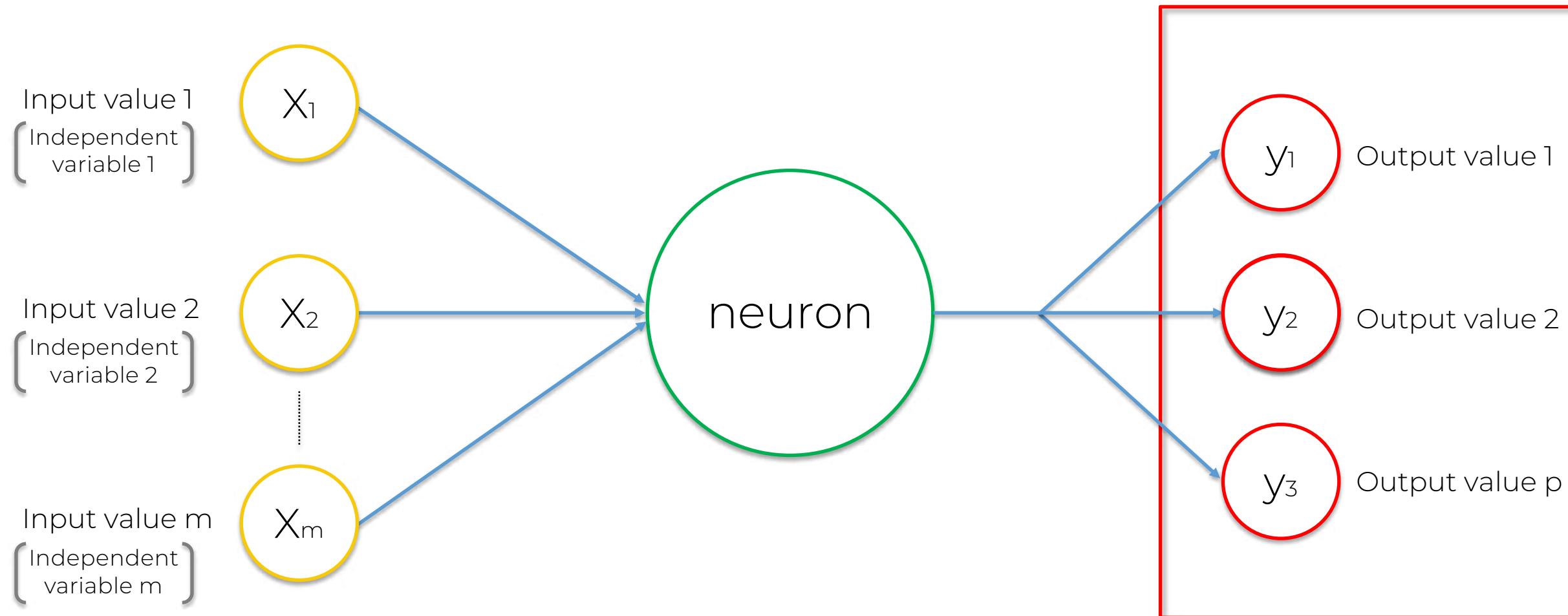
<http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf>



