ML & DL [1]

GDSC - AI-ML_Team_6

Oct. 13. 2021

Python Basics

Install Packages & Modules

- Use Anaconda prompt or Terminal
- pip install (package name)
- conda install (package name)
- To check installation, use (pip/conda) search (package name)

Activating Environment

- To create environment, conda create -n (environment name)
- To activate it, conda activate (environment name)
- To exit to base, conda deactivate
- Each environment do not share the packages -> should install again
- To work in various package versions, use environment

Useful Packages

- Numpy : linear algebra, random (np)
- Pandas : dataframe and managing file (pd)
- Matplotlib.pyplot : draw graphs (plt)
- Seaborn: more various graphs (sns)
- Os : interact with OS, ex) make dir, move dir, current working dir…

Class

-Blueprint of Object

Definition

```
In [1]:
class MyClass:
    def __init__(self, name, std_id):
        self.name = name
        self.std_id = std_id
        self.user_name_ = self.name + '-'  + str(self.std_id)

    def attendance(self):
        print(self.user_name_)
```

Initialization

Instances

Method

executed in 3ms, finished 00:32:09 2021-10-13

```
In [2]:
a = MyClass('yooseung', 20215047)
```

Object

executed in 2ms, finished 00:32:09 2021-10-13

```
In [3]:
a.attendance()
a.name
```

Use method & instance by .

executed in 7ms, finished 00:32:09 2021-10-13

```
yooseung-20215047

'yooseung'
```

map & sort

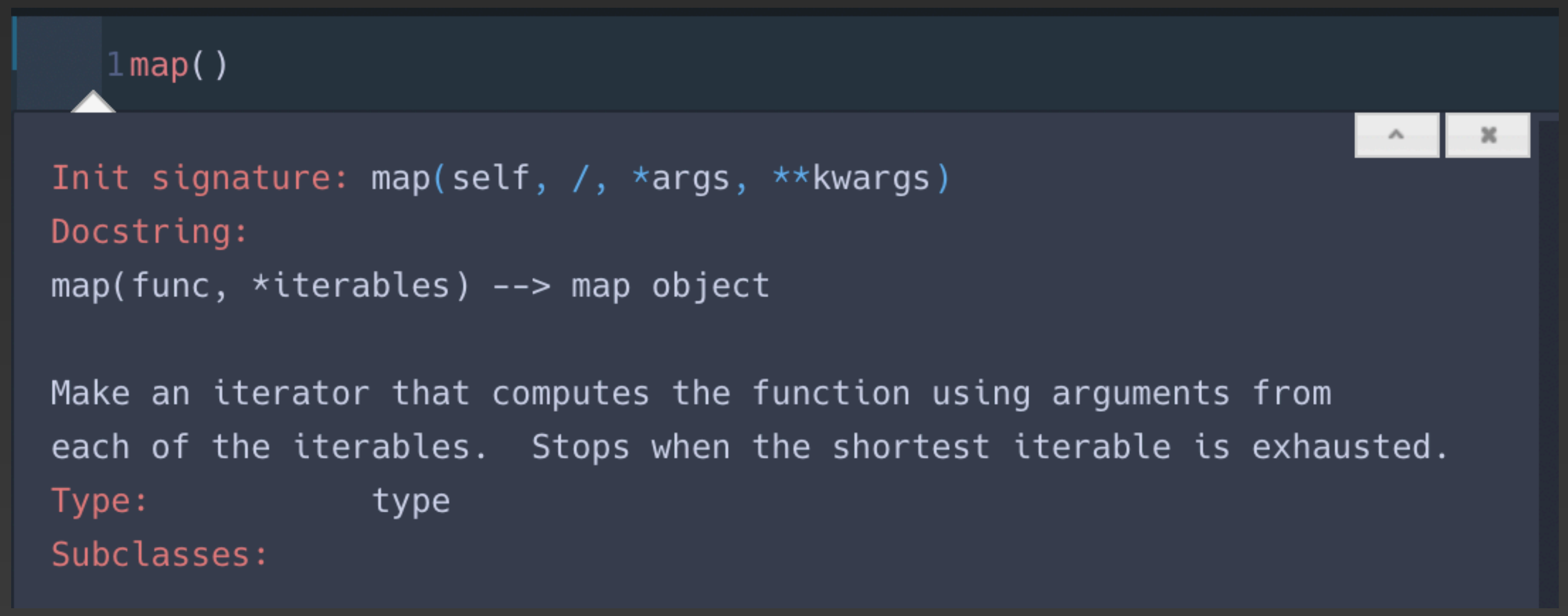
```
myList = [1,2,3,4,5]
myList = map(lambda x: x**2, myList)
print(myList)
myList = list(myList)
print(myList)                    Conditional can be used too
bool_list= list(map(lambda x: x % 2 == 1, myList))
print(bool_list)
myList.sort(reverse = True) In descending order
print(myList)
myList.sort()  In ascending order
print(myList)
```

executed in 9ms, finished 20:09:35 2021-10-13

```
<map object at 0x7fb5580e43d0> Map is a object
[1, 4, 9, 16, 25] To see result, use list()
[True, False, True, False, True]
[25, 16, 9, 4, 1]
[1, 4, 9, 16, 25]
```

