EE2211 Introduction to Machine Learning

Lecture 10

Wang Xinchao
xinchao@nus.edu.sg

© Copyright EE, NUS. All Rights Reserved.

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Linear Algebra, Probability and Statistics
- Fundamental Machine Learning Algorithms I (Helen)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Helen)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

[Important] In the Final, no coding questions for Xinchao's part!

Despite you will see some in the tutorial, they won't be testsed.

© Copyright EE, NUS. All Rights Reserved.

EE2211: Learning Outcome
A Summary of Module Content

- **I am able to understand the formulation of a machine learning task**
  - Lecture 1 (feature extraction + classification)
  - Lecture 4 to Lecture 9 (regression and classification)
  - Lecture 11 and Lecture 12 (clustering and neural network)
- **I am able to relate the fundamentals of linear algebra and probability to machine learning**
  - Lecture 2 (recap of probability and linear algebra)
  - Lecture 4 to Lecture 8 (regression and classification)
  - Lecture 12 (neural network)
- **I am able to prepare the data for supervised learning and unsupervised learning**
  - Lecture 1 (feature extraction) [For supervised and unsupervised]
  - Lecture 2 (data wrangling) [For supervised and unsupervised]
  - Lecture 10 (Training/Validation/Test) [For supervised]
  - Programming Exercises in tutorials
- **I am able to evaluate the performance of a machine learning algorithm**
  - Lecture 5 to Lecture 9 (evaluate the difference between labels and predictions)
  - Lecture 10 (evaluation metrics)
- **I am able to implement regression and classification algorithms**
  - Lecture 5 to Lecture 9

© Copyright EE, NUS. All Rights Reserved.

Outline

- Dataset Partition:
  - Training/Validation/Testing
- Cross Validation
- Evaluation Metrics
  - Evaluating the quality of a trained machine learning system

We will talk about many metrics:
It is OK you can't memorize them all
But intuition is important!

© Copyright EE, NUS. All Rights Reserved.

A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)

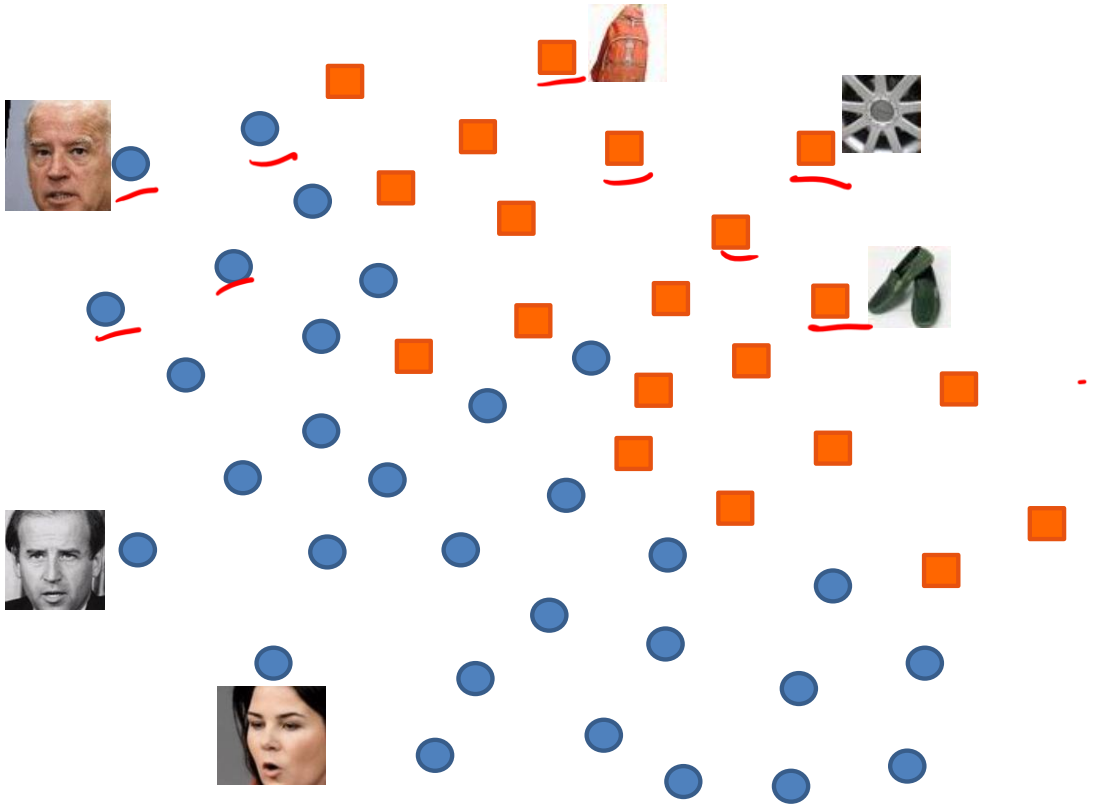
Faces ✓

Non-faces ✗

© Copyright EE, NUS. All Rights Reserved.