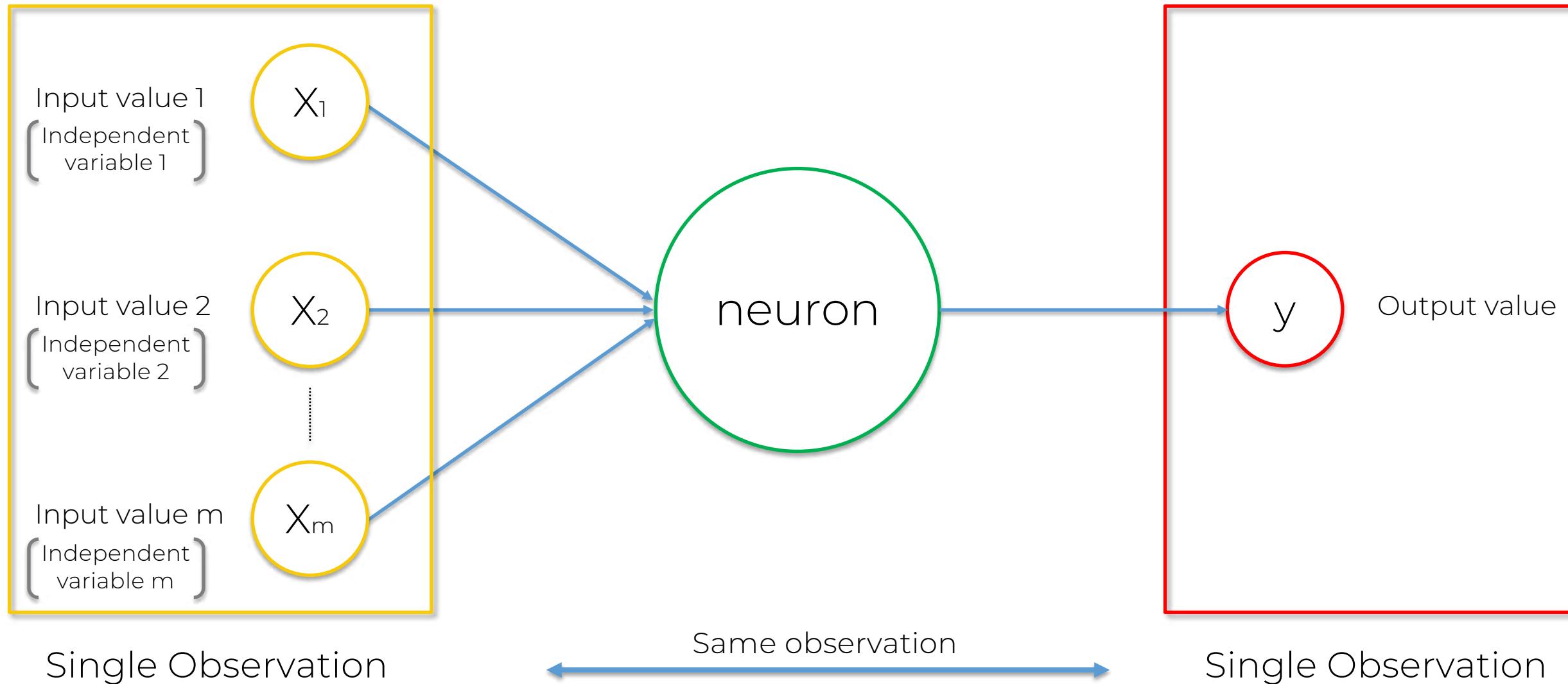
The Neuron



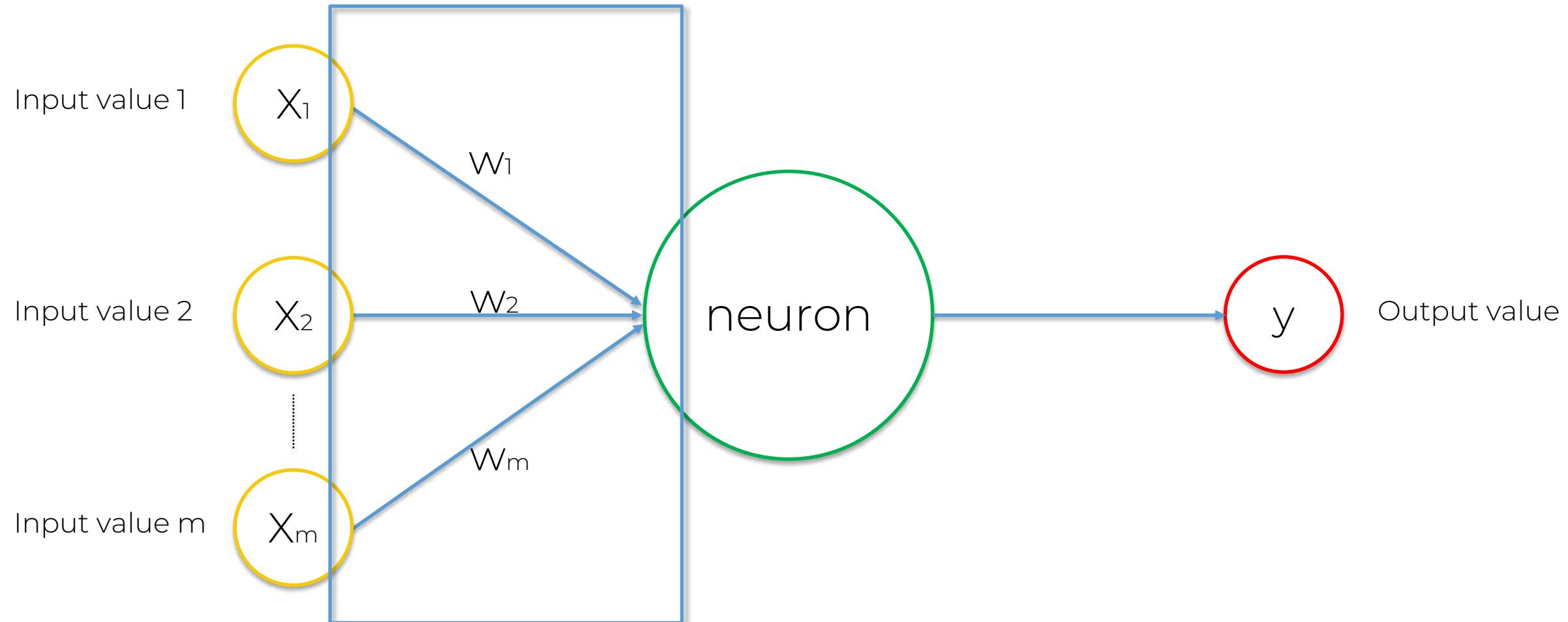
The Neuron



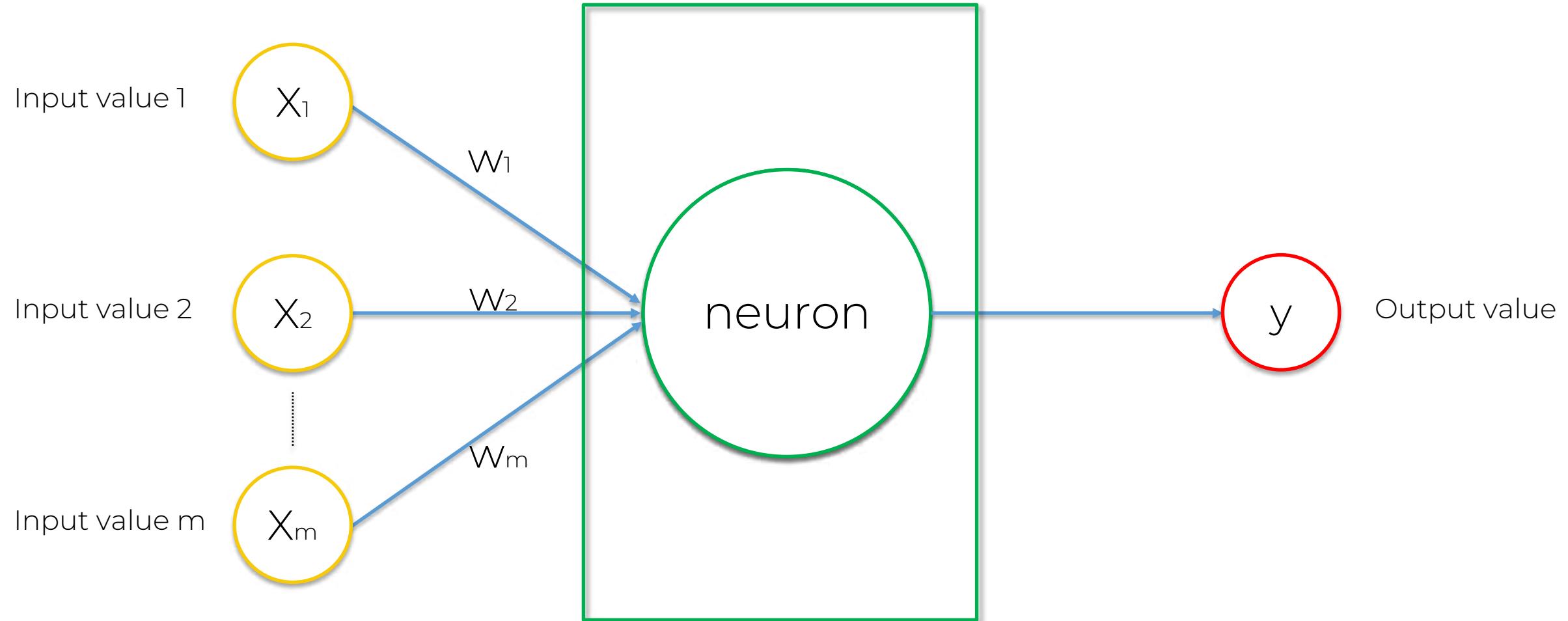
The Neuron



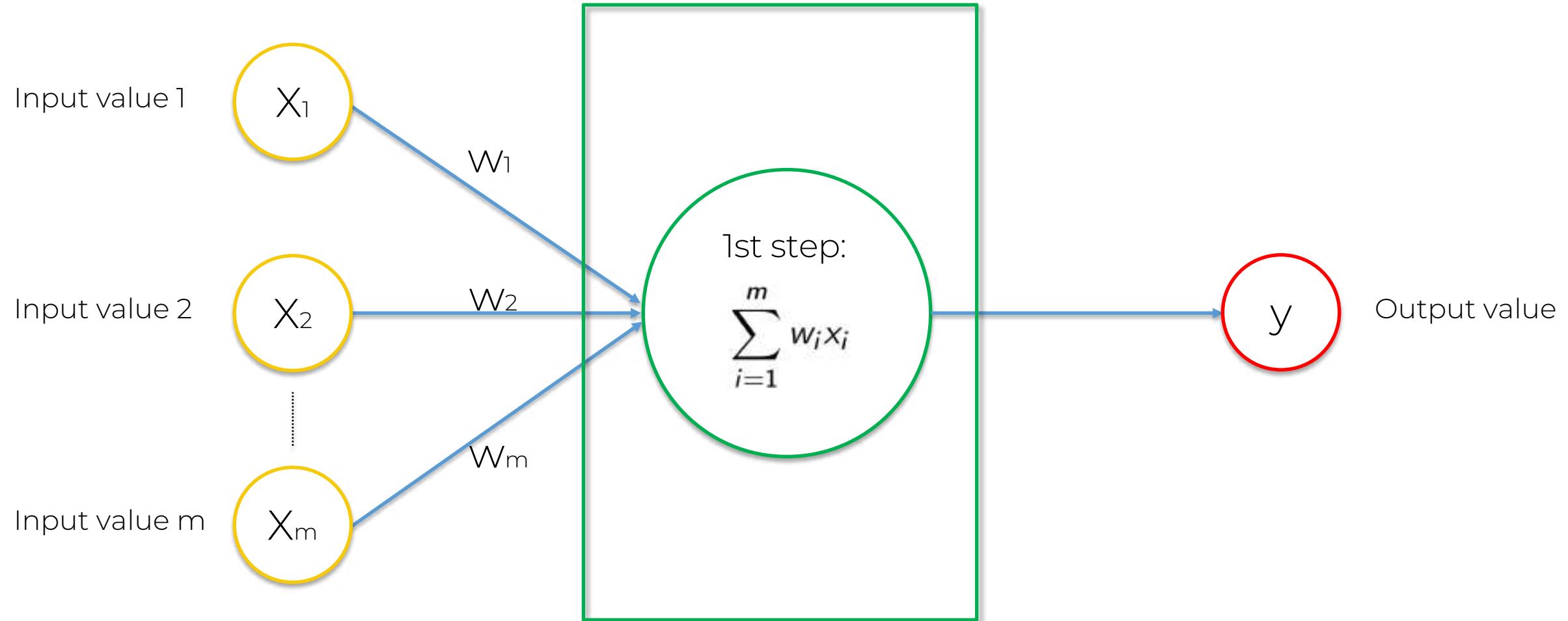
The Neuron



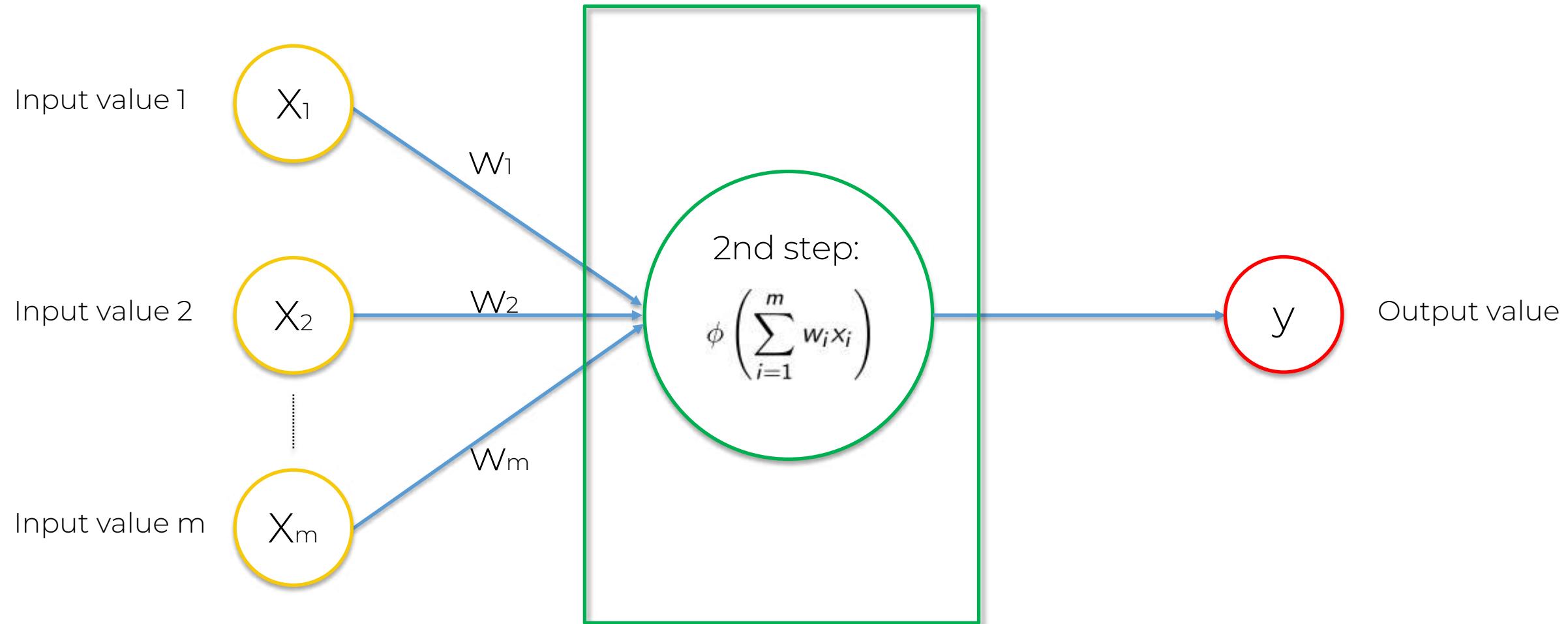
The Neuron



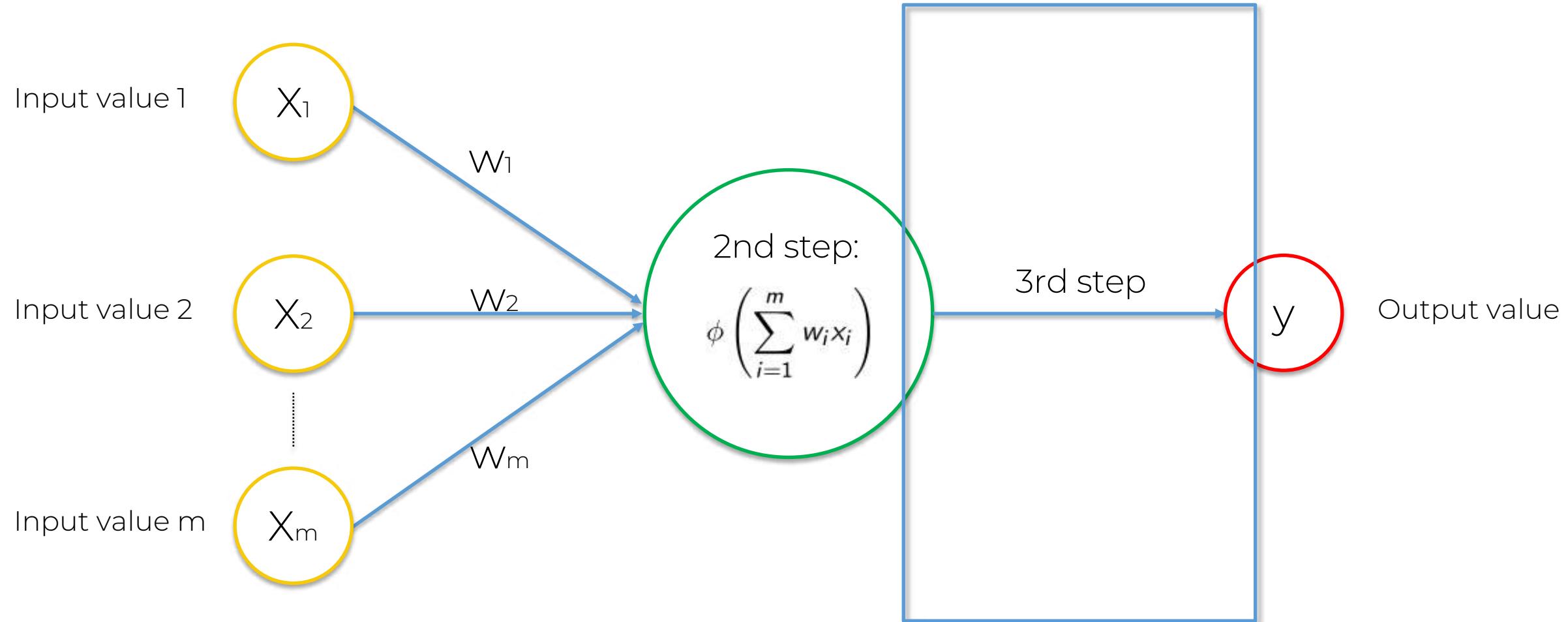
The Neuron



The Neuron

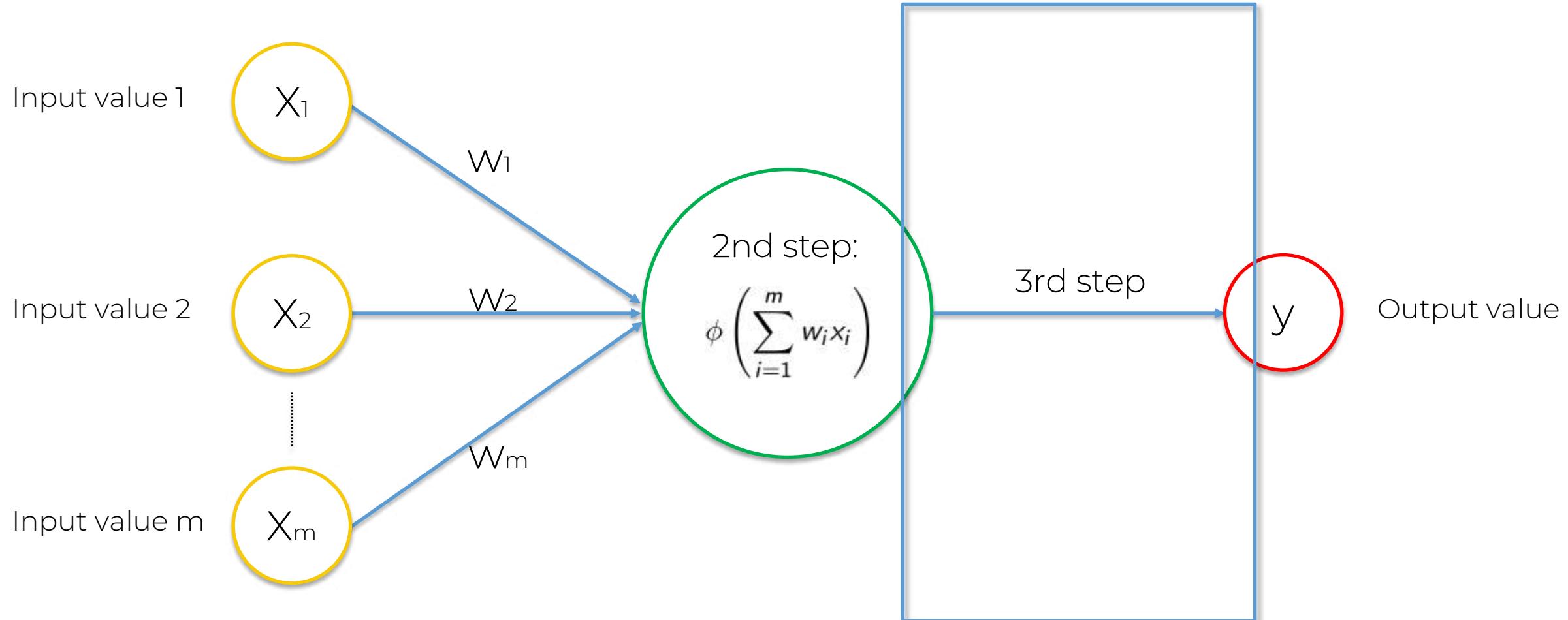


The Neuron

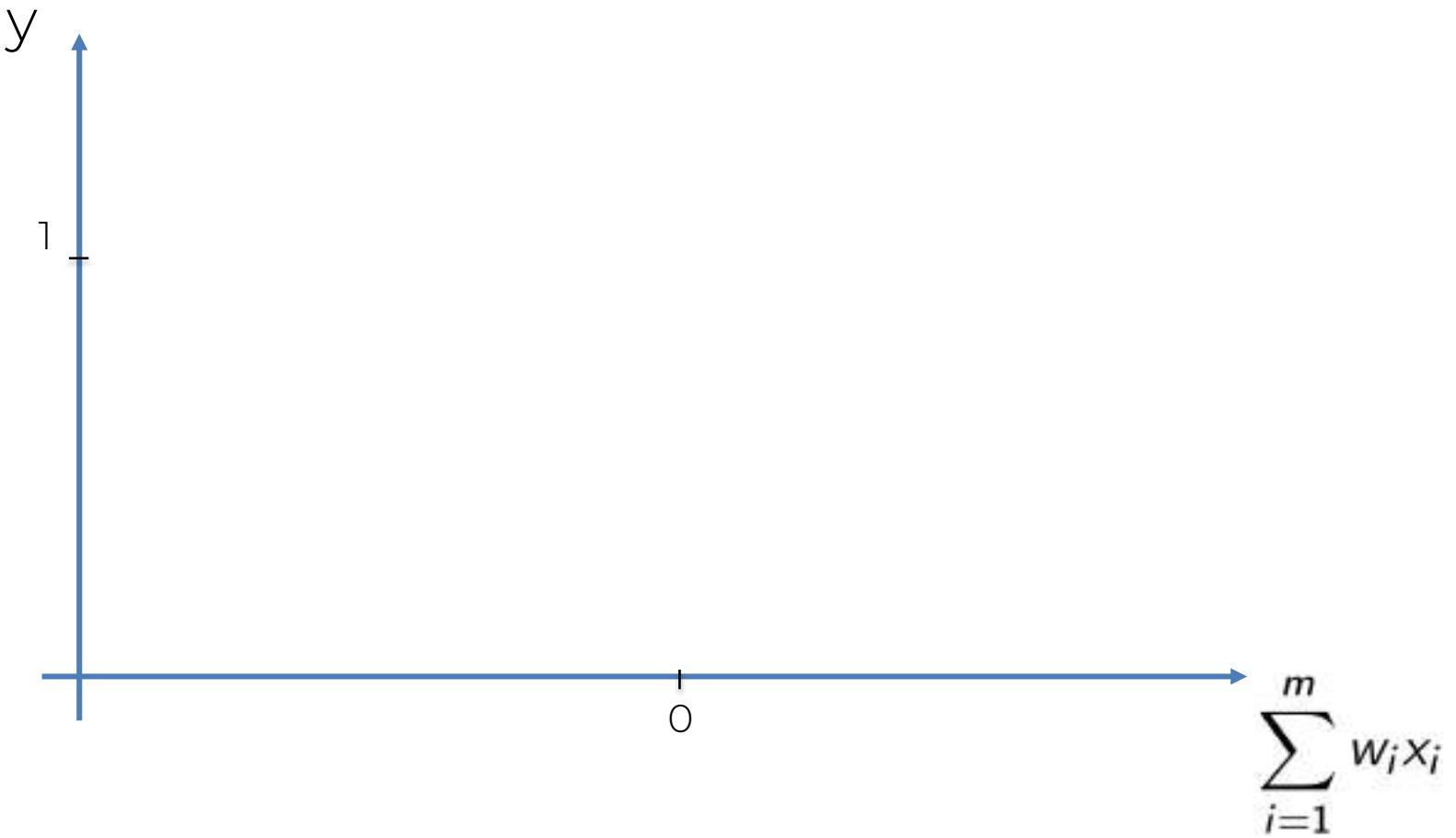


The Activation Function

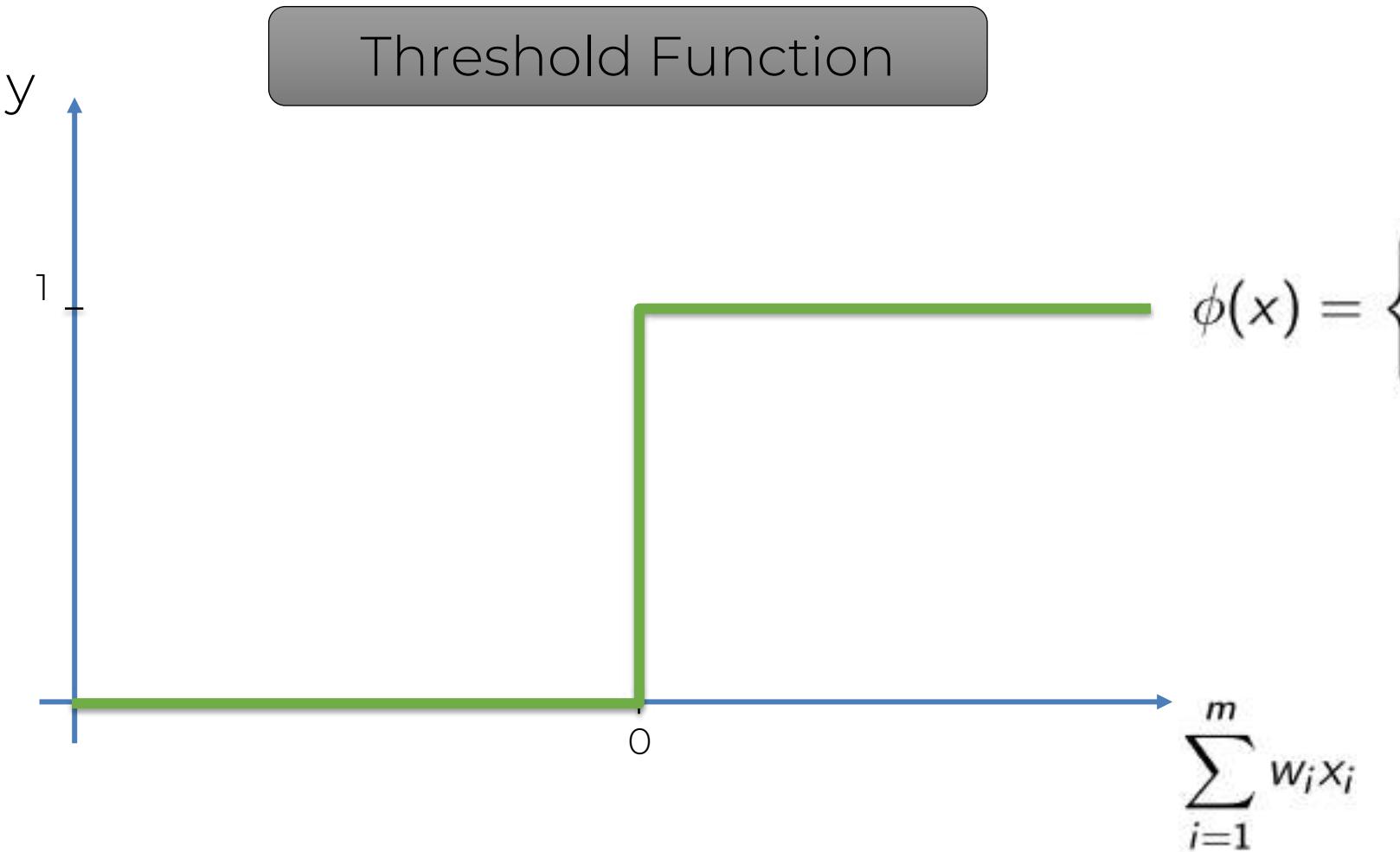
The Activation Function



The Activation Function

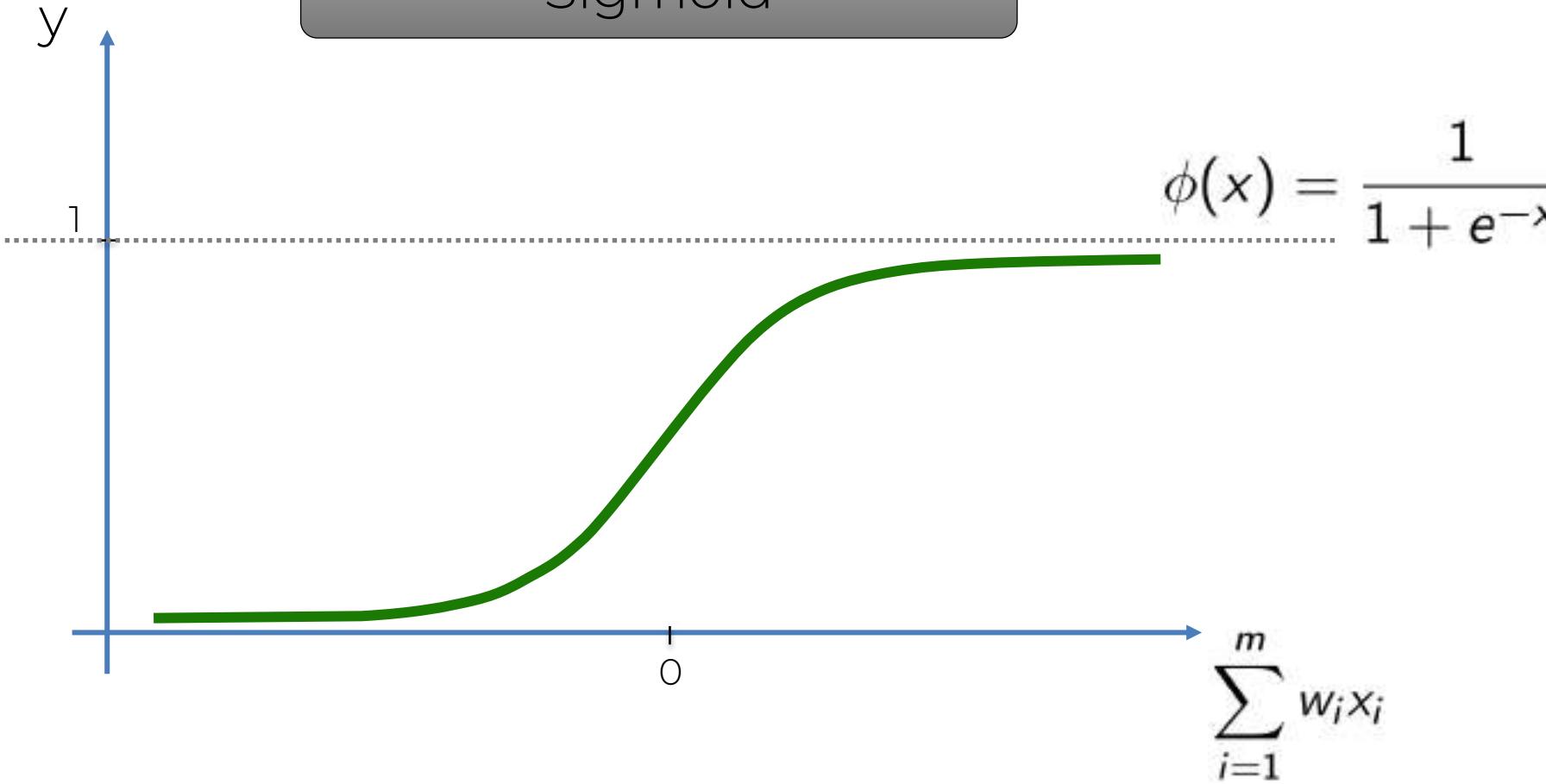


The Activation Function

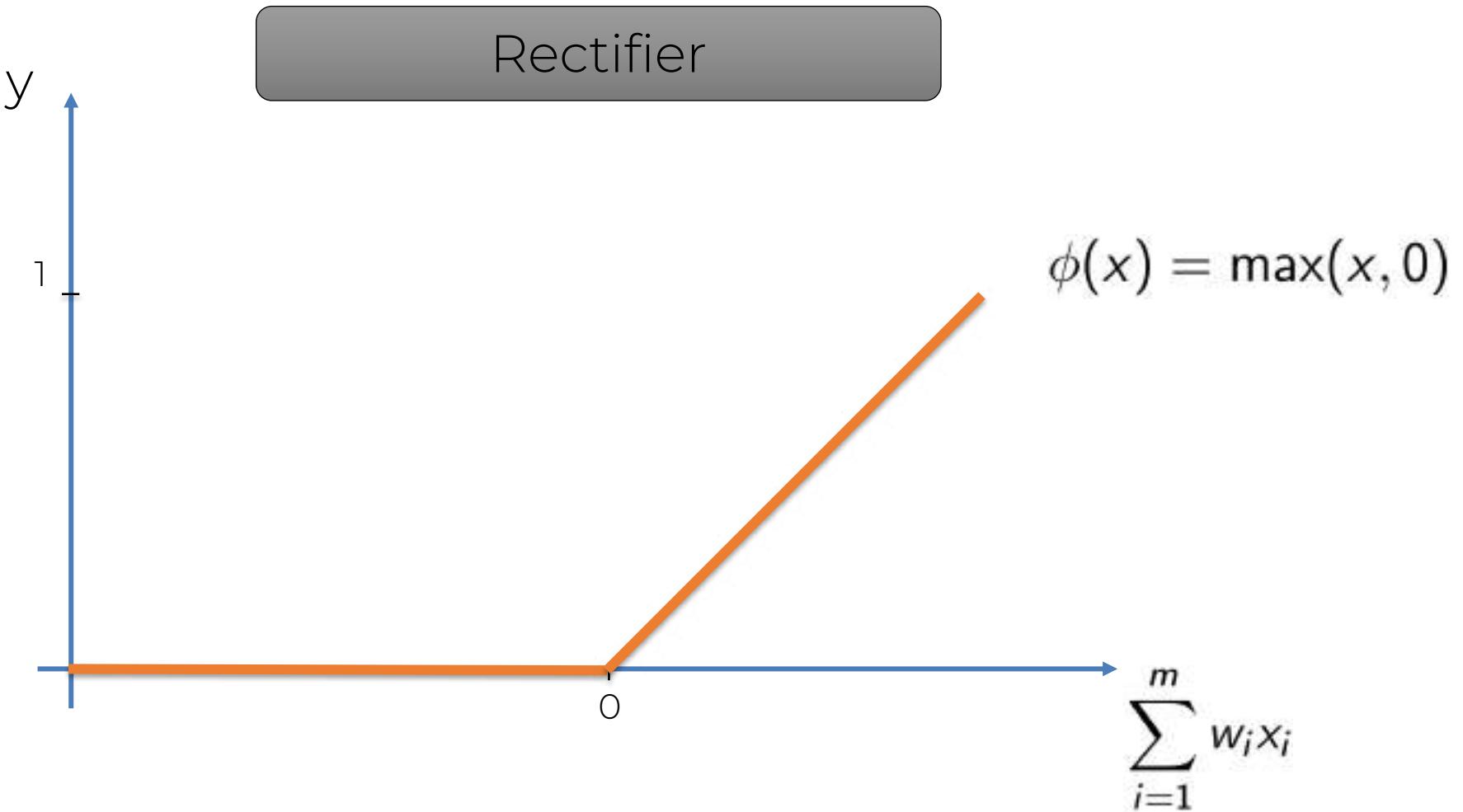


The Activation Function

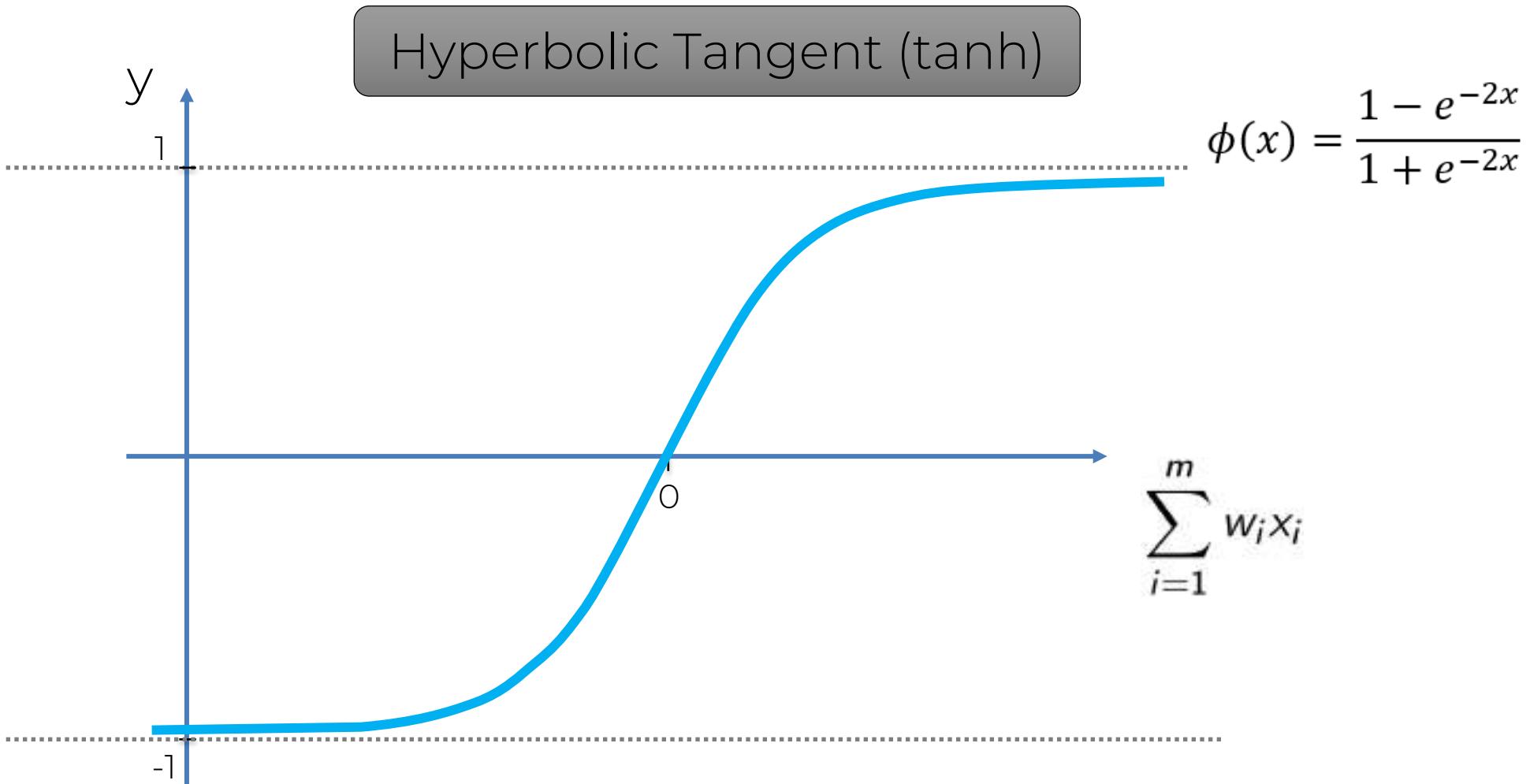
Sigmoid



The Activation Function



The Activation Function

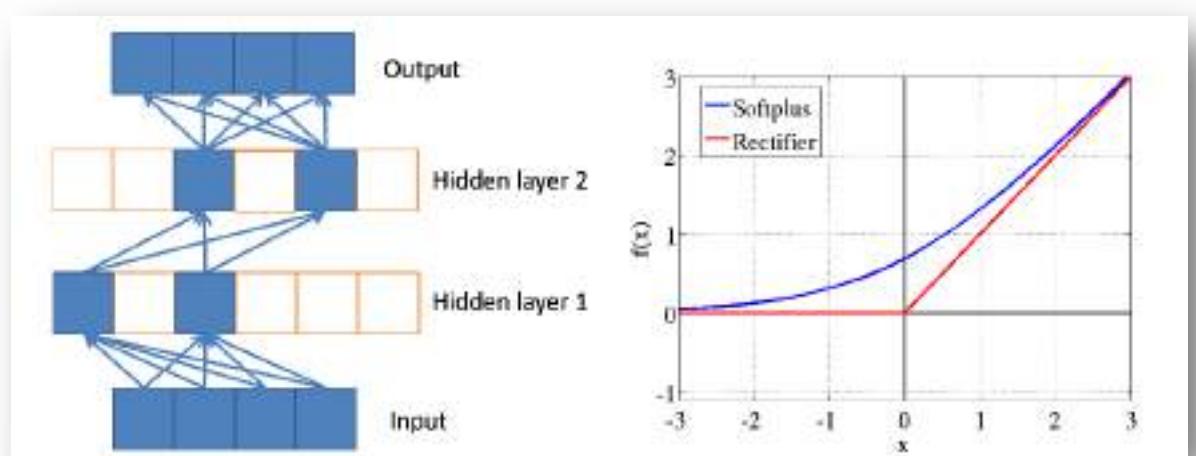


The Activation Function

Additional Reading:

*Deep sparse rectifier
neural networks*

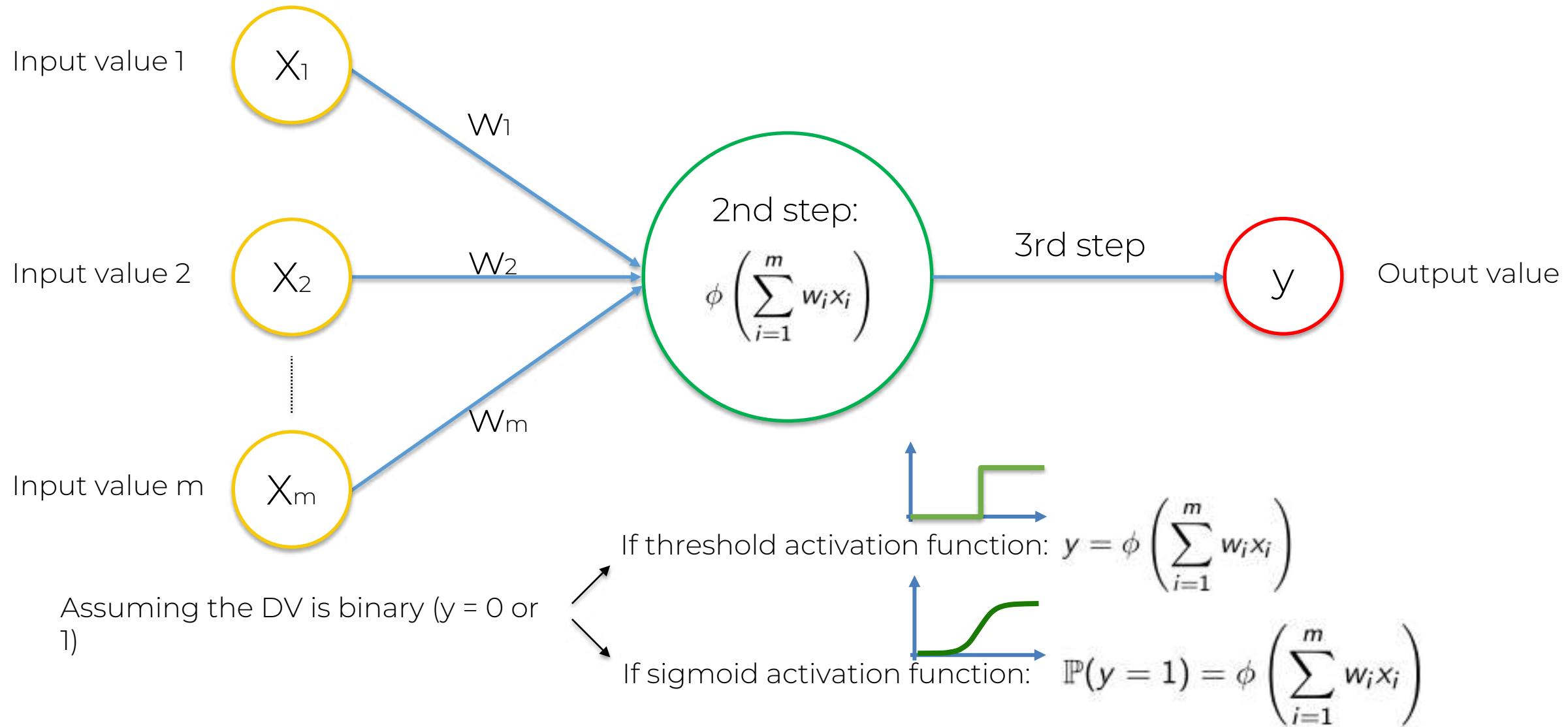
By Xavier Glorot et al. (2011)



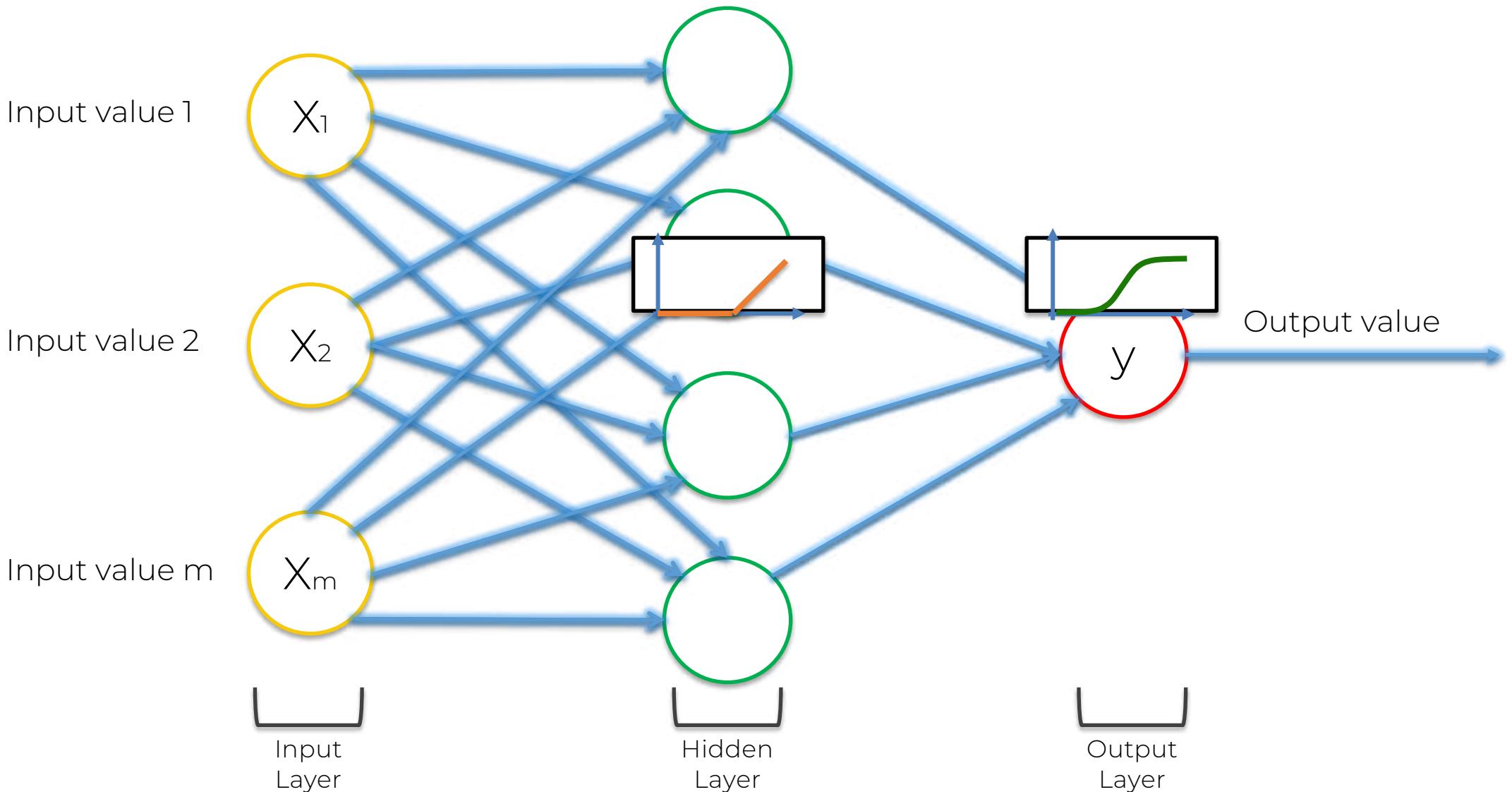
Link:

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

The Activation Function



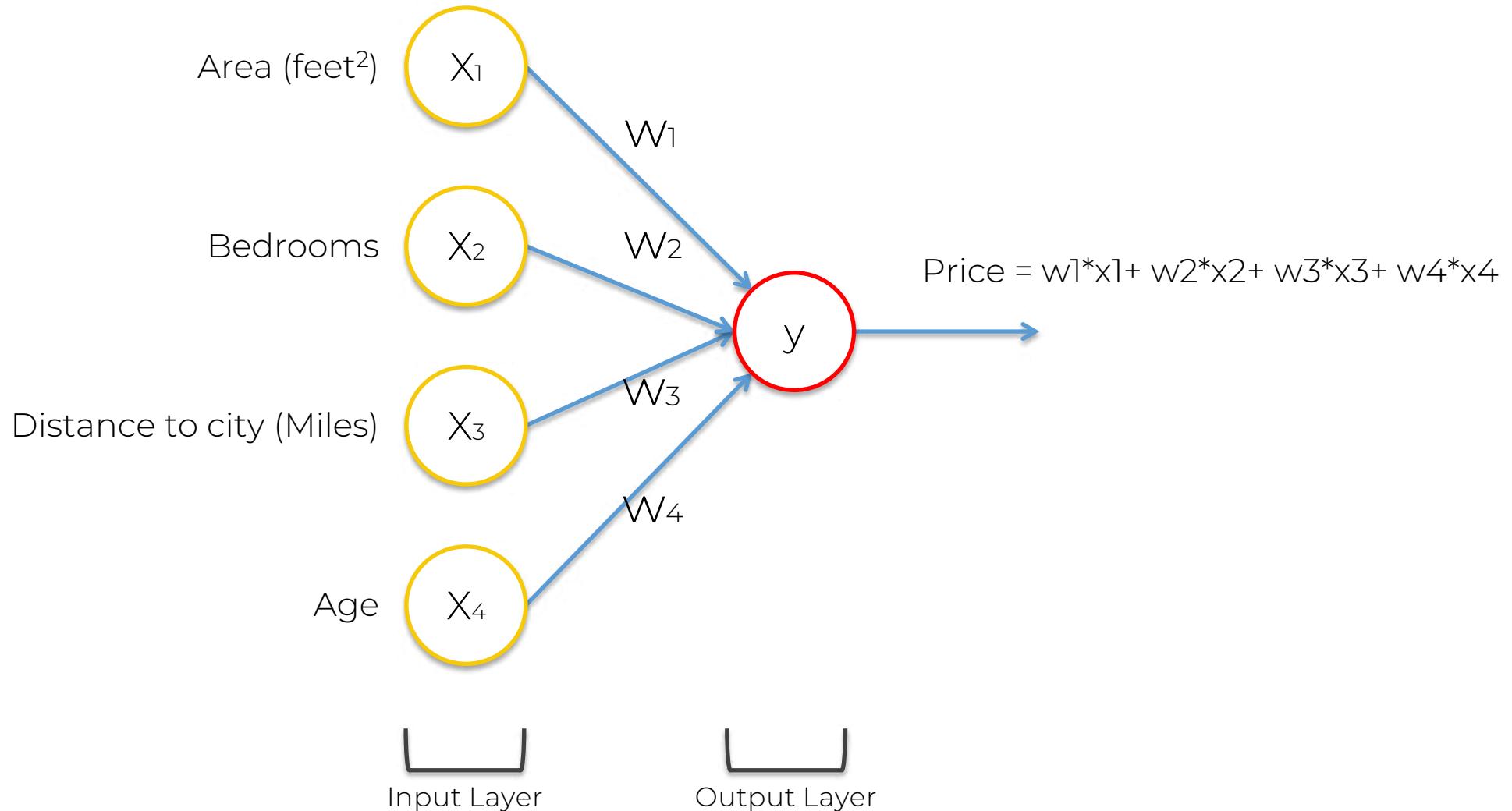
The Activation Function



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?



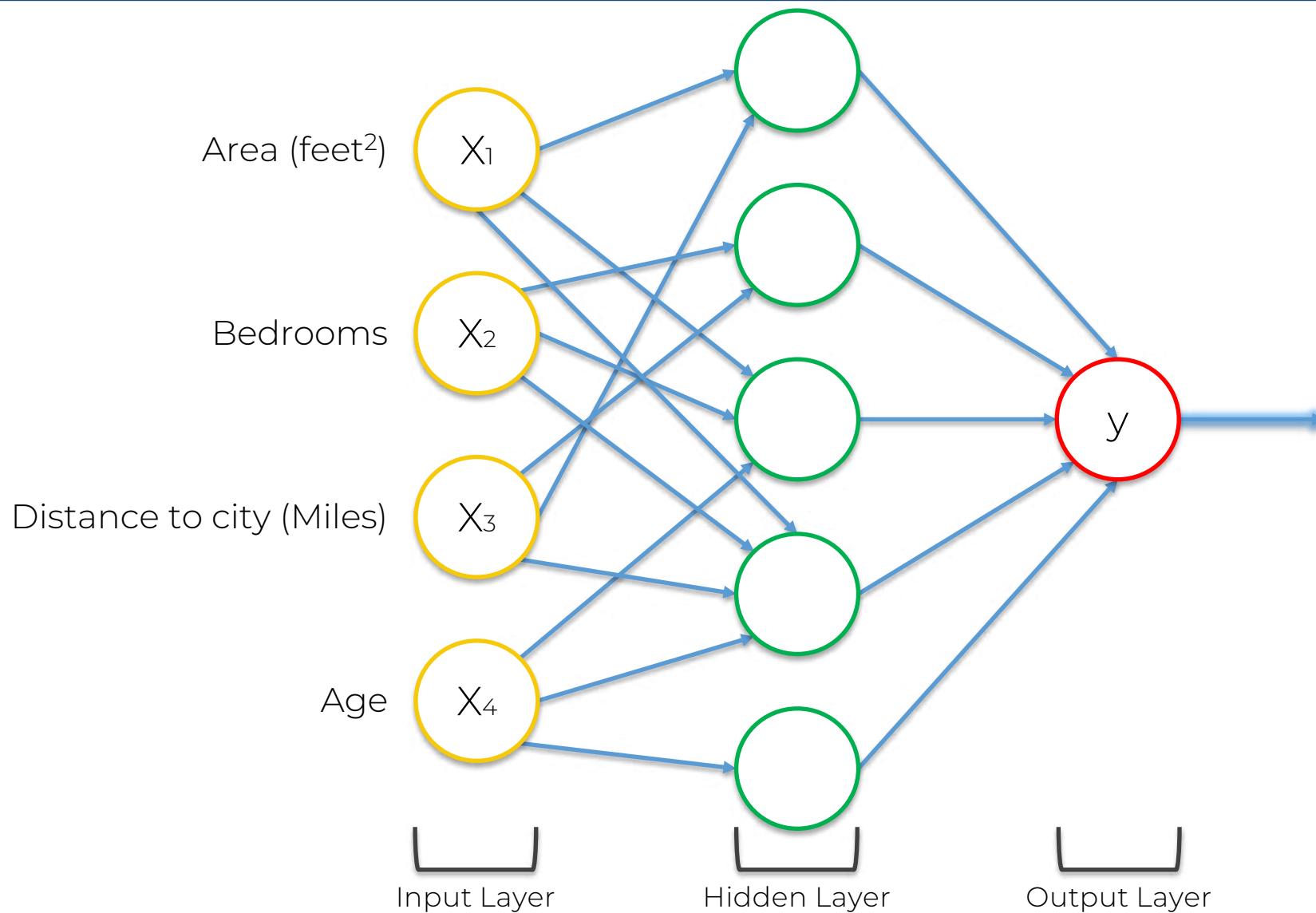
How Do Neural Networks Work?



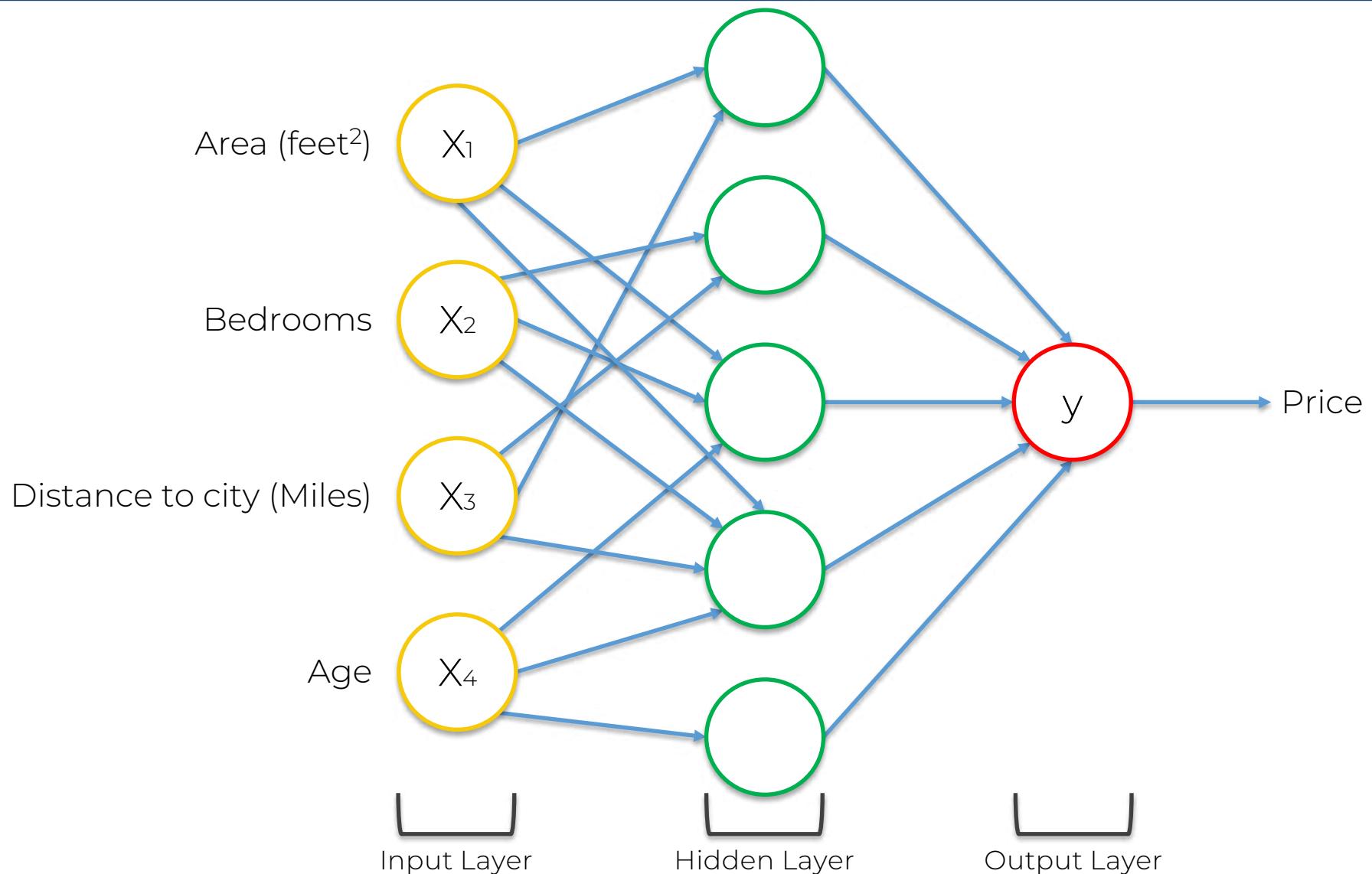
How Do Neural Networks Work?



How Do Neural Networks Work?



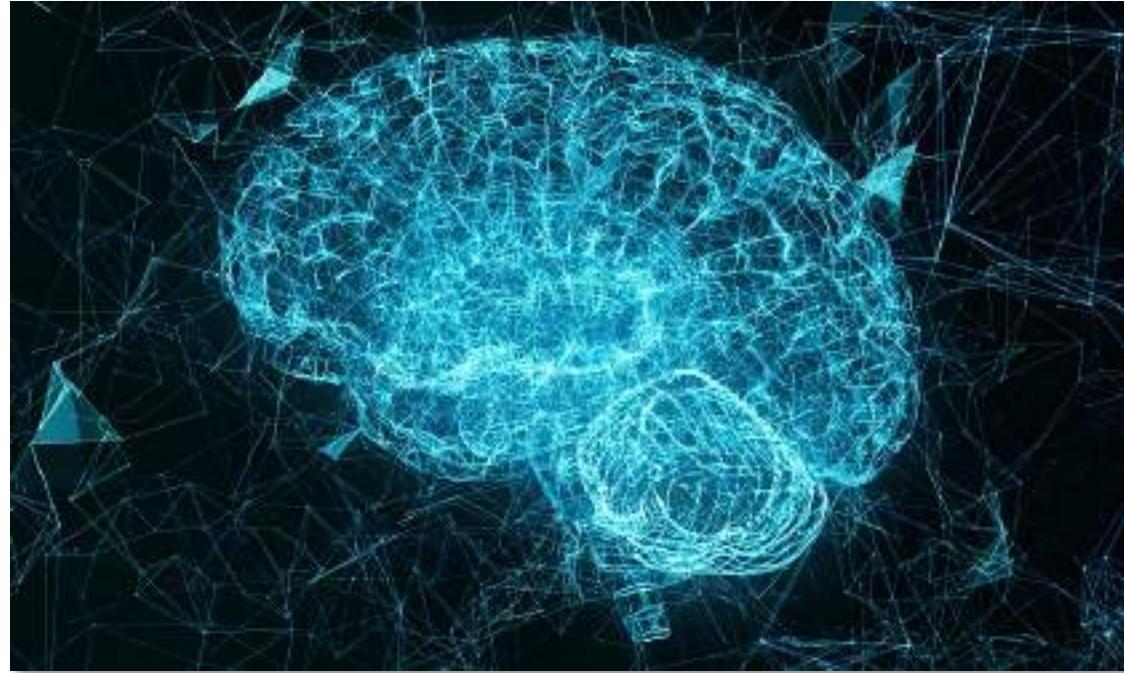
How Do Neural Networks Work?



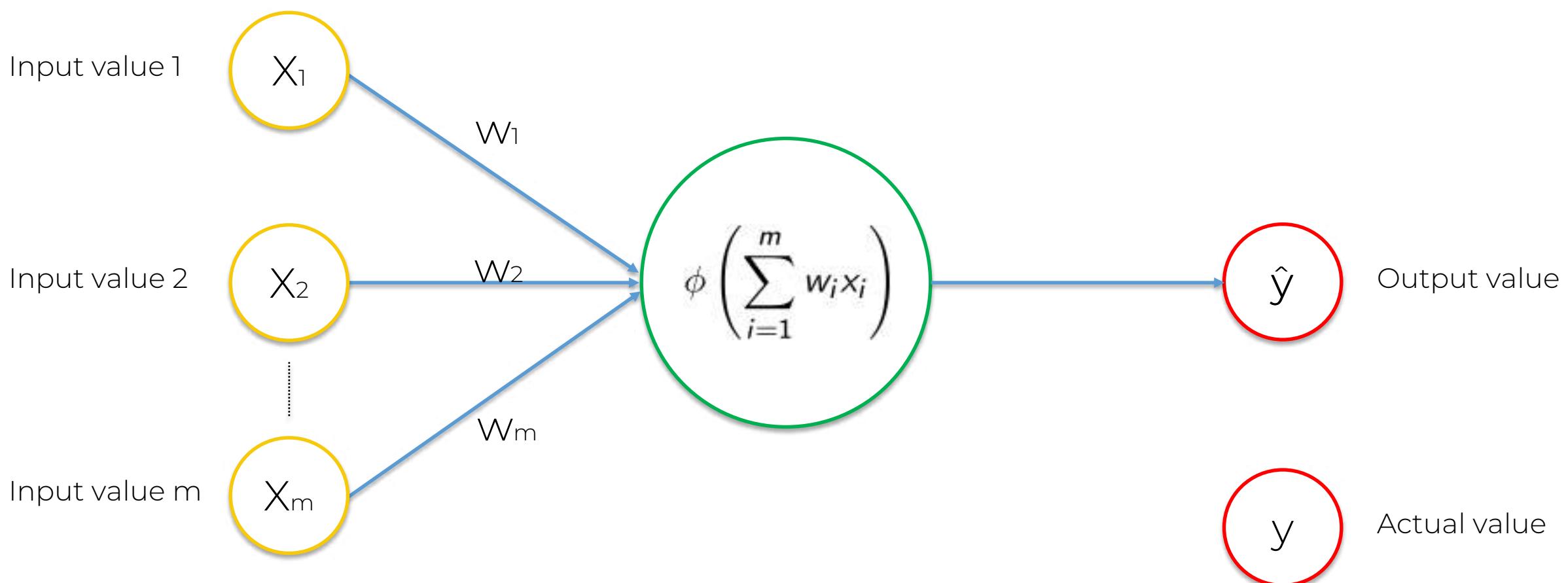
How do NNs Learn?