Shortcuts

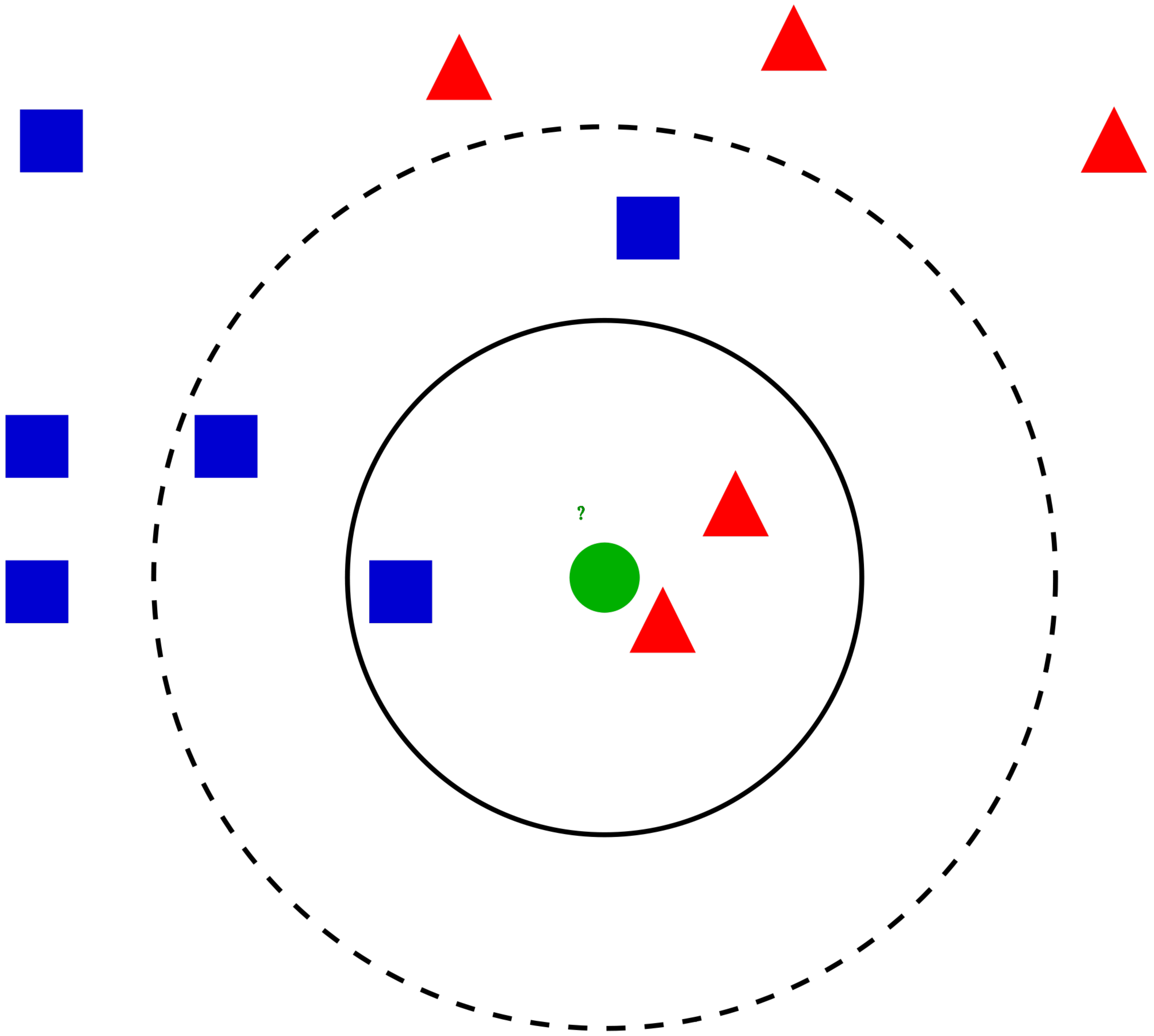
- Esc : command mode
  - M : markdown
  - Y : code
  - A/B : add cell Above/Below
  - DD : delete cell
  - Z : undo delete cell
  - C + V : copy and paste
- Enter : edit mode
  - Ctrl + enter : run selected cell
  - Shift + enter : run and move next
  - Shift + tab : cancel tab
  - Shift + tab (inside the word): show detail