A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)
  - We will have **one dataset** to **train** the Random Forest



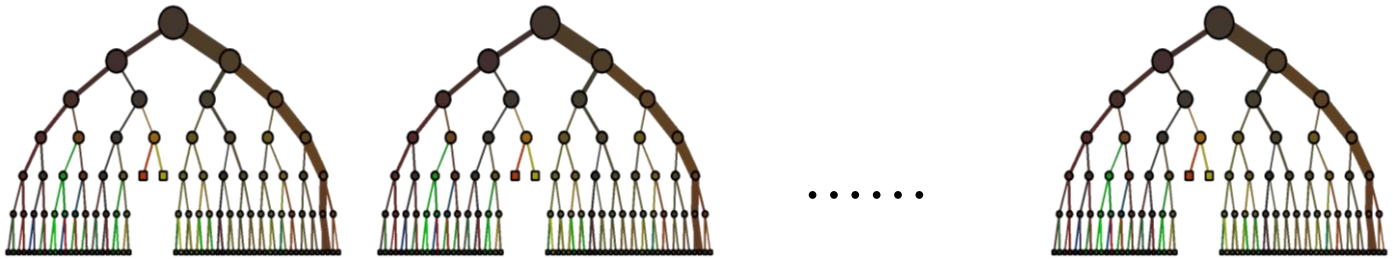
© Copyright EE, NUS. All Rights Reserved.

A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)
  - We will have **one dataset** to **train** the Random Forest
  - We will have tunable **(hyper)parameters** for the Random Forest. For example, ***the number of trees*** in the Random Forest
    - Shall we use 100 trees?
    - Shall we use 200 trees?
    - …

    We need to decide on the parameter

pyerparameters are parameters assigned by the ML practitioner and not by the ML model themselves. e.g. max_depth for random forest



© Copyright EE, NUS. All Rights Reserved.

A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)
  - We will have **one dataset** to **train** the Random Forest
  - We will have tunable **(hyper)parameters** for the Random Forest. For example, ***the number of trees*** in the Random Forest
    - Shall we use 100 trees?
    - Shall we use 200 trees?
    - …

    We need to decide on the parameter
  - Once we decide the number of trees, we will apply the Random Forest with the **selected parameter** on **unseen** test data.

Test Data

Yes!

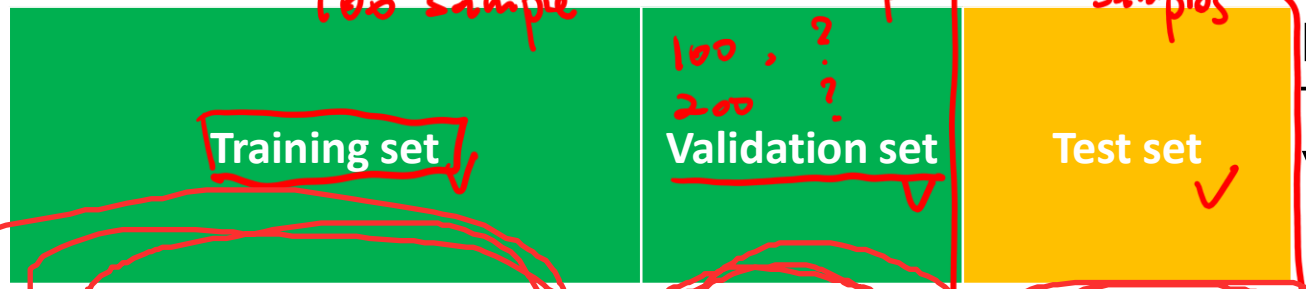
No!

© Copyright EE, NUS. All Rights Reserved.