How do Neural Networks learn?

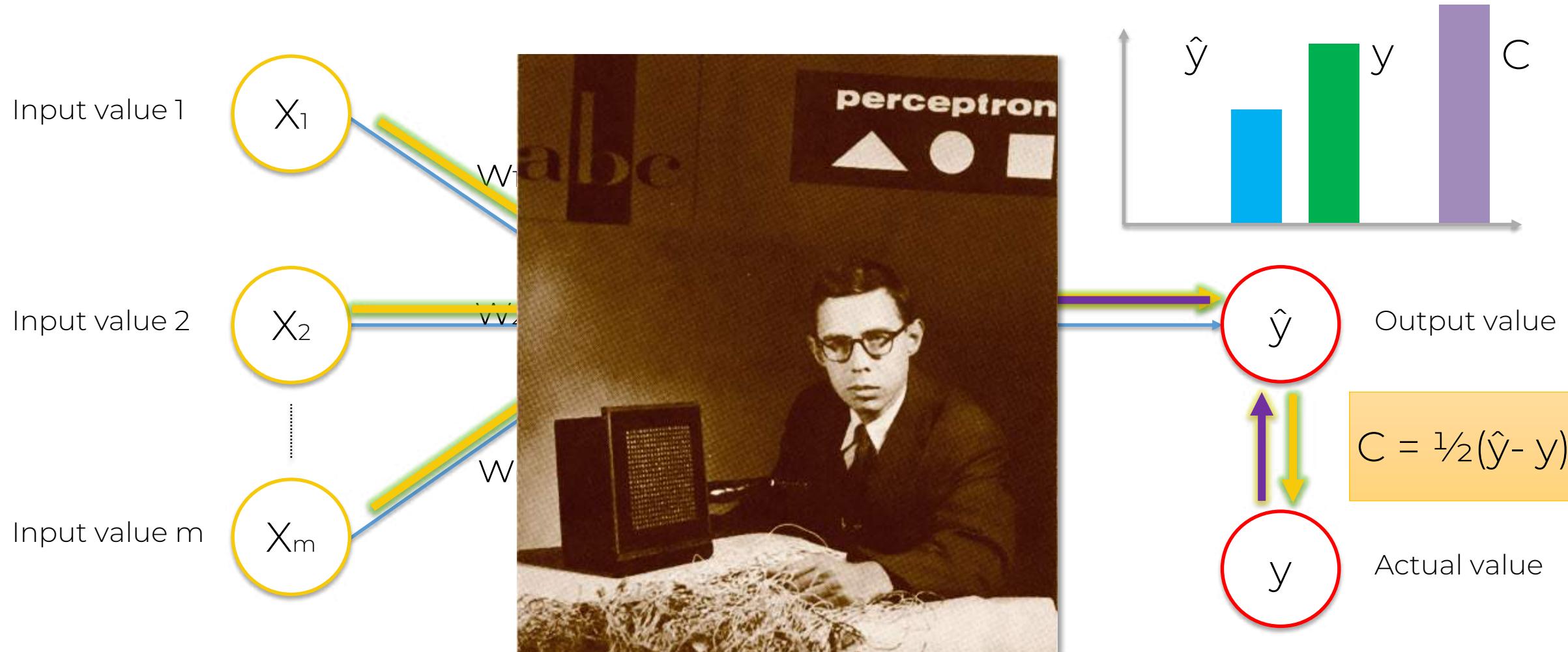
```
function use_array(a, b) {  
    if (a < -1 || b < -1) return [];  
    let c = new Array(b - a + 1);  
    for (let i = a; i <= b; i++) c[i - a] = a[i];  
    return c;  
}  
  
function use_keyword(a, b) {  
    let c = new Array(b - a + 1);  
    for (let i = a; i <= b; i++) c[i - a] = a[i];  
    return c;  
}  
  
function use_string(a, b) {  
    let c = new Array(b - a + 1);  
    for (let i = a; i <= b; i++) c[i - a] = a[i];  
    return c;  
}  
  
function use_number(a, b) {  
    let c = new Array(b - a + 1);  
    for (let i = a; i <= b; i++) c[i - a] = a[i];  
    return c;  
}
```



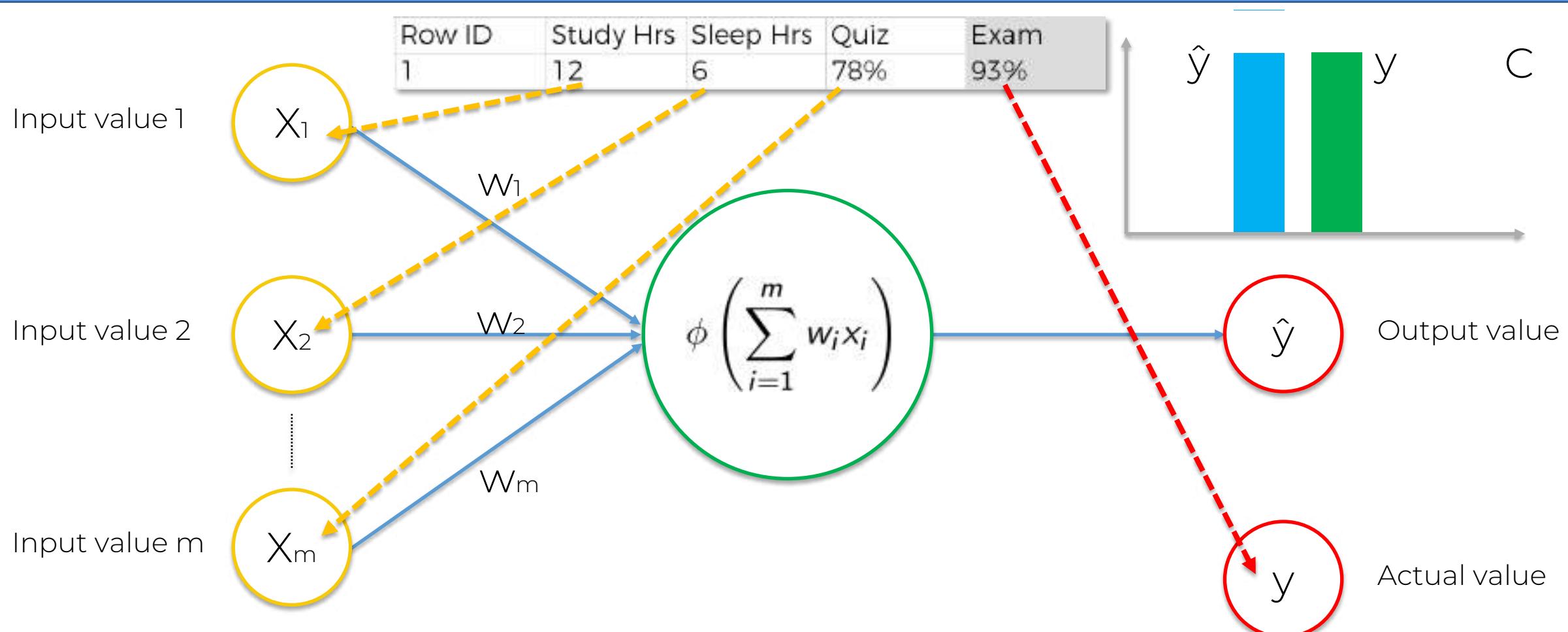
How do Neural Networks learn?



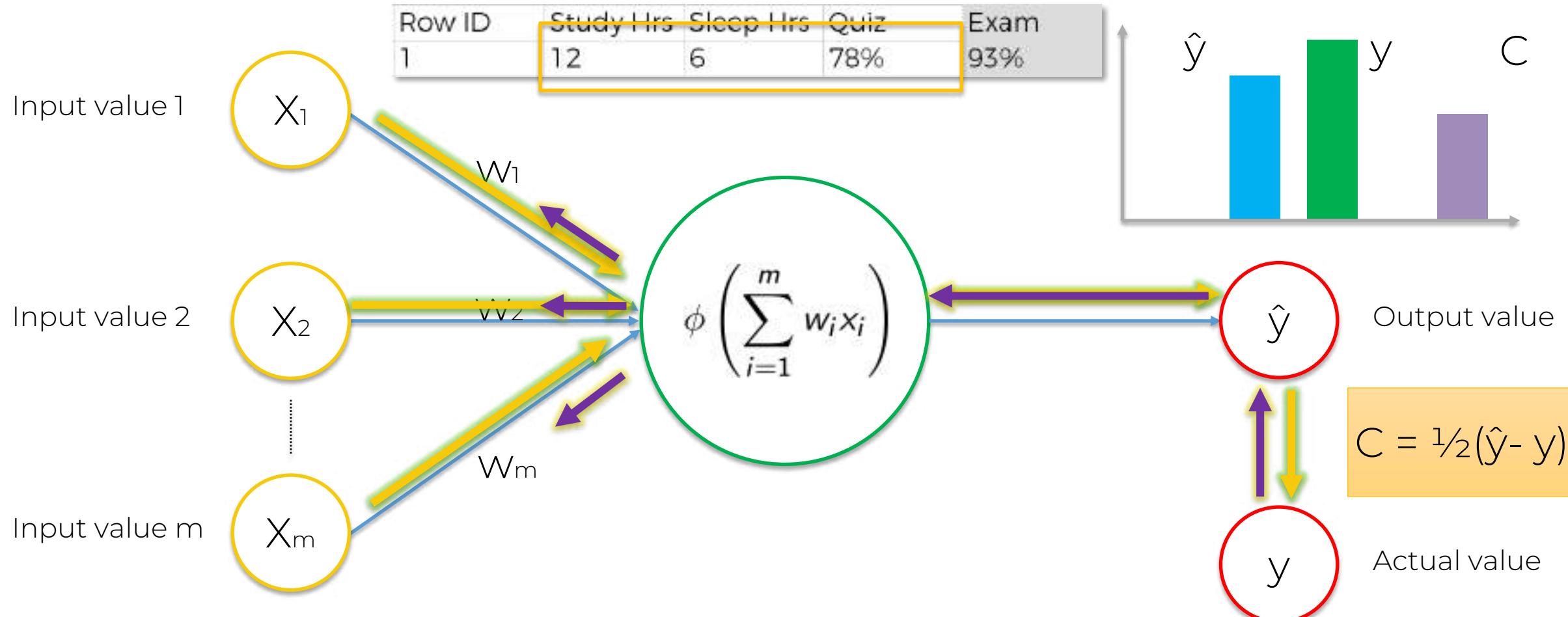
How do Neural Networks learn?



How do Neural Networks learn?

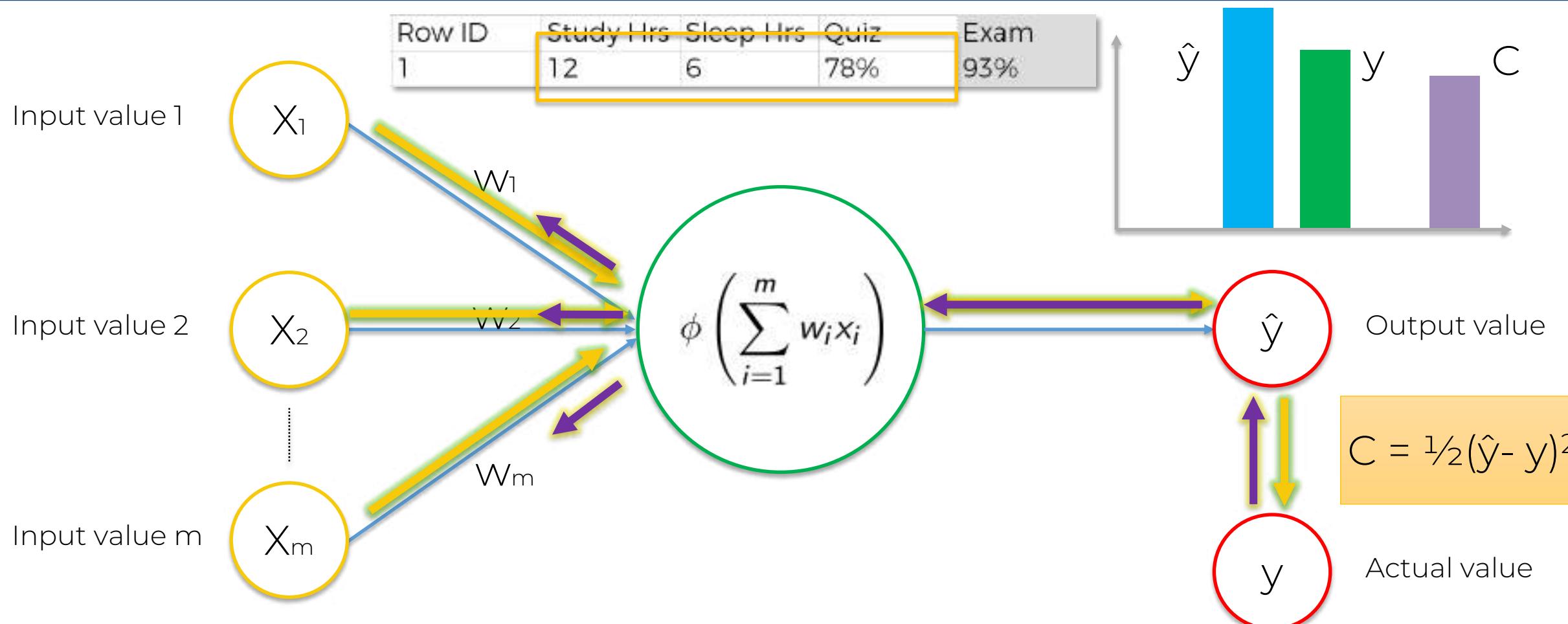


How do Neural Networks learn?

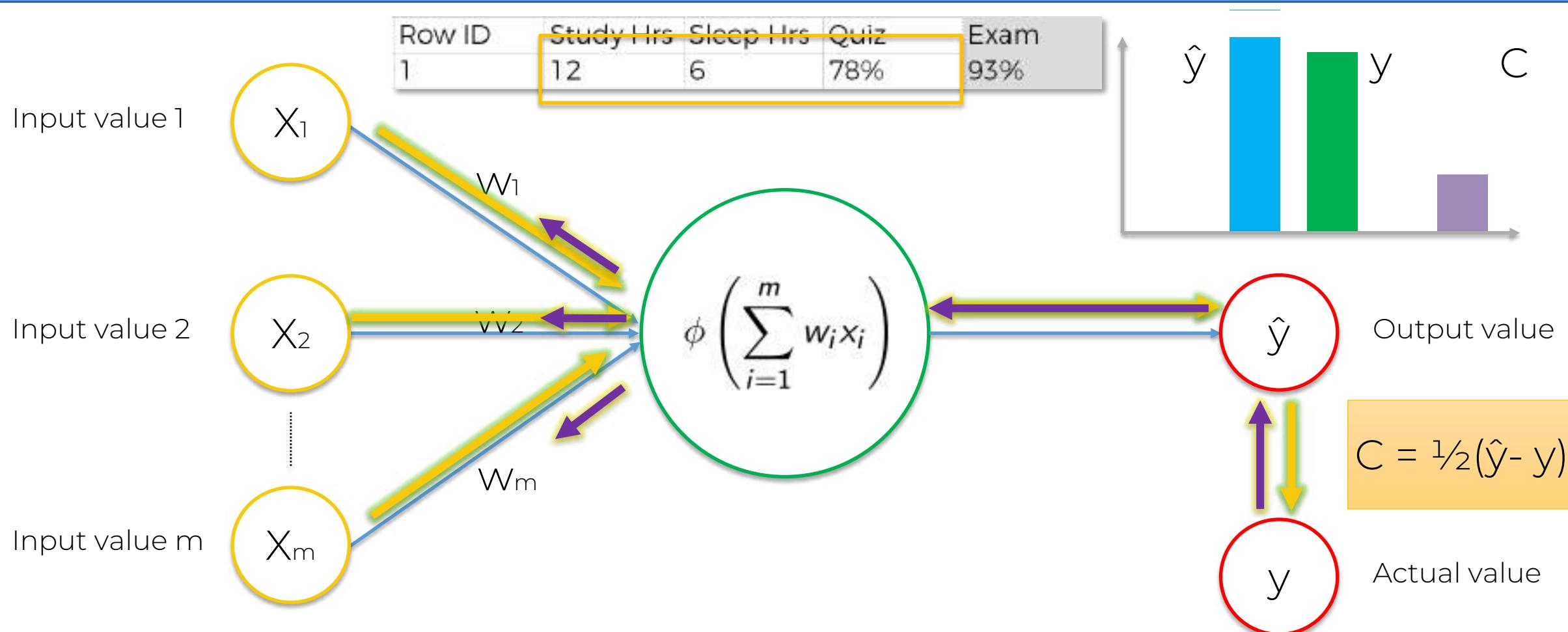


How do Neural Networks learn?

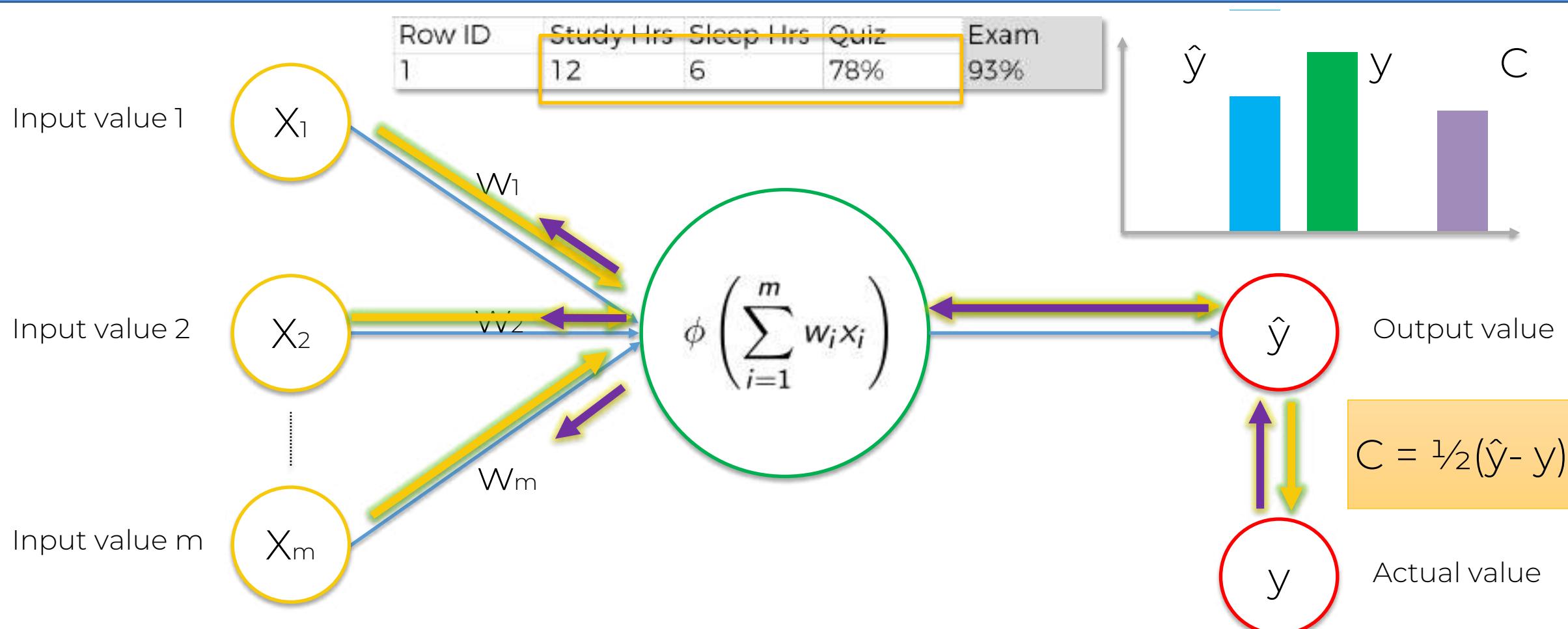
NOT FOR DISTRIBUTION @ SUPERDATASCIENCE www.Superdatascience.com



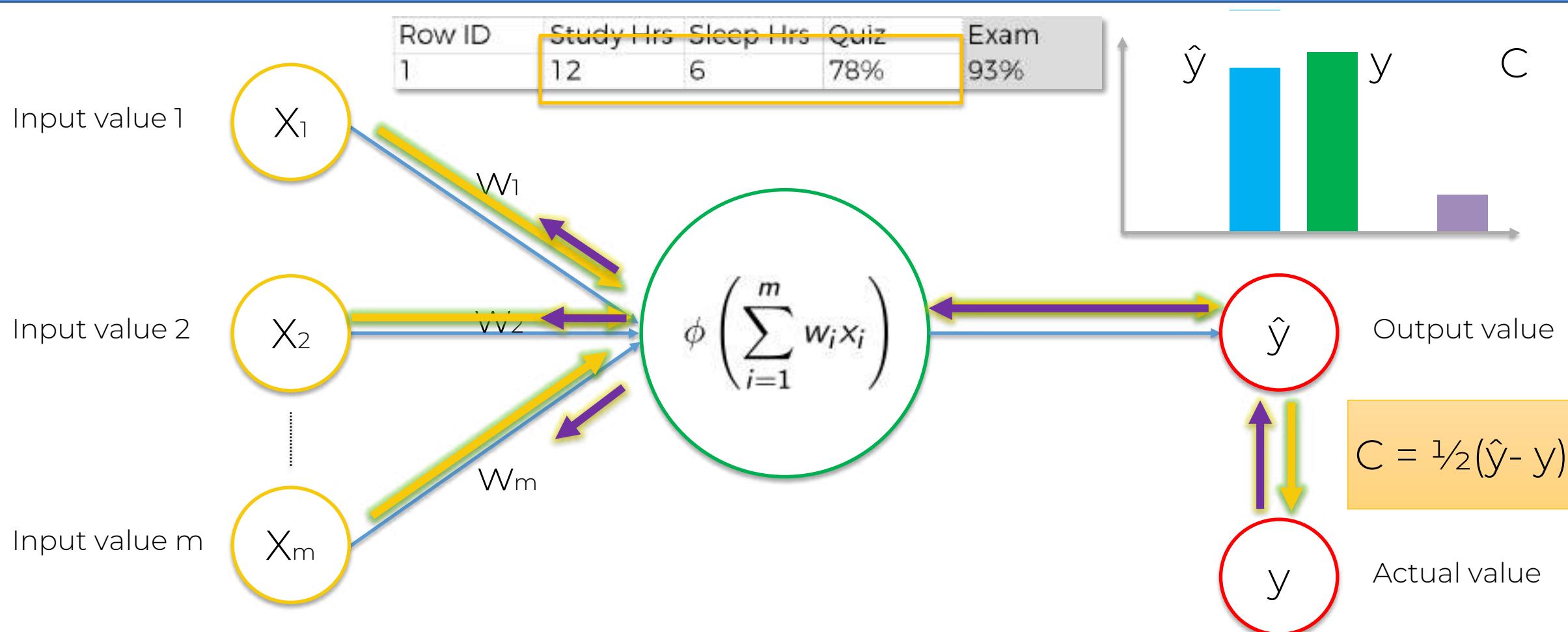
How do Neural Networks learn?



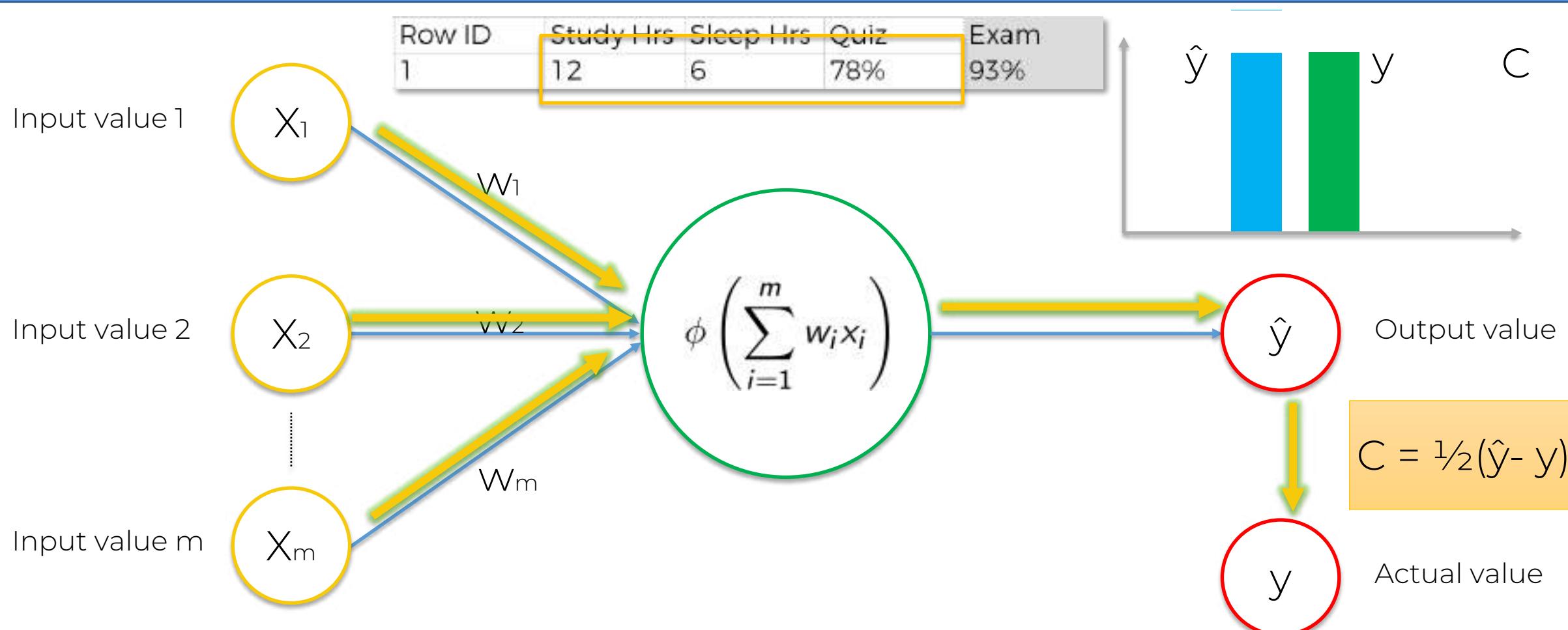
How do Neural Networks learn?



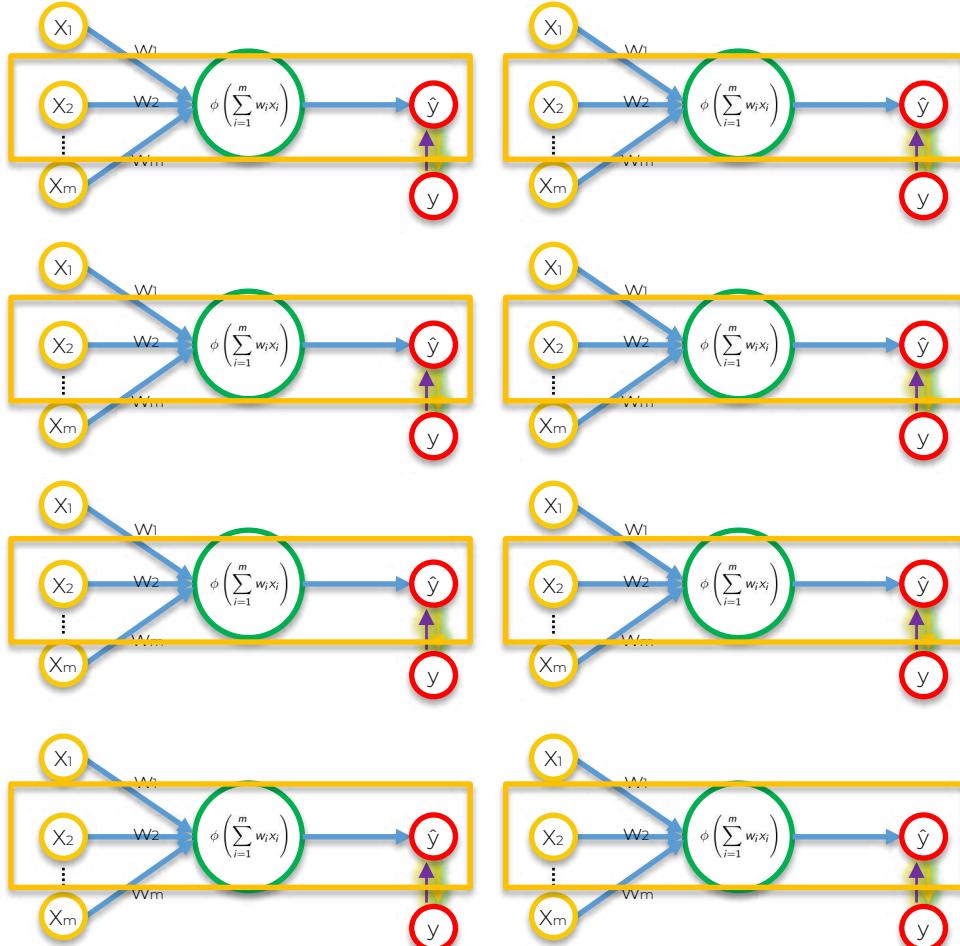
How do Neural Networks learn?



How do Neural Networks learn?



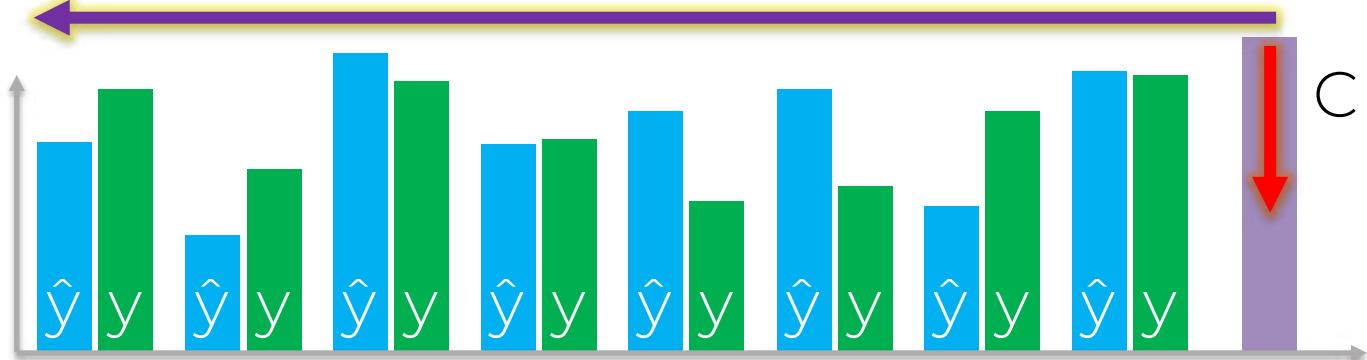
How do Neural Networks learn?



Row ID	study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	51	3	57%	75%
5	9	10	58%	51%
6	4	0	70%	60%
7	63	5	92%	89%
8	57	8	91%	97%

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

Adjust w_1, w_2, w_3



How do Neural Networks learn?