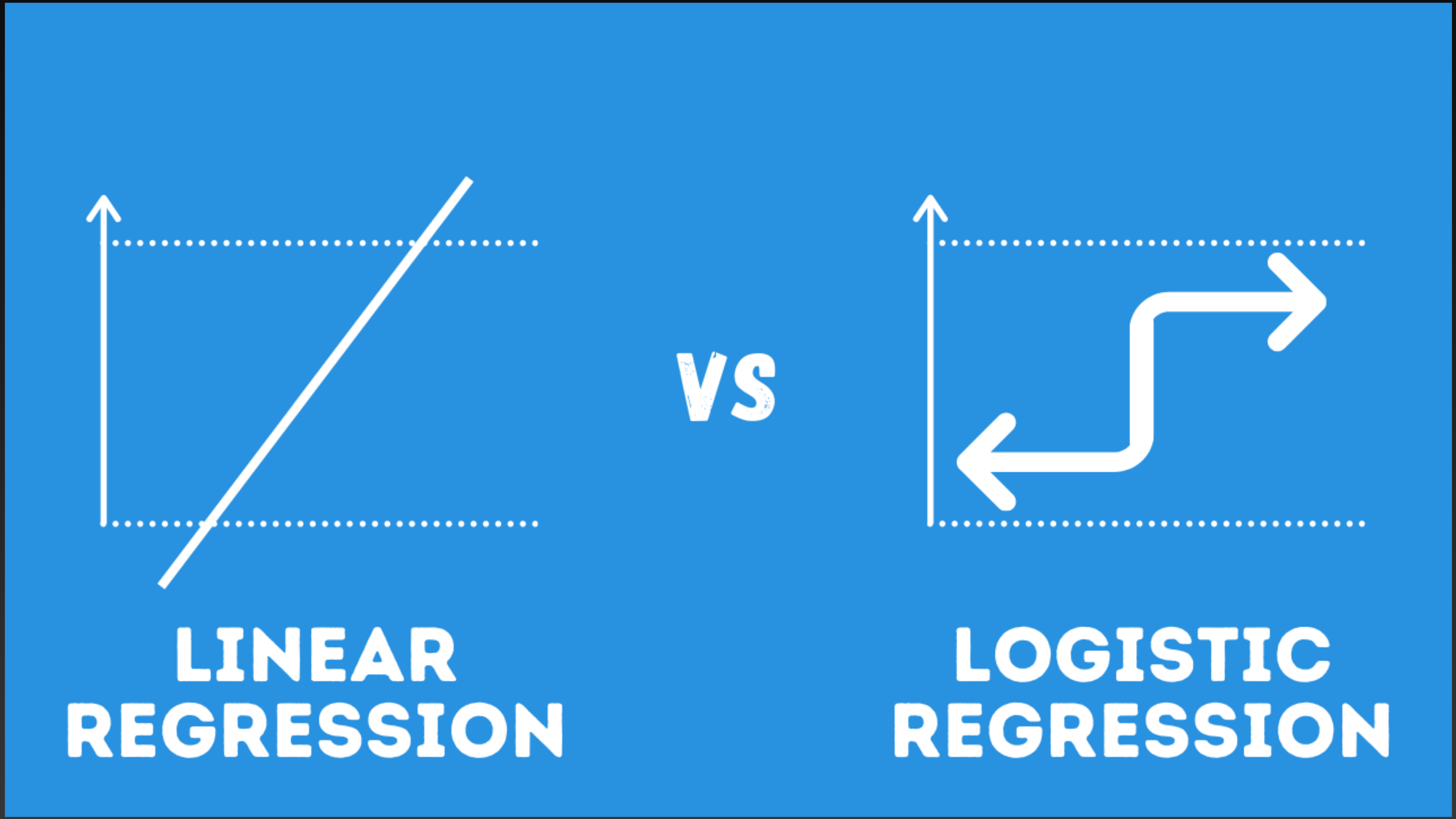
1. Machine Learning

KNN

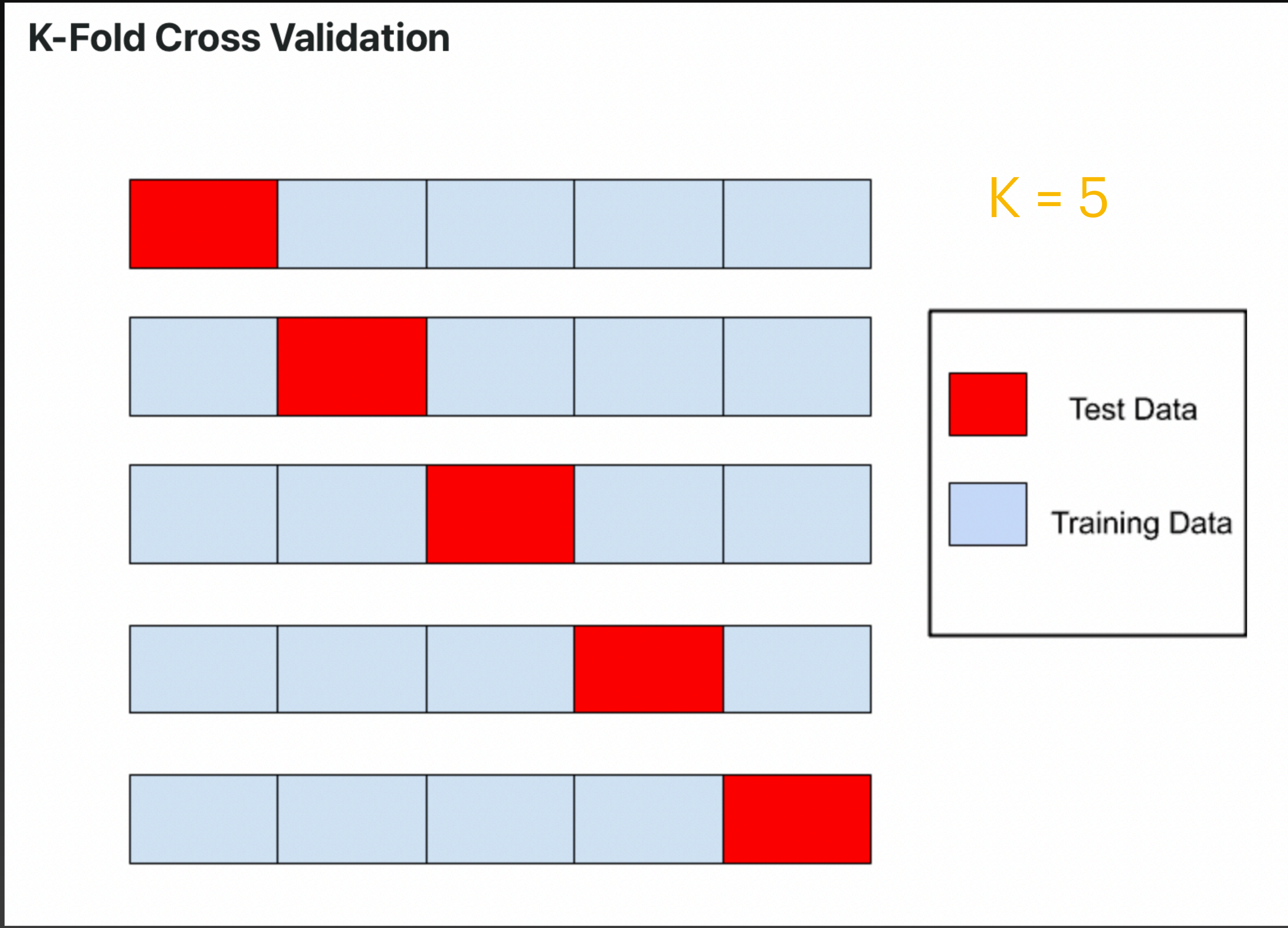
- Find K nearest Neighbors
- Determine the value with them
- For example, circle will be predicted as red triangle