Training, Validation, and Test

Always need these three data sets

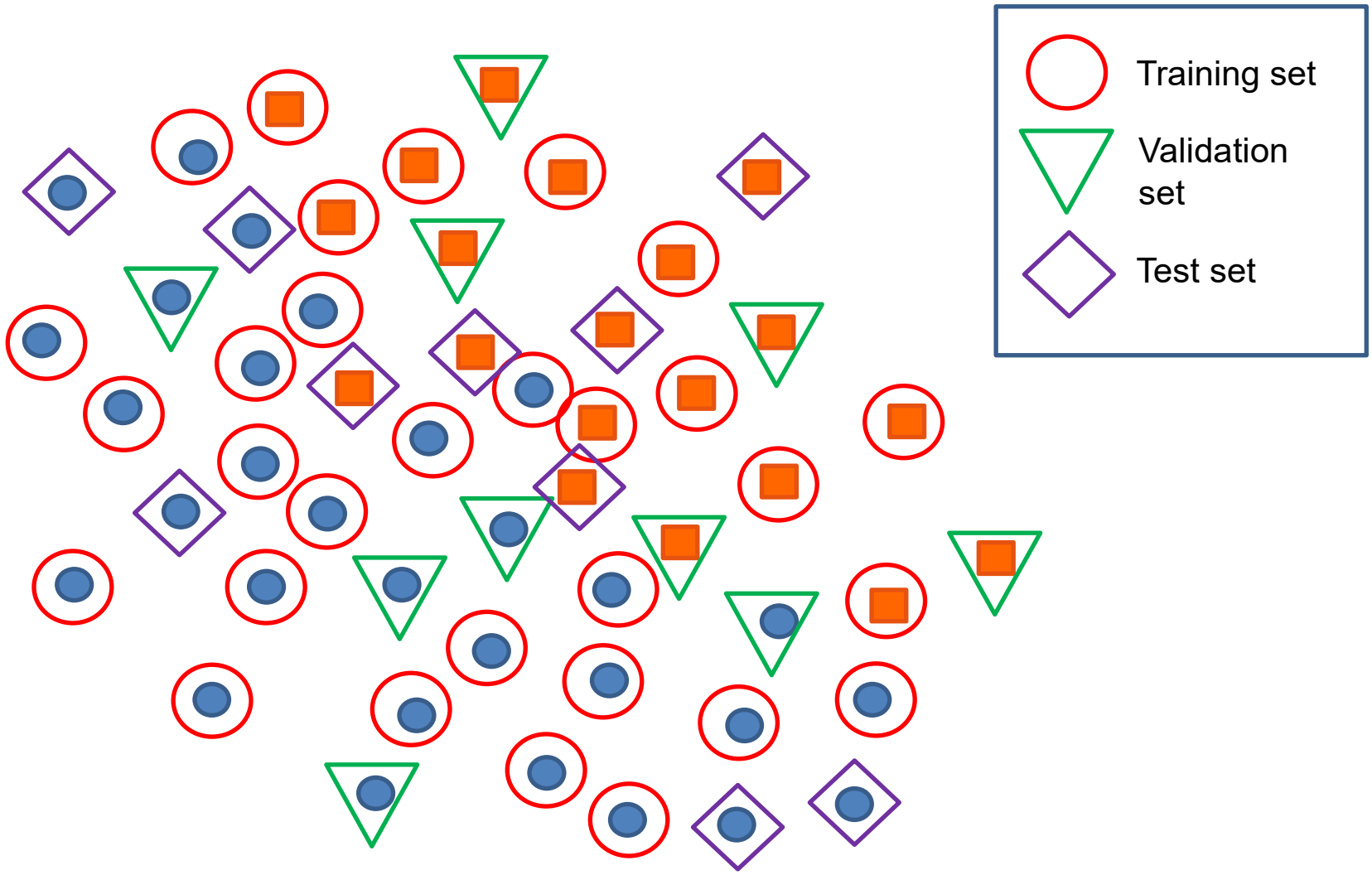
- In real-world application,
  - We don't have test data, since they are unseen
  - Imagine you develop a face detector app, you don't know whom you will test on
- In lab practice,
  - We divide the dataset into three parts



  - **NEVER touch test data during training/validation!!!**

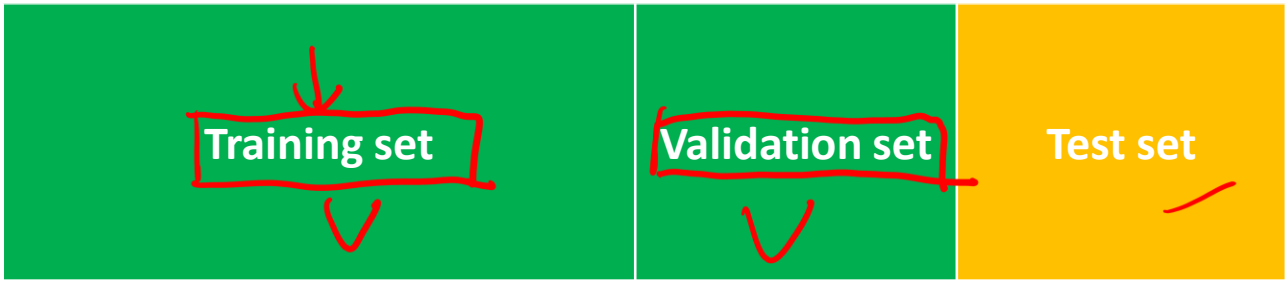
© Copyright EE, NUS. All Rights Reserved.

Training, Validation, and Test



© Copyright EE, NUS. All Rights Reserved.