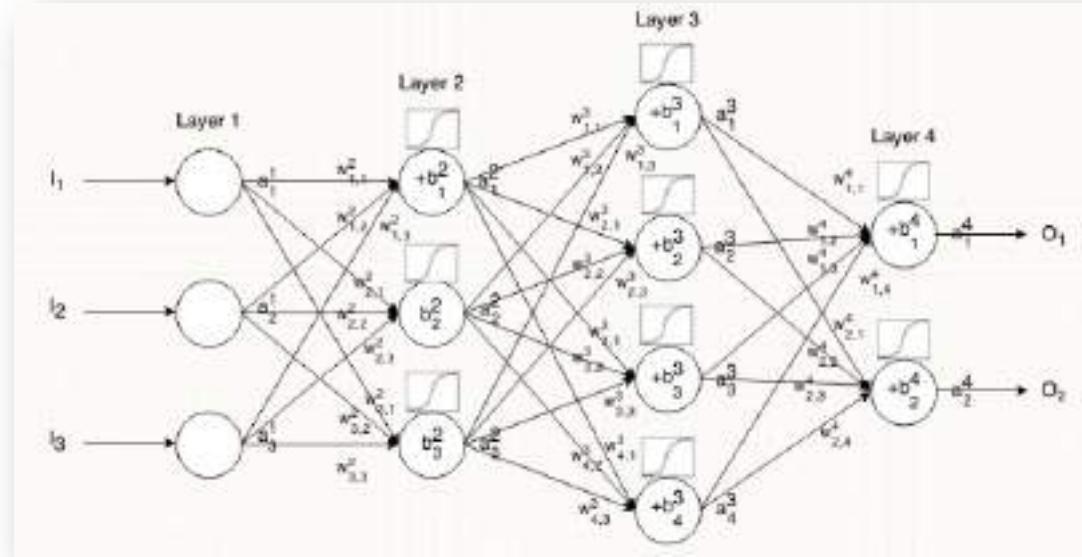
Additional Reading:

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

CrossValidated (2015)

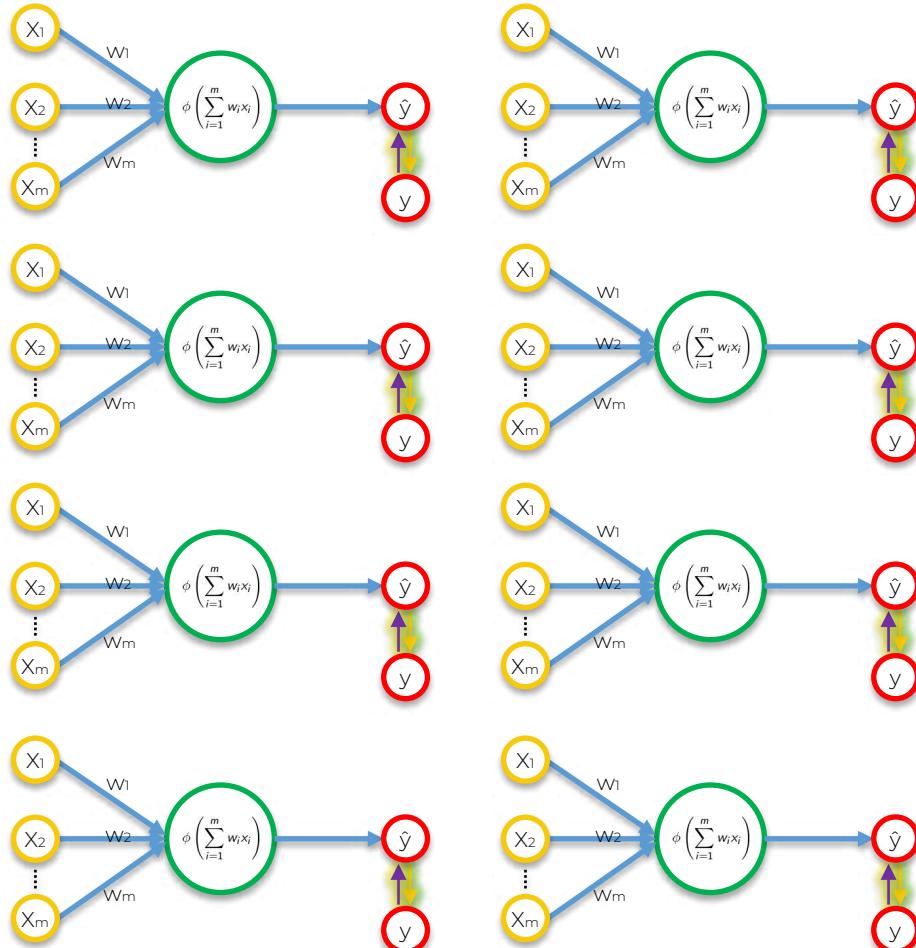
Link:

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



Gradient Descent

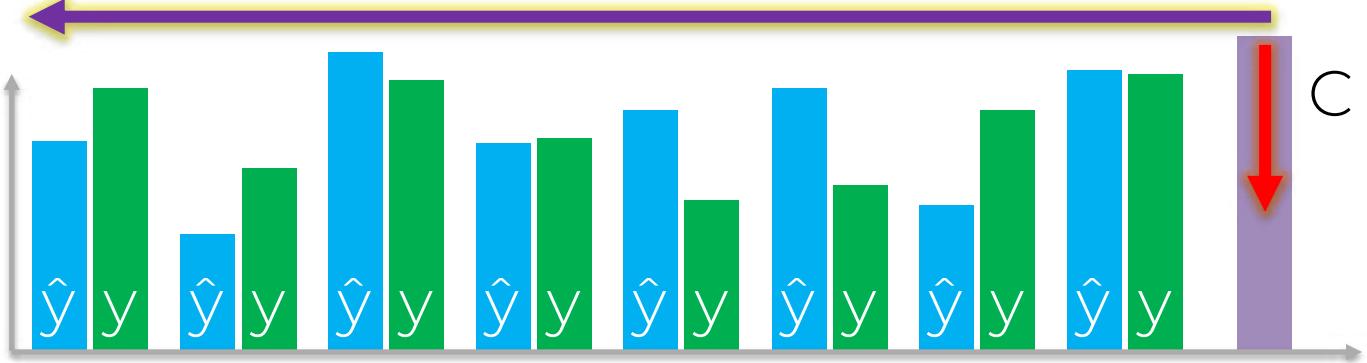
Gradient Descent



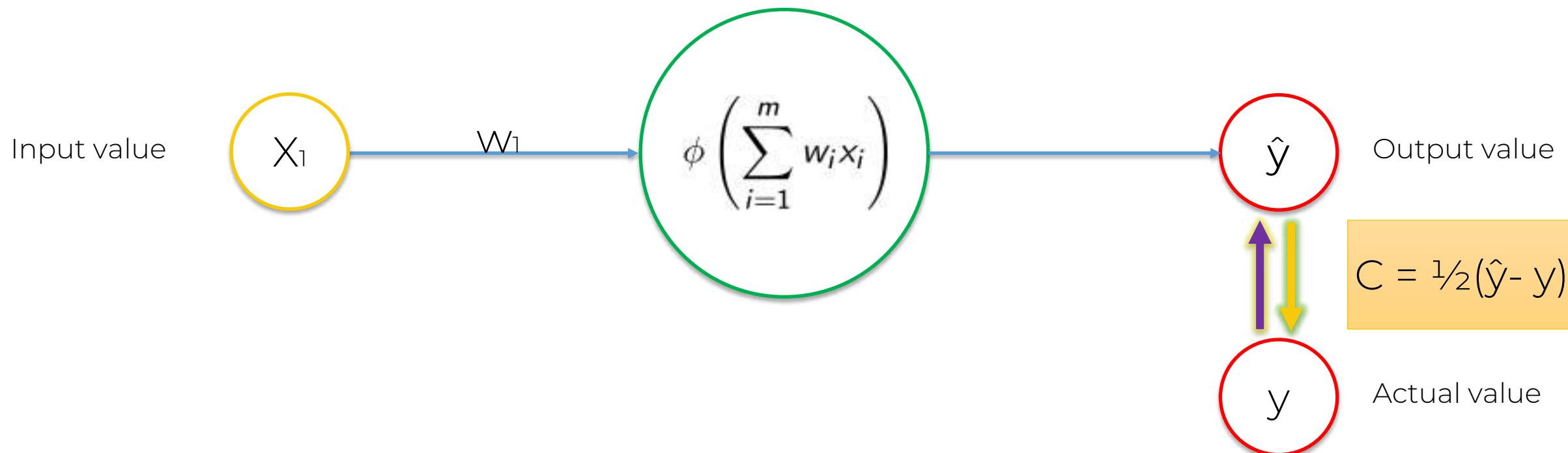
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

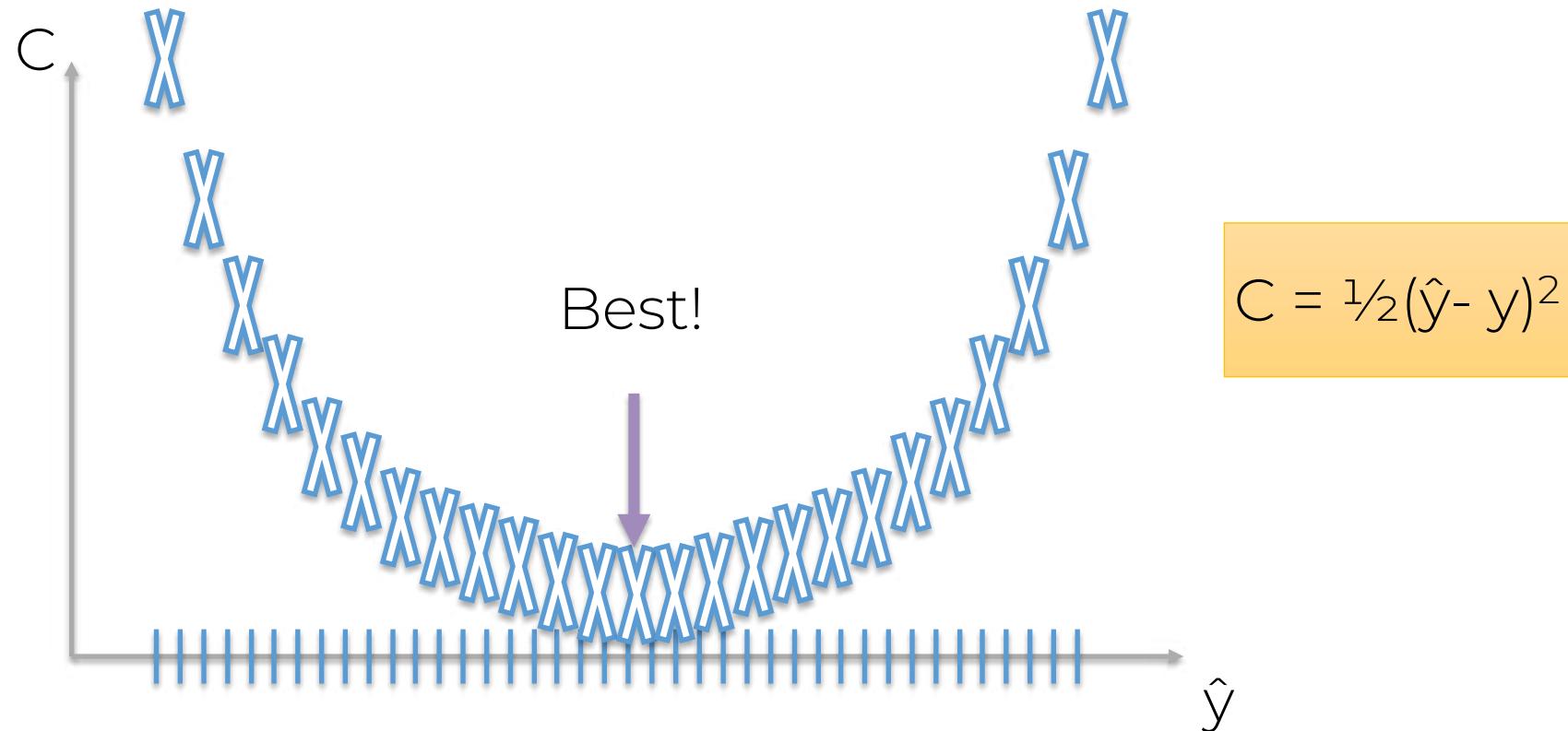
Adjust w_1, w_2, w_3



Gradient Descent



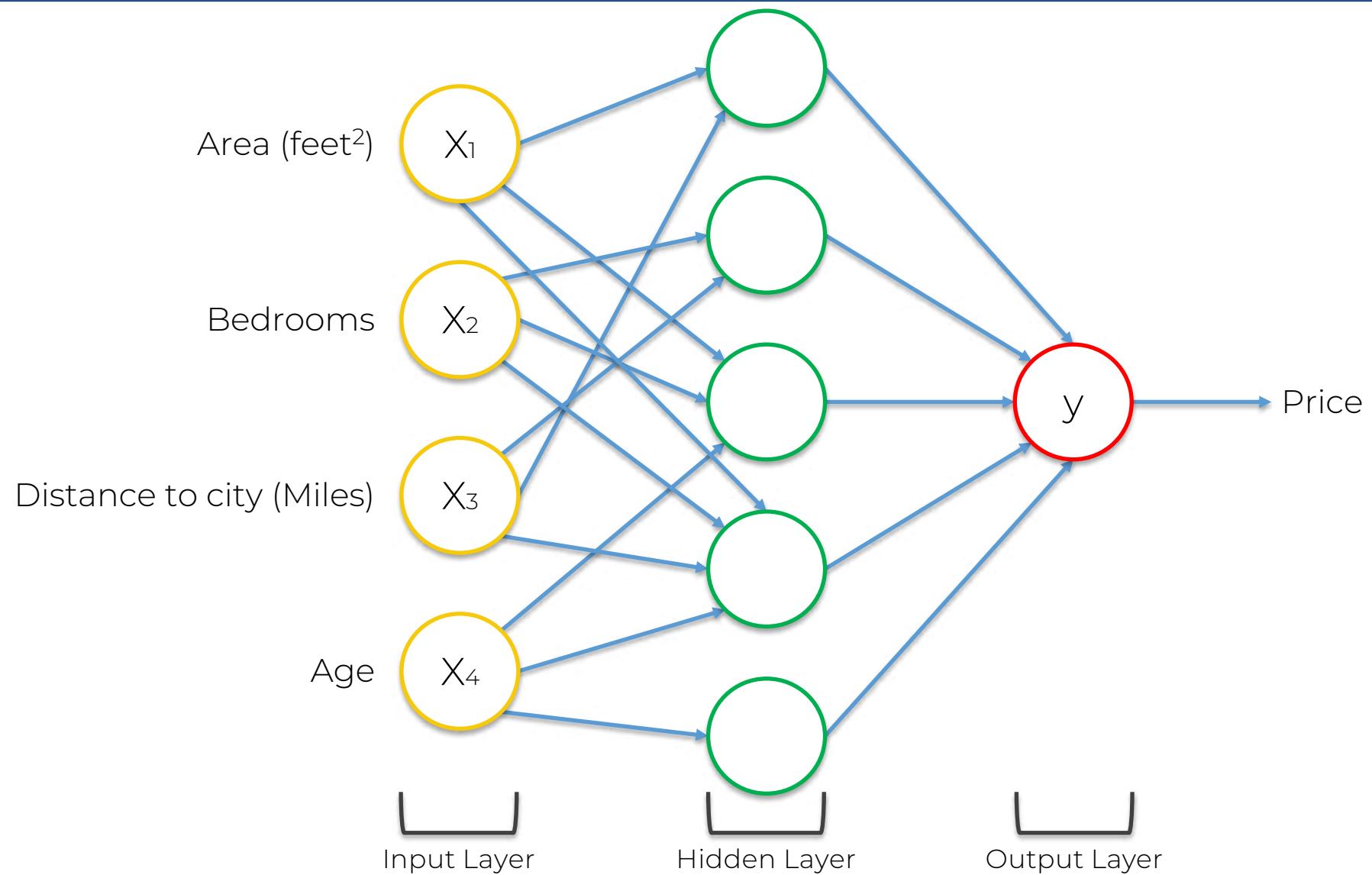
Gradient Descent



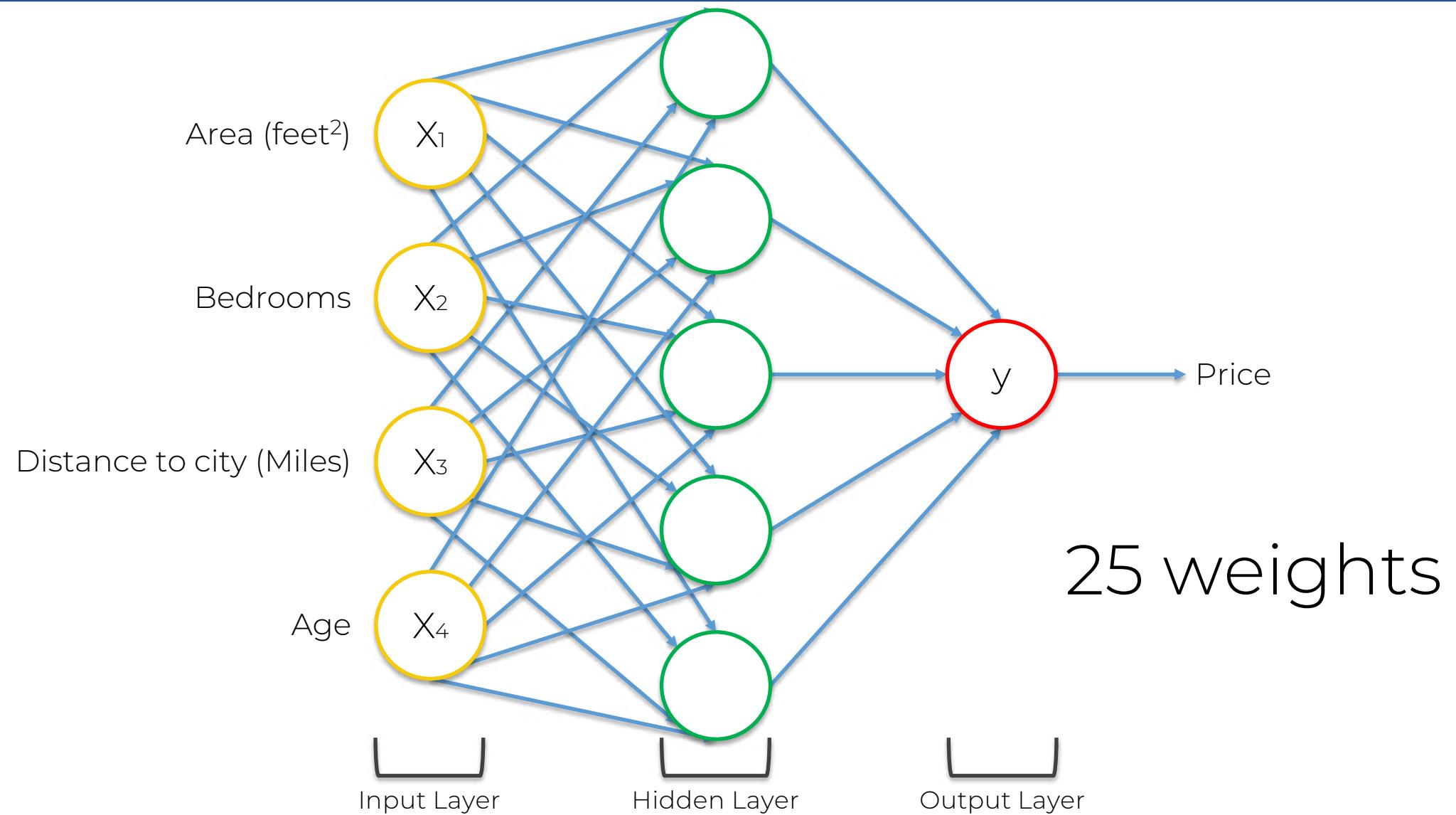
Gradient Descent

Curse of Dimensionality

Gradient Descent



Gradient Descent



Gradient Descent

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

Sunway TaihuLight: World's fastest Super Computer

93 PFLOPS

93×10^{15}

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

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

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



Gradient Descent

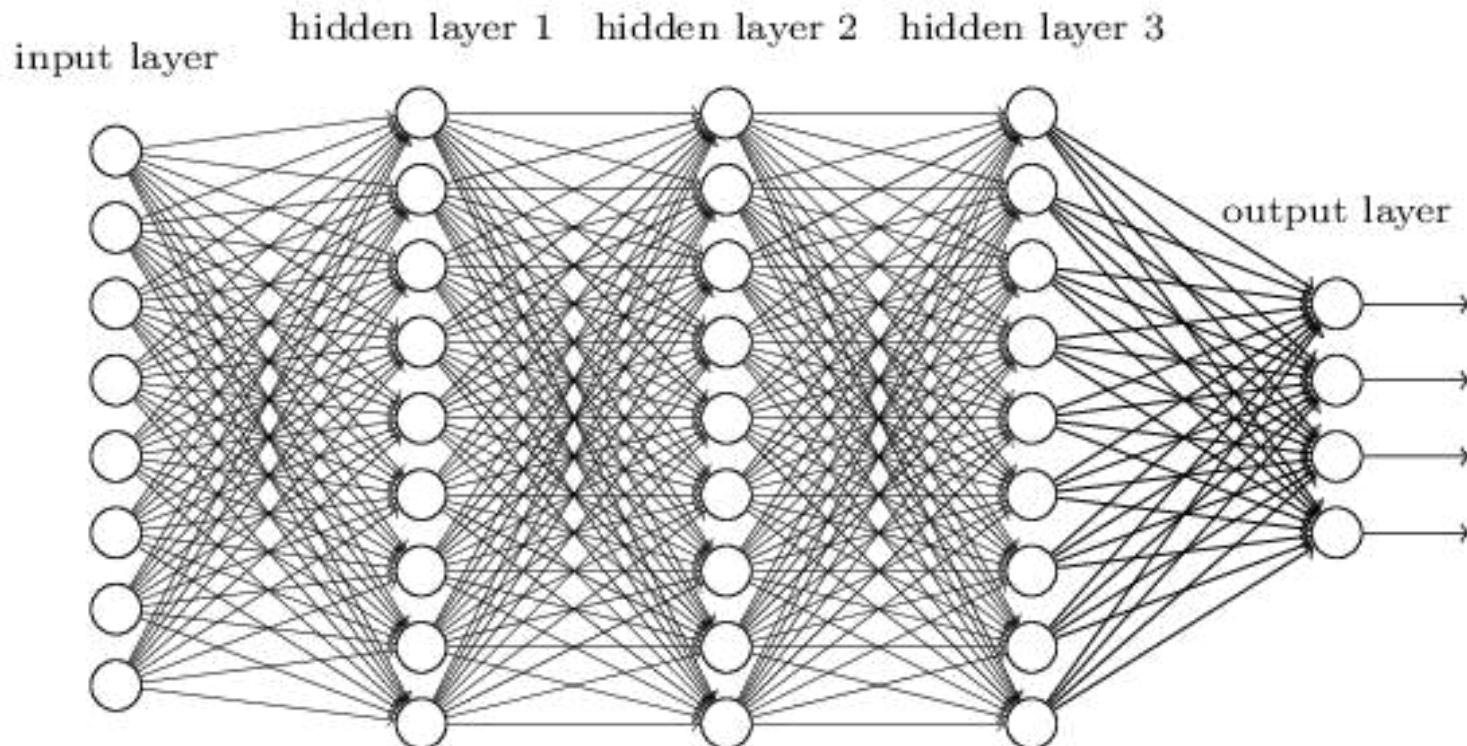
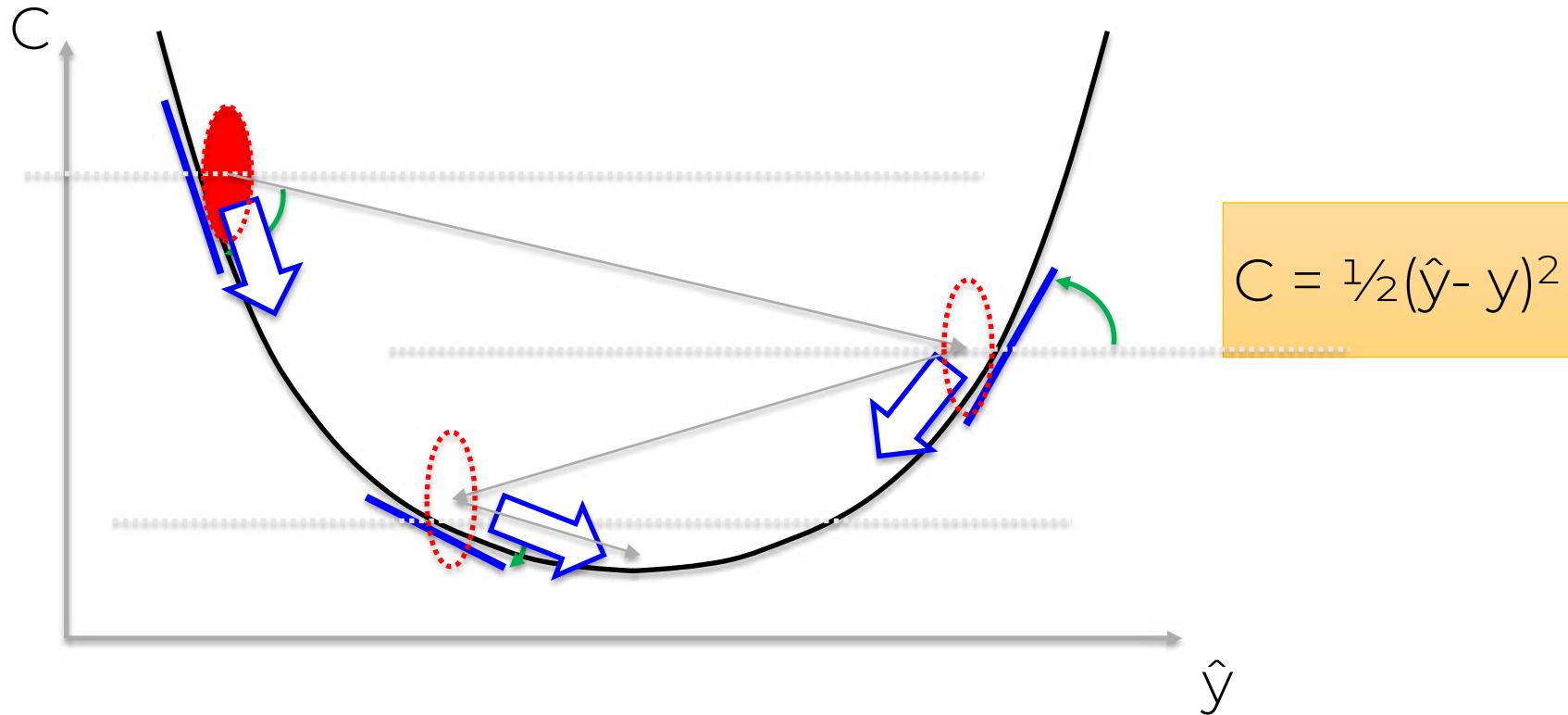


Image Source: neuralnetworksanddeeplearning.com

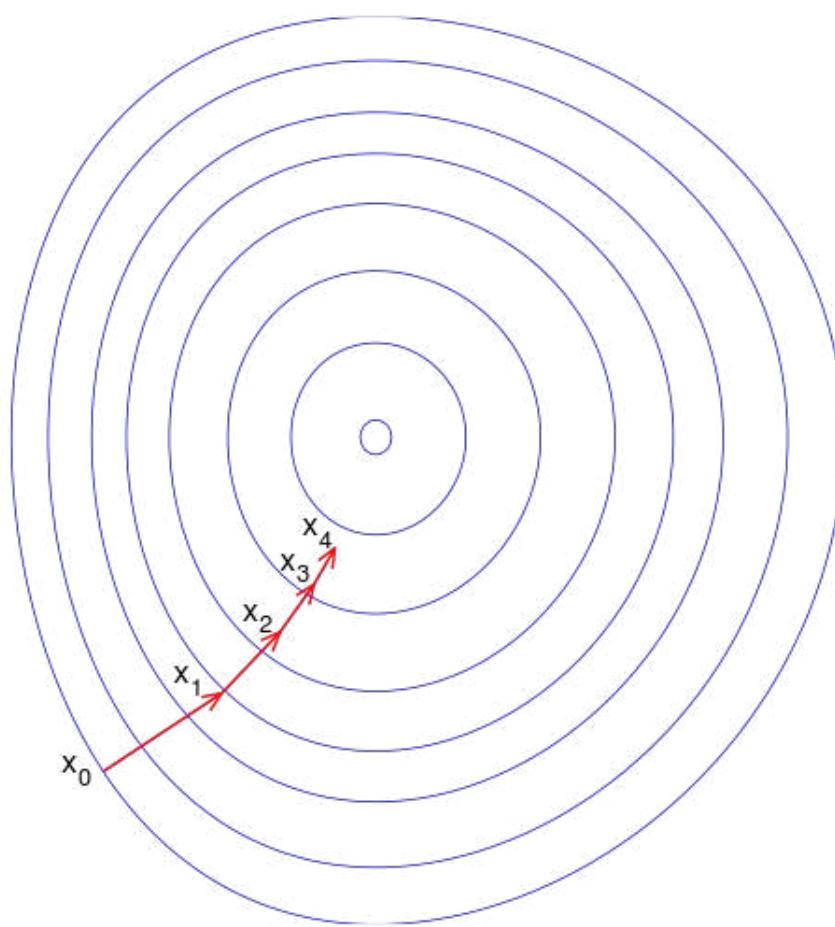
Gradient Descent

Gradient Descent

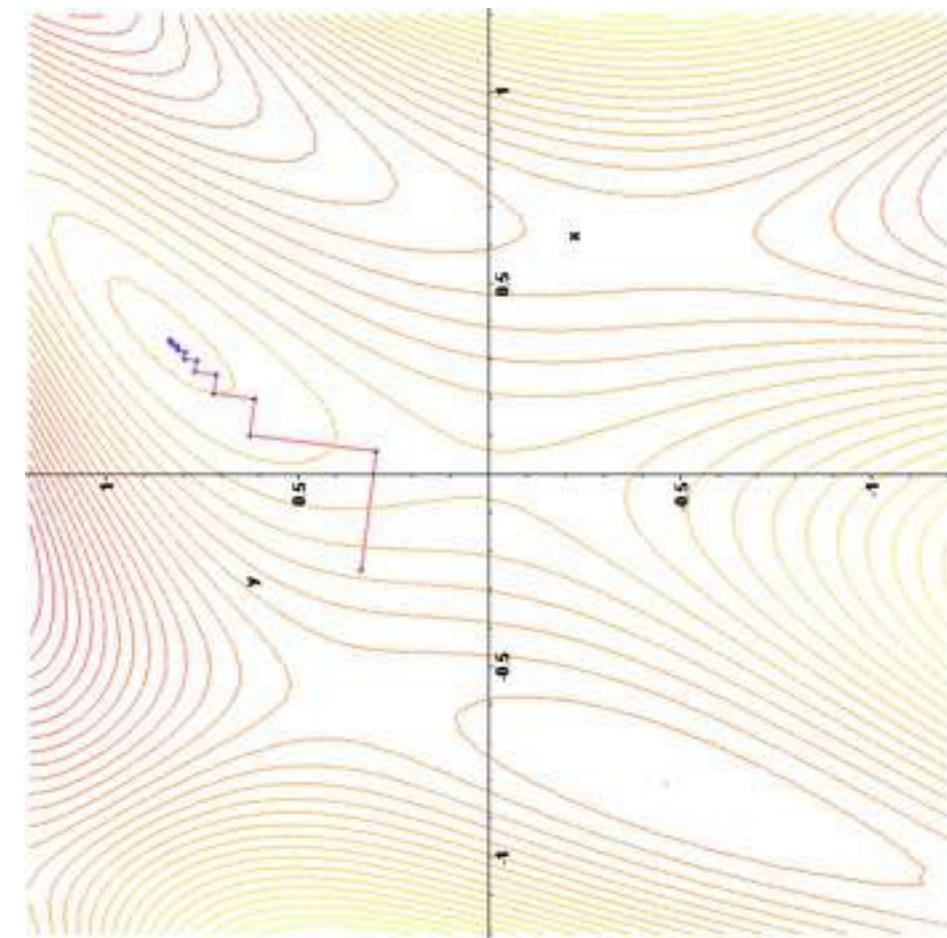
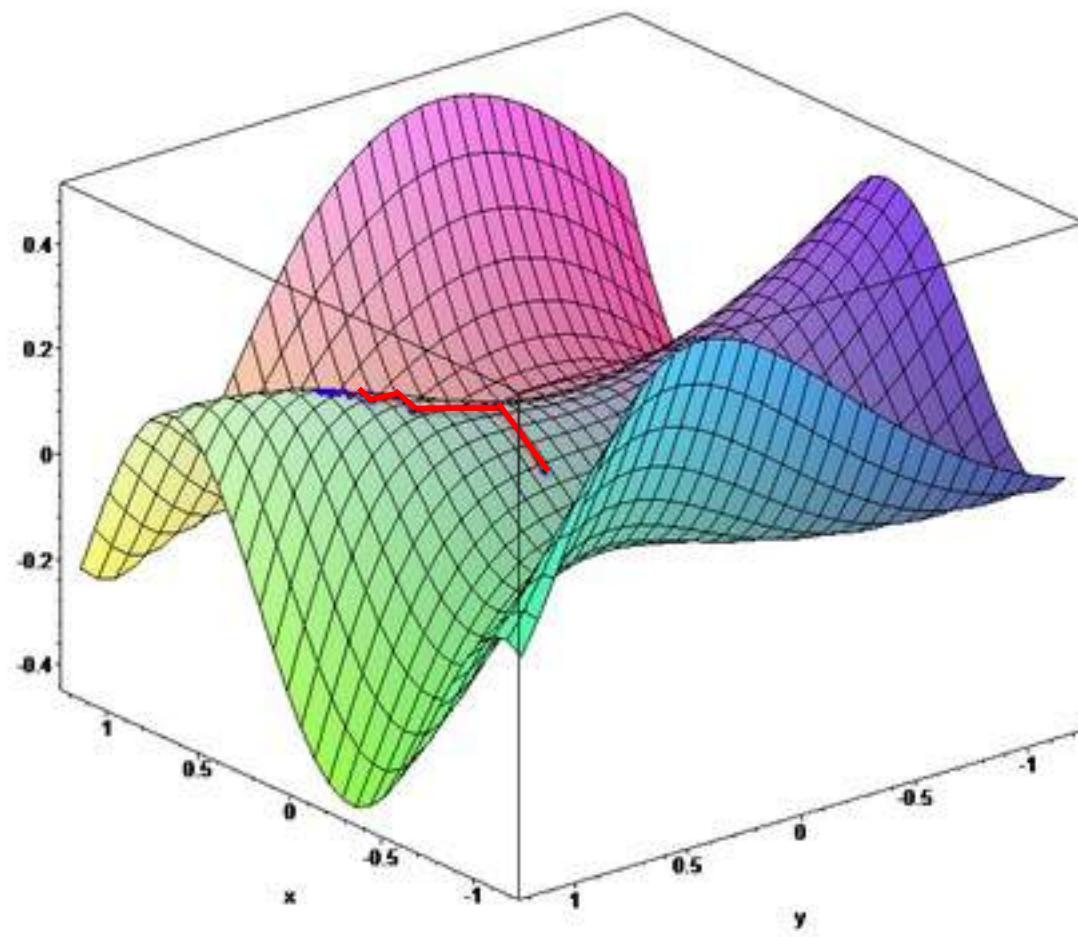
Gradient Descent