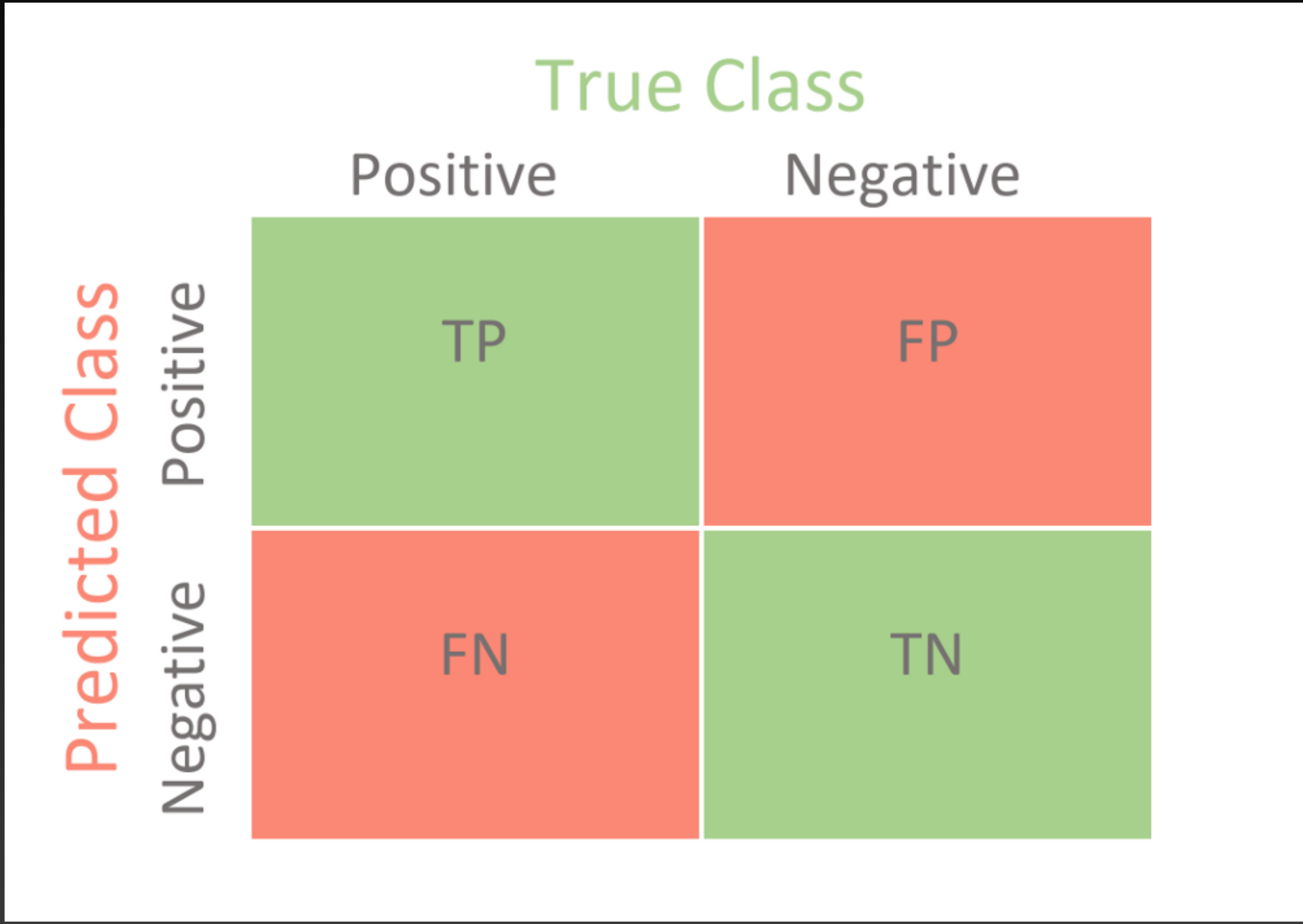
Regression



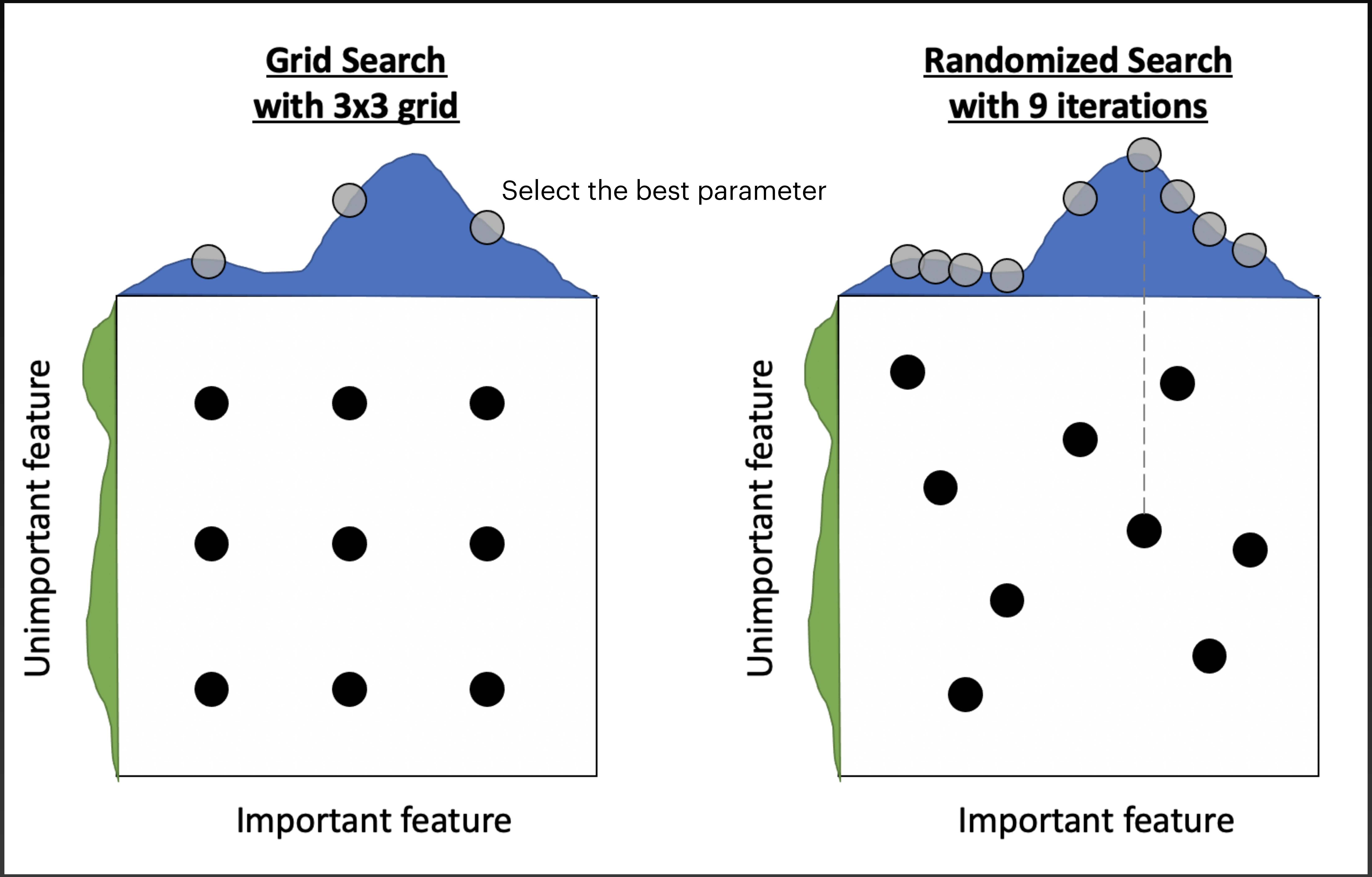
Cross Validation



Confusion Matrix



Grid Search


Grid Search
with 3x3 grid
Select the best parameter
Randomized Search
with 9 iterations
Unimportant feature
Important feature
Unimportant feature
Important feature

GridSearchCV

- knn = KNeighborsClassifier()

Array 1 ~ 50

- knn_cv = GridSearchCV (knn, grid, cv = 3)