Training, Validation, and Test



For training the ML models

For validation: choosing the parameter or model

For testing the "real" performance and generalization

Example: Assume I want to build a Random Forest. I have a parameter to decide: shall I have

- 100 Trees?
- 200 Trees?

What we do next is to use the **training set** to train two classifiers,
1) C_1: Random Forest with 100 trees, and 2) C_2: Random Forest with 200 trees

Apply C_1, C_2 on the validation set.

They have the following accuracy:

1. **C_1: Random Forest with 100 trees**: validation accuracy 90%
2. **C_2: Random Forest with 200 trees**: validation accuracy 88%

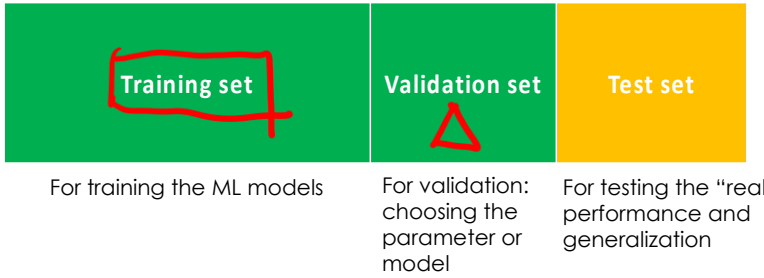
to choose model, look at validation accuracy, then training accuracy, then find the one that used FEWER number of parameters.

Which one to choose for real application, i.e., testing?
The one with higher validation accuracy, i.e., Random Forest with 100 trees!

C_1 → test set

© Copyright EE, NUS. All Rights Reserved.

Python Demo: lec10.ipynb



Load Dataset Split it into Train:Val:Test = 100:25:25

[76]:
```
##--- load data from scikit ---##
import numpy as np
import pandas as pd
print("pandas version: {}".format(pd.__version__))
import sklearn
print("scikit-learn version: {}".format(sklearn.__version__))
from sklearn.datasets import load_iris

## Set Seed
seed = 20

## Load dataset
iris_dataset = load_iris()
X = np.array(iris_dataset['data'])
y = np.array(iris_dataset['target'])

## one-hot encoding
Y = list()
for i in y:
    letter = [0, 0, 0]
    letter[i] = 1
    Y.append(letter)
Y = np.array(Y)

## Random shuffle data and train-test split
test_Idx = np.random.RandomState(seed=seed).permutation(Y.shape[0])
X_test = X[test_Idx[:25]]
Y_test = Y[test_Idx[:25]]
X = X[test_Idx[25:]]
Y = Y[test_Idx[25:]]
```

- Problem Setup
  - Dataset used: IRISdataset
    - Link: https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-dataset
  - Training/Validation/Test: 100/25/25
  - Machine Learning Task and Model: Polynomial regression
  - Parameters to select: Order 1 to 10

In the Final, no coding questions for Xinchao's part!

© Copyright EE, NUS. All Rights Reserved.

k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

k = 4.

4-fold cross validation

Test

Step 1: take out ***test set*** from the dataset

means dont use test set

partition into k parts and run k times to use each part as validation (remainder as training)

© Copyright EE, NUS. All Rights Reserved.

k-fold Cross Validation

Dataset 1,000.

- In practice, we do the **k-fold cross validation**

4-fold cross validation

Test

200

Step 2: We partition the ***remaining part of the dataset*** (after taking out the test set), into ***k*** equal parts (equal in terms of number of samples).

k=4

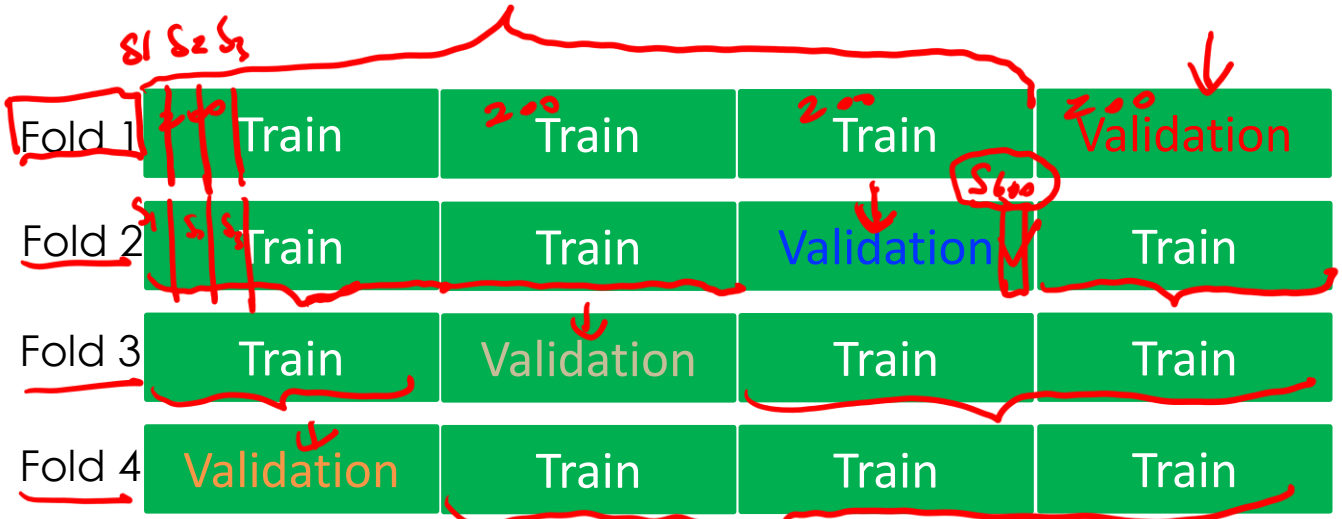
© Copyright EE, NUS. All Rights Reserved.