Gradient Descent

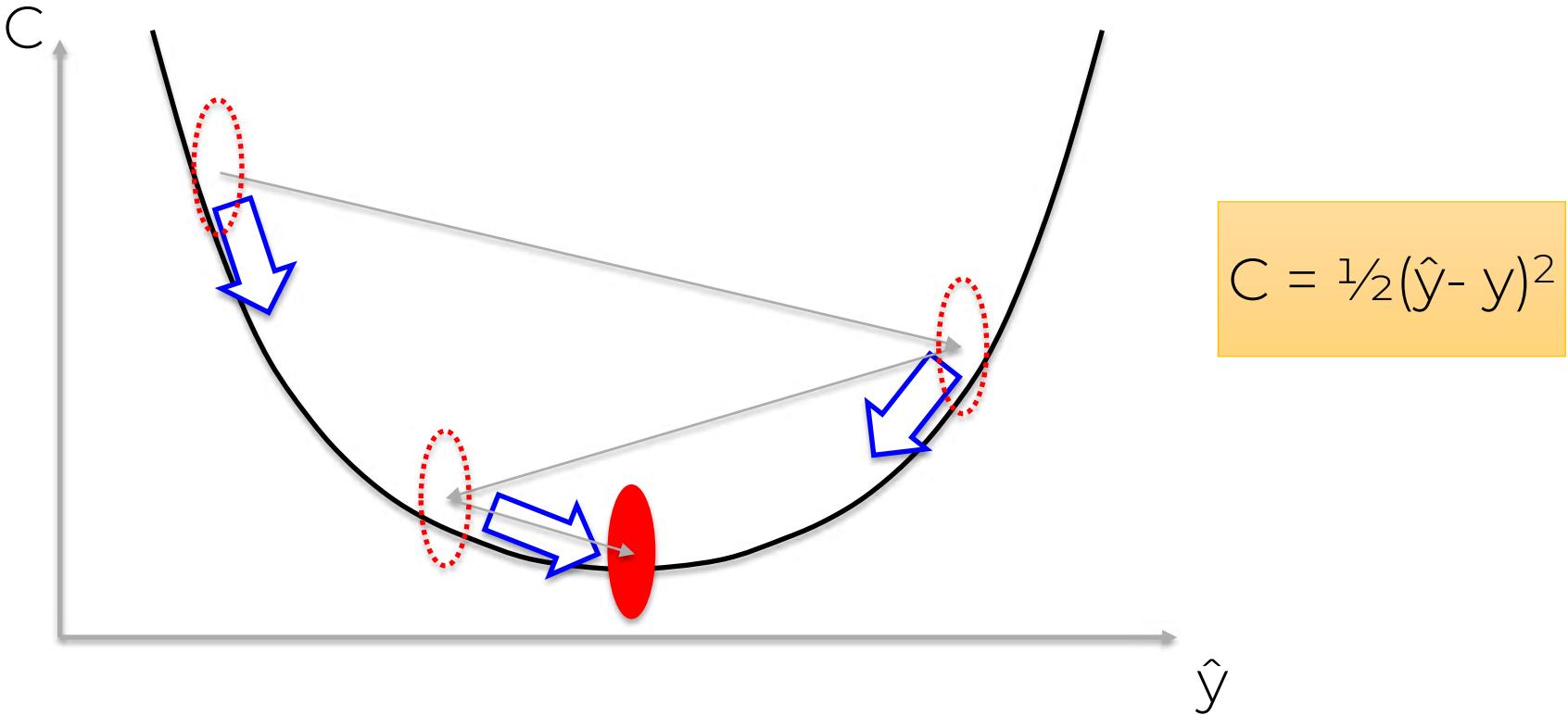


Gradient Descent

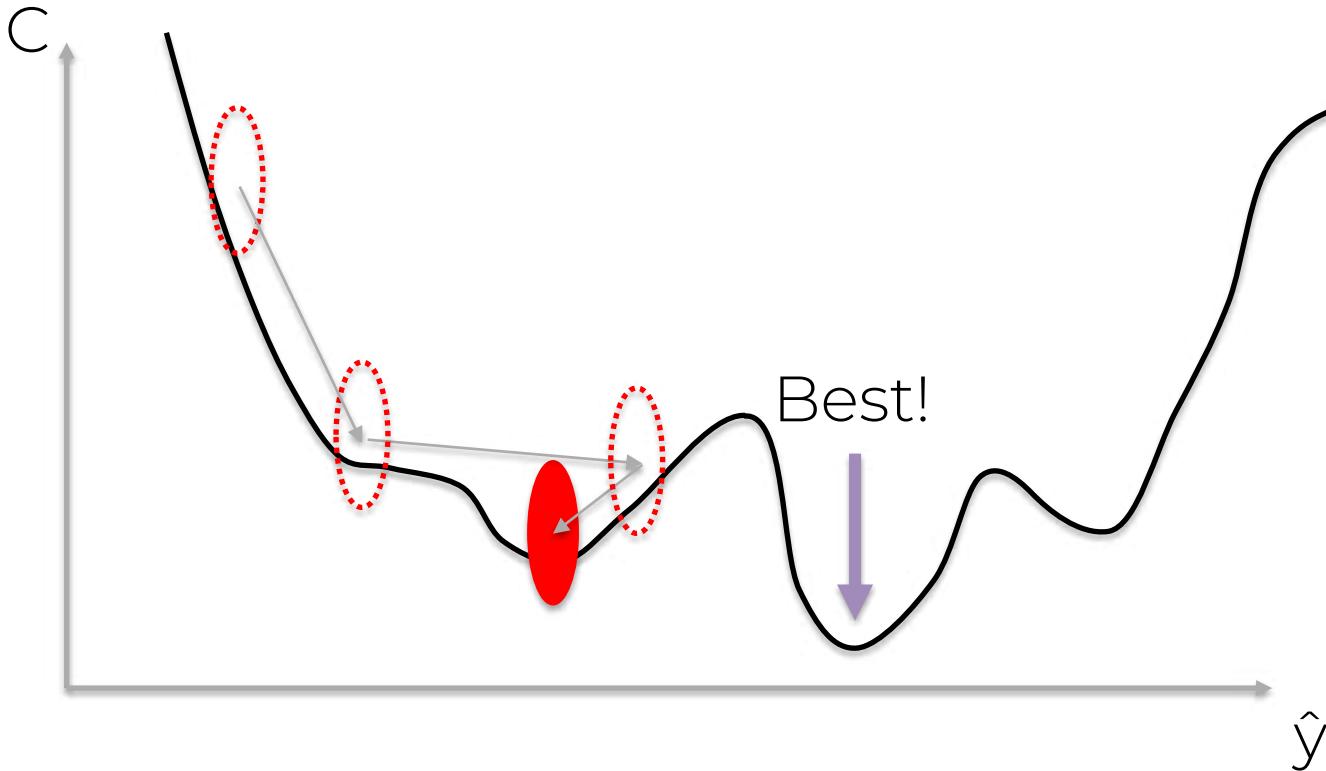


Stochastic Gradient Descent

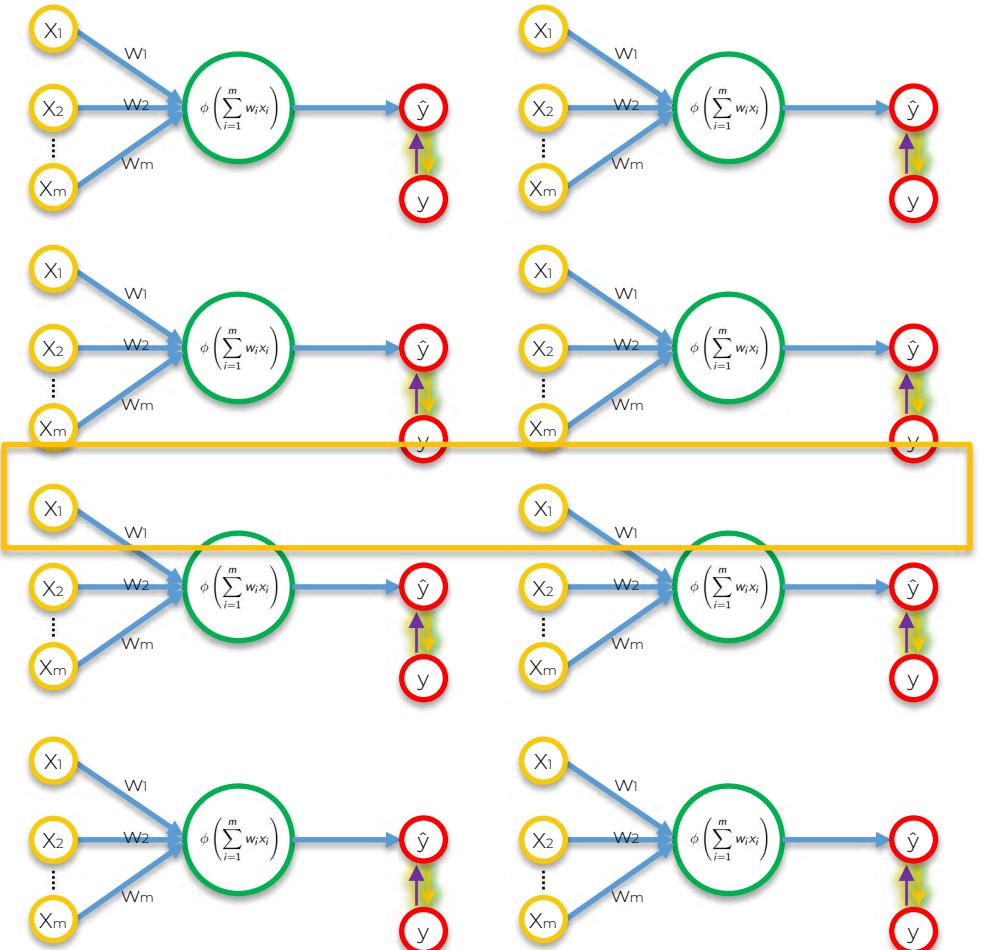
Stochastic Gradient Descent



Stochastic Gradient Descent



Stochastic Gradient Descent



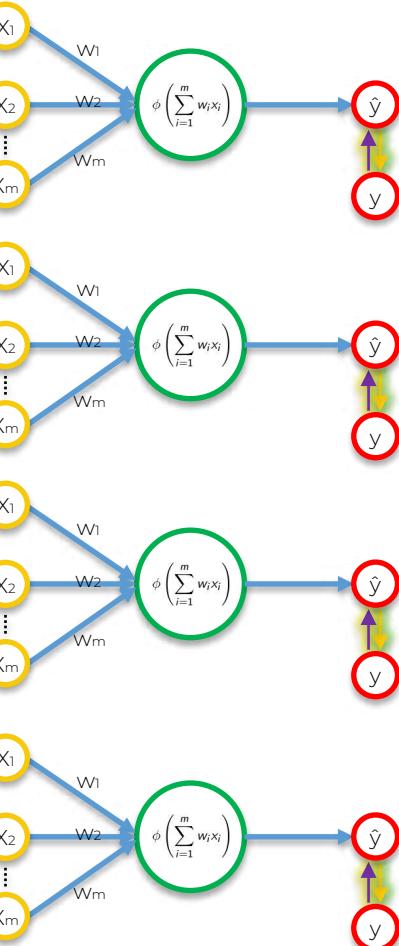
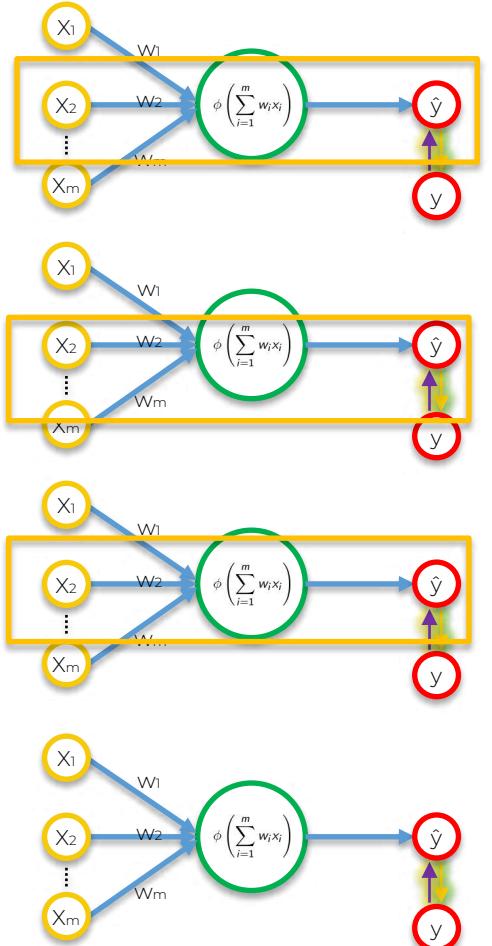
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

Adjust w_1, w_2, w_3



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%

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

Adjust w_1, w_2, w_3



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%

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

Stochastic Gradient Descent

Stochastic Gradient Descent

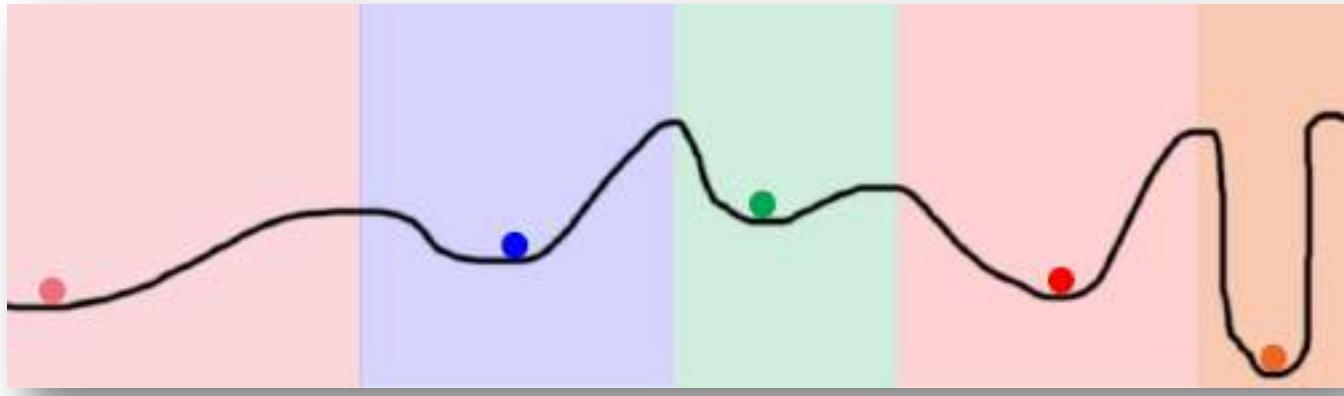
Additional Reading:

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

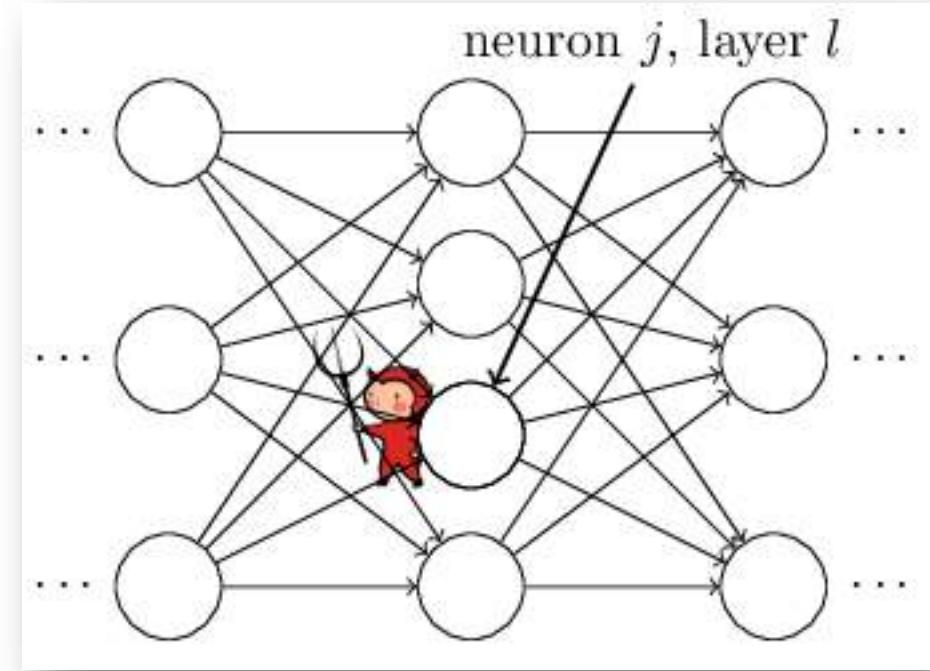
Additional Reading:

Neural Networks and Deep Learning

Michael Nielsen (2015)

Link:

<http://neuralnetworksanddeeplearning.com/chap2.html>



Backpropagation

Gradient Descent

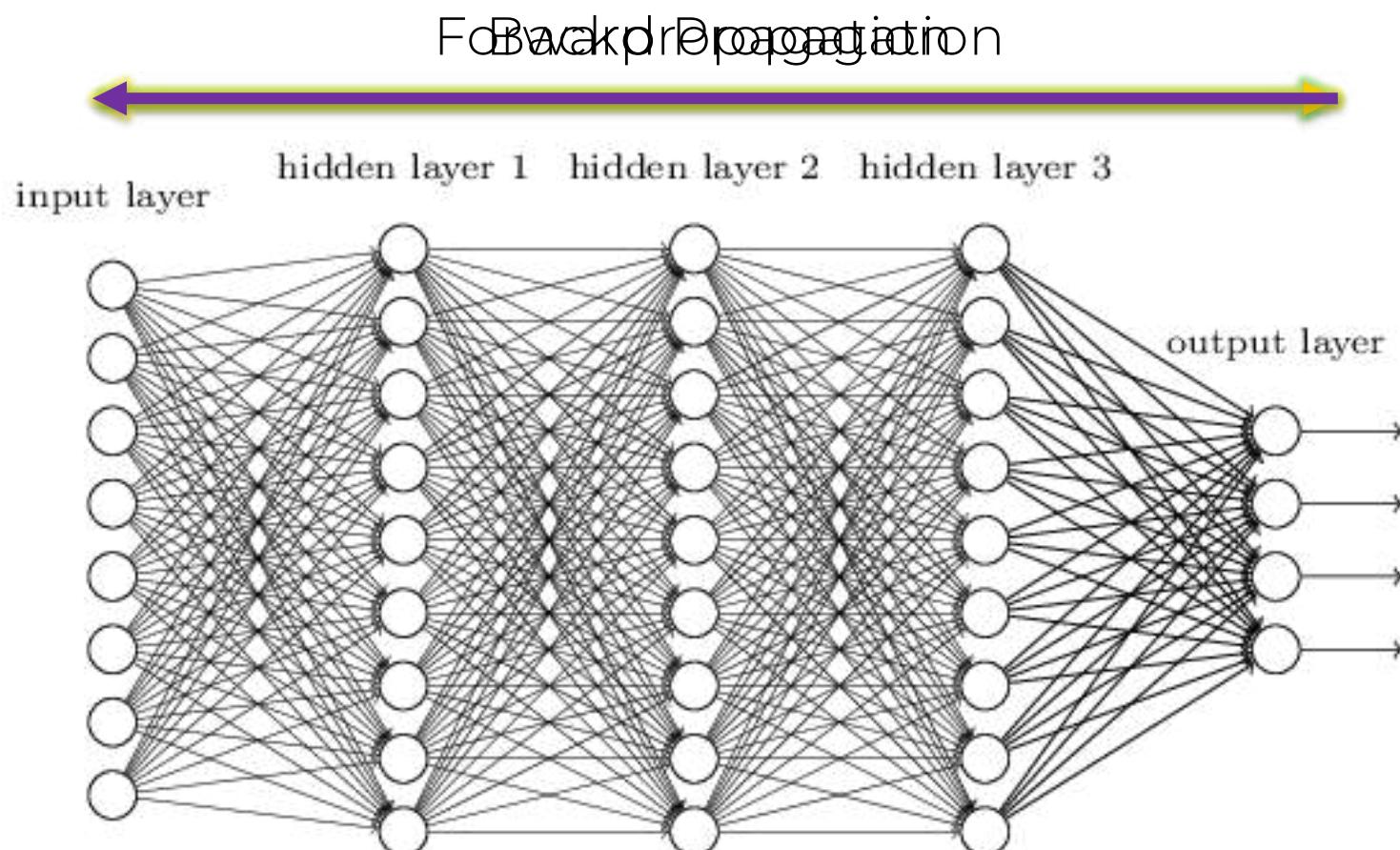


Image Source: neuralnetworksanddeeplearning.com

Stochastic Gradient Descent

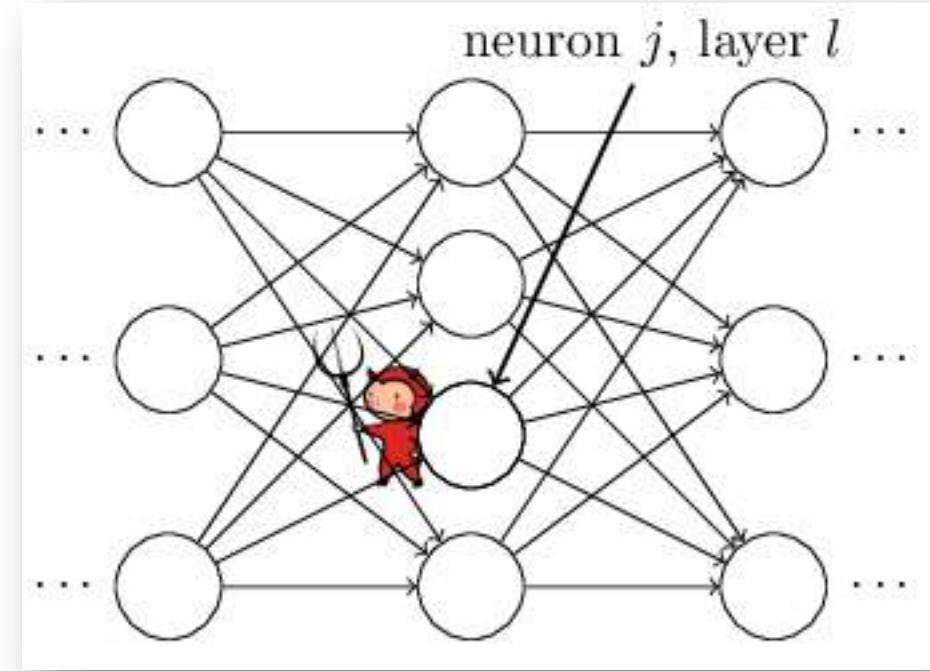
Additional Reading:

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

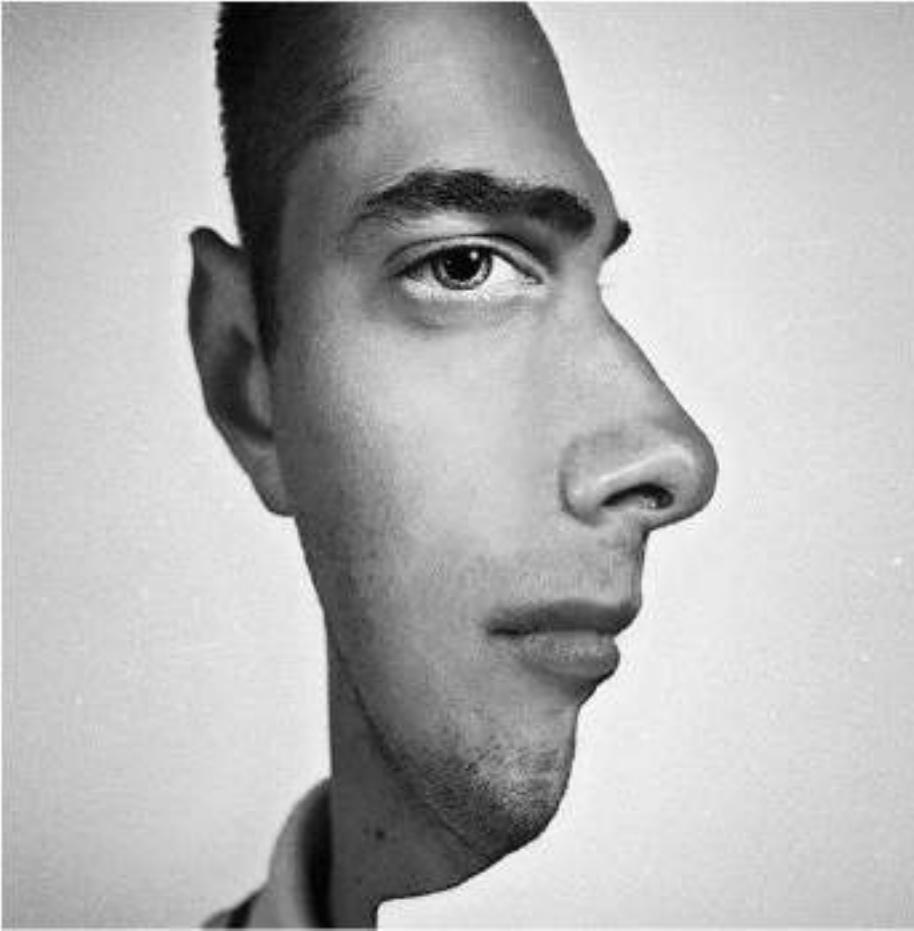
Plan of Attack

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

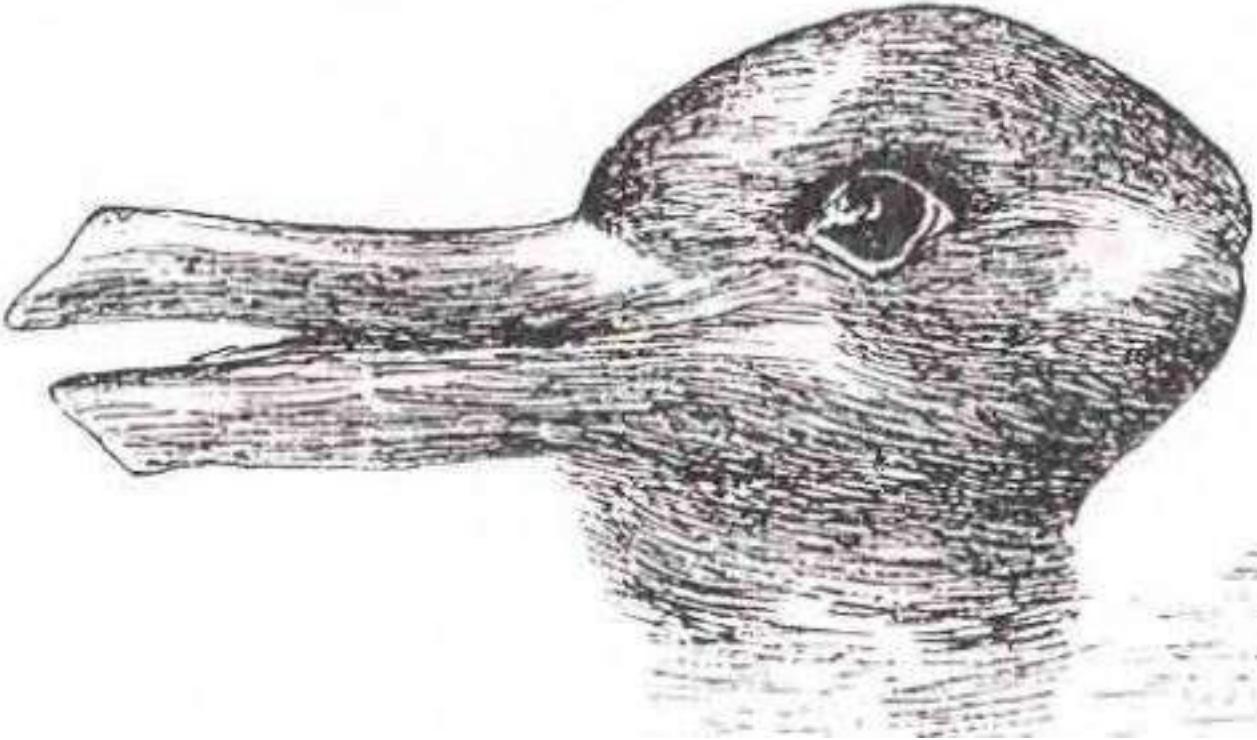
Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks

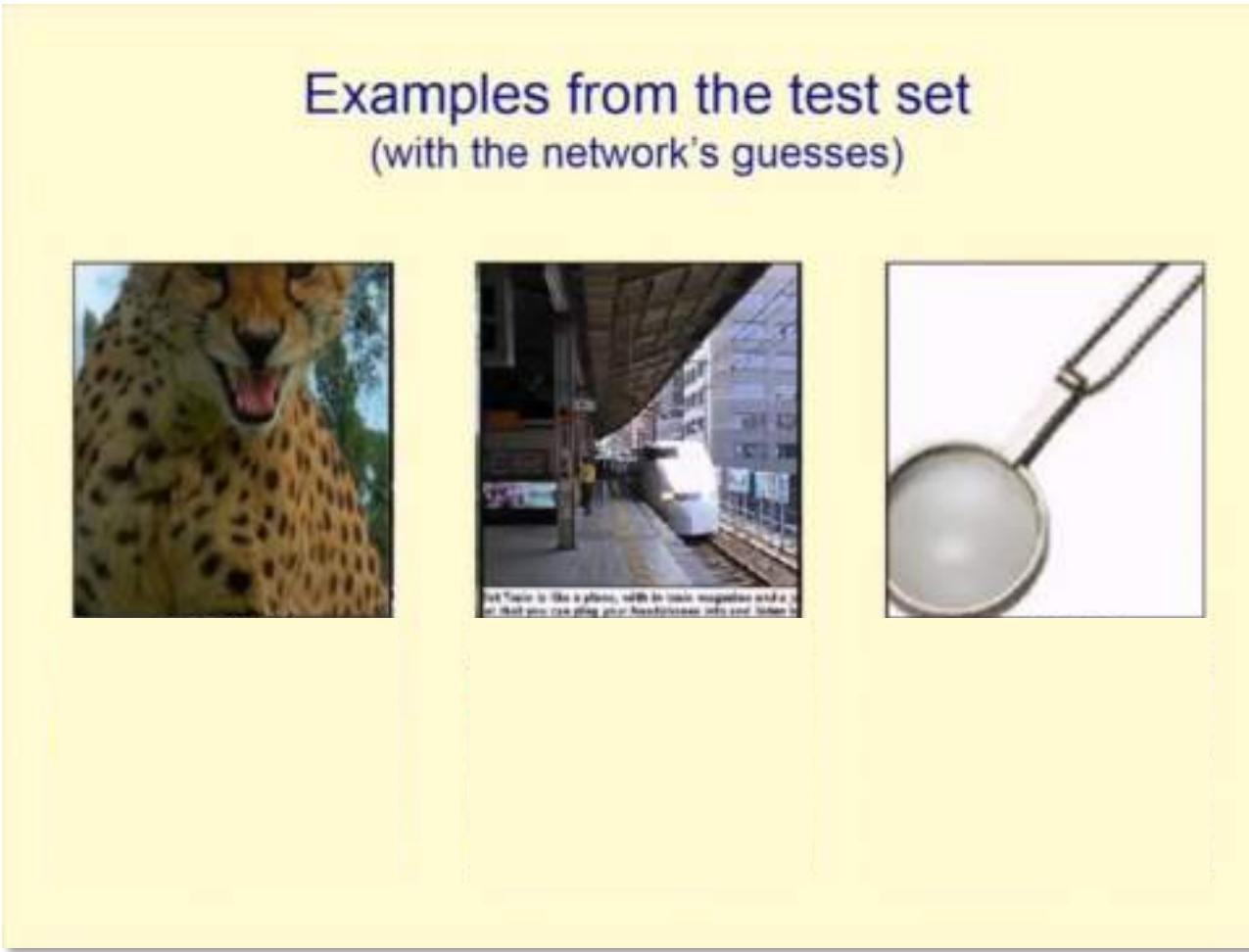
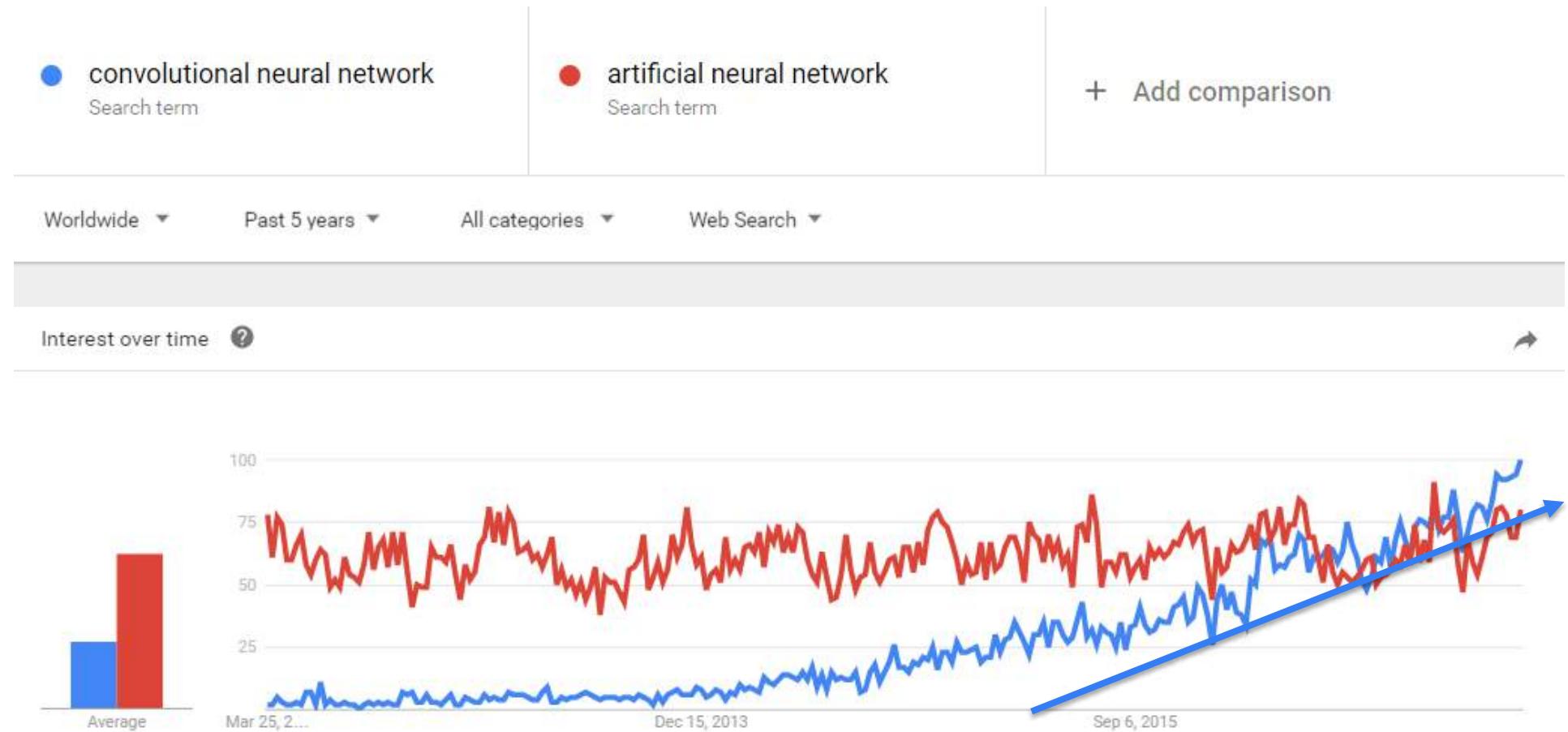


