AI in Built Environment
DCP4300

Lec07-08: Deep Learning

Part B

Dr. Chaofeng Wang
Jianhao Gao (TA)

University of Florida
College of Design Construction and Planning

Outline:

0. Recap

1. Python

2. ML algorithms and demos

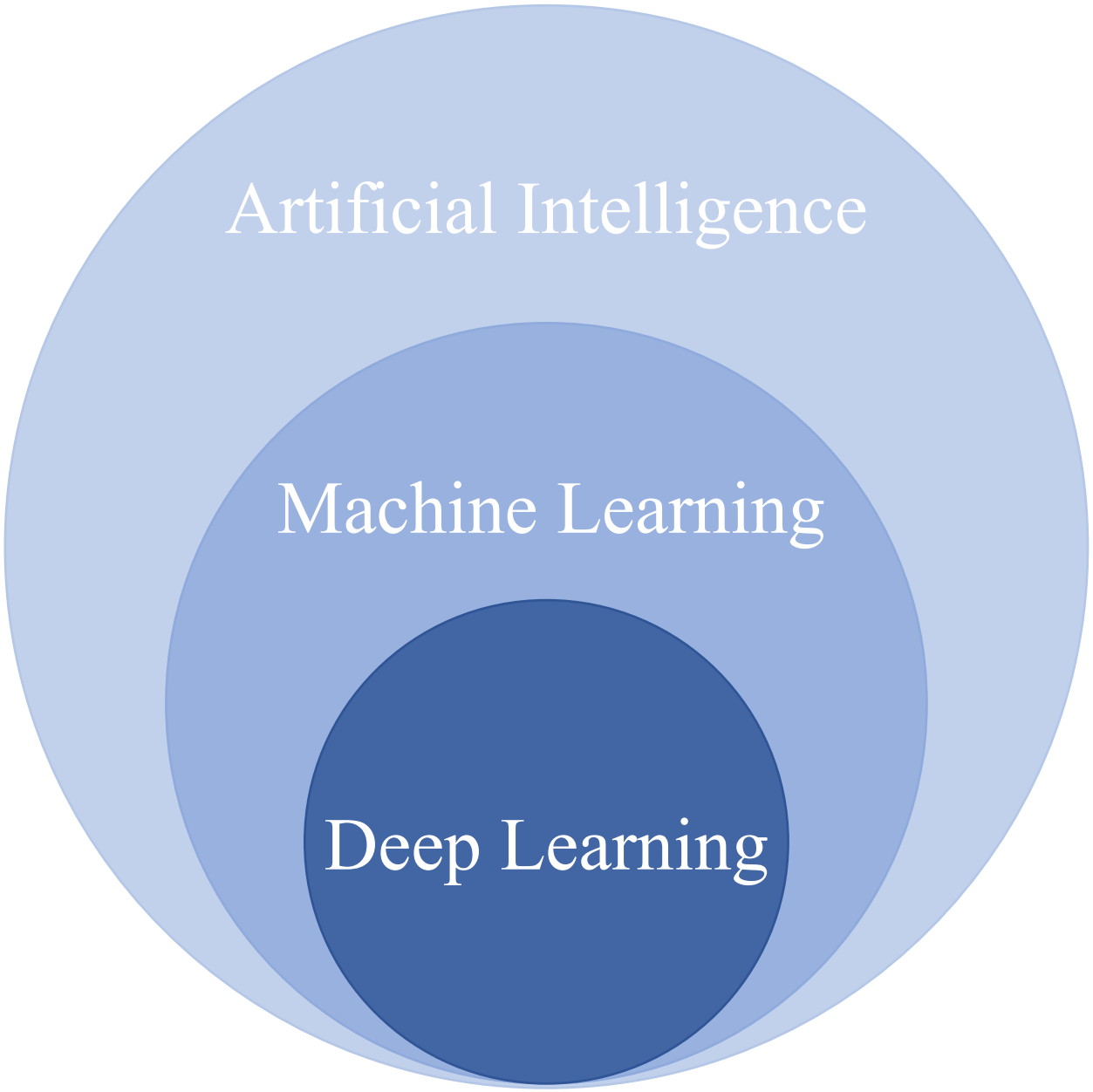
3. Build a neural network in TensorFlow / PyTorch (Part C)

You can install Python and Jupyter notebook on your computer,
or use Google Colab in the browser instead, which needs your Google account.

For deep learning demonstrations, we'll always use Google Colab.

0

Recap



AI: Techniques that enable machines to mimic human.

ML: Techniques that enable machines to **learn from data**, **without being explicitly programmed**.

DL: Techniques that enable machines to learn from data, **hierarchically**, using **neural networks**.

Types of Machine Learning:

Supervised Learning: Learn a function from **labeled** data.

- Classification
- Regression

Semi-supervised Learning

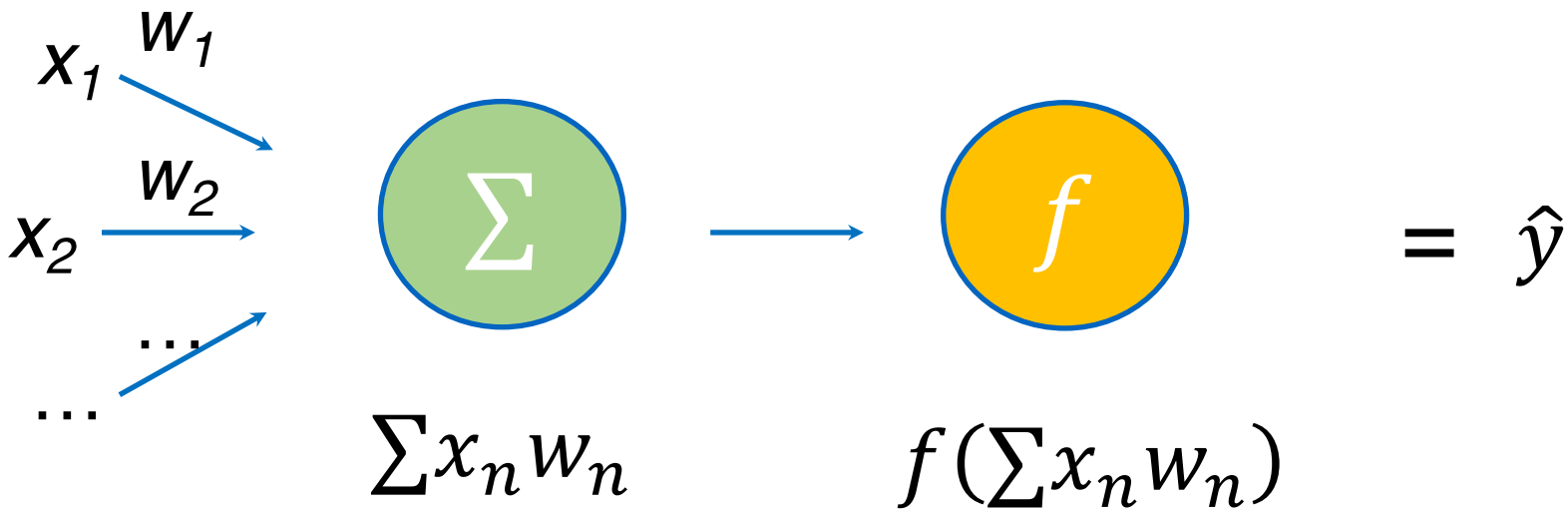
Unsupervised Learning: Learn the pattern from **unlabeled** data.

- Clustering
- Dimension reduction

Reinforcement Learning: Learn to react to an environment by **trial and error**.

- Decision making
- Robotics
- …

What does a perceptron do?



Sum and Activate

Neural network (Multilayer perceptron)

Weights

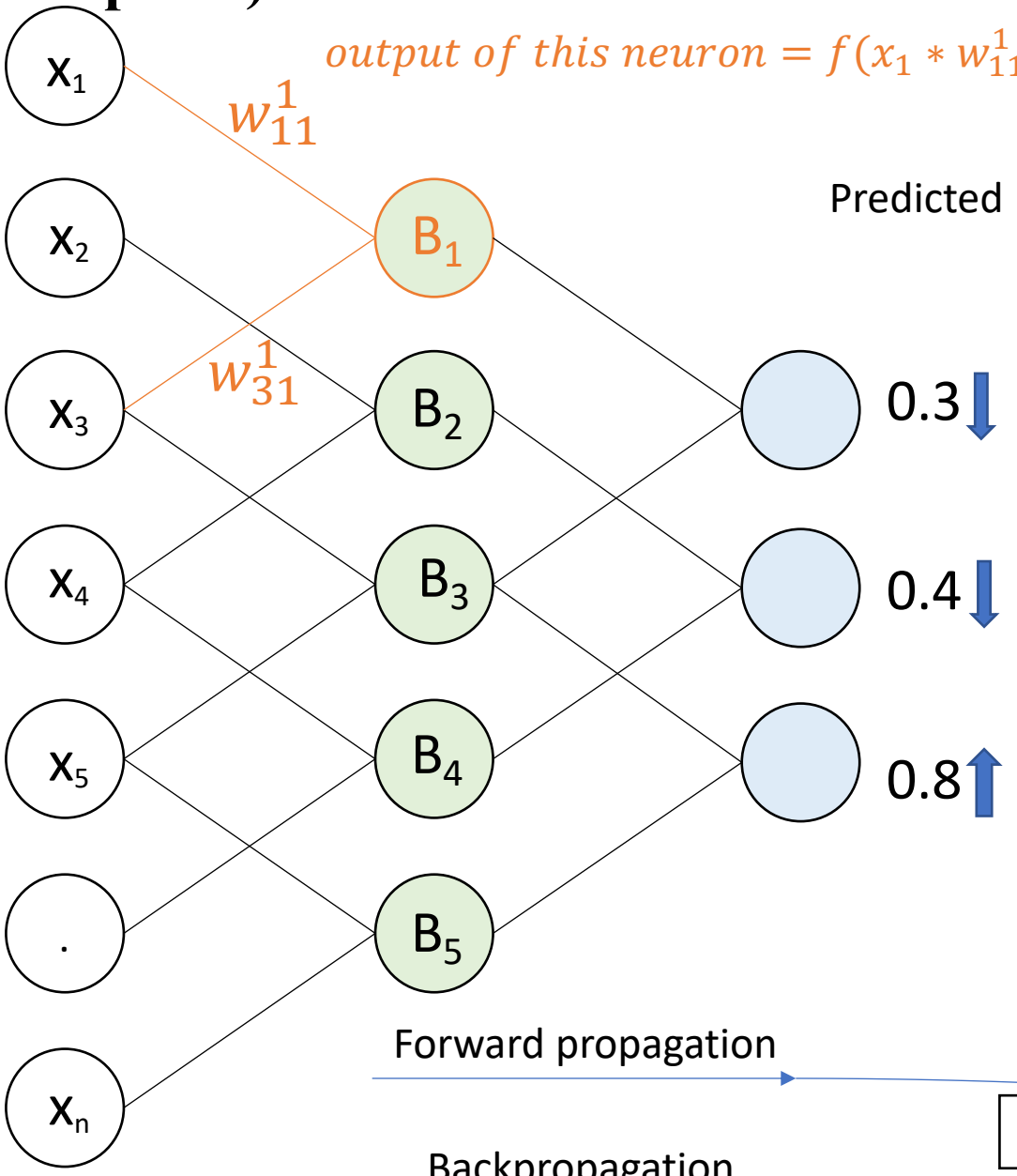
The weight linking the 1st hidden layer and its prior layer