k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

Test

4-fold cross validation



Order of the samples are kept the same across all folds.

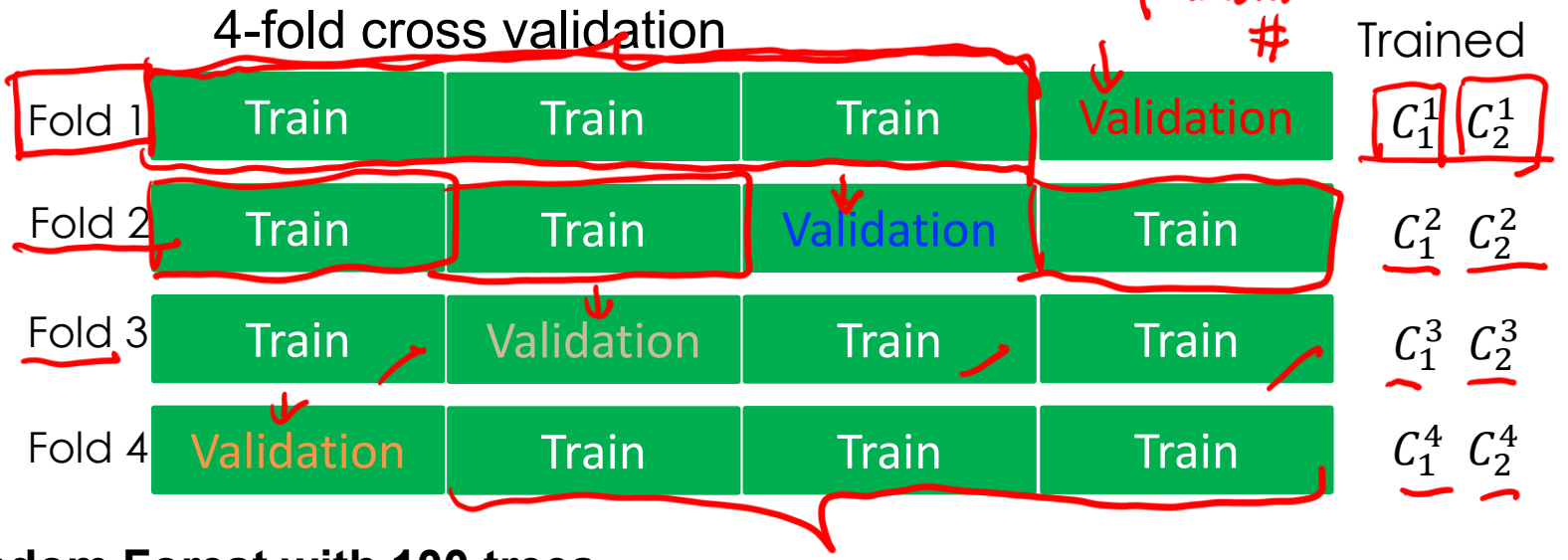
Step 3: We run ***k folds*** (i.e., k times) of experiments.
Within each fold, we use ***one part*** as ***validation set***, and the ***k-1 remaining parts*** as ***training set***. We use different validation sets for different folds.