Image Source: a talk by Geoffrey Hinton

Convolutional Neural Networks



Source: google trends

Convolutional Neural Networks



Yann Lecun

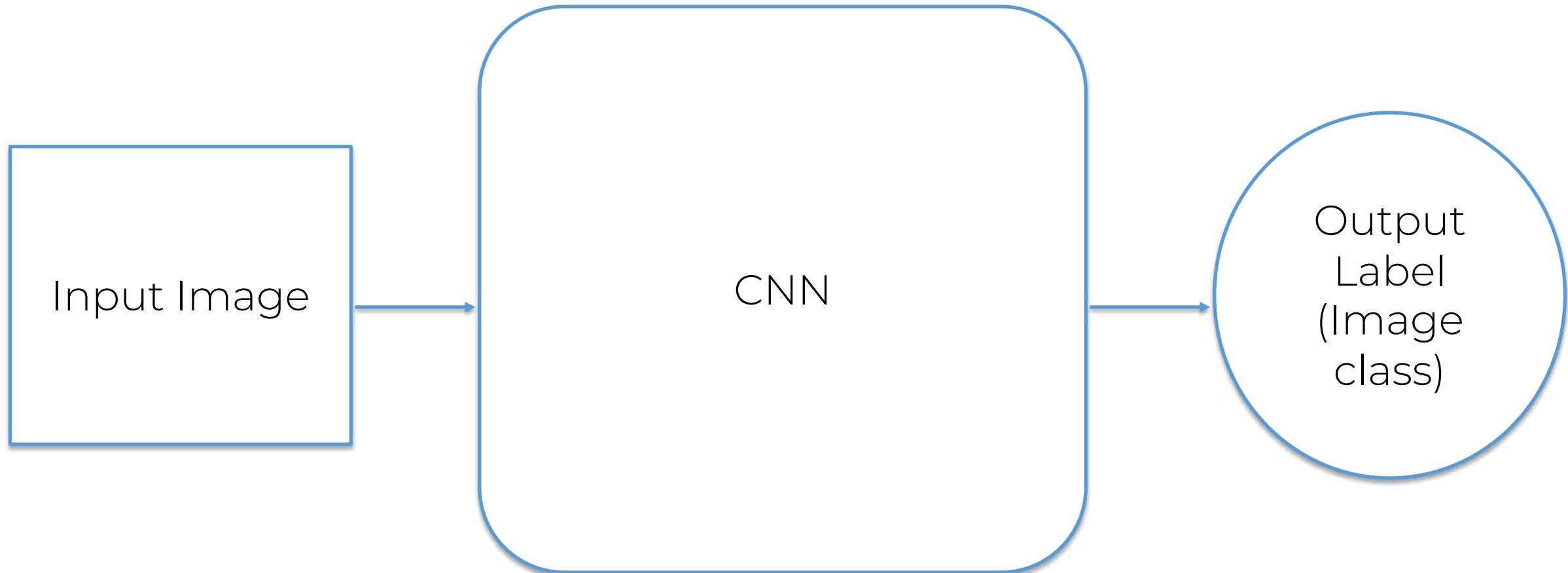
Convolutional Neural Networks

Google

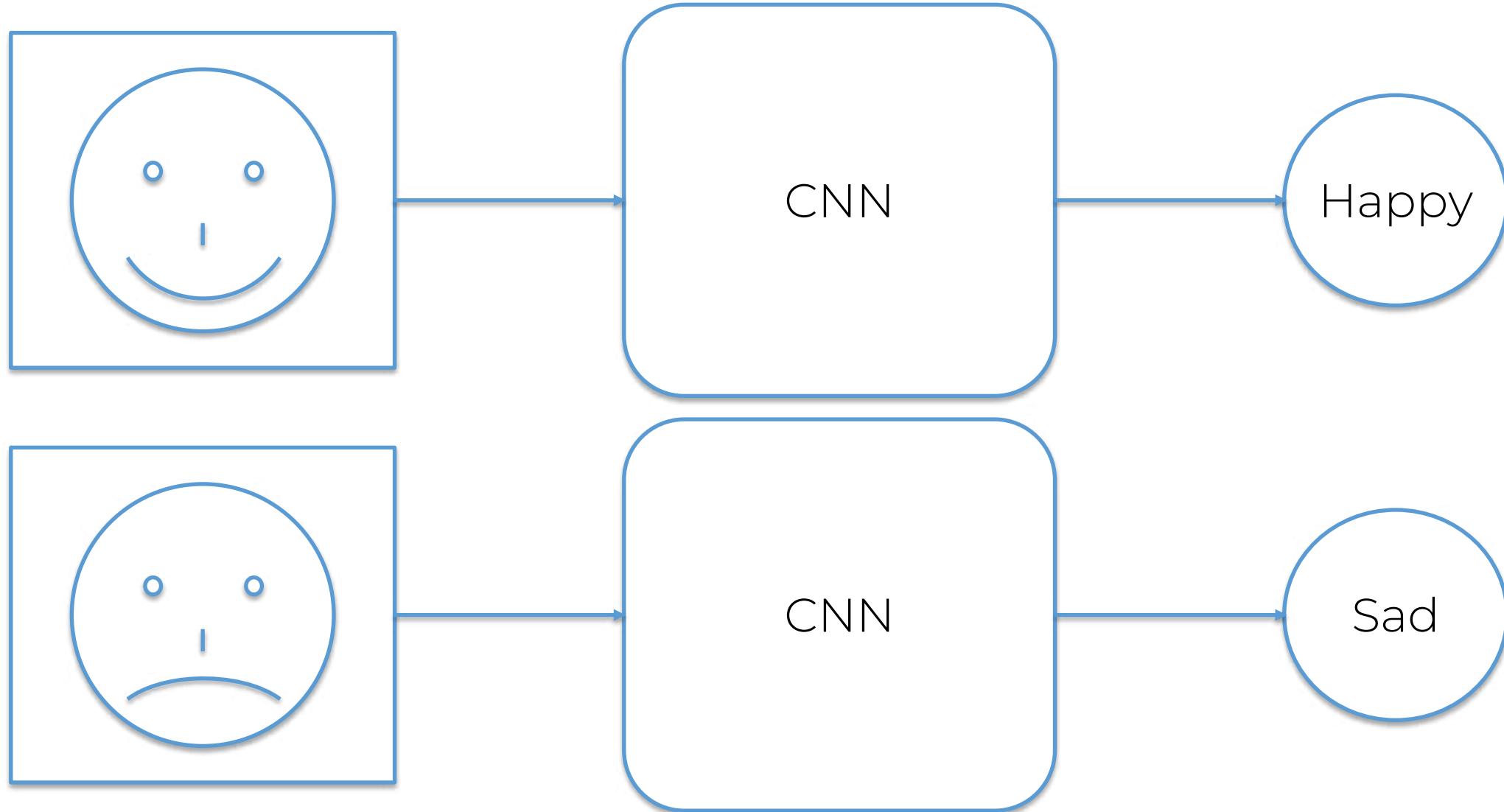
Facebook



Convolutional Neural Networks

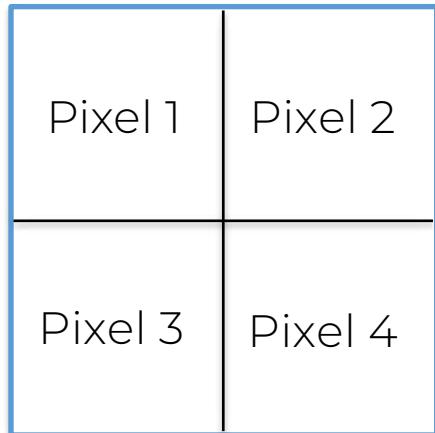


Convolutional Neural Networks

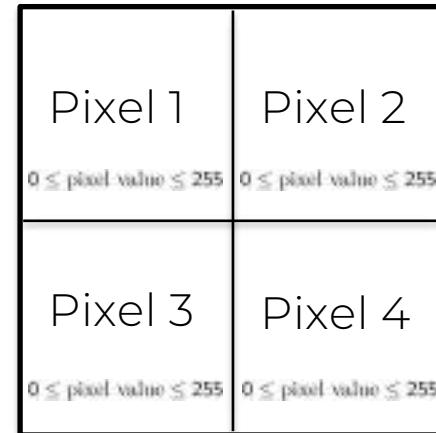


Convolutional Neural Networks

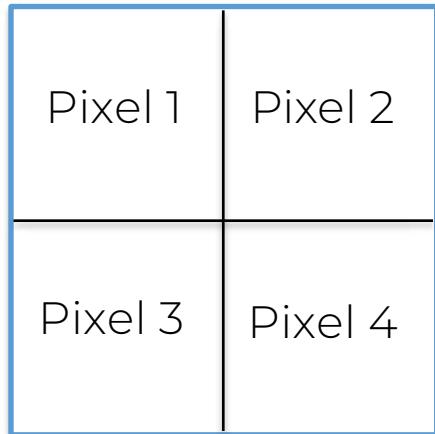
B / W Image 2x2px



2d array



Colored Image 2x2px

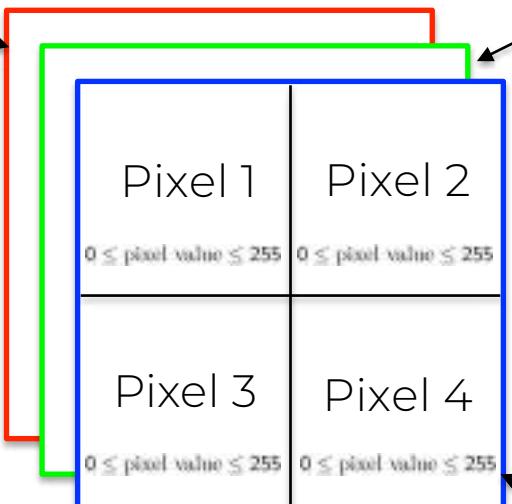


3d array

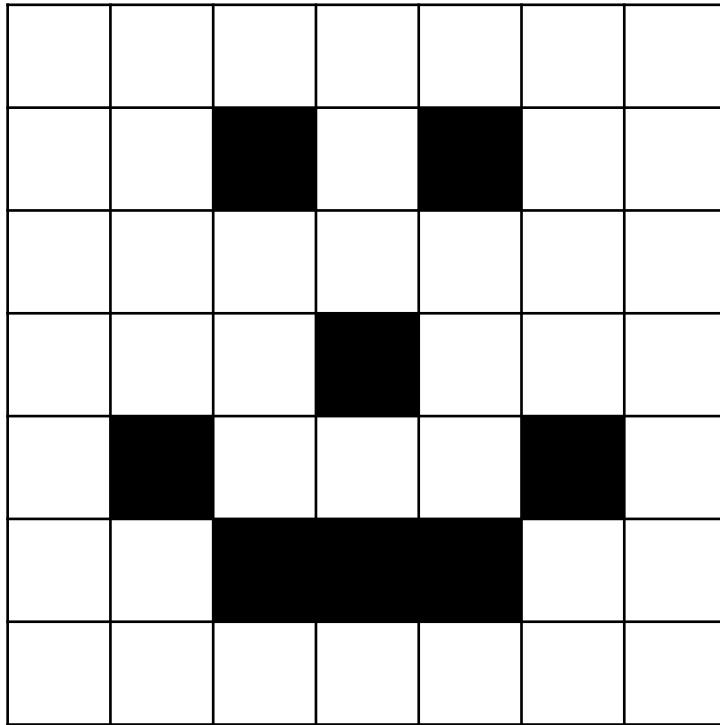
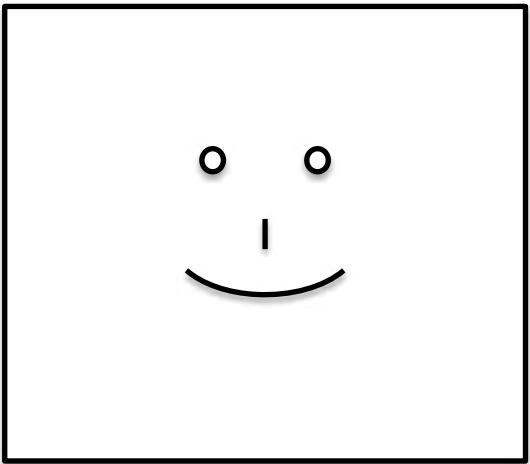
Red channel

Green channel

Blue channel

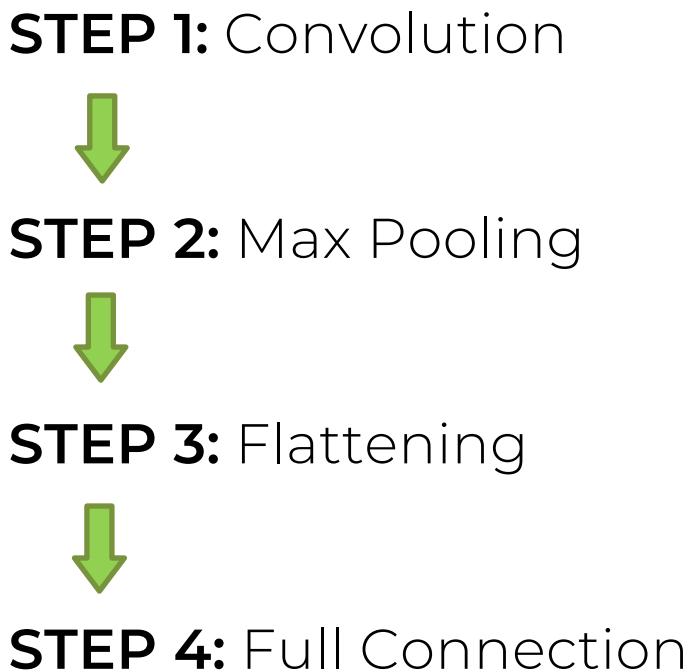


Convolutional Neural Networks



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

Convolutional Neural Networks



Convolutional Neural Networks

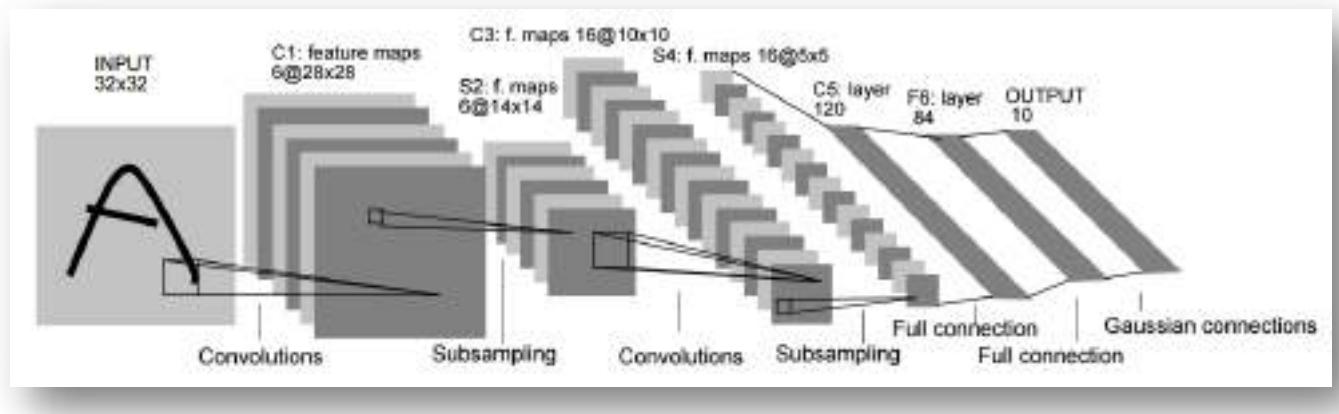
Additional Reading:

*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)

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

Link:

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

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

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



Input
Image

0	0	1
1	0	0
0	1	1

Feature
Detector



0				

Feature Map

Step 1 - Convolution

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



Input
Image

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



Input
Image

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

0	0	1
1	0	0
0	1	1

Feature
Detector



0	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

0	0	1
1	0	0
0	1	1

Feature
Detector



0	1	0	0	0

Feature Map

Step 1 - Convolution

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



Input
Image

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



Input
Image

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

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

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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



Input
Image

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

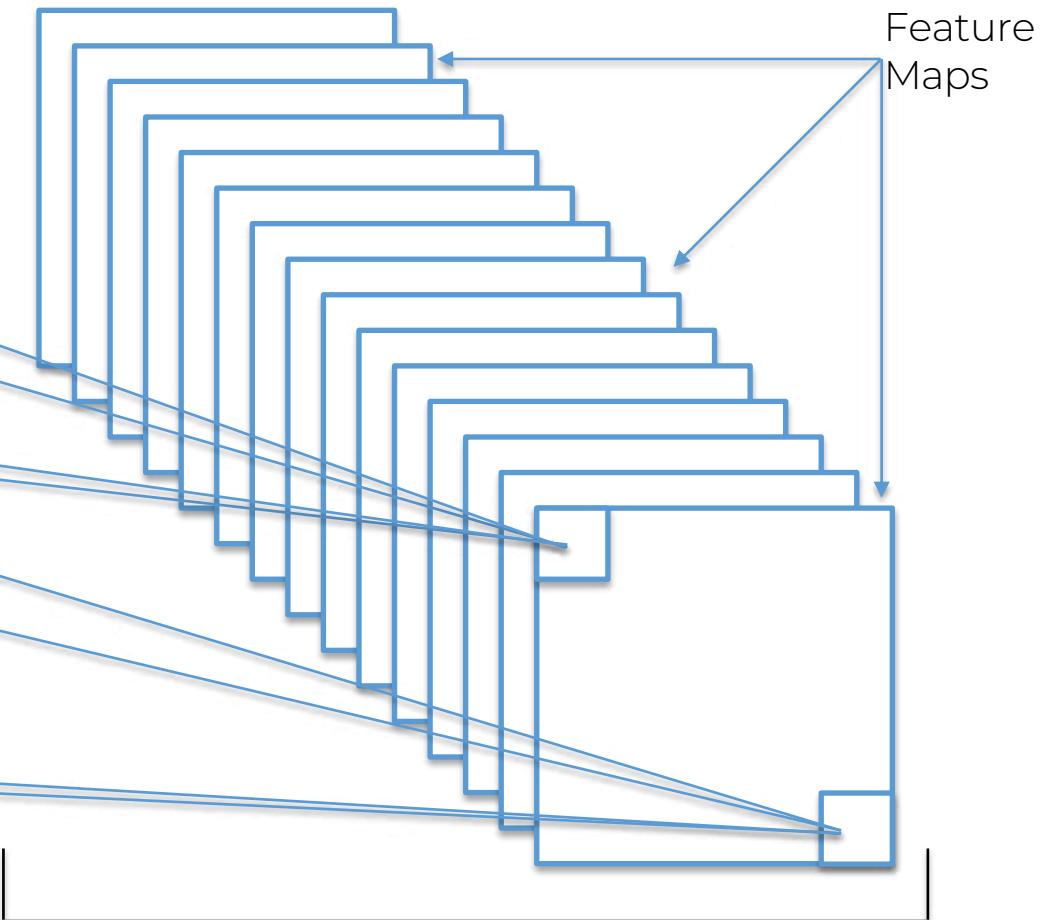
Feature Map

Step 1 - Convolution

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

Input Image

We create many feature maps to obtain our first convolution layer



Convolutional Layer

Step 1 - Convolution

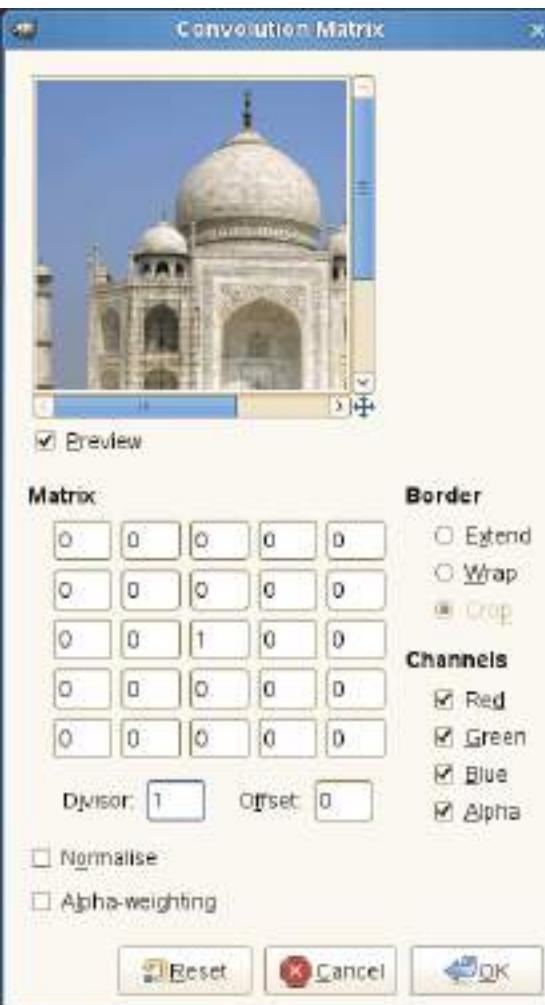


Image Source: docs.gimp.org/en/plug-in-convmatrix.html

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:

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



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

Step 1 - Convolution

Edge Detect:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



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



*

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



Image Source: eonardoaraujosantos.gitbooks.io

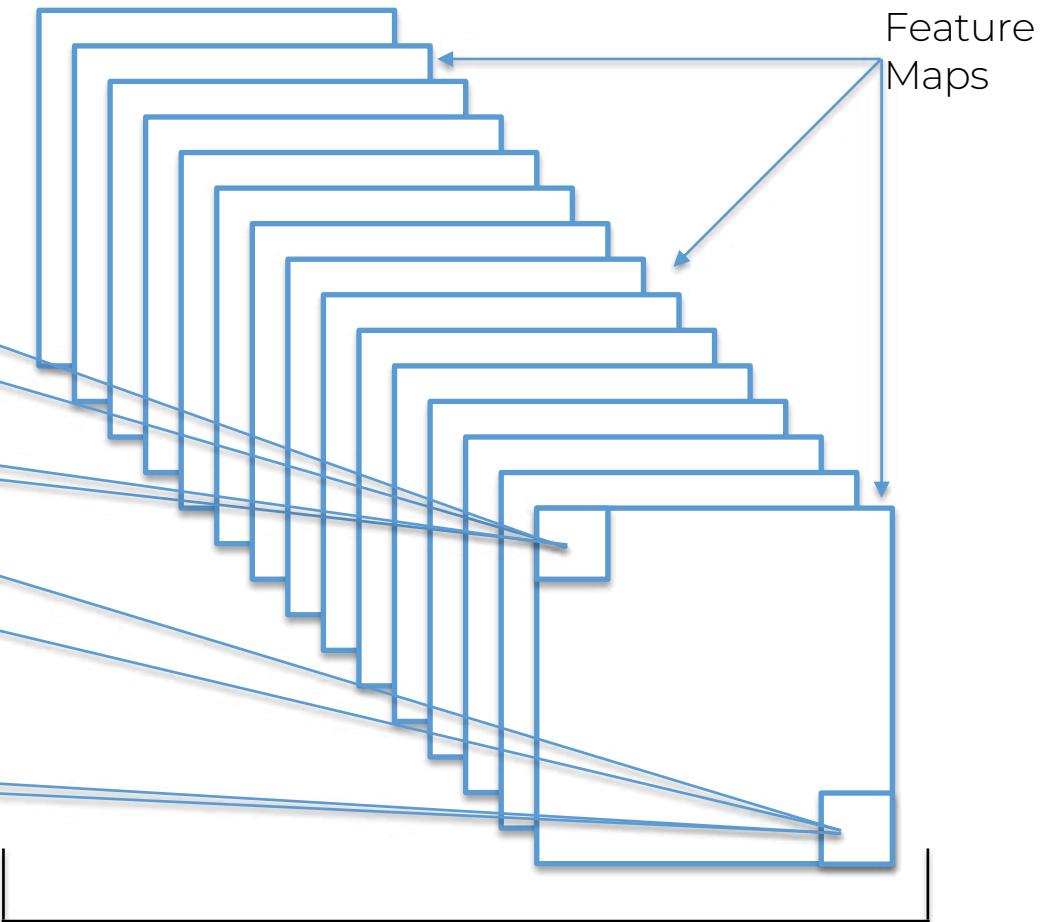
Step 1(B) - ReLU Layer

Step 1(B) – ReLU Layer

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

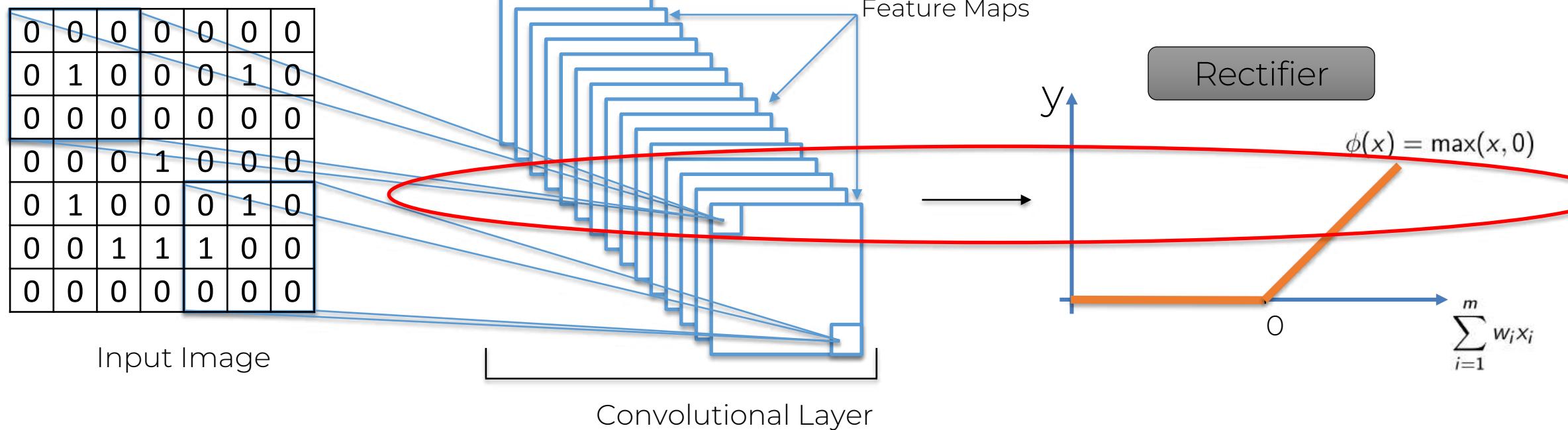
Input Image

We create many feature maps to obtain our first convolution layer



Convolutional
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



Black = negative; white = positive values

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

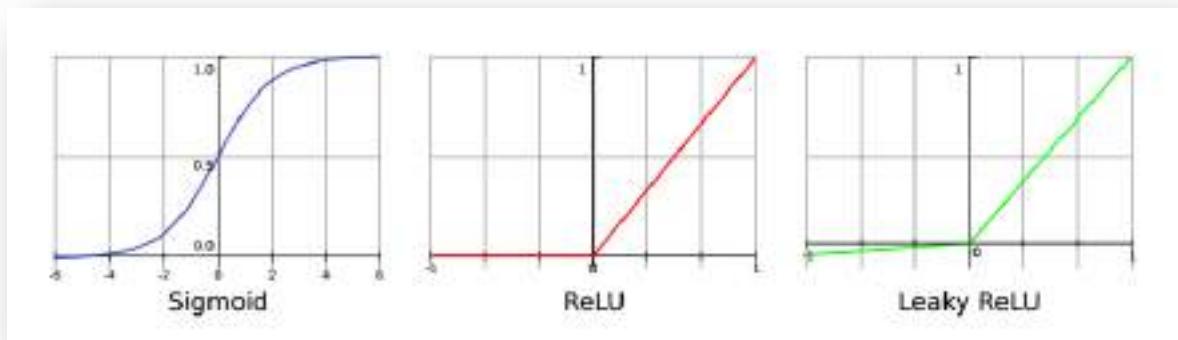
Additional Reading:

*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

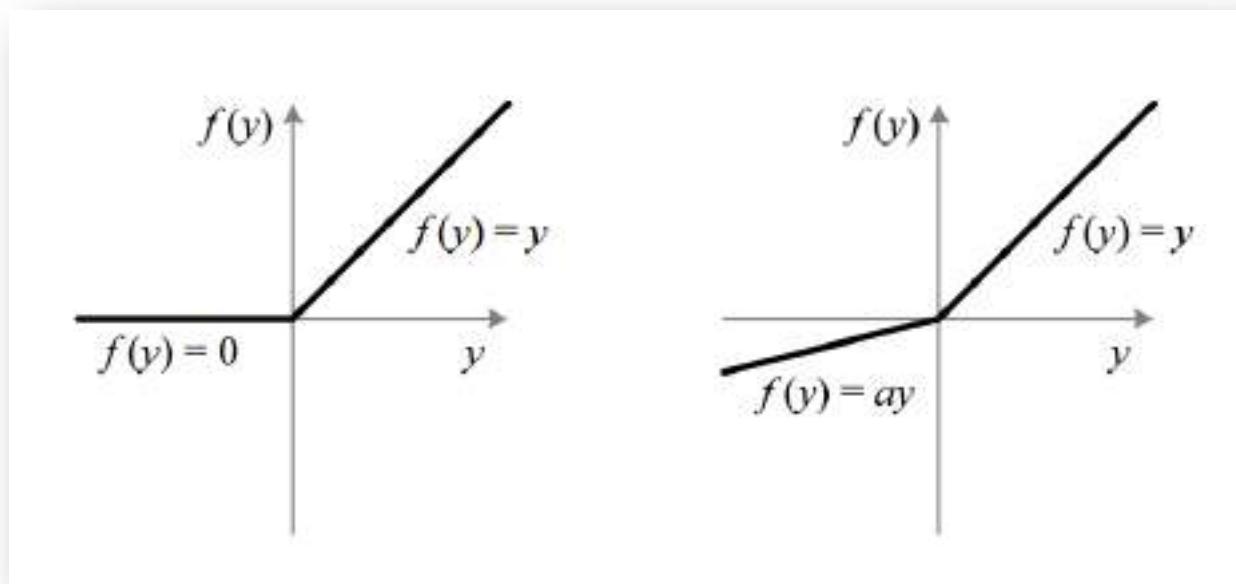
Additional Reading:

*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

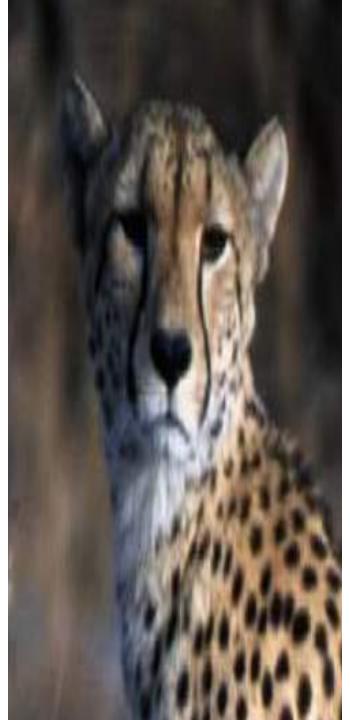


Image Source: Wikipedia

Step 2 - Max Pooling

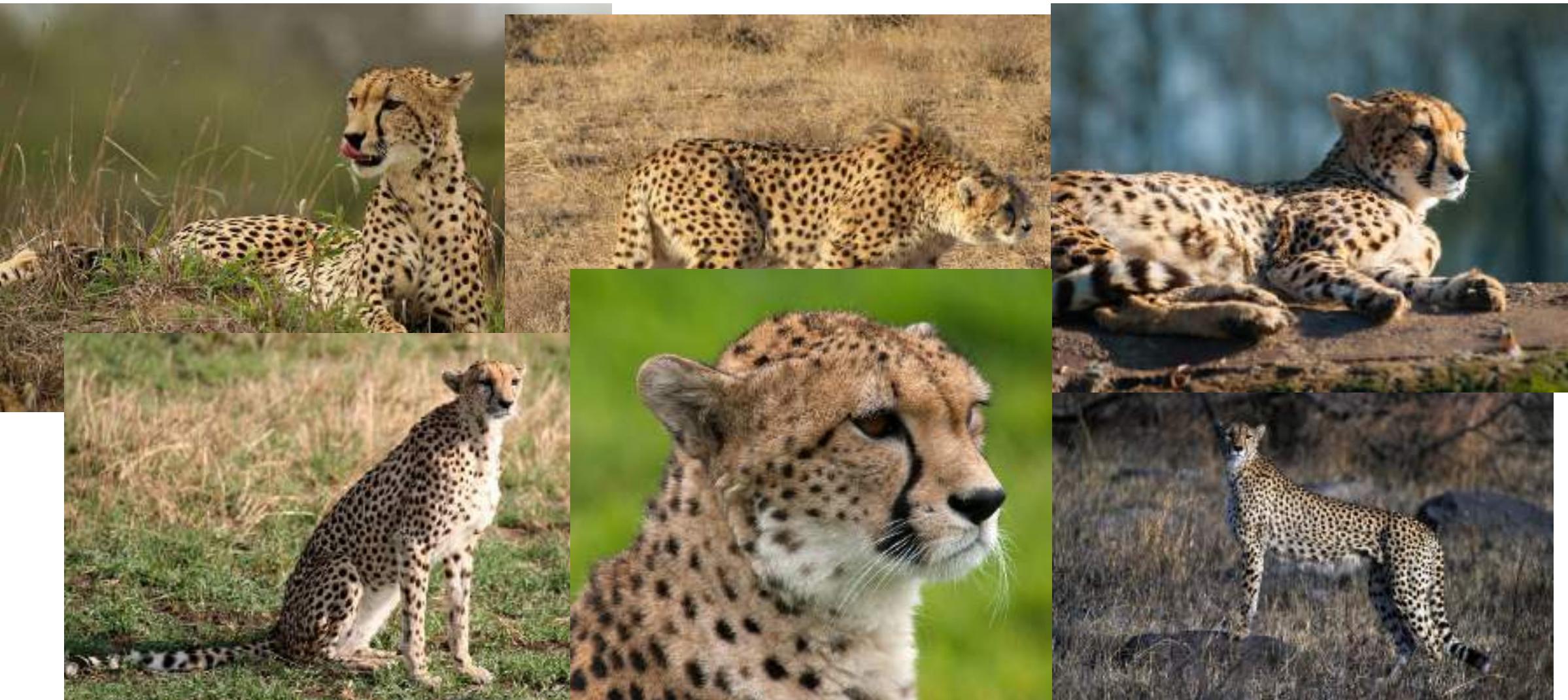


Image Source: Wikipedia

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	

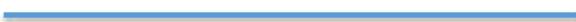
Pooled Feature Map

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

The diagram illustrates the process of Max Pooling. A blue arrow points from the bottom-right corner of a 2x2 subgrid in the Feature Map to the corresponding cell in the Pooled Feature Map. This subgrid contains the values [2, 1], [4, 0]. The maximum value in this subgrid is 4, which is then placed in the bottom-right cell of the Pooled Feature Map.