Estimator

- knn_cv.cv_results_

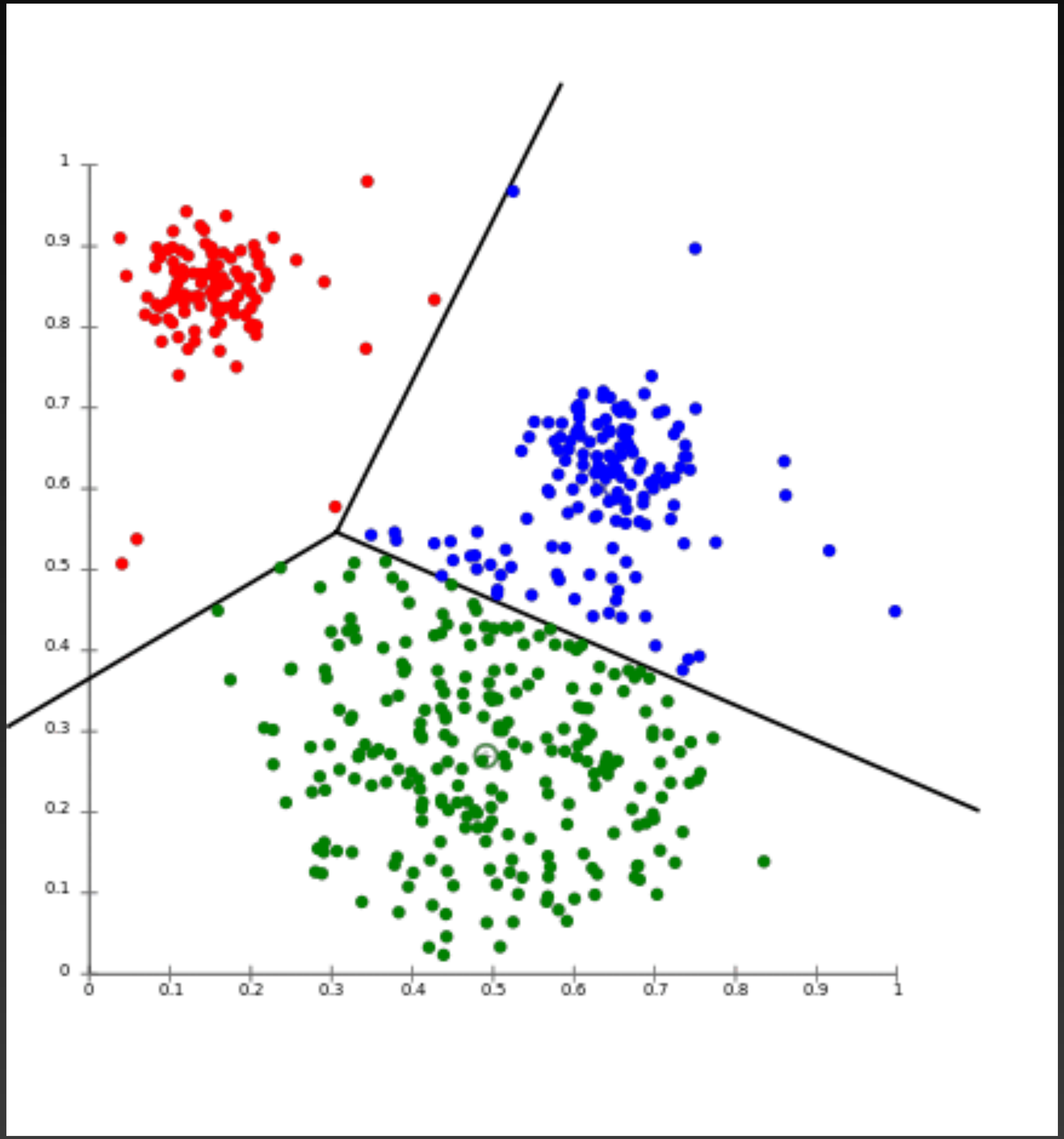
K = 4 -> Rank 1

```
'rank_test_score': array([25,  9,  3,  1,  5,  2, 25,  4, 40, 10, 27,  8, 35, 17, 24, 17, 38,
       29, 29, 11, 29, 11, 19, 11, 19, 11, 39, 11, 19,  6, 11,  6, 29, 19,
       29, 19, 29, 36, 36, 28, 41, 41, 48, 41, 48, 41, 41, 41, 47],
      dtype=int32)}
```

```
'split0_test_score': array([0.41346154, 0.41346154, 0.43269231, 0.41346154, 0.41346154,
        0.40384615, 0.41346154, 0.42307692, 0.41346154, 0.42307692,
        0.42307692, 0.40384615, 0.40384615, 0.40384615, 0.40384615,
        0.40384615, 0.39423077, 0.39423077, 0.39423077, 0.39423077,
        0.39423077, 0.39423077, 0.39423077, 0.39423077, 0.39423077,
        0.39423077, 0.40384615, 0.39423077, 0.39423077, 0.39423077,
        0.39423077, 0.39423077, 0.39423077, 0.39423077, 0.39423077,
        0.39423077, 0.39423077, 0.38461538, 0.38461538, 0.38461538,
        0.38461538, 0.38461538, 0.38461538, 0.38461538, 0.38461538,
        0.38461538, 0.38461538, 0.38461538, 0.39423077]),
'split1_test_score': array([0.90291262, 0.88349515, 0.88349515, 0.91262136, 0.90291262,
        0.9223301 , 0.88349515, 0.88349515, 0.87378641, 0.87378641,
        0.87378641, 0.89320388, 0.88349515, 0.88349515, 0.88349515,
        0.88349515, 0.87378641, 0.88349515, 0.88349515, 0.89320388,
        0.87378641, 0.89320388, 0.89320388, 0.90291262, 0.89320388,
        0.90291262, 0.88349515, 0.90291262, 0.90291262, 0.90291262,
        0.91262136, 0.91262136, 0.89320388, 0.89320388, 0.88349515,
        0.89320388, 0.88349515, 0.88349515, 0.88349515, 0.88349515,
        0.86407767, 0.86407767, 0.85436893, 0.86407767, 0.85436893,
        0.86407767, 0.86407767, 0.86407767, 0.85436893]),
'split2_test_score': array([0.86407767, 0.90291262, 0.9223301 , 0.94174757, 0.89320388,
        0.91262136, 0.88349515, 0.91262136, 0.87378641, 0.90291262,
        0.88349515, 0.90291262, 0.88349515, 0.90291262, 0.89320388,
        0.90291262, 0.89320388, 0.89320388, 0.89320388, 0.90291262,
        0.90291262, 0.90291262, 0.89320388, 0.89320388, 0.89320388,
        0.89320388, 0.87378641, 0.89320388, 0.88349515, 0.90291262,
        0.88349515, 0.89320388, 0.88349515, 0.89320388, 0.89320388,
        0.89320388, 0.89320388, 0.89320388, 0.89320388, 0.90291262,
        0.89320388, 0.89320388, 0.88349515, 0.89320388, 0.88349515,
        0.89320388, 0.89320388, 0.89320388, 0.88349515]),
```