© Copyright EE, NUS. All Rights Reserved.

k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

Test



1) C_1: **Random Forest with 100 trees**
2) C_2: **Random Forest with 200 trees**

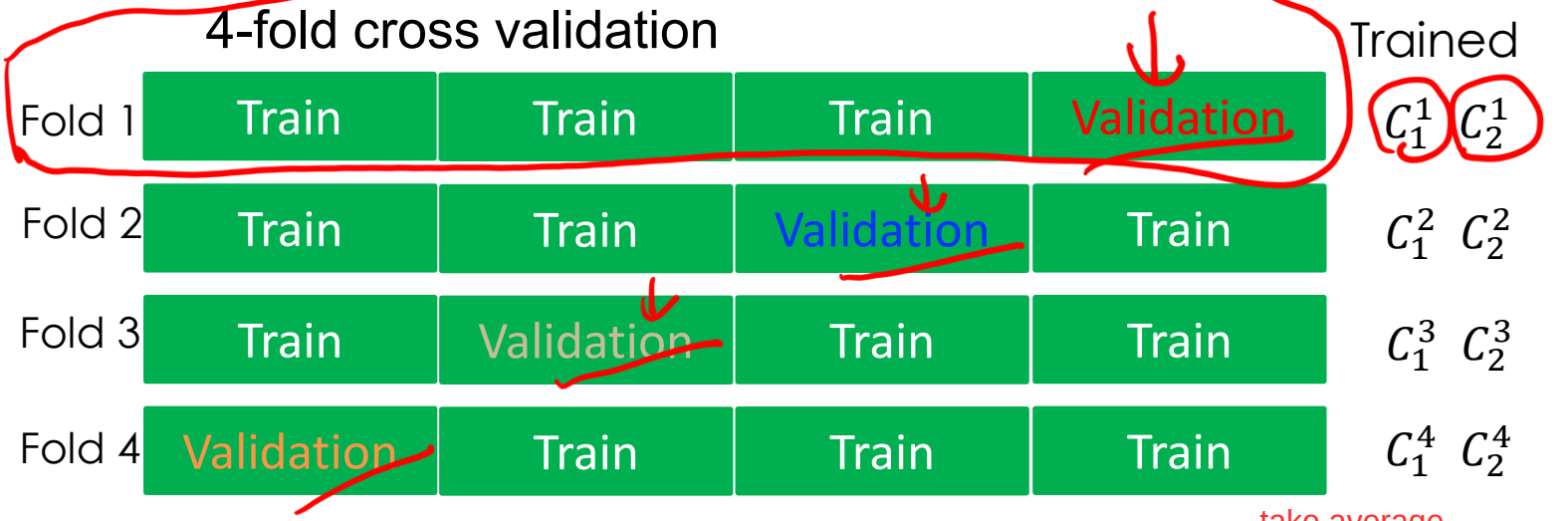
Step 3.1: Within each fold, if we have ***n*** parameter/model candidates, we will train ***n*** models, and we check their validation performance.

© Copyright EE, NUS. All Rights Reserved.

k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

Test



Example: which one to select for test?

take average accuracy

	Fold 1 Accuracy on Validation Set 1	Fold 2 Accuracy on Validation Set 2	Fold 3 Accuracy on Validation Set 3	Fold 4 Accuracy on Validation Set 4	Average Accuracy on All Validation Sets
Classifier with Param1 (e.g. 100 trees)	88% C_1^1	89% C_1^2	93% C_1^3	92% C_1^4	90.5%
Classifier with Param2 (e.g. 200 trees)	90% C_2^1	88% C_2^2	91% C_2^3	91% C_2^4	90%

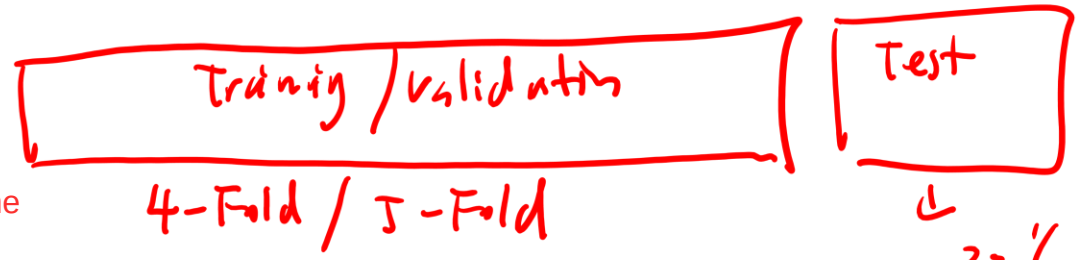
Step 4: We select the parameter/model with best average validation performance over k folds.

k-fold Cross Validation

CV

Other common partitioning:

- 10-Fold CV
- 5-Fold CV
- 3-Fold CV



partition out the test set first, remove it, and use the remainder to do this kfold cross validation

We may decide on the size of the test set, for example, 15%, 20%, 30% of the whole dataset, the rest for training/validation.

20%
30%

From CV (4-Fold), I know 100 trees is better.

1) C^3_1, val accuracy 93%

2) train another RF with 800 green samples.

© Copyright EE, NUS. All Rights Reserved.

POST!

k-fold Cross Validation

- The test set contains the examples that the learning algorithm has never seen before,
- So test performance shows how well our model ***generalizes***.

there is no connection betwee training and validation accura one does not lead to another

this is two step. after cross validation, then use test set to get test accuracy score

Example:

depth = 5, 6, 7 ?