w_{31}^{1}

Neuron 3 from previous layer

Neuron 1 in the current layer

$$output\ of\ this\ neuron = f(x_1 * w_{11}^1 + x_3 * w_{31}^1 + B_1)$$



True Value (target)	Error
0	-0.3
0	-0.4
1	0.2

Learning rate

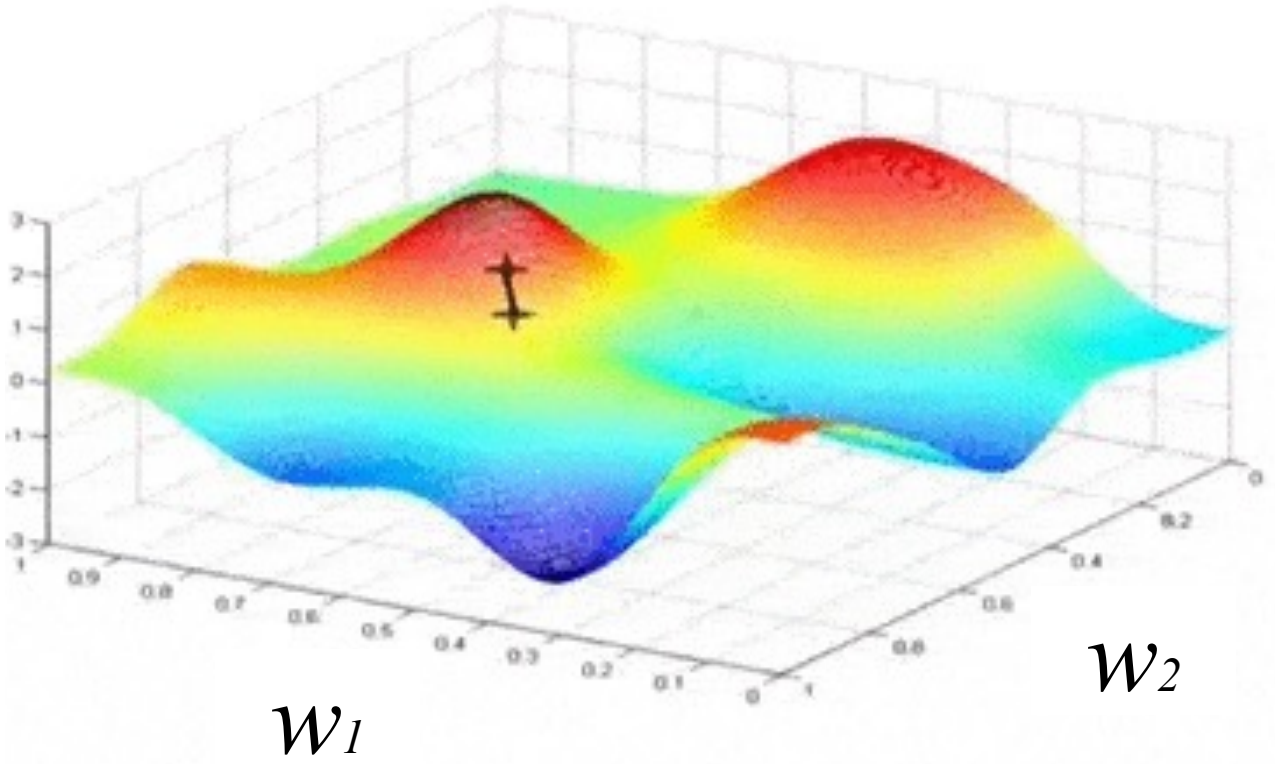
Gradient descent:

1. Compute the slope (gradient) at the current step $\frac{\partial J}{\partial w}$

2. Make a move in the direction opposite to the slope

The move = $-\eta \frac{\partial J}{\partial w}$

Learning rate



At the end, what is learned?

The weights: W

1

Python

Based on
https://www.w3schools.com/python/python_getstarted.asp

Run codes on your computer

1.Install Python: https://www.python.org/downloads/

2.Install Jupyter: Run this in the command line

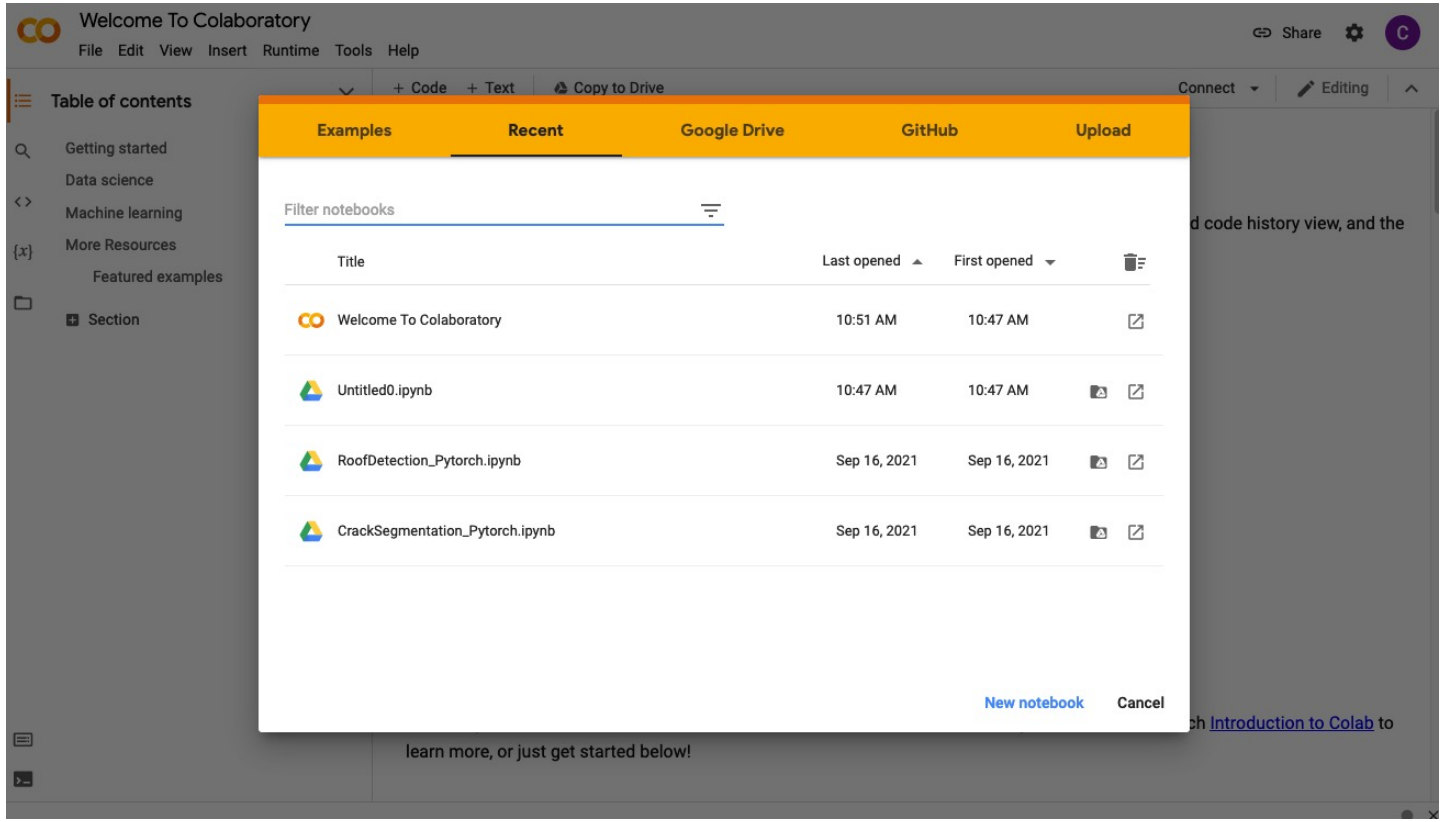
```
pip3 install --upgrade pip
pip3 install jupyter
```

3.How to use Jupyter: https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html

Run codes in the cloud: Google Colab

Google Colab:

1. First you need a Google Account
2. Go to https://colab.research.google.com
3. Create a New notebook

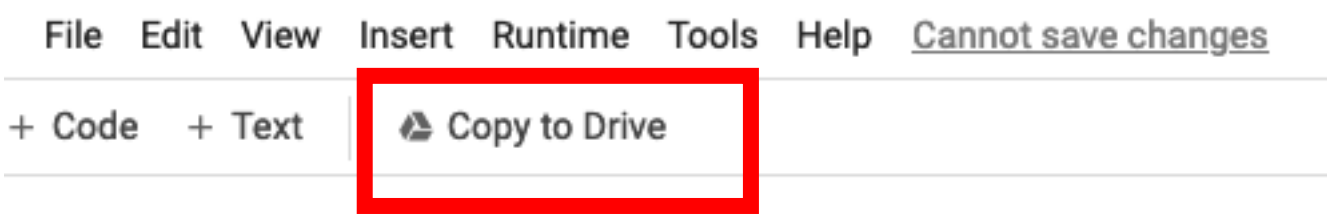


Run codes in the cloud: Google Colab

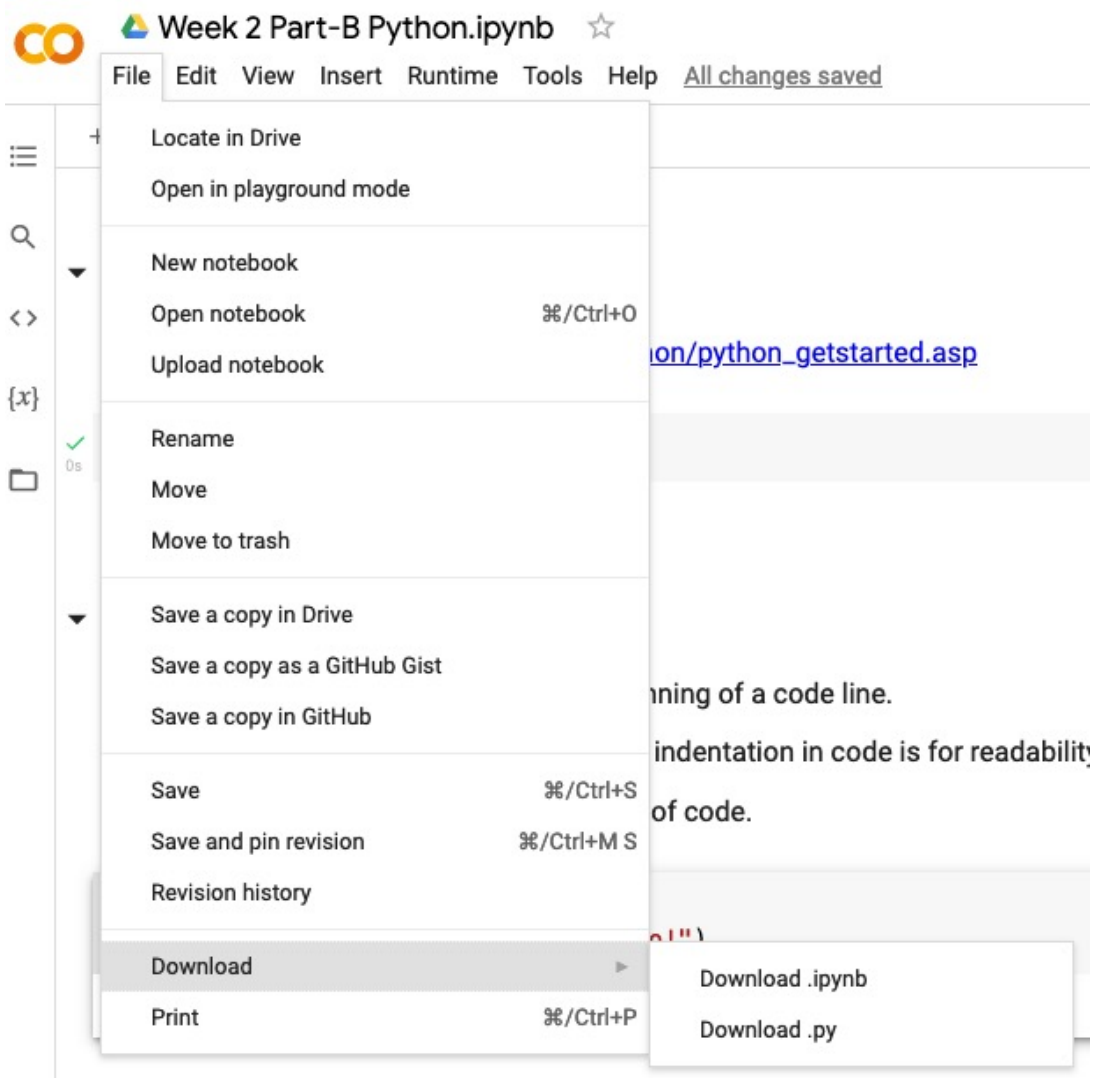
Now we run the Python tutorial in a Jupyter notebook that is hosted on Google Colab:

https://colab.research.google.com/drive/1FsaIHg4G91zNJWarShbd_HrzBnKYL7c5?usp=sharing

Run codes in the cloud: Google Colab



You can also download the notebook from Colab



Run codes on your computer (more than two ways…)

1

*.py file:

contains Python code, mainly for the machine to execute

To execute a py file on your computer, go to the command line, and
run the command in a folder that contains your file:
python filename.py

2.

*.ipynb file:

a notebook, for you to write and run Python code, and visualize results interactively

If you run ‘jupyter notebook’ in the command line,
Jupyter will open in your browser

2

2. ML algorithms and demos

Common algorithms in **Supervised Machine Learning**

Regression
Decision tree / Random forest
K-nearest neighbor
Support vector machines
Multilayer perceptron (Neural network)
…

Keywords:
Data Model Training

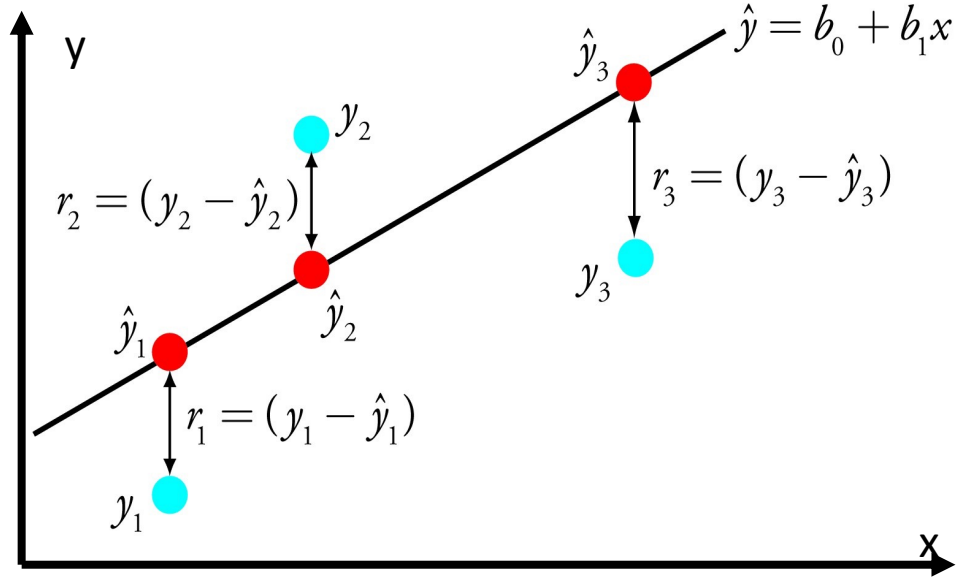
Basic frame:
Use data to set the parameters of a model to fit the labels.

Linear Regression

x	y
0.1	0.05
0.2	0.3
0.4	0.2
x_i	?

Fit the relation by a **linear function**:

$$y = \boldsymbol{X\beta} + \epsilon$$



Math:

Find coefficient $\boldsymbol{\beta}$ *and error* ϵ for

$y = \boldsymbol{X\beta} + \epsilon$

that minimize the residual:

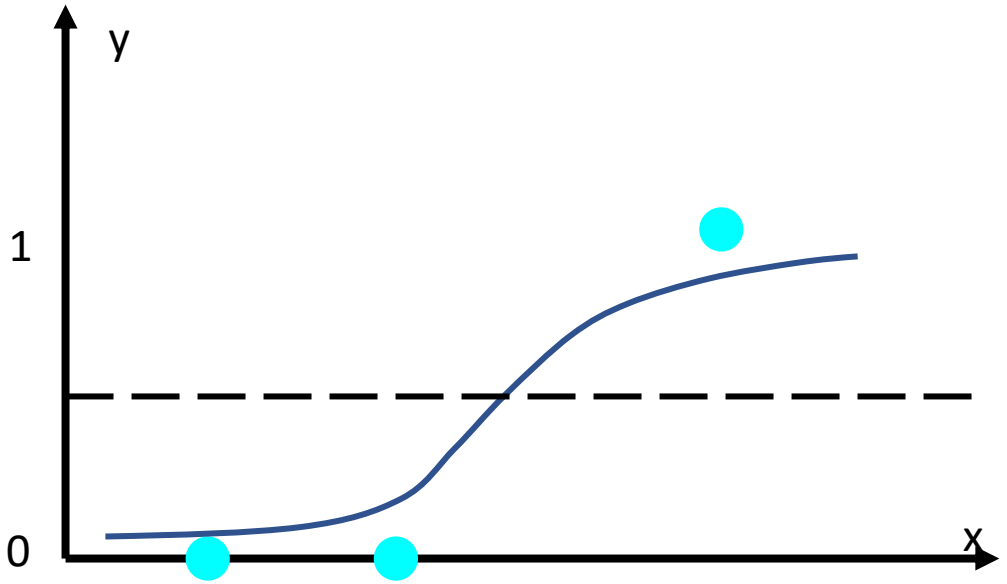
$$\mathcal{R} = \sum_1^n r_i^2$$

Logistic Regression

x	y
0.1	0
0.2	0
0.4	1
x_i	?

Fit the relation by a **logistic function**:

$$P(y = 1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$



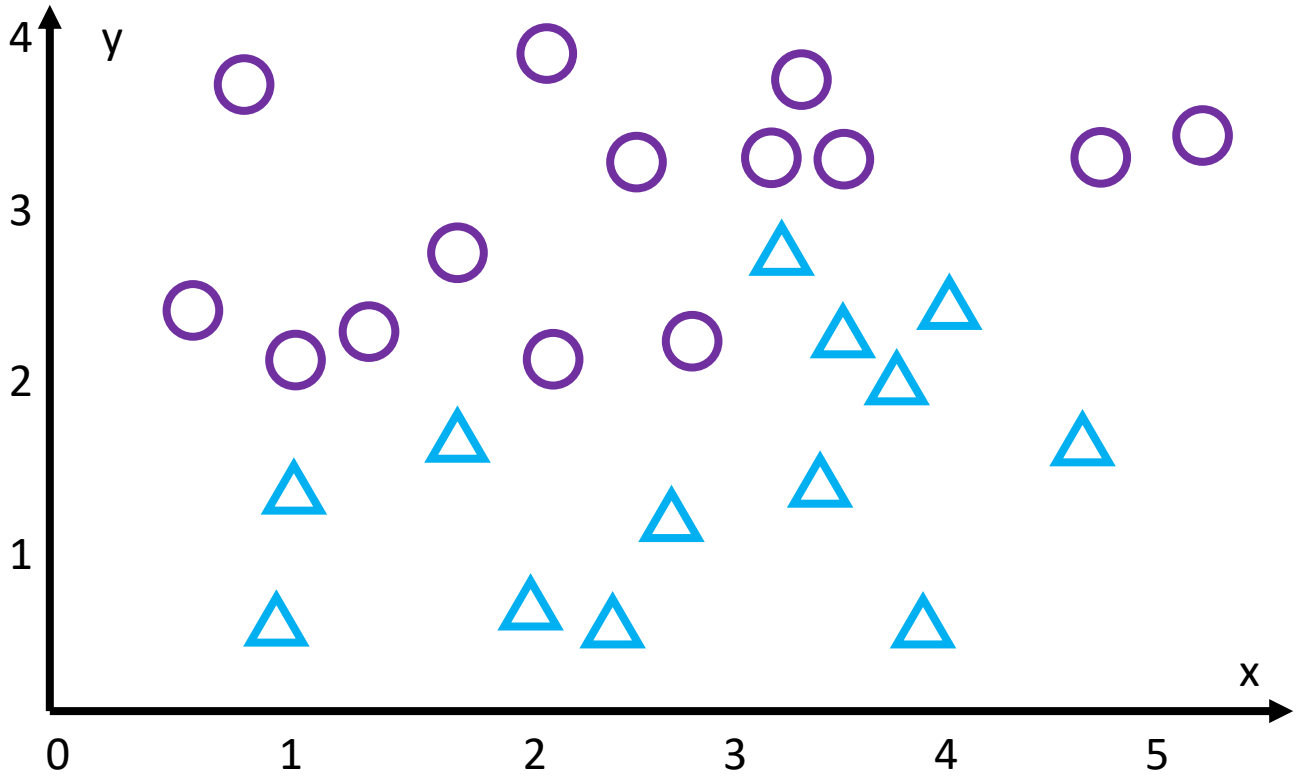
Math:

Find coefficient $\boldsymbol{\beta}$ for

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}$$

that minimize the residual between the prediction and the ground truth

Decision Tree

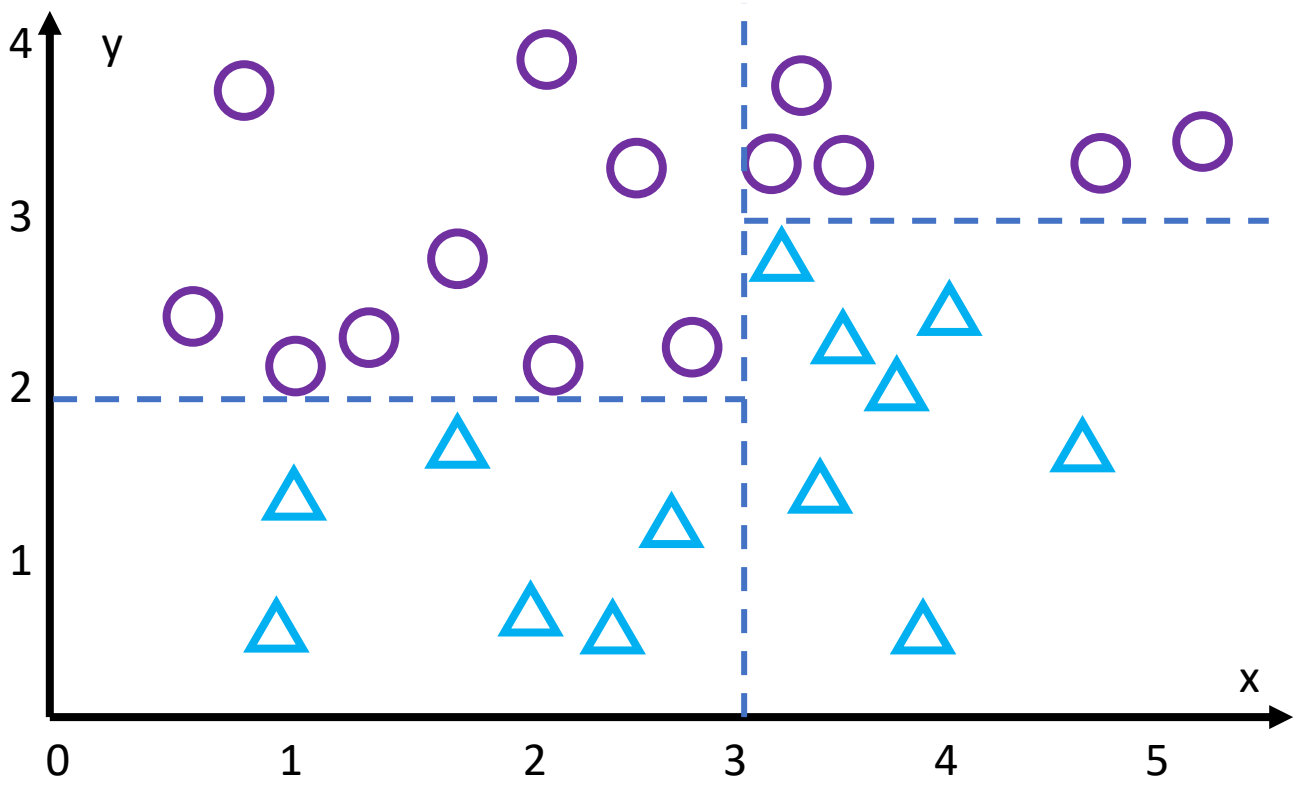


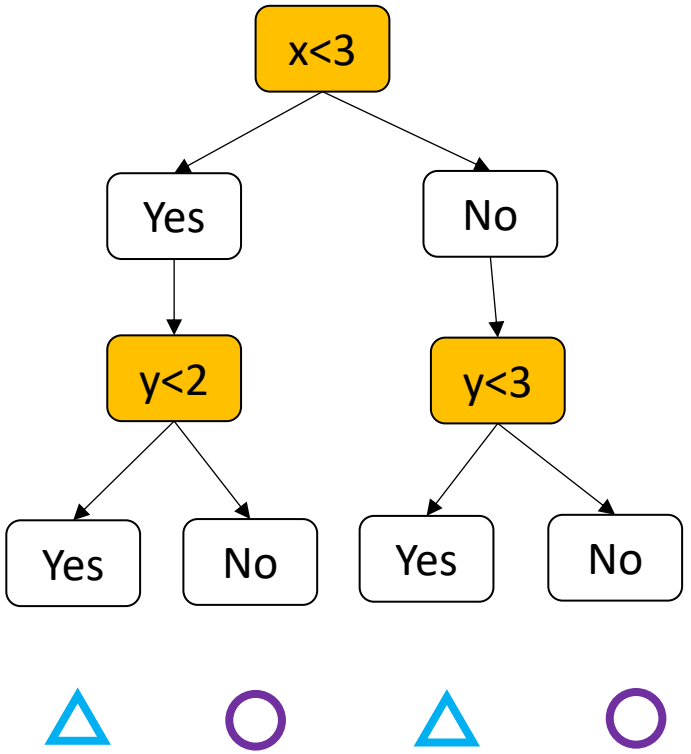
x and y are called features, each data point can be represented by features:

○ (x, y)

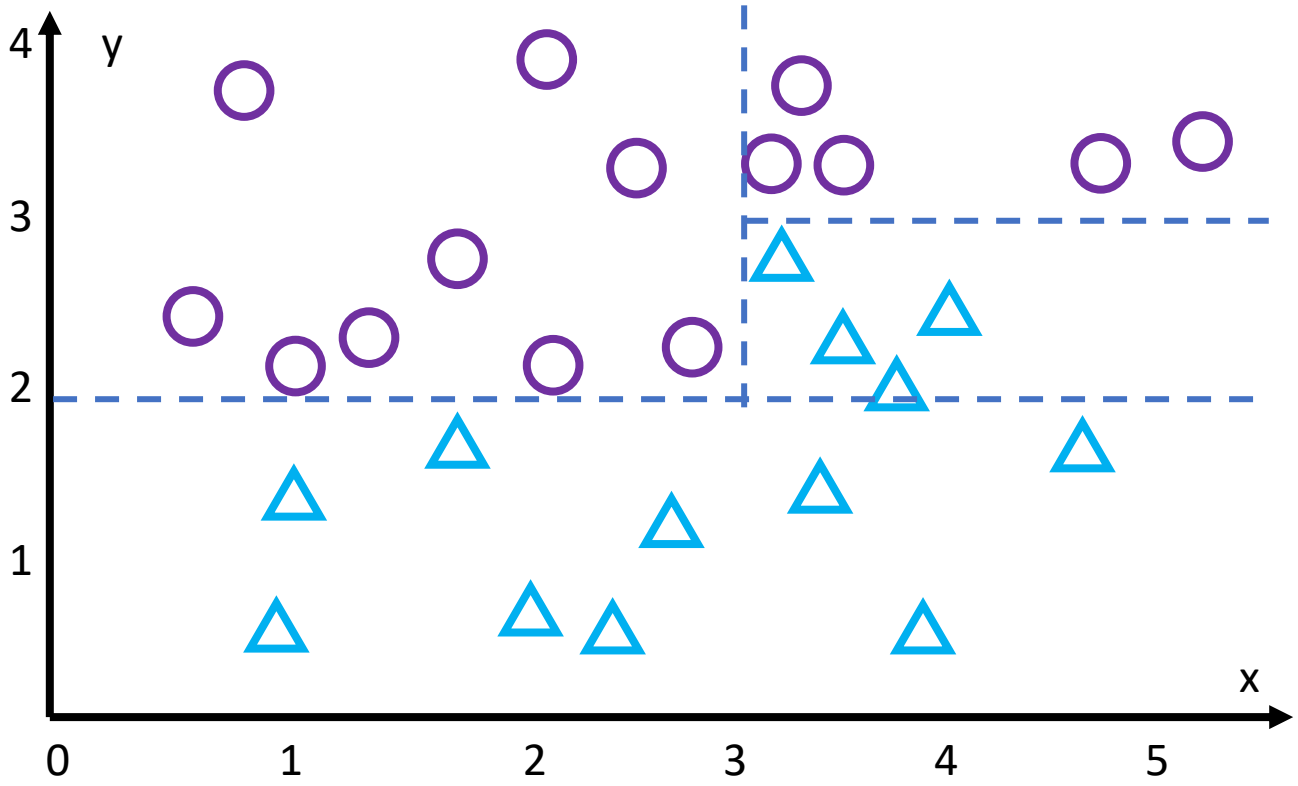
△ (x, y)