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	

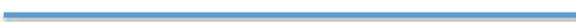
Pooled Feature Map

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



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:

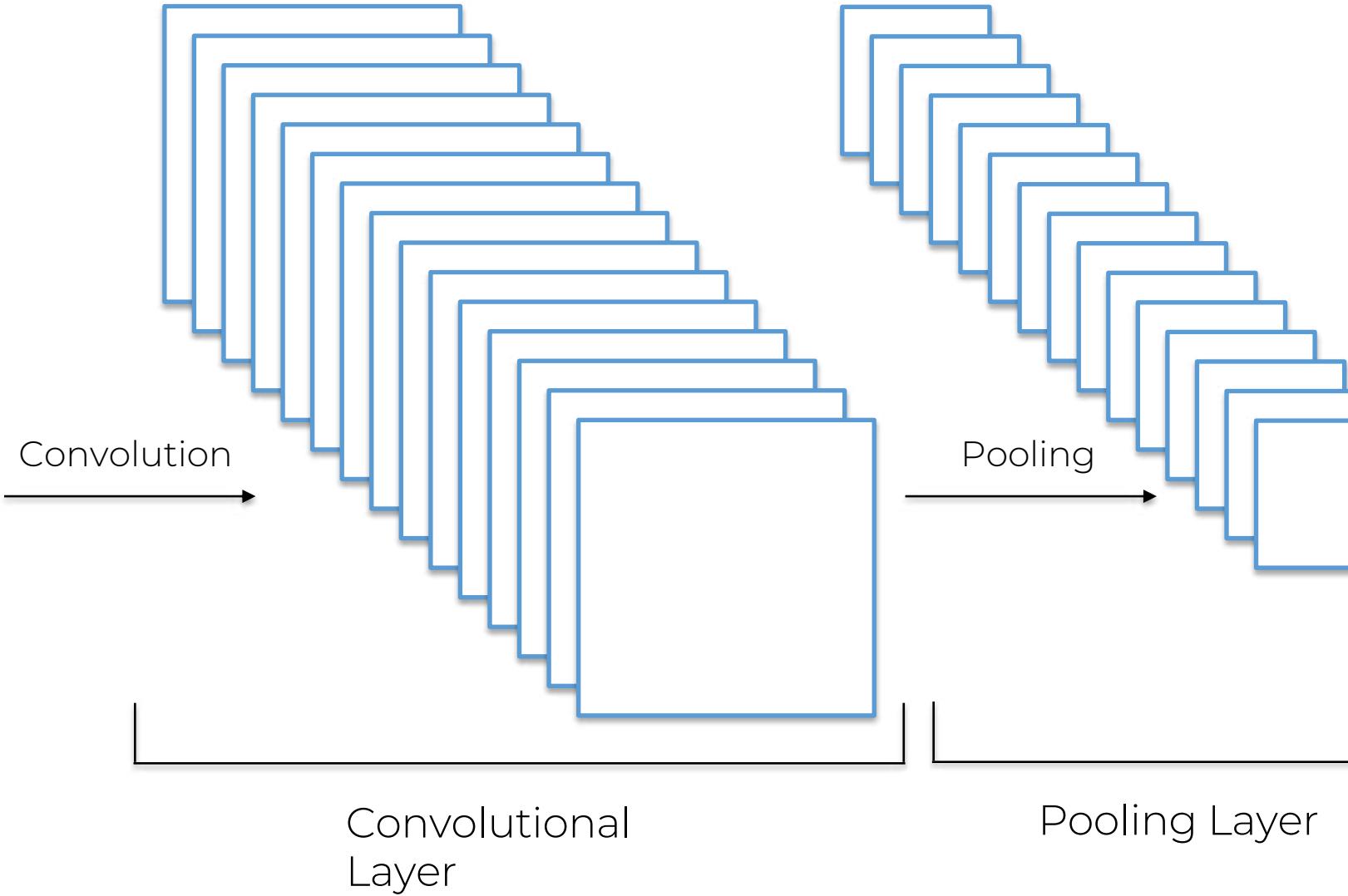
http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf



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

1	1	0
4	2	1
0	2	1

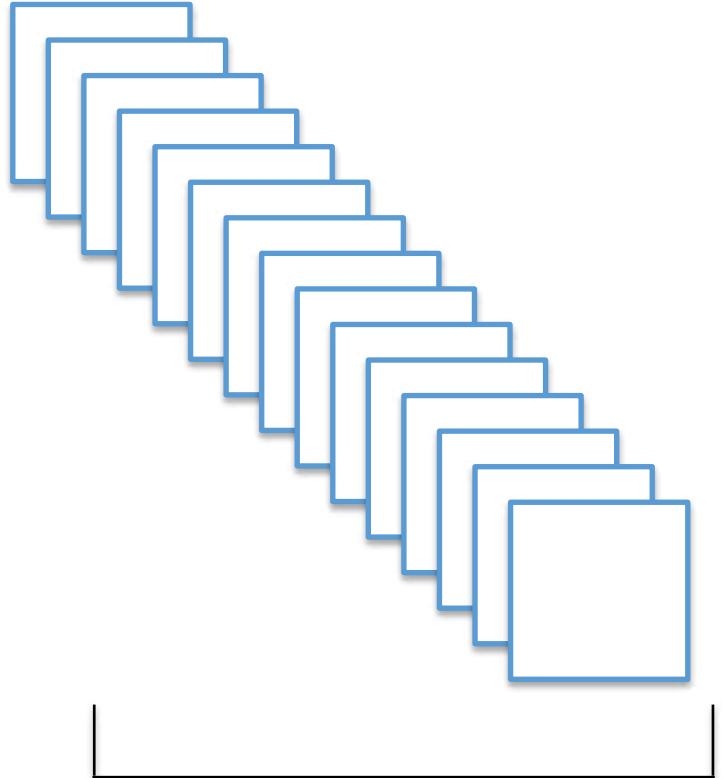
Pooled Feature
Map

Flattening



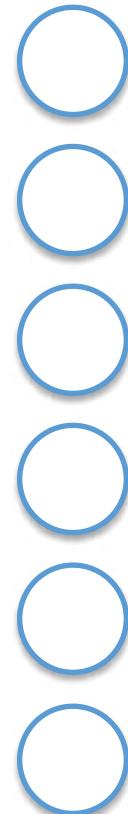
1	1	0	4	2	1	0	2	1
---	---	---	---	---	---	---	---	---

Step 3 - Flattening



Pooling Layer

Flattening

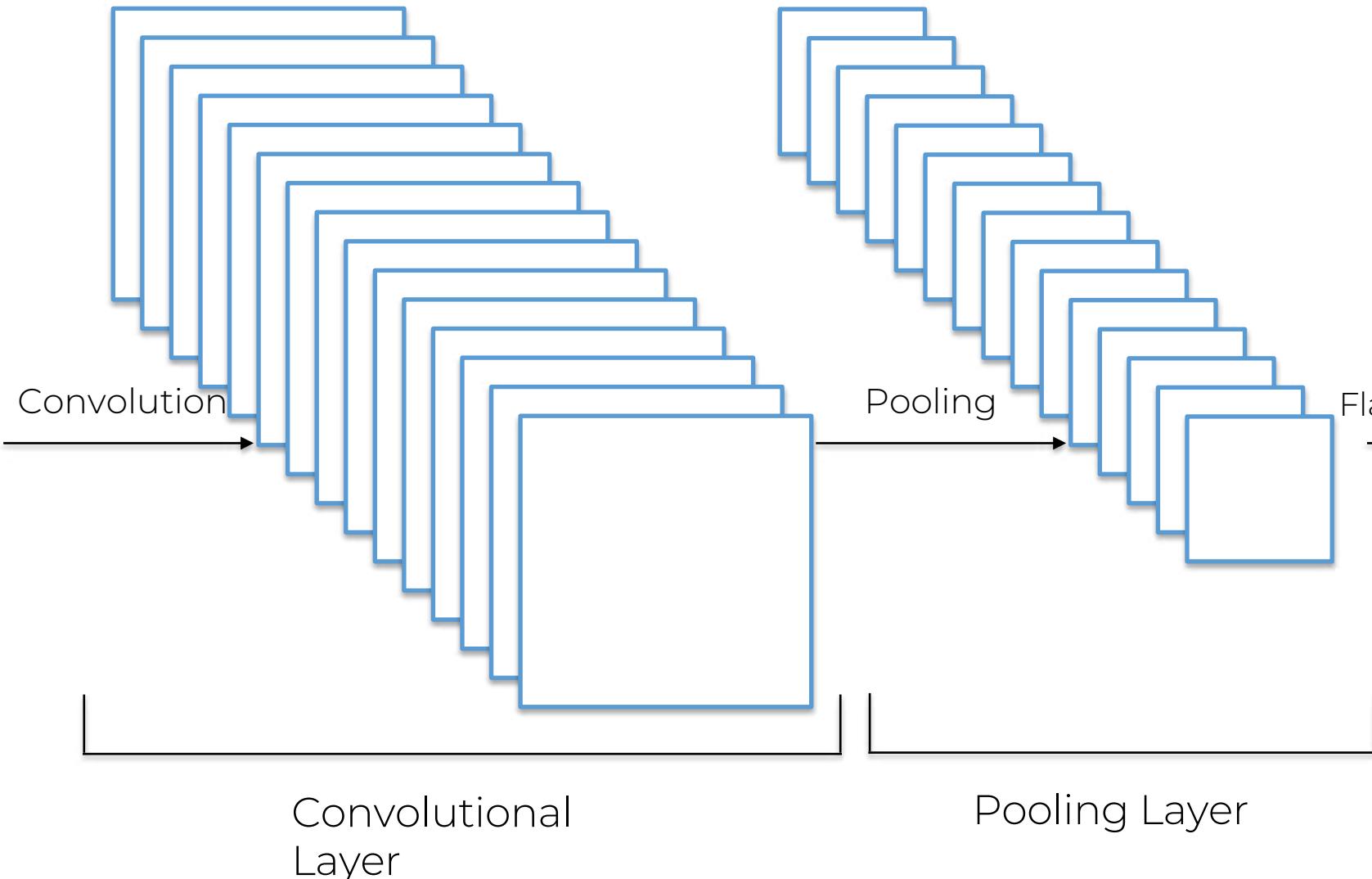


Input layer of a future ANN

Step 3 - Flattening

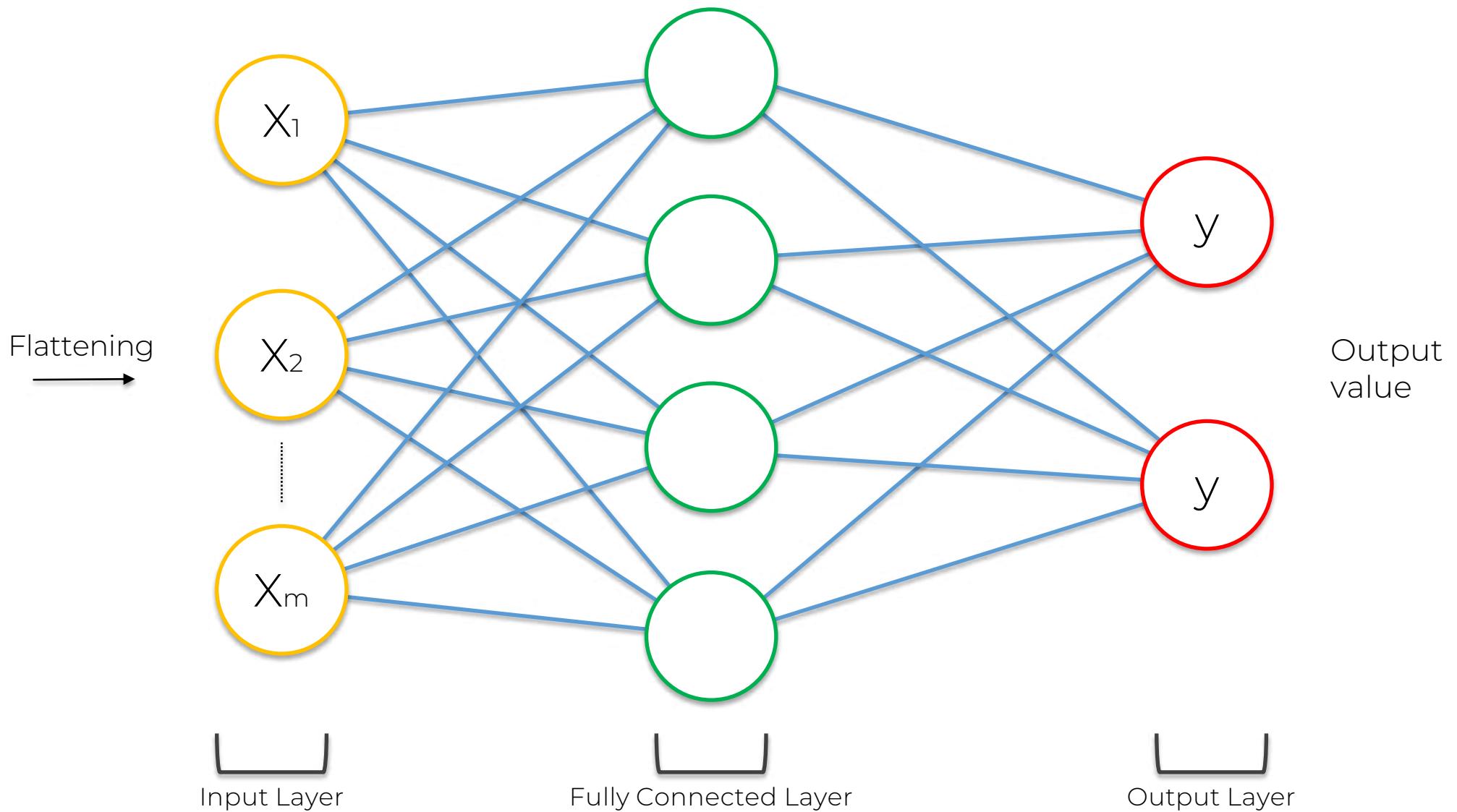
0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0