Xinchao uses k-fold cross validation to obtain an optimal parameter for his model (e.g., decision tree). This parameter, on the test set, achieves accuracy of 0.8.

trees = 100, 200, 300?

Helen uses uses k-fold cross validation to obtain an optimal parameter for her model (e.g., random forest). This parameter, on the test set, achieves accuracy of 0.9.

We can say, Helen's model ***generalizes*** better than Xinchao's model.

Benchmarking

Take home message:

Validation performance -> Selecting parameters!

Test performance -> The "real" performance of a model with selected parameter!

© Copyright EE, NUS. All Rights Reserved.

Training, Validation, and Test

- Validation is however not always used:
  - Validation is used when you need to **pick parameters or models**
  - If you have no models or parameters to compare, you may consider partition the data into only training and test

© Copyright EE, NUS. All Rights Reserved.

Outline

- Dataset Partition:
  - Training/Validation/Testing
- Cross Validation
- Evaluation Metrics
  - Evaluating the quality of a trained machine learning system

We will talk about many metrics:
It is OK you can't memorize them all
But intuition is important!

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Regression

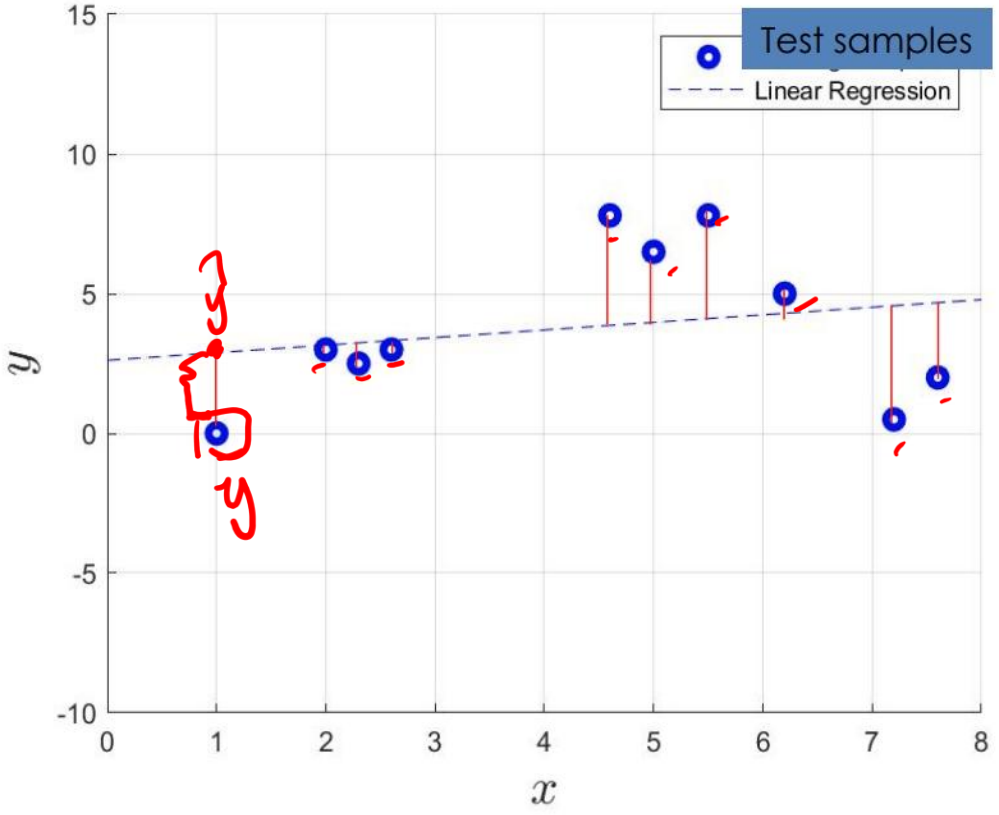
Mean Square Error

$$(\text{MSE} = \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n})$$

Mean Absolute Error

$$(\text{MAE} = \frac{\Sigma_{i=1}^{n}|y_i - \hat{y}_i|}{n})$$

where y_i denotes the target output and $\hat{y}_i$ denotes the predicted output for sample i.



© Copyright EE, NUS. All Rights Reserved.