Cv = 3 -> Three test sets

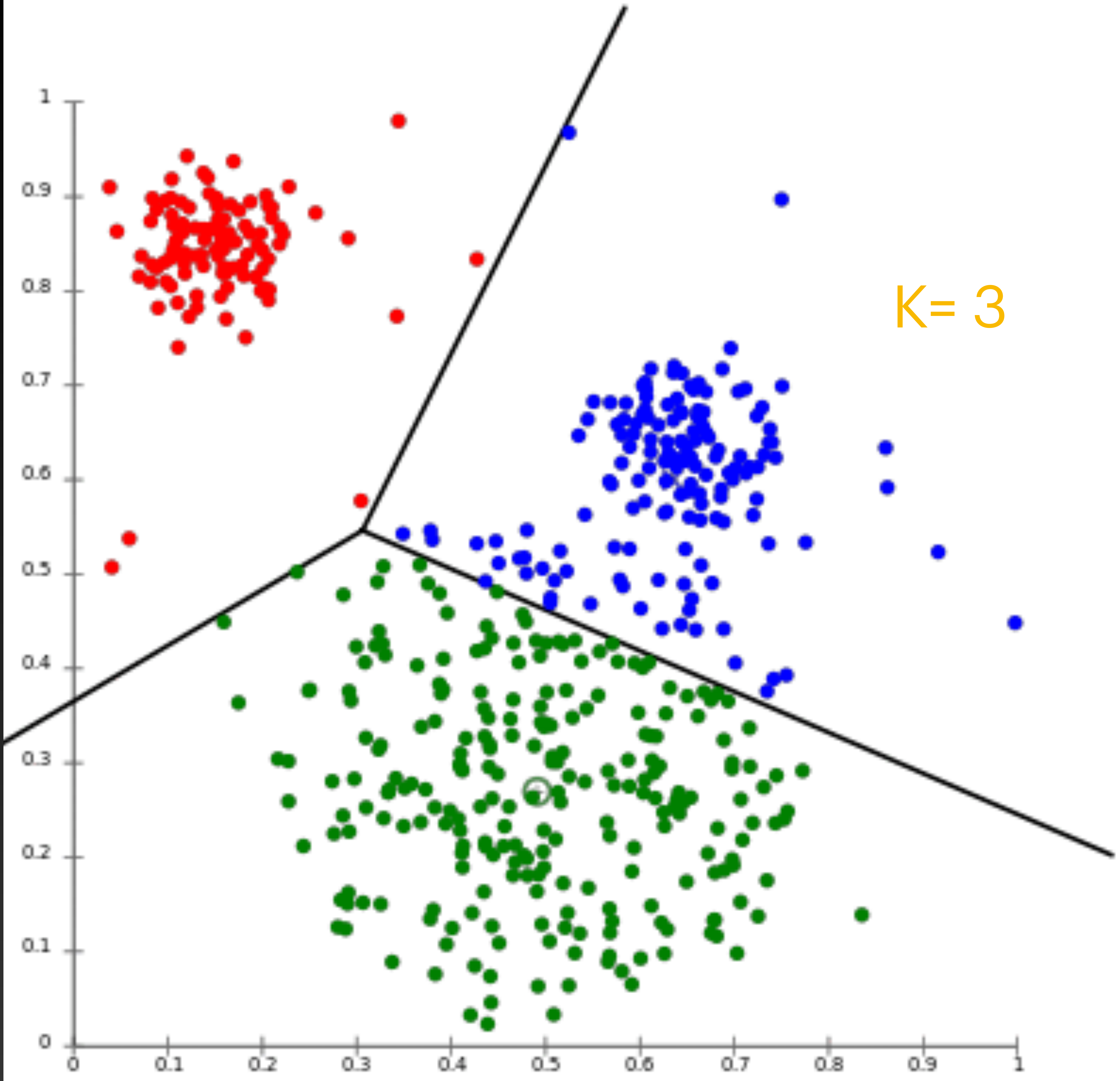
K-Mean Clustering



K = 3

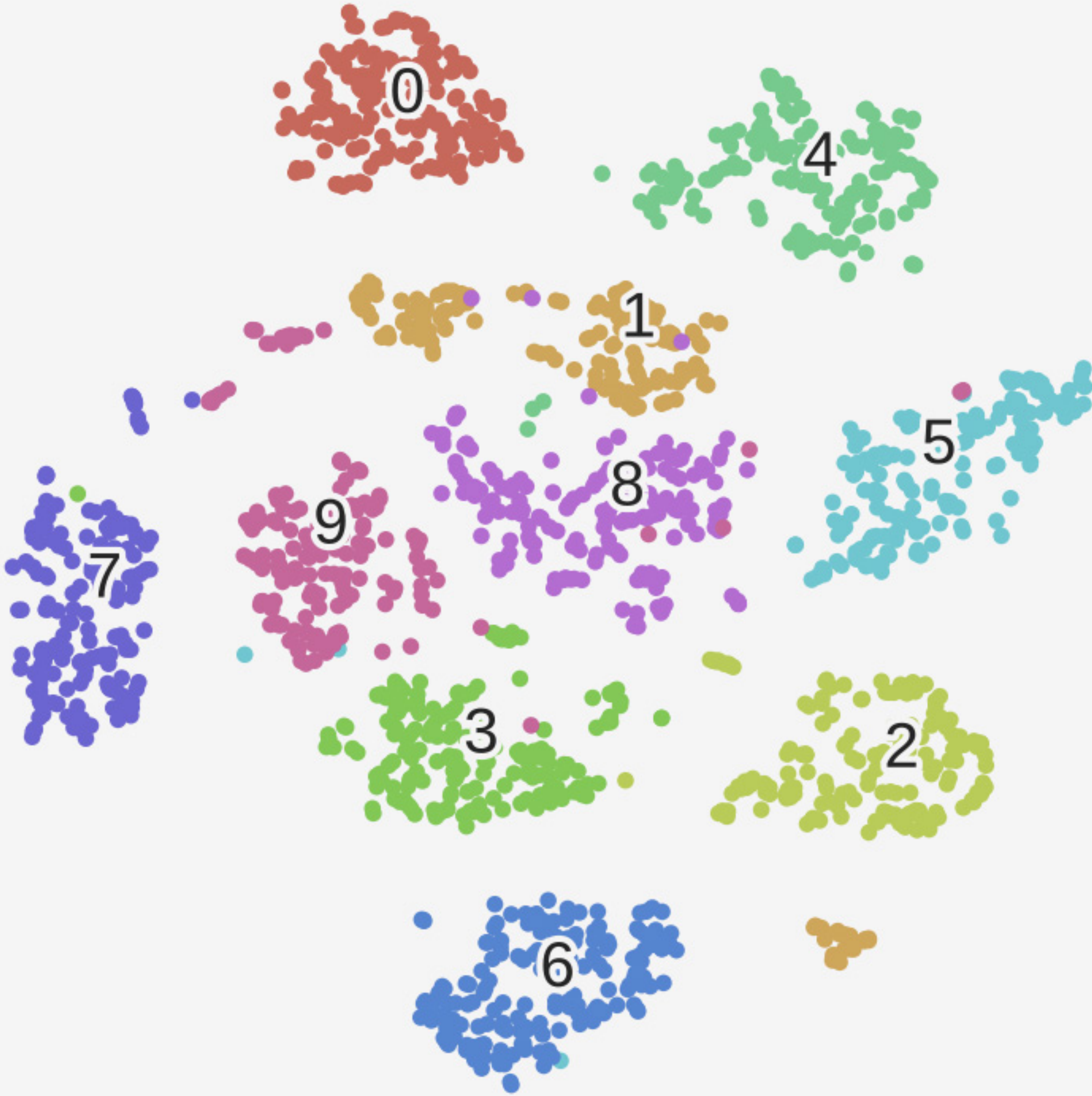
K-Mean Clustering

- Cluster in to K groups
- In the way of minimizing the variance of each points



T-SNE

- Convert high-dimension graph to low-dimension graph
- Using T-distribution



2. Deep Learning Tutorial

Forward Propagation

input	weight	activation	output.
$[x_1, x_2, \cdots, x_n]$	$[w_1, w_2, \cdots, w_n]$	sigmoid $(\sigma(z))$	$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$

$$\Rightarrow z = W^T \cdot x = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \cdot [x_1, x_2, \cdots x_n] = \begin{bmatrix} w_1 x_1 \\ w_2 x_2 \\ \vdots \\ w_n x_n \end{bmatrix}$$

$$\Rightarrow \sigma(z) = \begin{bmatrix} \sigma(z_1) \\ \sigma(z_2) \\ \vdots \\ \sigma(z_n) \end{bmatrix} = \underbrace{\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}}_{\text{labels}}$$

Error Loss Function

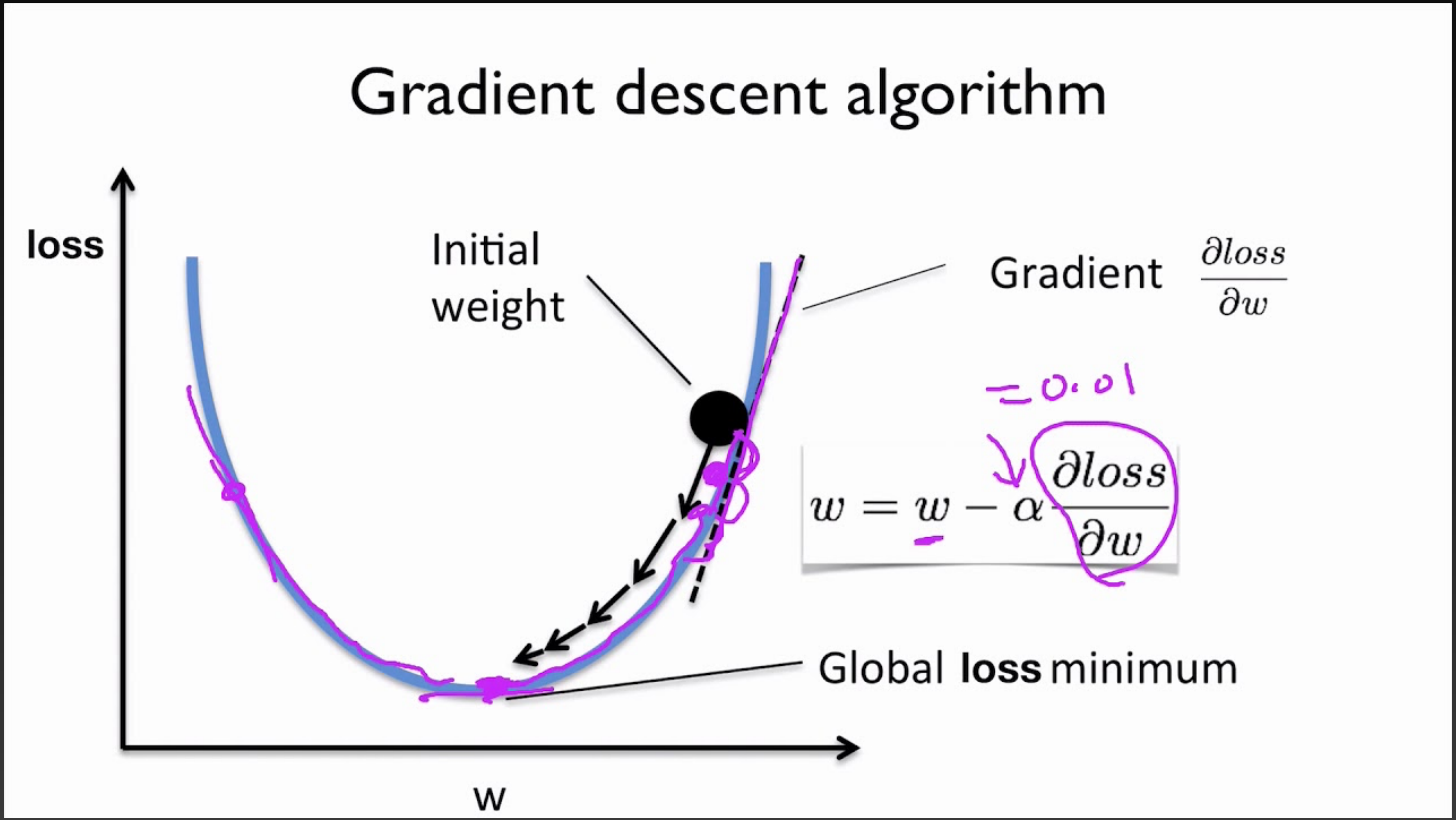
$$f(y,\hat{y}) := -(1-y)\log(1-\hat{y}) - y\log\hat{y}$$