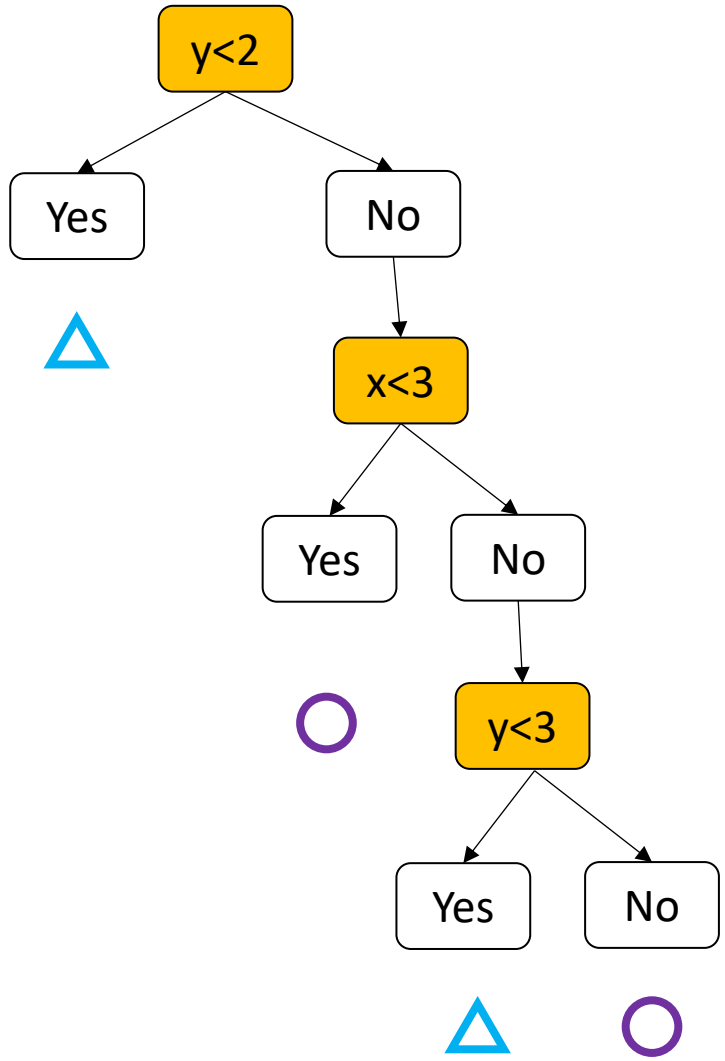
Decision Tree





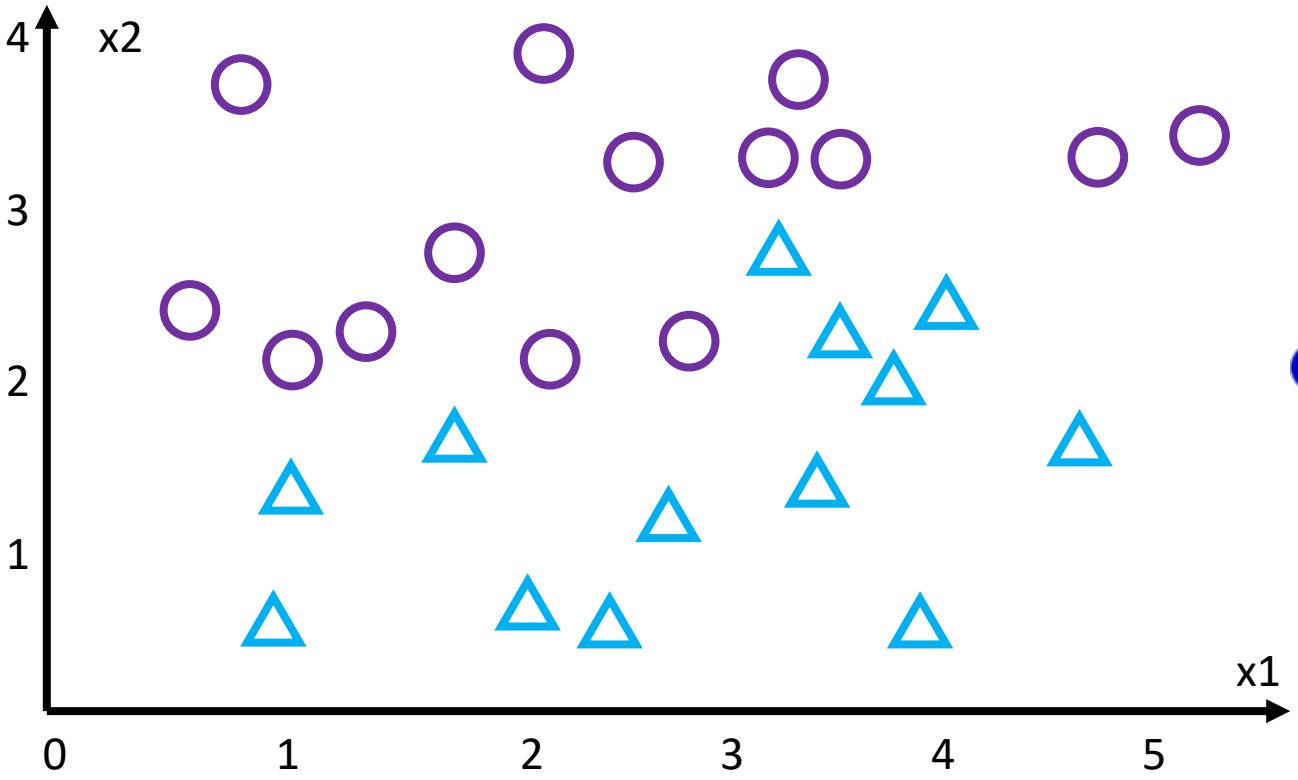
Decision Tree

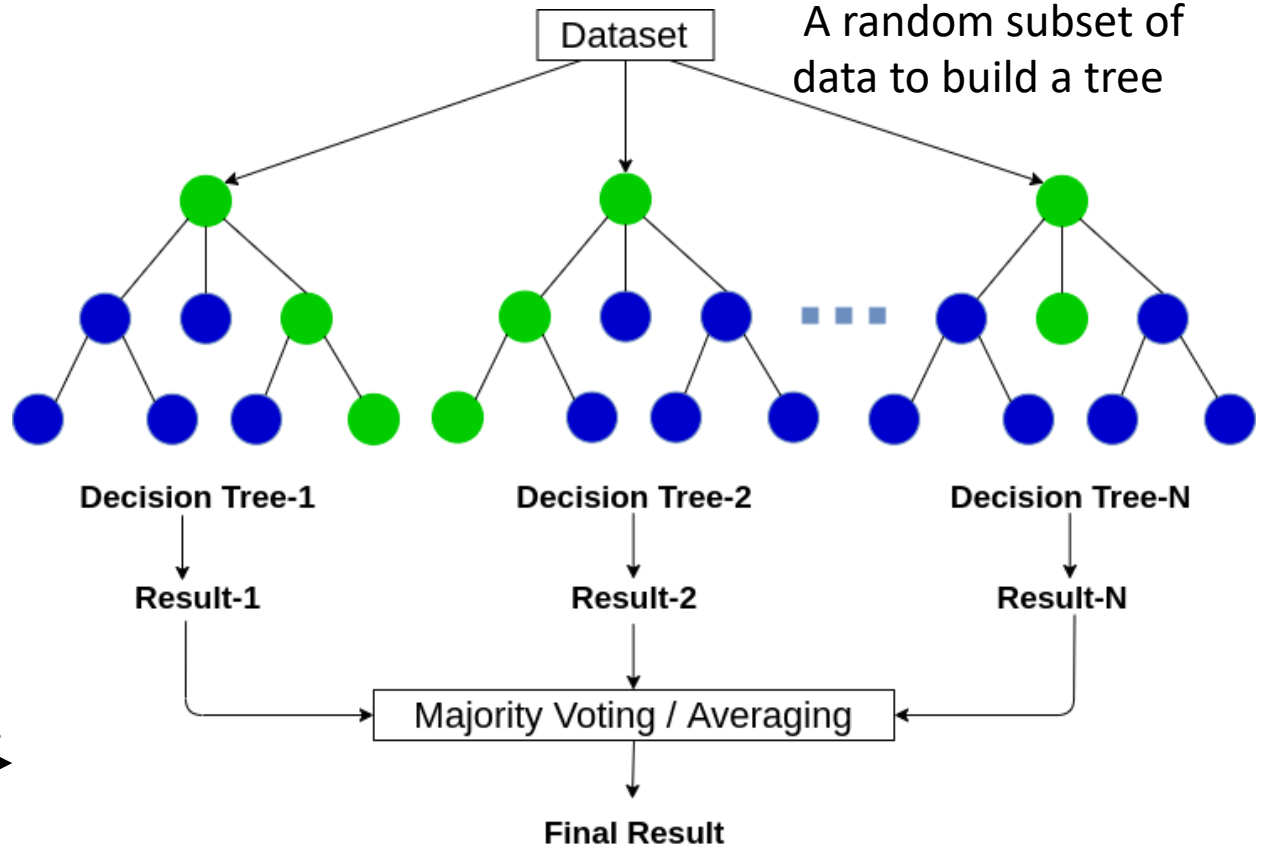




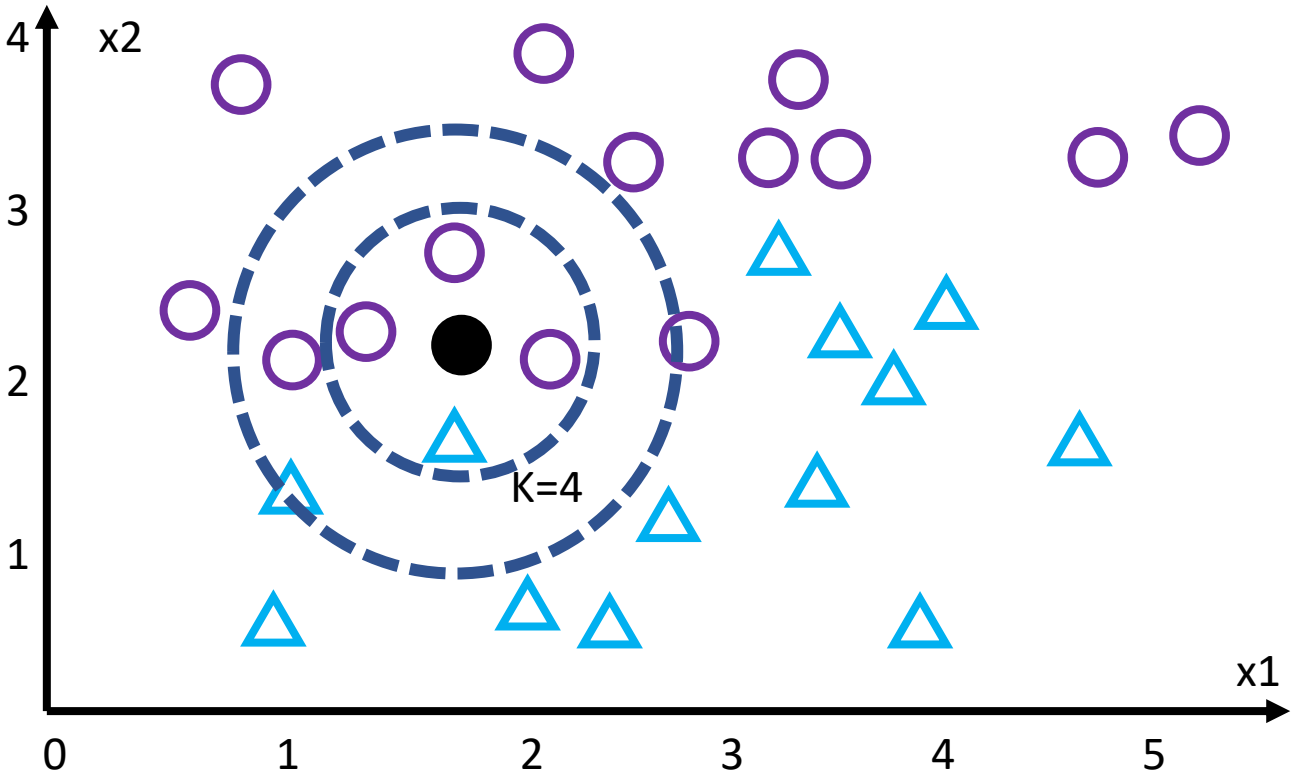
Another tree