Input Image

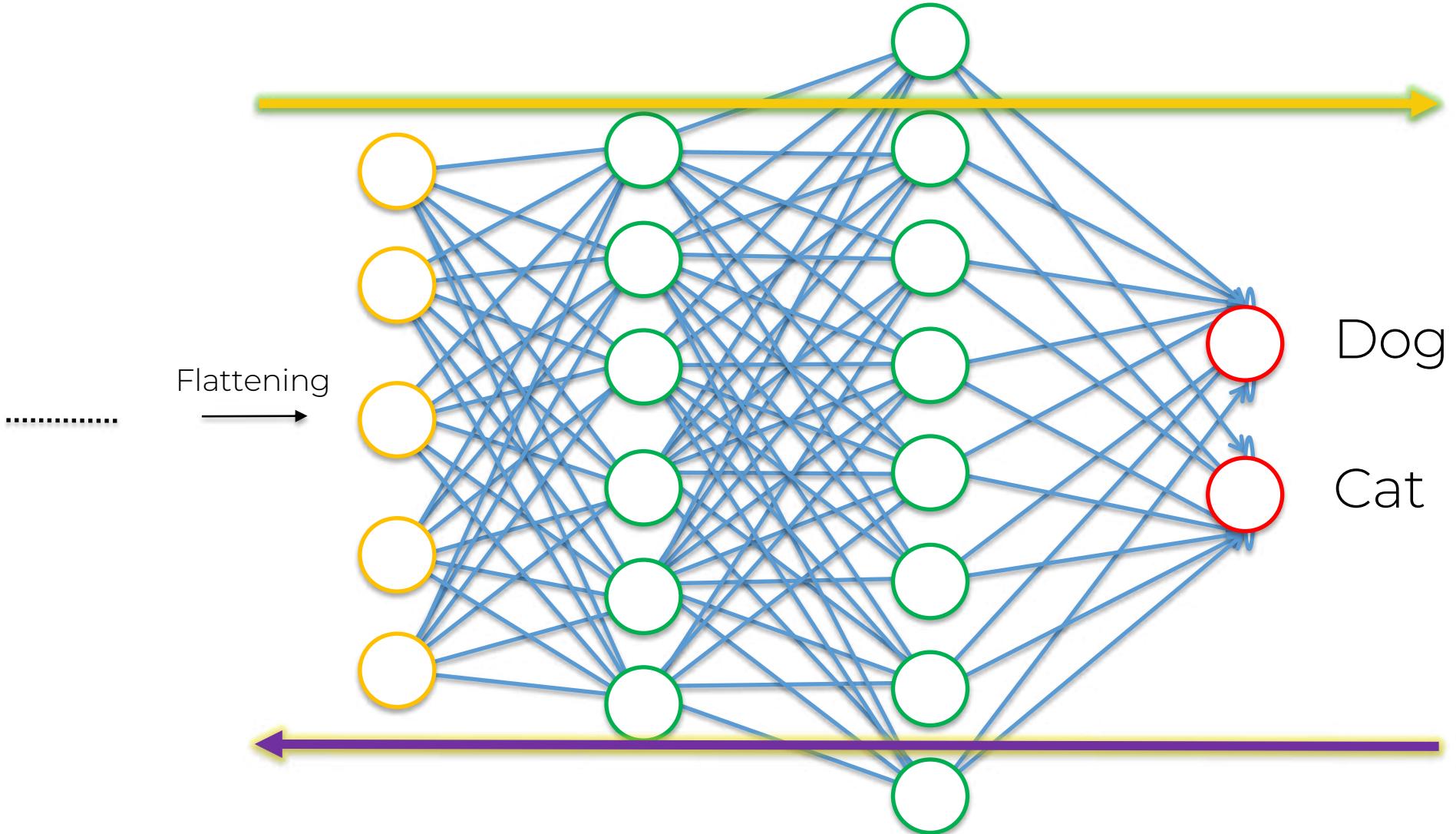


Step 4 - Full Connection

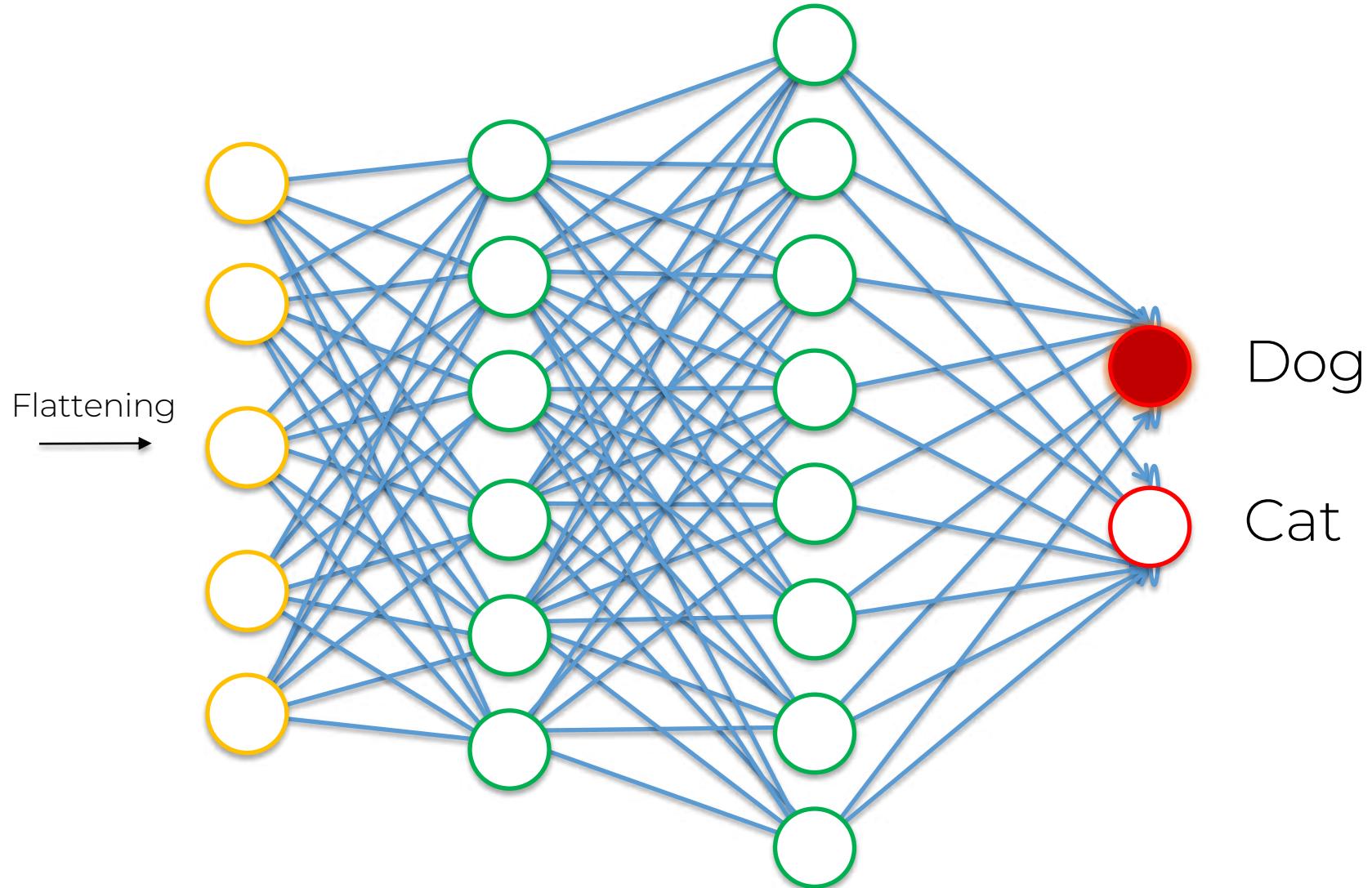
Step 4 - Full Connection



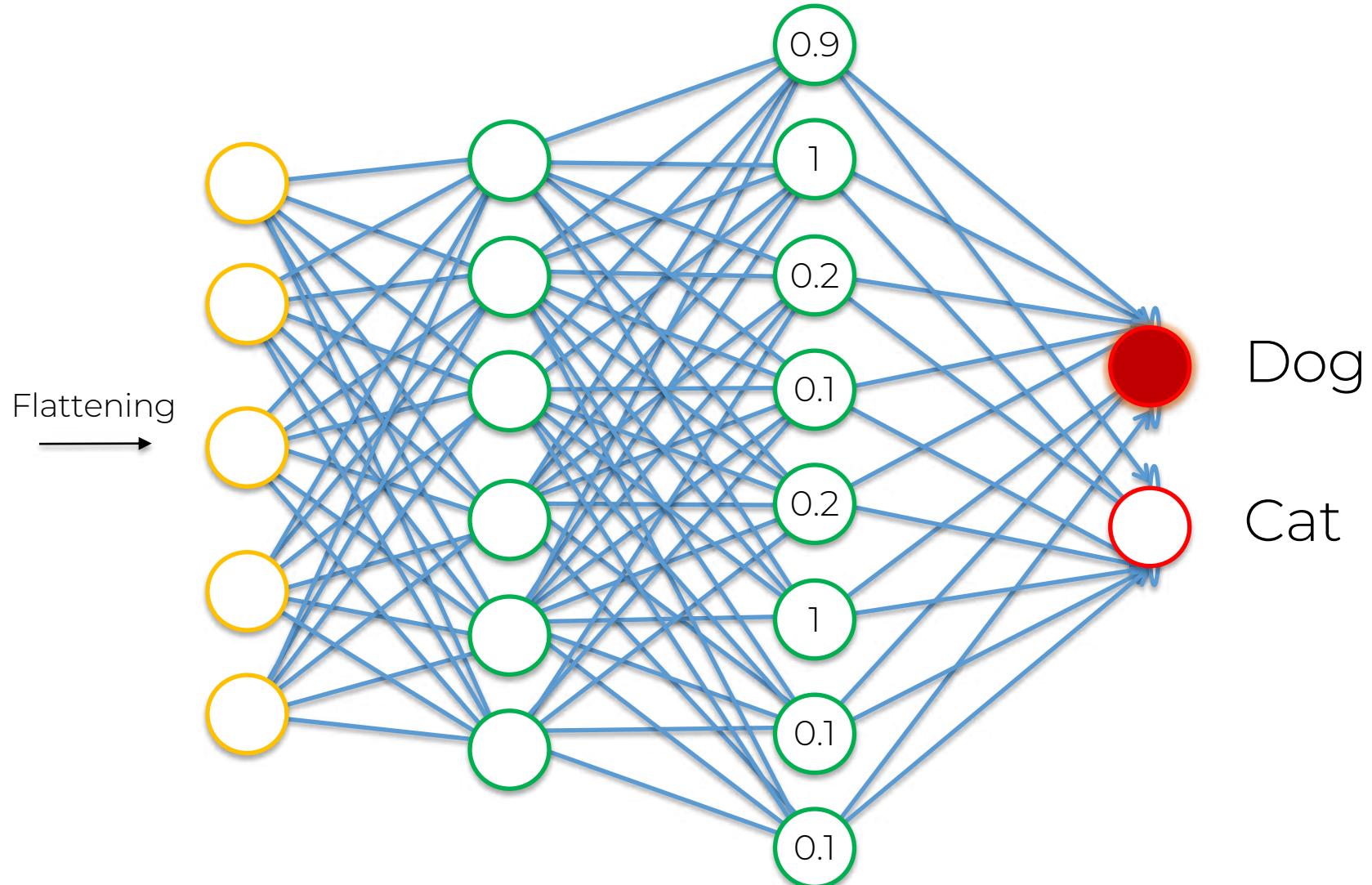
Step 4 - Full Connection



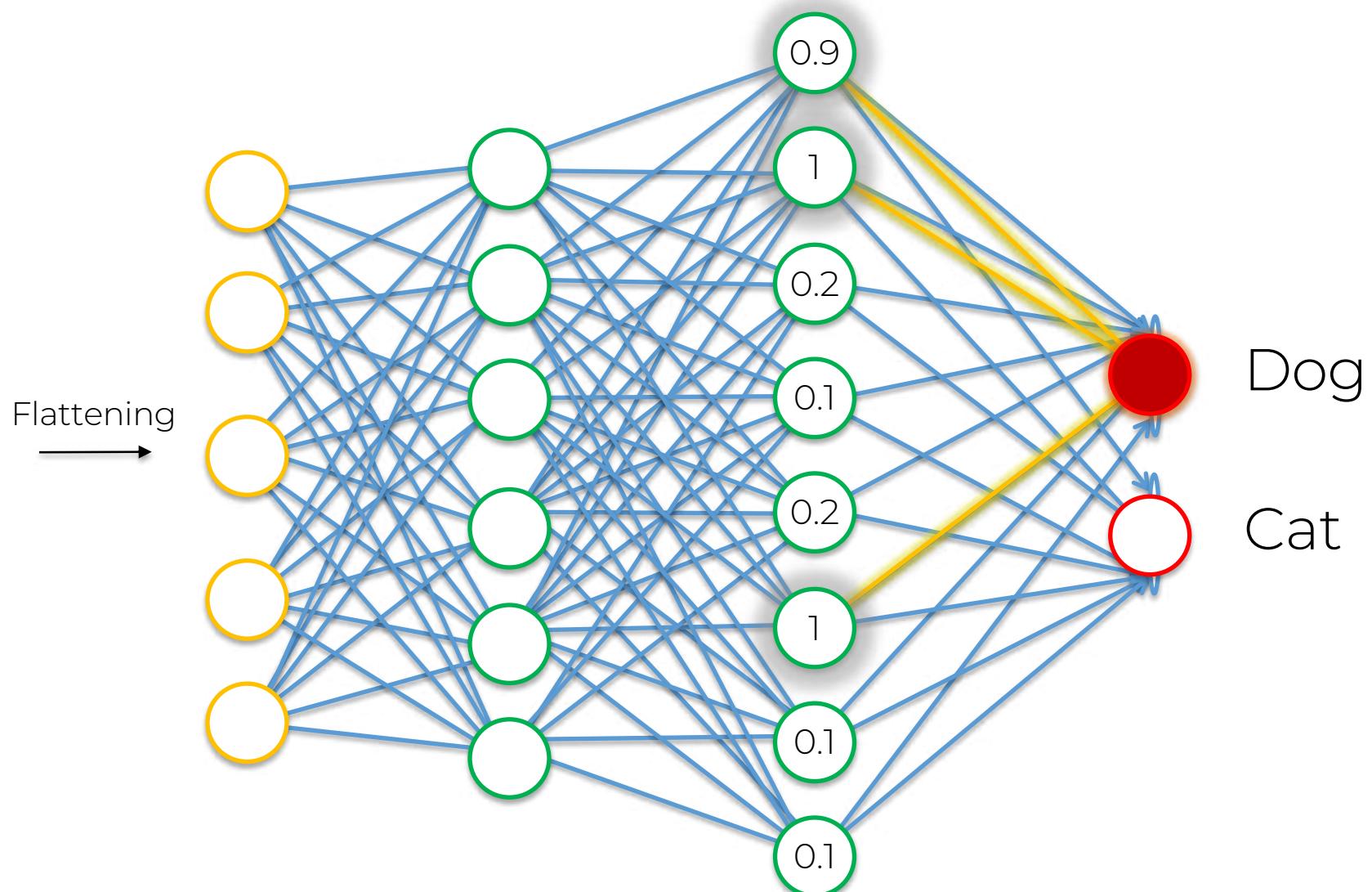
Step 4 - Full Connection



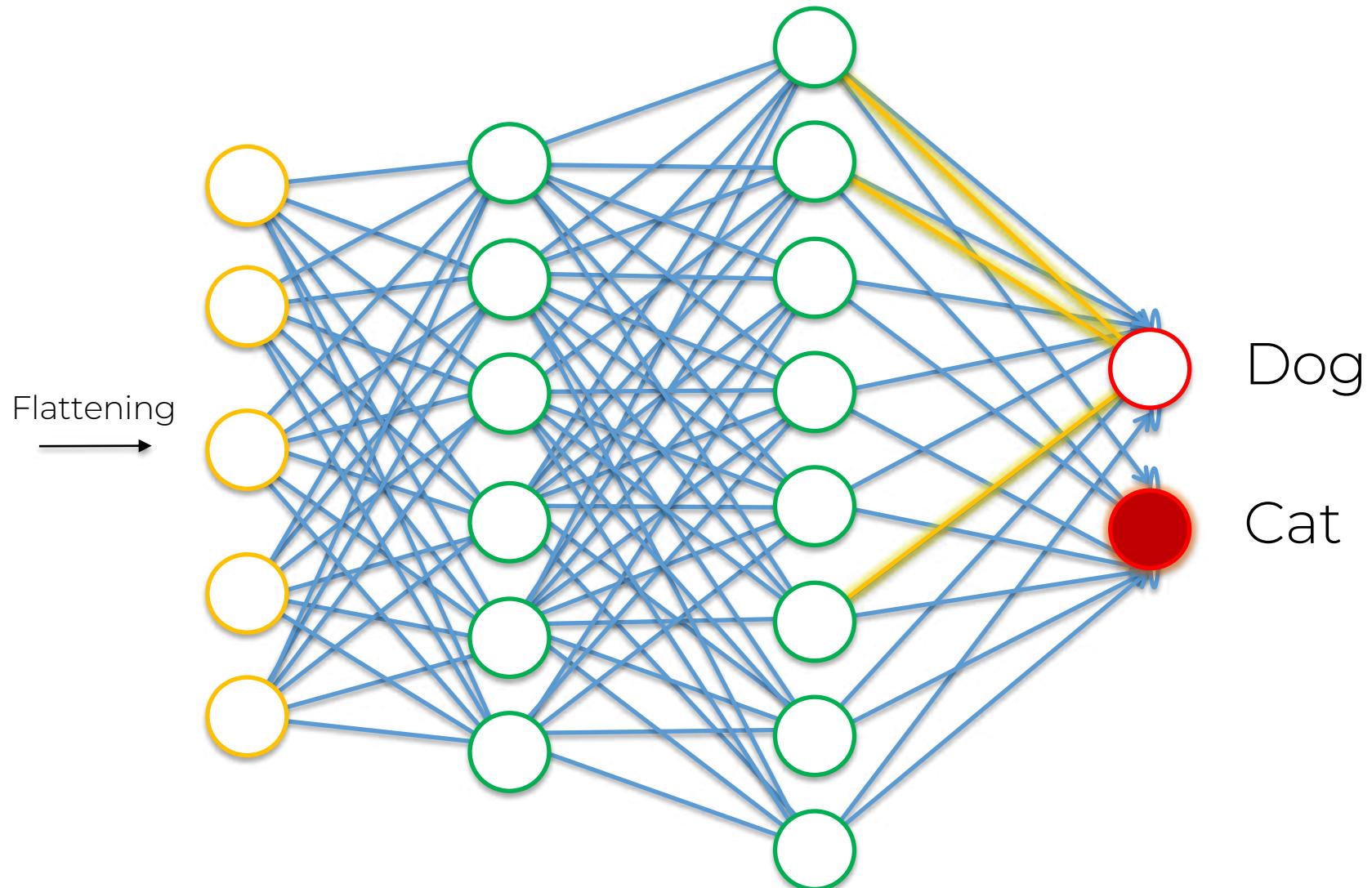
Step 4 - Full Connection



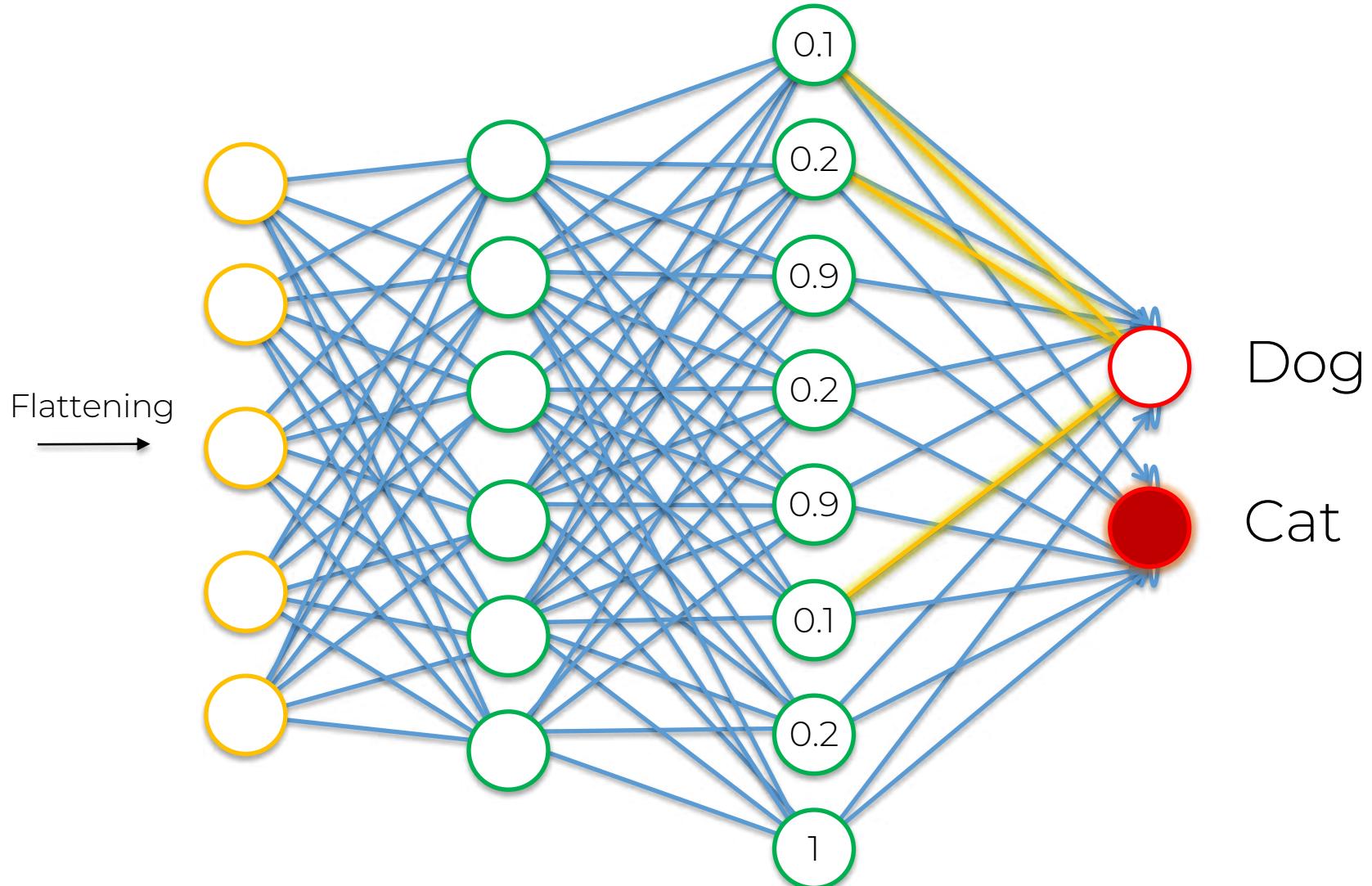
Step 4 - Full Connection



Step 4 - Full Connection

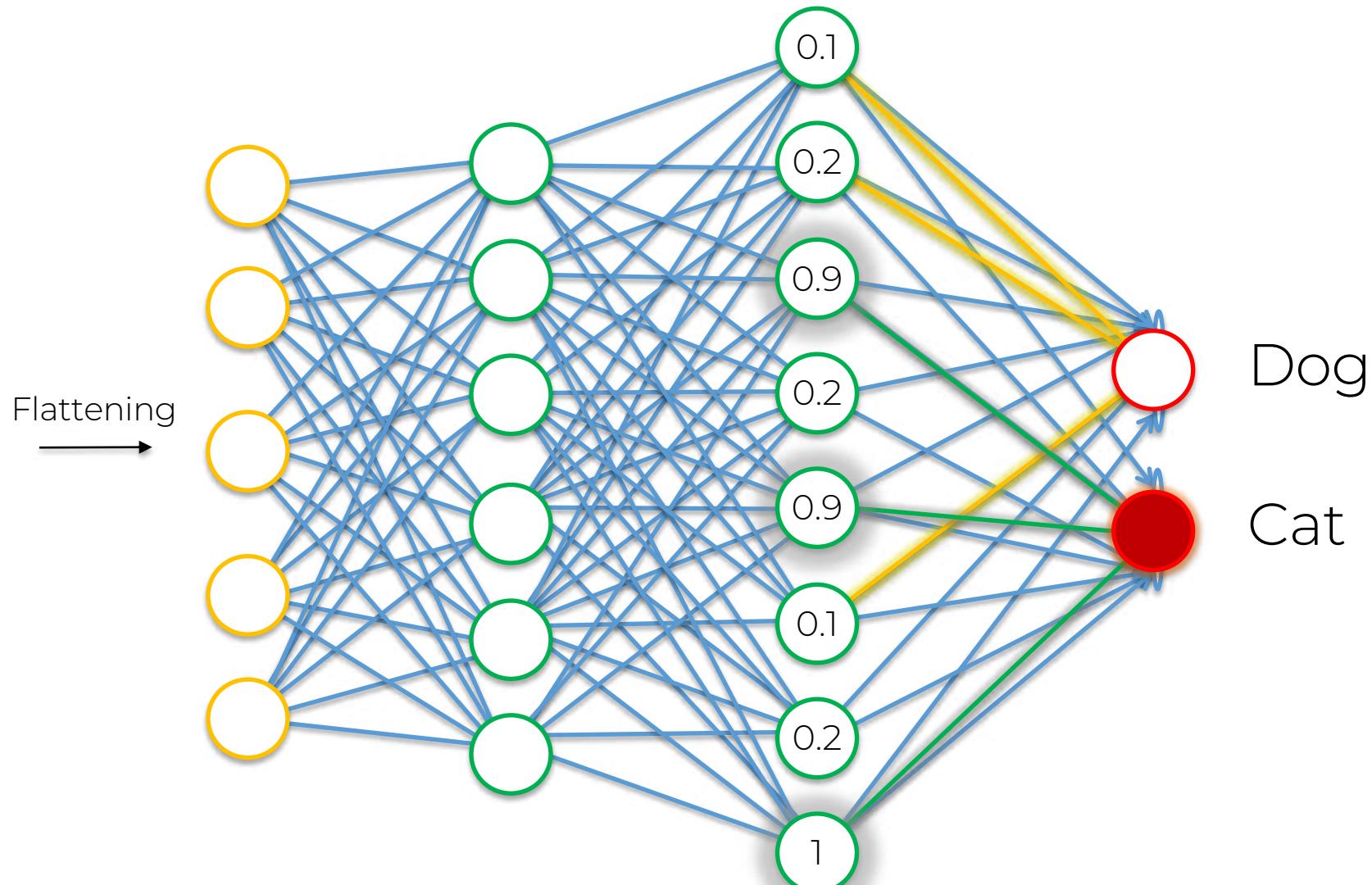


Step 4 - Full Connection



Step 4 - Full Connection

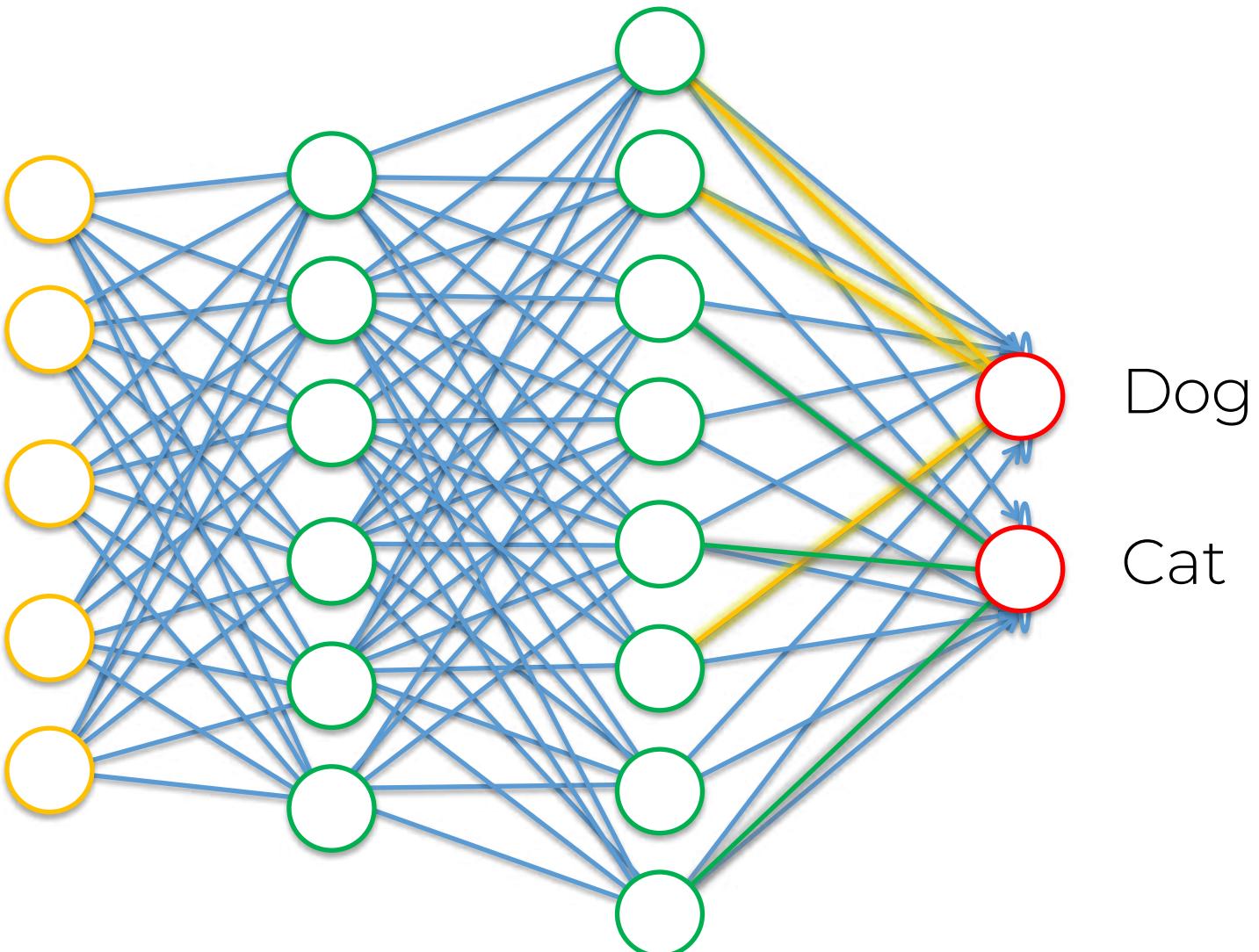
NOT FOR DISTRIBUTION @ SUPERDATASCIENCE www.superdatascience.com



Step 4 - Full Connection



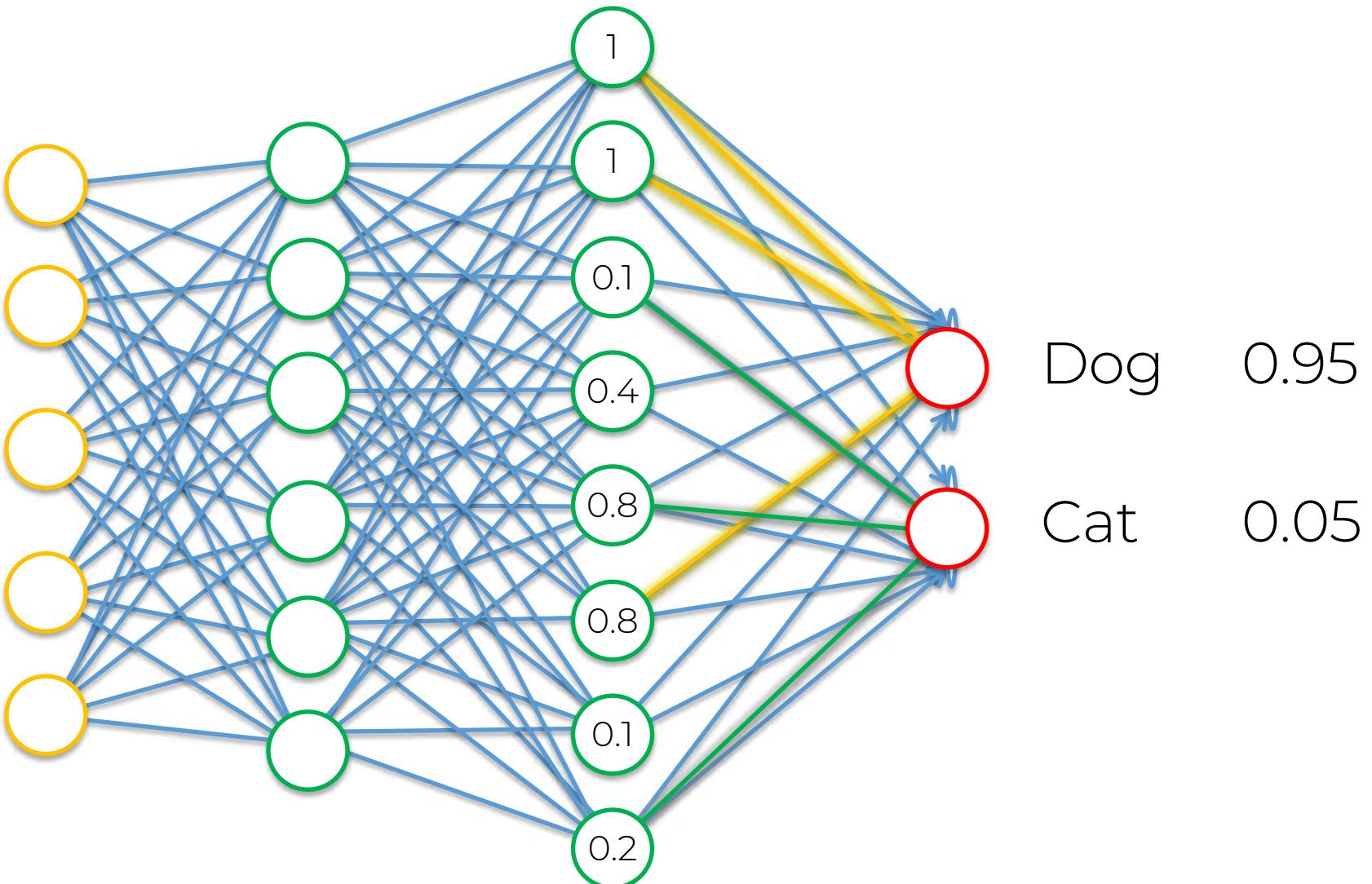
Flattening
→



Step 4 - Full Connection



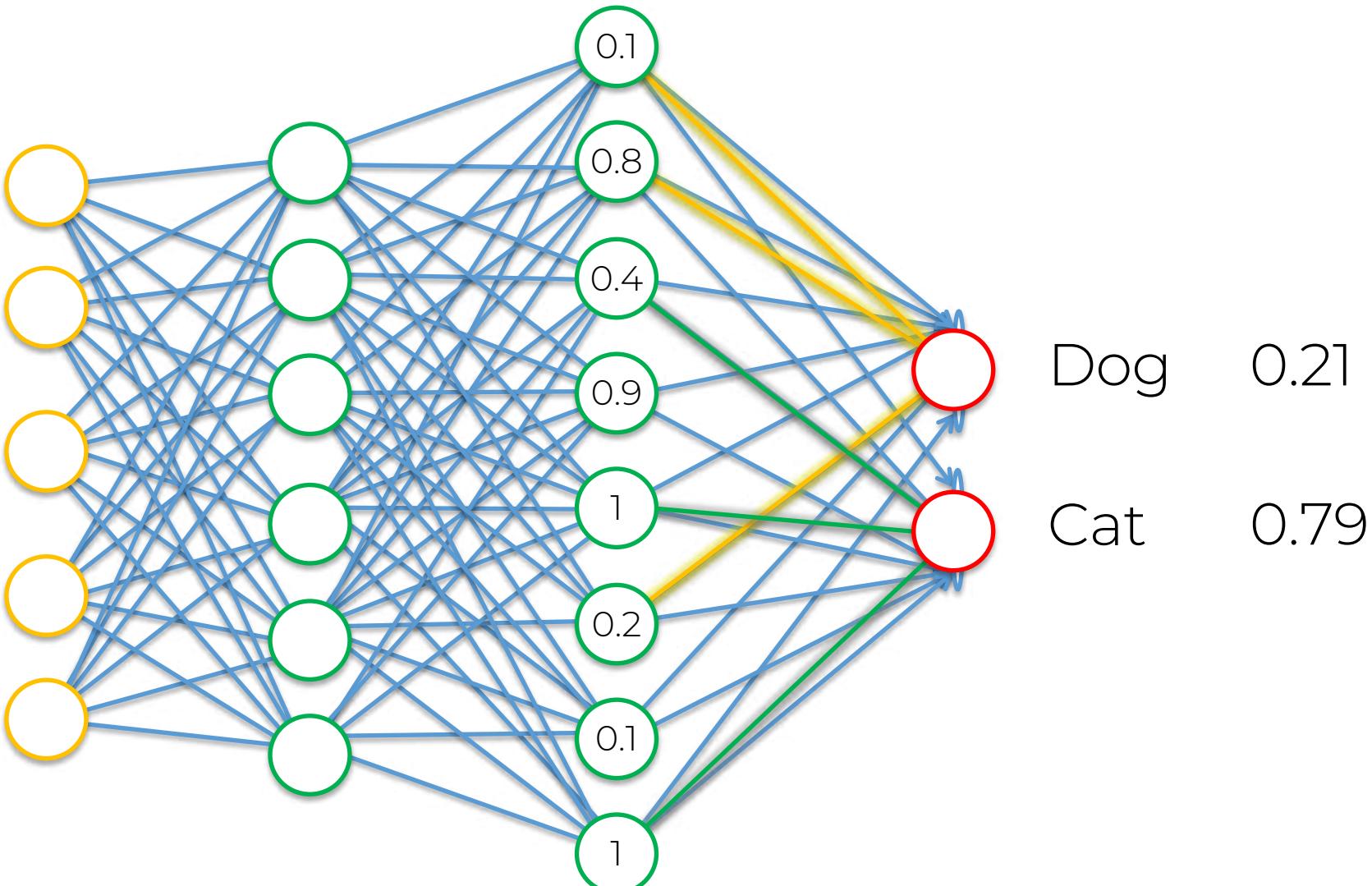
Flattening
→



Step 4 - Full Connection



Flattening
→



Step 4 - Full Connection

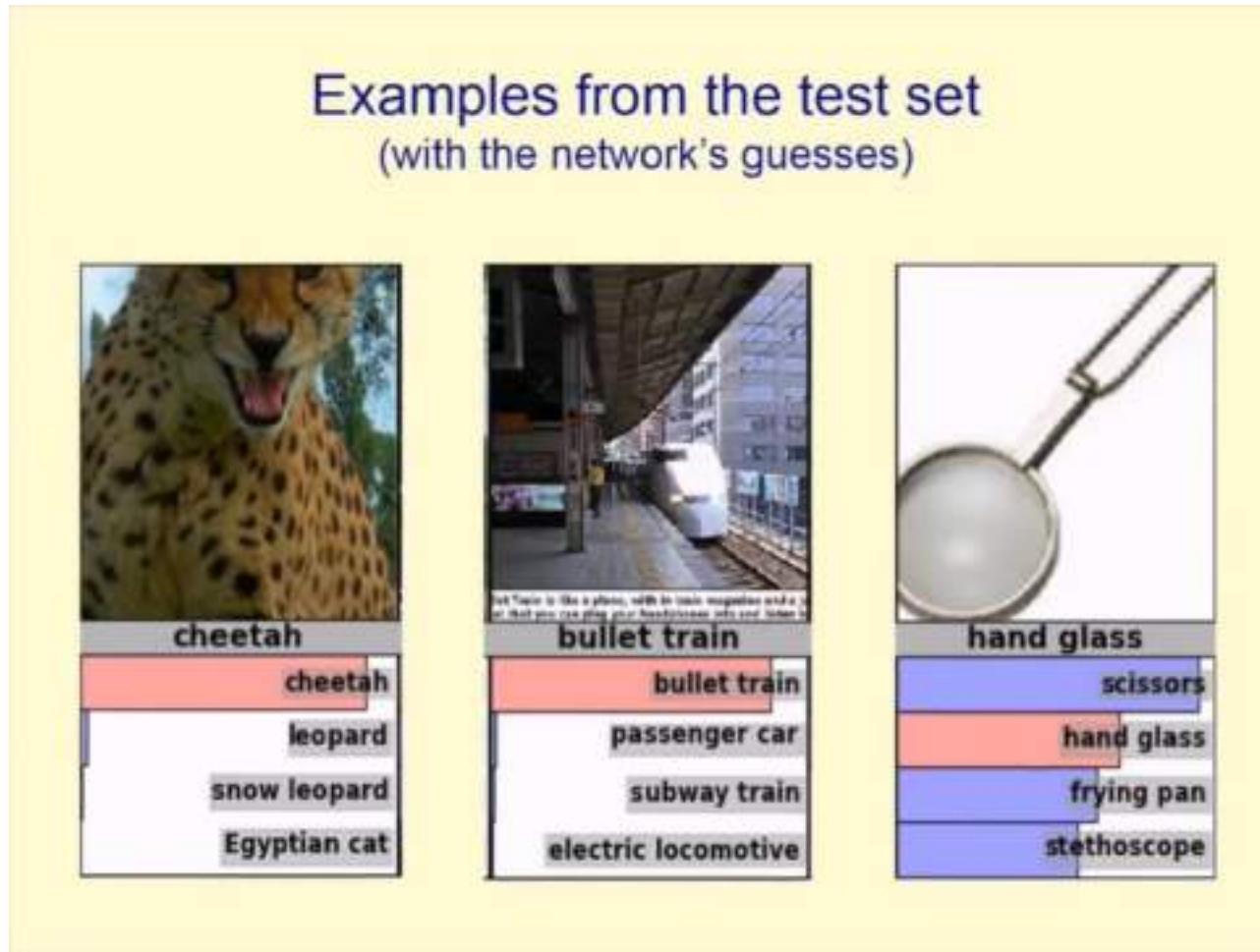
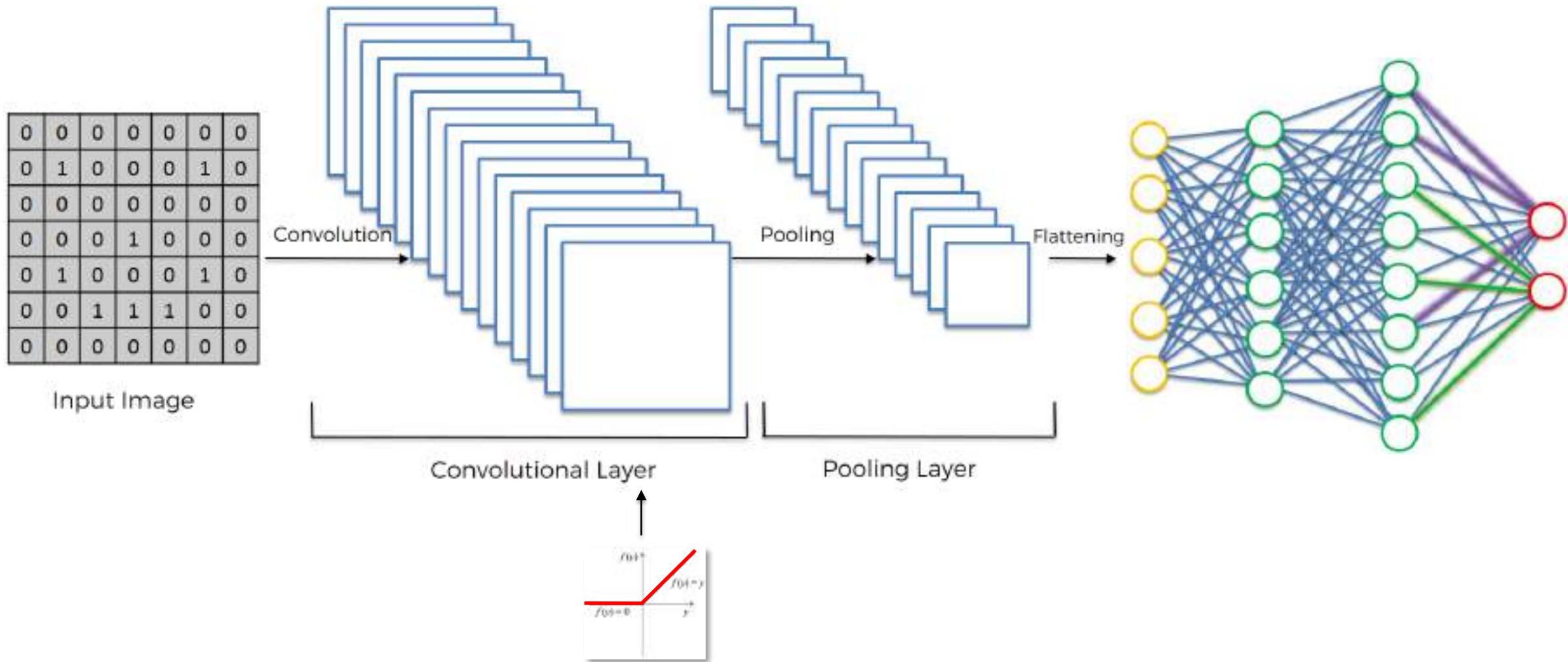


Image Source: a talk by Geoffrey Hinton

Summary

Summary



Summary

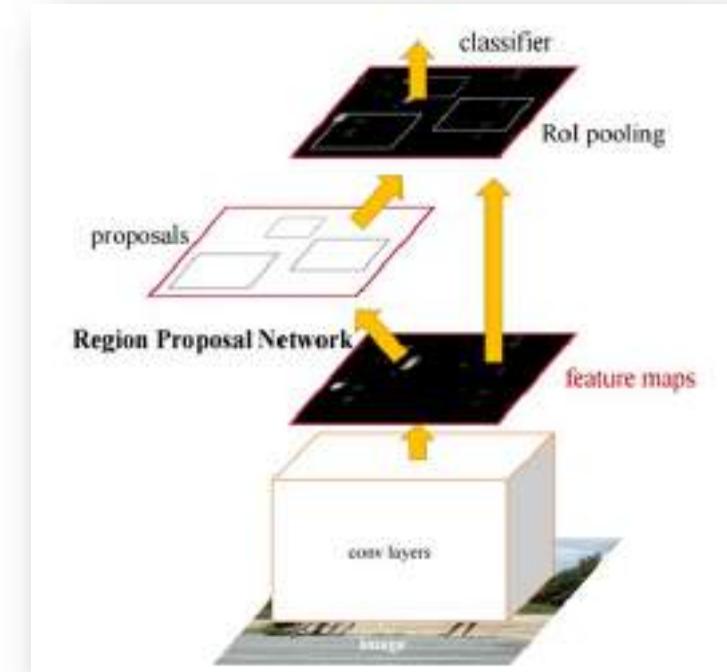
Additional Reading:

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

Adit Deshpande (2016)

Link:

<https://adethpande3.github.io/adethpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>

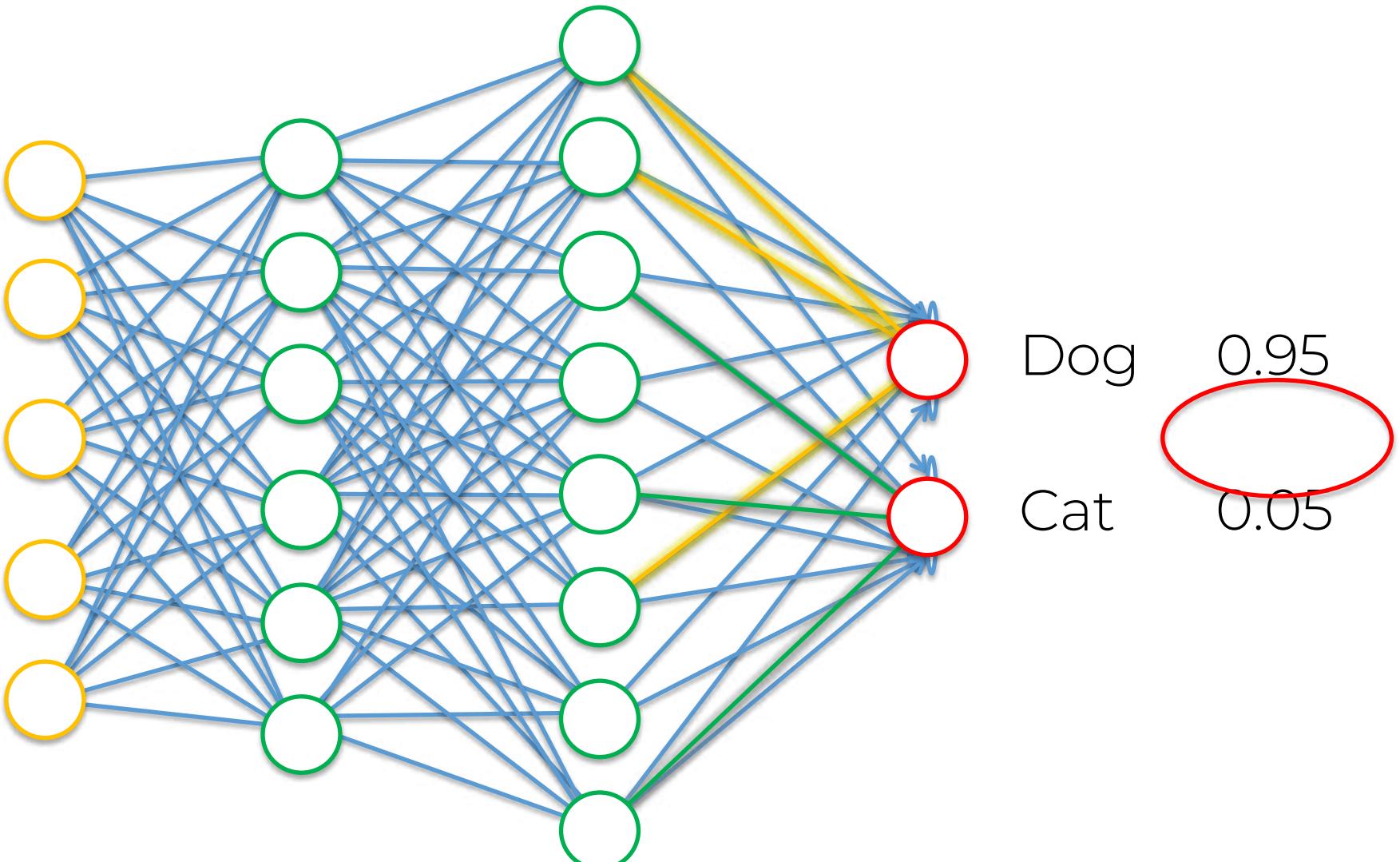


Softmax & Cross-Entropy

Softmax & Cross-Entropy



Flattening
→

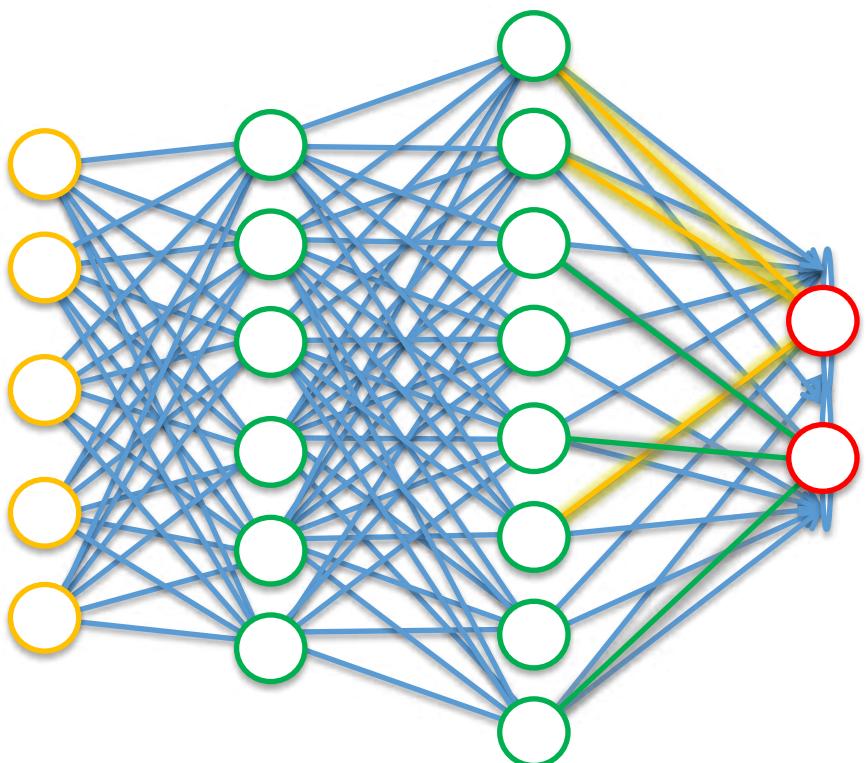


Softmax & Cross-Entropy



Dog

Flattenin
g



Dog \rightarrow $z_1 \rightarrow 0.95$
Cat \rightarrow $z_2 \rightarrow 0.05$

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



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

Cat

0.9

0.1

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

1
0

Softmax & Cross-Entropy

NN1 NN2



Dog	<table border="1"><tr><td>1</td></tr><tr><td>0</td></tr></table>	1	0
1			
0			
Cat			

0.9
0.1

0.6
0.4



Dog	<table border="1"><tr><td>0</td></tr><tr><td>1</td></tr></table>	0	1
0			
1			
Cat			

0.1
0.9

0.3
0.7



Dog	<table border="1"><tr><td>1</td></tr><tr><td>0</td></tr></table>	1	0
1			
0			
Cat			

0.4
0.6

0.1
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

Classification Error

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

Mean Squared Error

0.25

0.71

Cross-Entropy

0.38

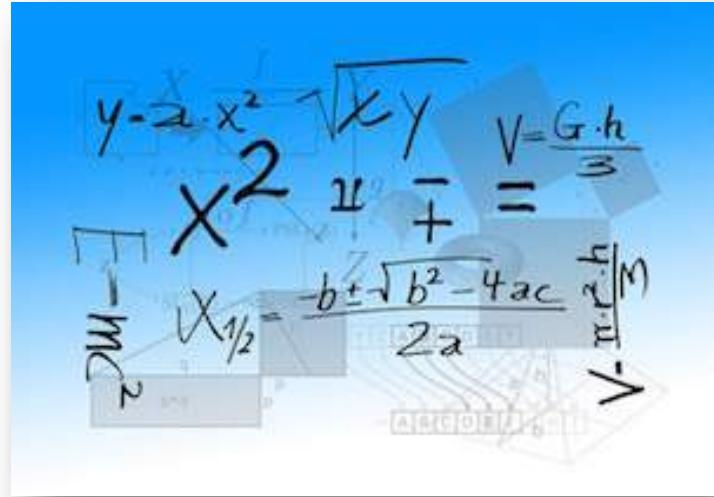
1.06

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/