→ where y : actual value

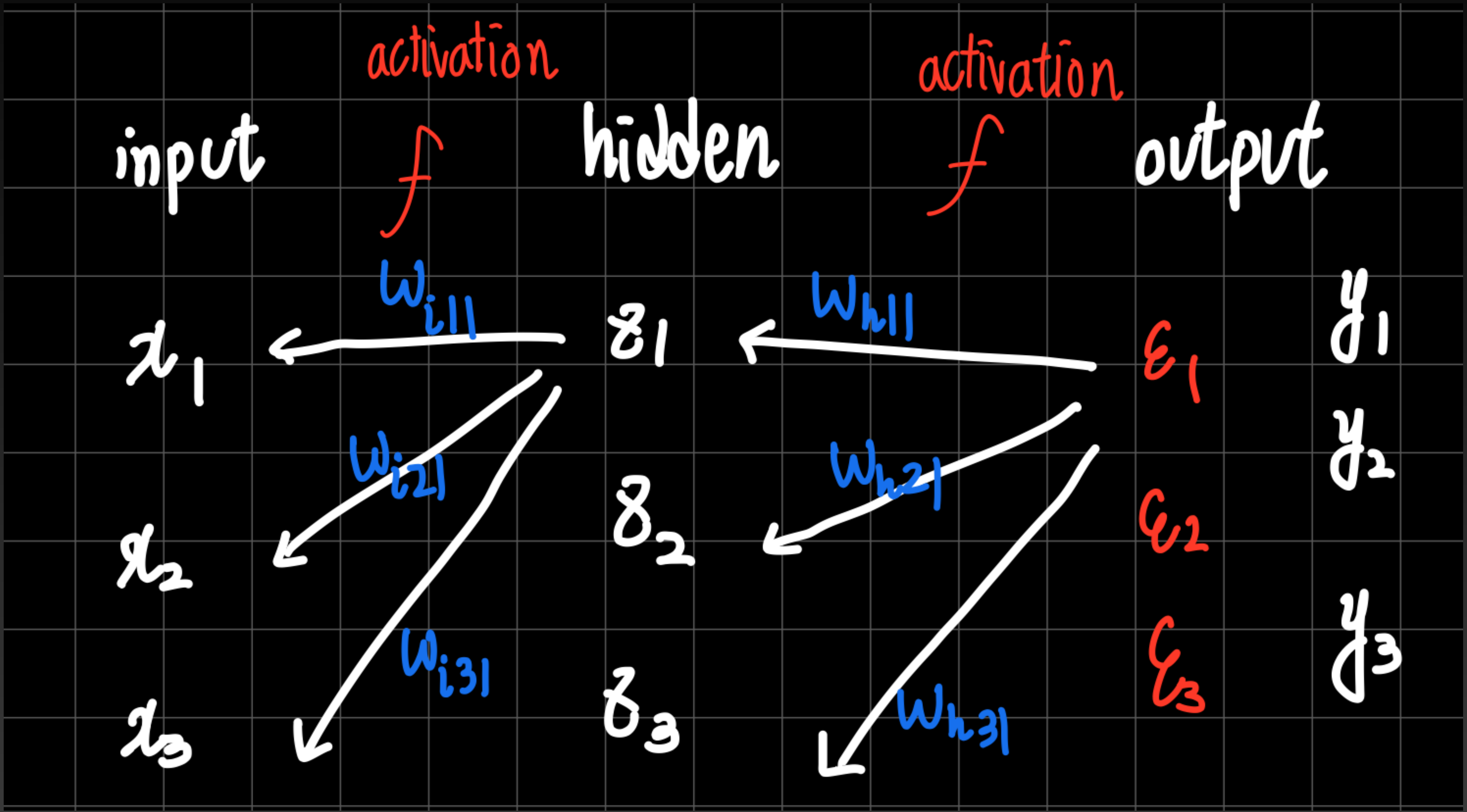
$\hat{y}$: predicted value.

output	actual	loss	cost
$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$	$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$	$\begin{bmatrix} f(y_1,\hat{y}_1) \\ f(y_2,\hat{y}_2) \\ \vdots \\ f(y_n,\hat{y}_n) \end{bmatrix}$	$\sum_{i=1}^{n} f(y_n,\hat{y}_n)$