Random Forest

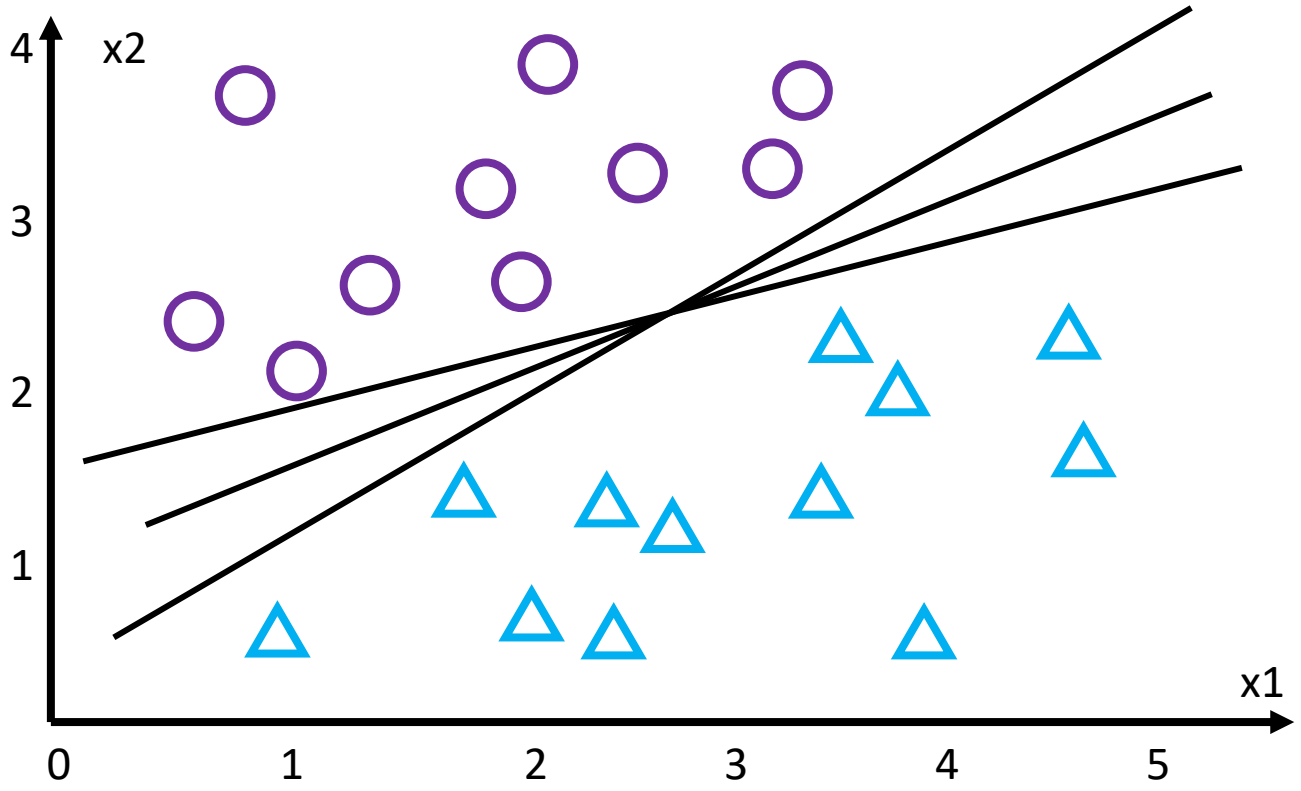




K-nearest neighbor

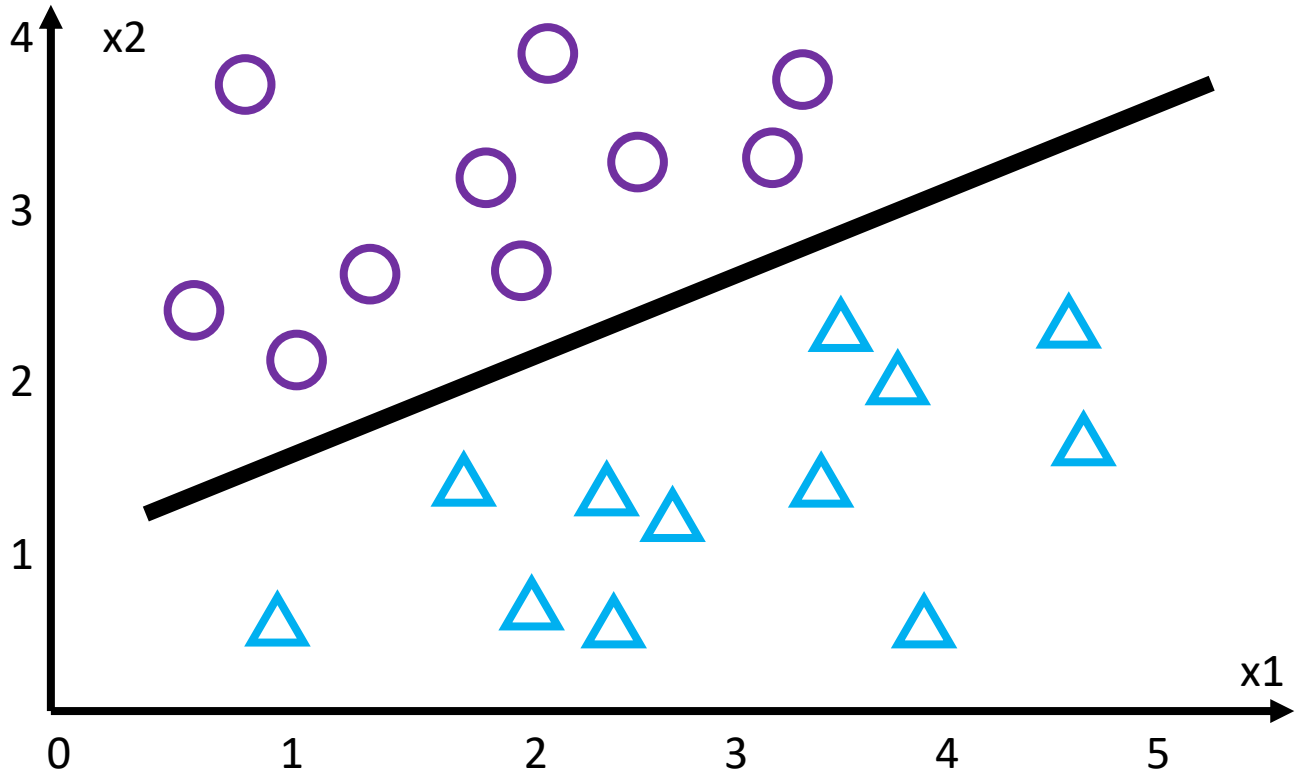


Support vector machines

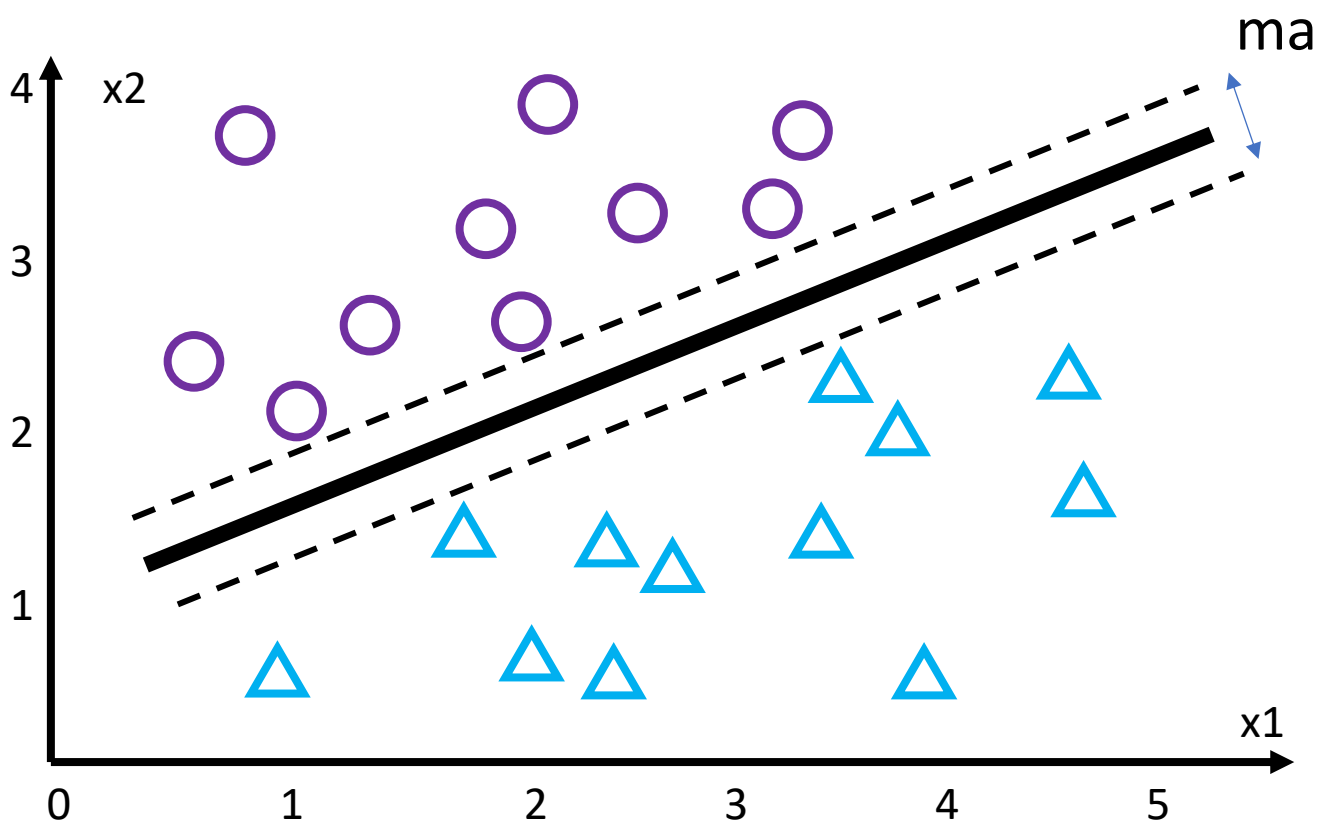


Which is the best?

Support vector machines

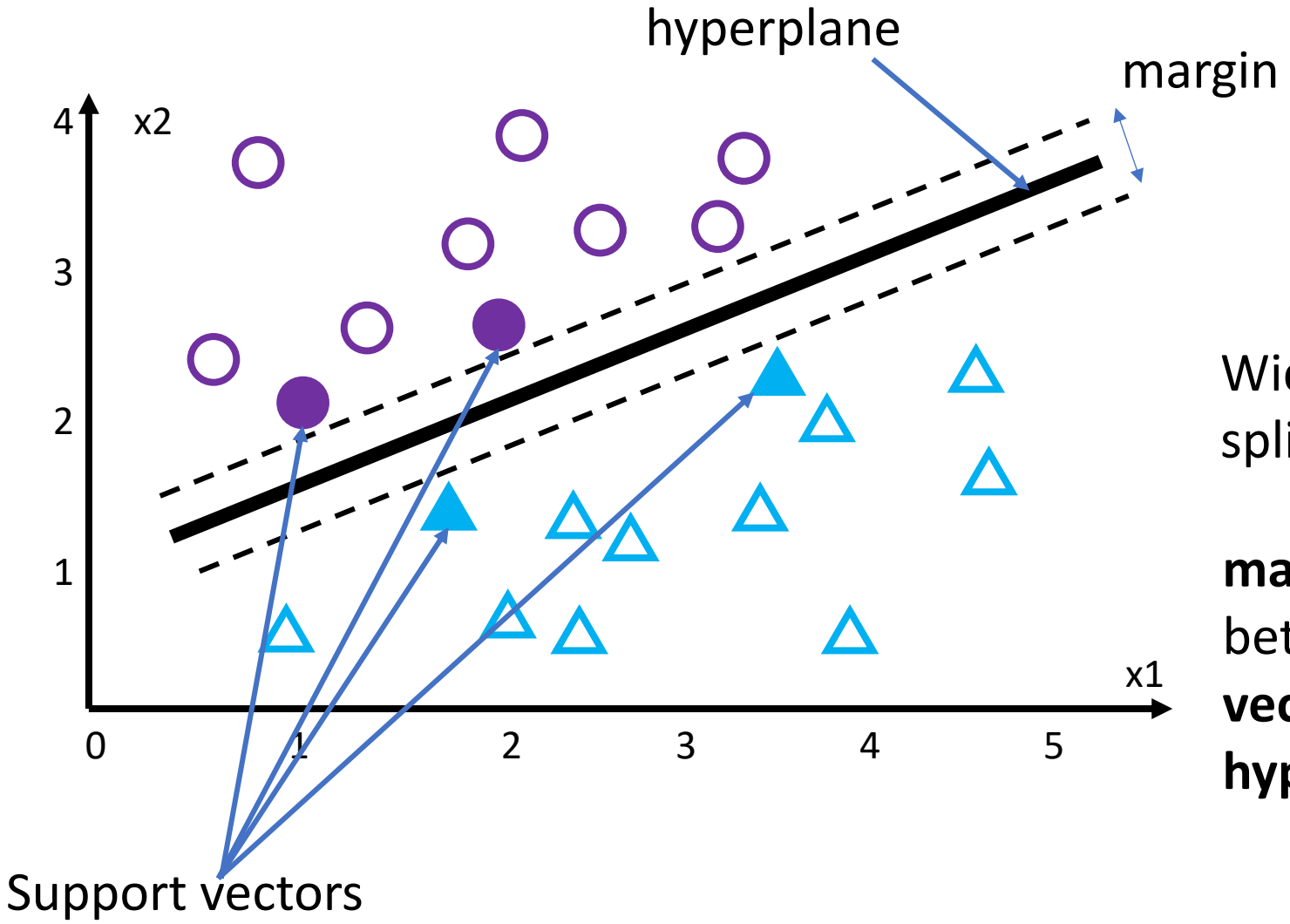


Support vector machines



Widest **margin** to split the groups.

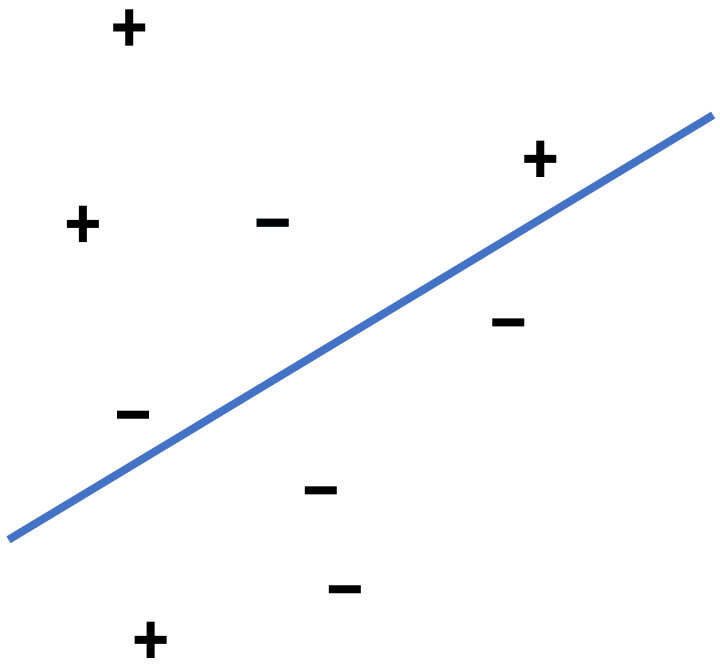
Support vector machines



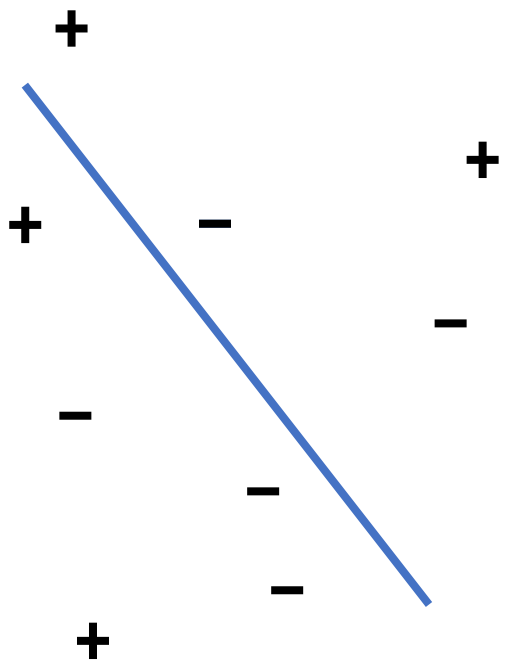
Widest **margin** to split the groups.

margin = distance between the **support vectors** and the **hyperplane**.