Gradient Descent

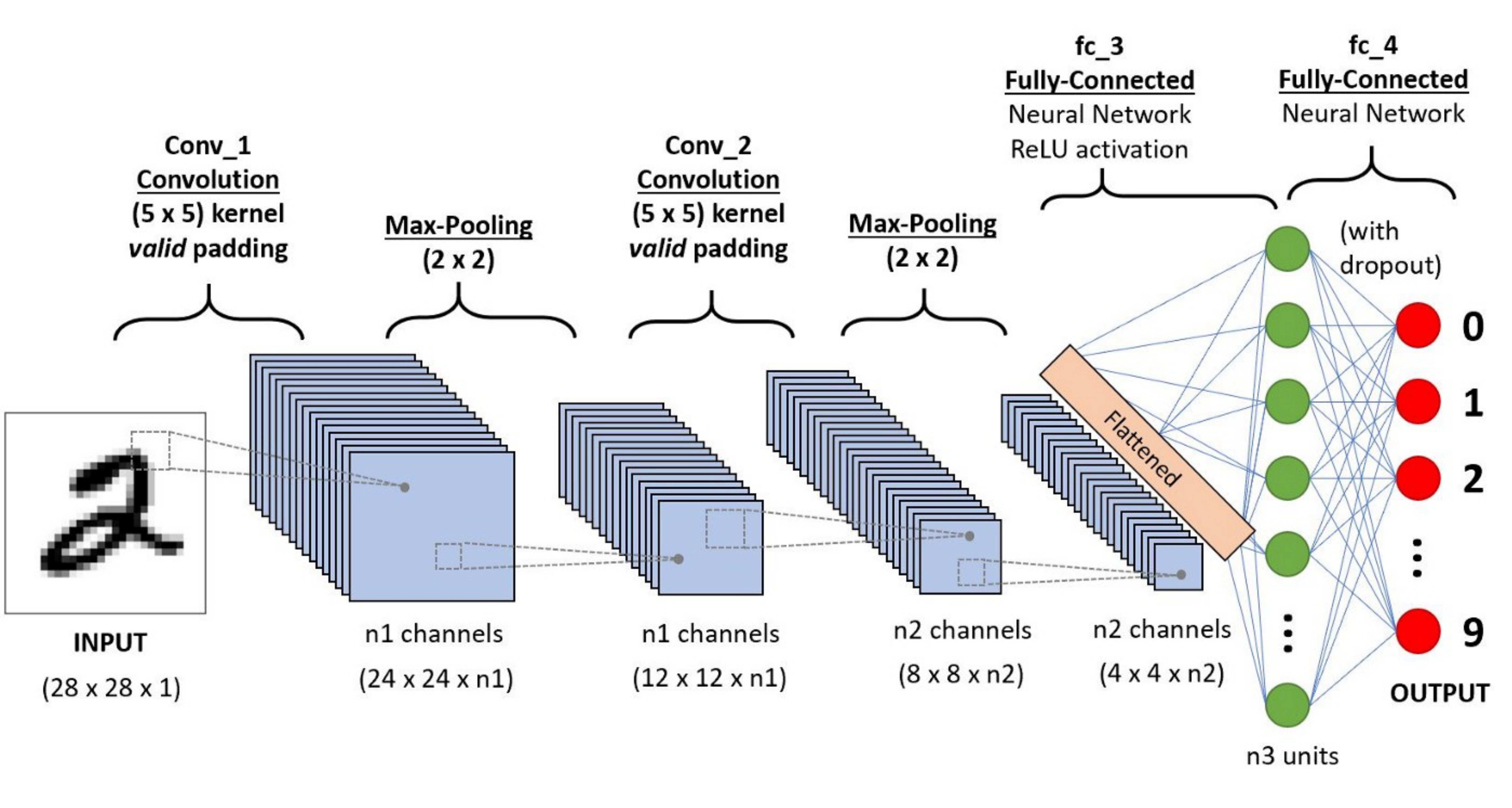


Back propagation

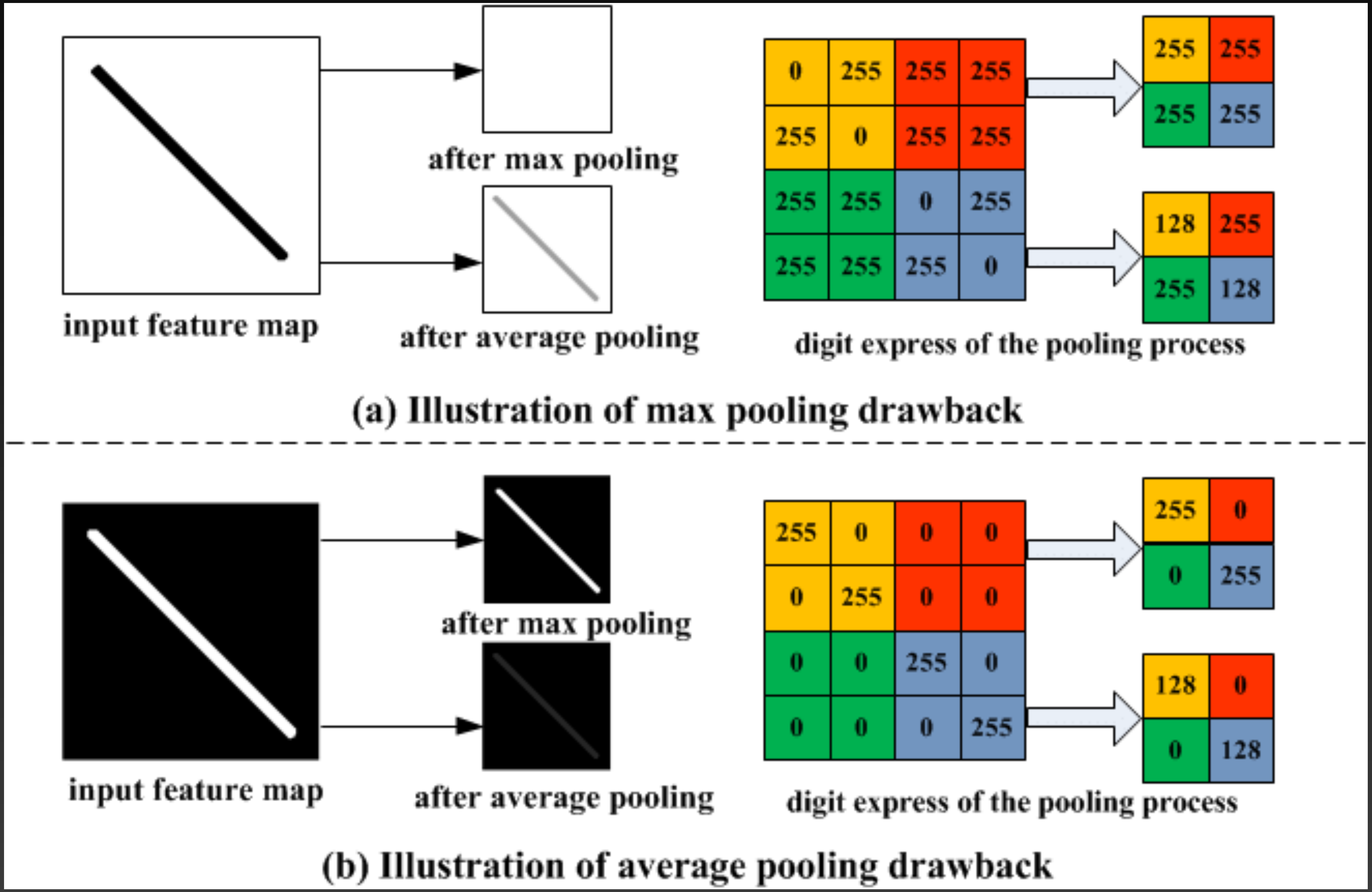


3. Pytorch Tutorial

Convolutional Neural Network



Pooling


after max pooling
input feature map
after average pooling
0 255 255 255
255 0 255 255
255 255 0 255
255 255 255 0
255 255
255 255
128 255
255 128
digit express of the pooling process
(a) Illustration of max pooling drawback
after max pooling
input feature map
after average pooling
255 0 0 0
0 255 0 0
0 0 255 0
0 0 0 255
255 0
0 255
128 0
0 128
digit express of the pooling process
(b) Illustration of average pooling drawback

Just an Implementation on Pytorch!