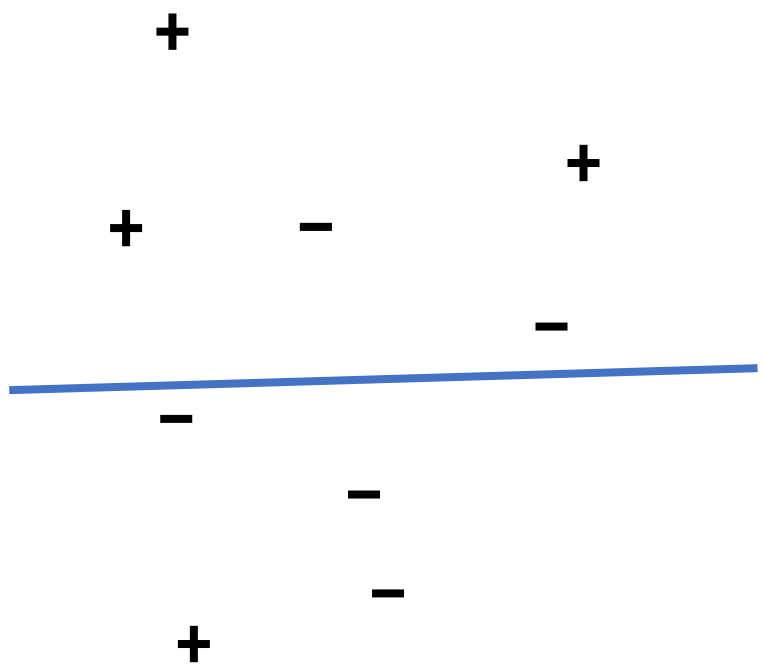
Support vector machines: Kernel Trick



Not a good separator

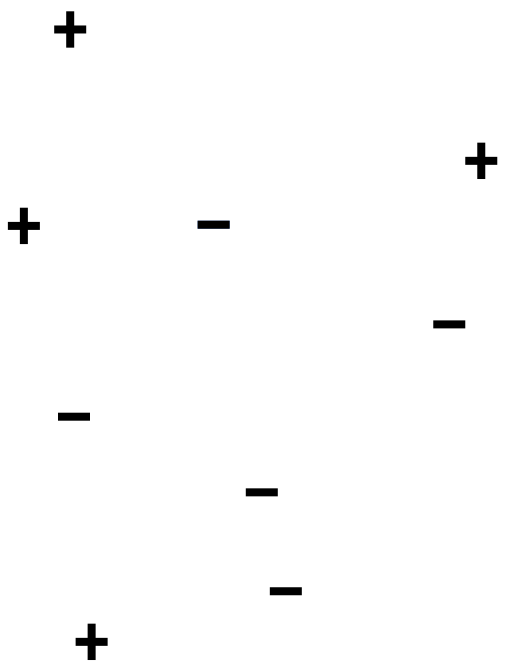


Not a good separator



Not a good separator

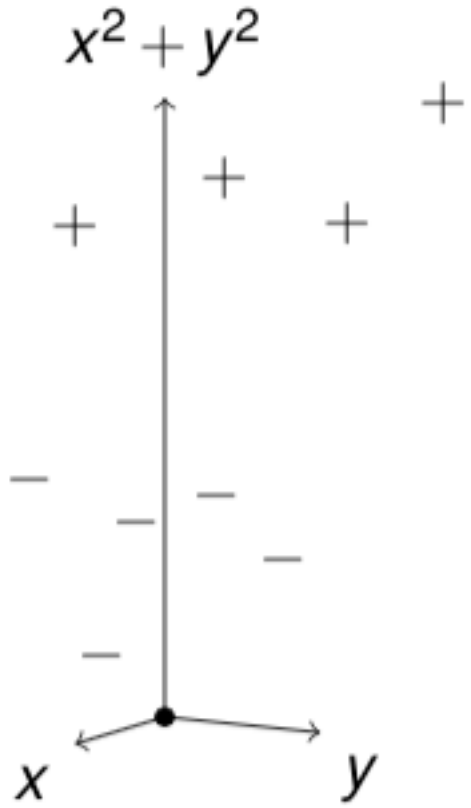
Support vector machines: Kernel Trick



Kernelization

To map point (x,y) to point $(x,y,x2 + y2)$

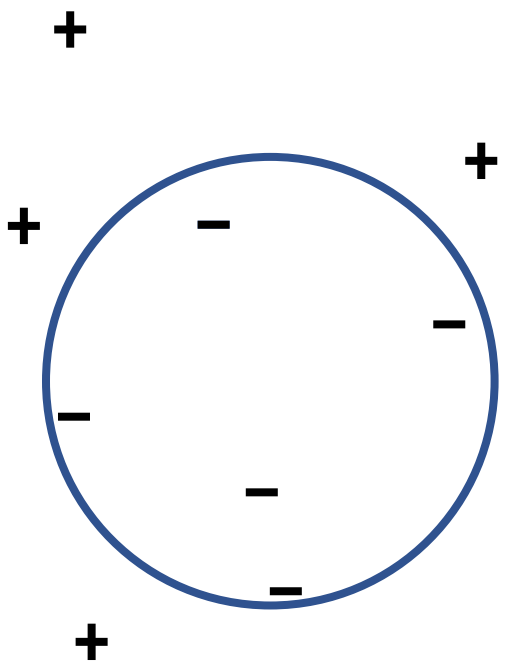
Kernel functions:
Linear
Radial Basis Function (RBF)
Polynomial
Sigmoid



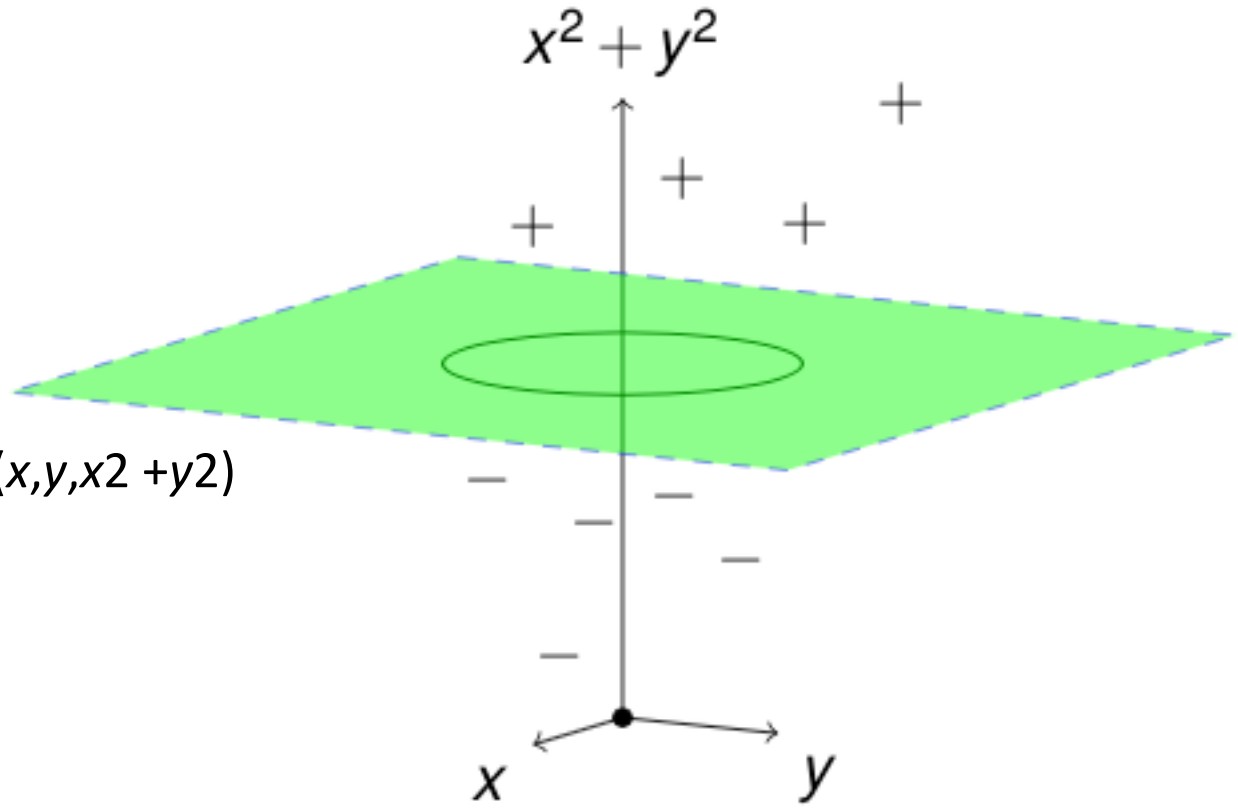
Adapted from Satish Rao, CS270

Support vector machines: Kernel Trick

A visualization: https://www.youtube.com/watch?v=3liCbRZPrZA



Kernelization

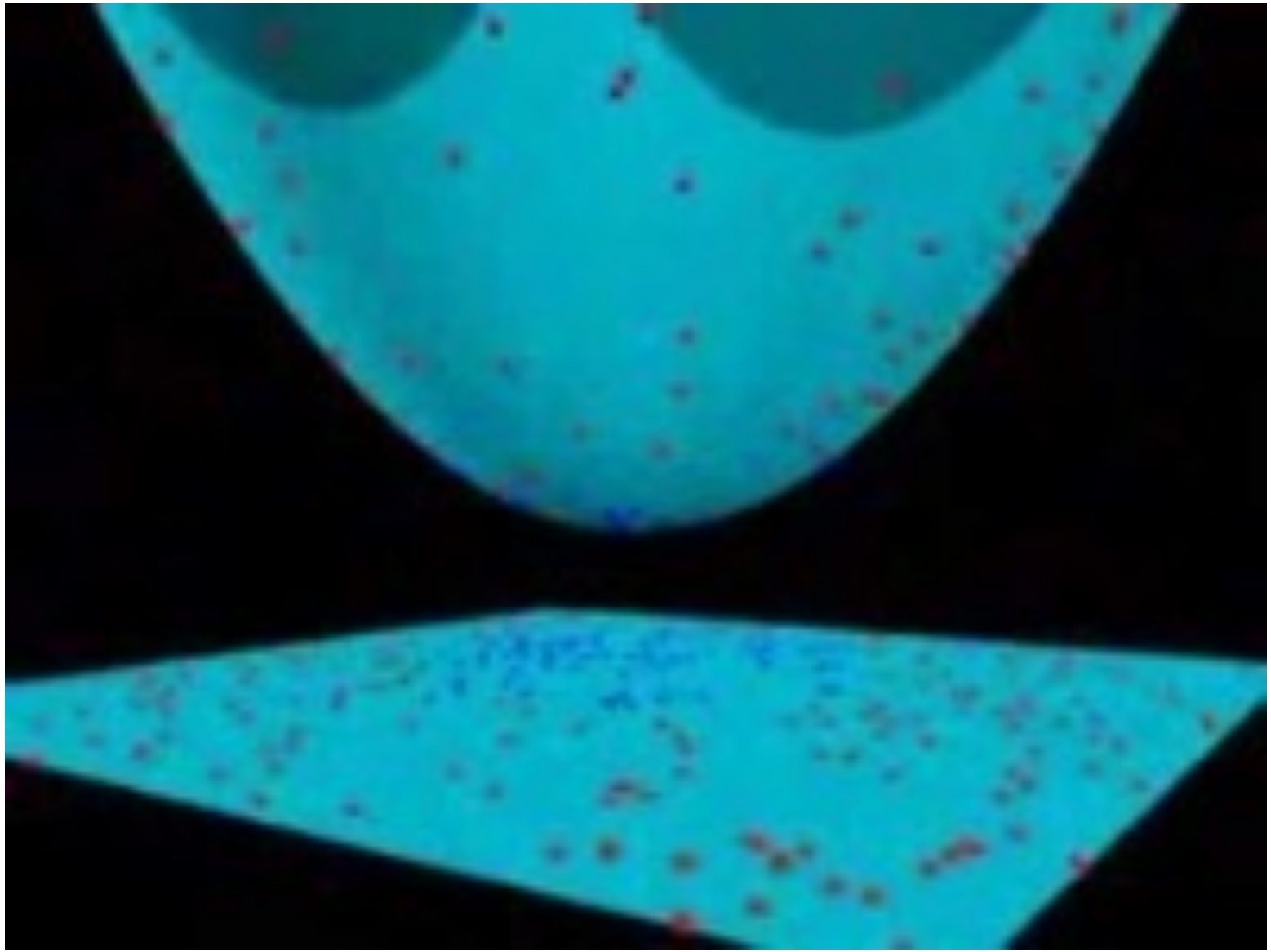


To map point (x,y) to point $(x,y,x2 + y2)$

Kernel functions:
Linear
Radial Basis Function (RBF)
Polynomial
Sigmoid

Adapted from Satish Rao, CS270

A visualization: https://www.youtube.com/watch?v=3liCbRZPrZA



Train:

Search for the optimal parameters of the model.

Infer:

Use the trained model to do the calculation.

How to evaluate the performance of a model?

	Prediction		
Label		1	0
	1	TP	FN
	0	FP	TN

TP: True Positive
TN: True Negative
FP: False Positive
FN: False Negative

Accuracy = (TP+TN)/all
Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Let’s practice using this notebook:
https://colab.research.google.com/drive/1_979SLMyXRD_zTr9mdy32EBKUDKrn7VF?usp=sharing