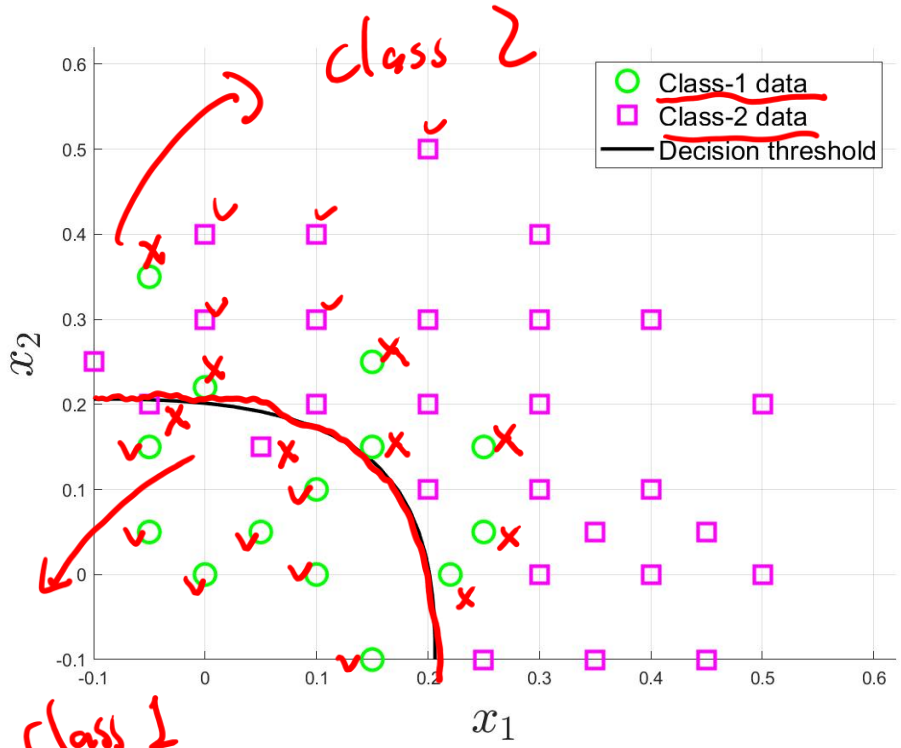
Evaluation Metrics

Classification

Class-1: Positive Class
Class-2: Negative Class

Confusion Matrix

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	7 (TP)	7 (FN)
Class-2 (actual)	2 (FP)	25 (TN)



TP: True Positive
FN: False Negative (i.e., **Type II Error**)
FP: False Positive (i.e., **Type I Error**)
TN: True Negative

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	7 (TP)	7 (FN)
Class-2 (actual)	2 (FP)	25 (TN)

Classification

- How many samples in the dataset have the real label of Class-2?

$2+25 = 27$

- How many samples are there in total?

$7+7+2+25 = 41$

- How many sample are correctly classified? How many are incorrectly classified?

$7+25 = 32$

$2+7 = 9$

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification

Confusion Matrix for Binary Classification

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	TP	FN
N (actual)	FP	TN

Recall = TP/(TP+FN)

Precision = TP/(TP+FP)

Accuracy = (TP+TN)/(TP+TN+FP+FN)

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification

Cost Matrix for Binary Classification

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$C_{p,p}$ * TP	$C_{p,n}$ * FN
N (actual)	$C_{n,p}$ * FP	$C_{n,n}$ * TN

Total cost:

$C_{p,p}$ * TP +
$C_{p,n}$ * FN +
$C_{n,p}$ * FP +
$C_{n,n}$ * TN

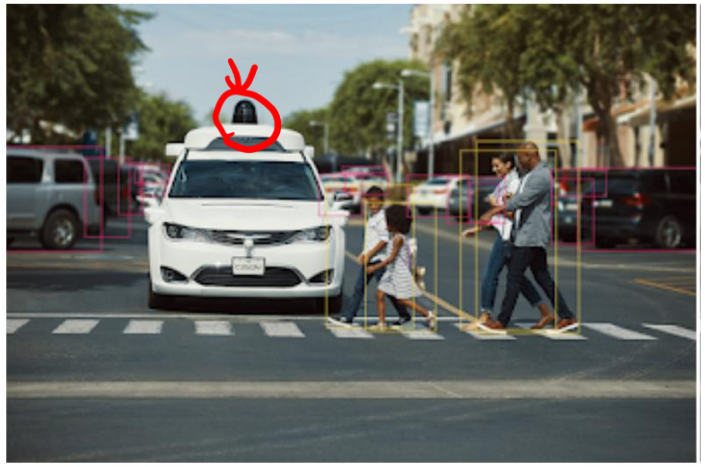
Main Idea: To assign different ***penalties*** for different entries. Higher penalties for more severe results. Smaller costs are preferred.

Usually, $C_{p,p}$ and $C_{n,n}$ are set to 0; $C_{n,p}$ and $C_{p,n}$ may and may not equal

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

- Example of cost matrix
  - Assume we would like to develop a self-driving car system
  - We have an ML system that detects the pedestrians using camera, by conducing a binary classification
    - When it detects a person (positive class), the car should stop
    - When no person is detected (negative class), the car keeps going



Credit: automotiveworld.com

True Positive (cost $C_{p,p}$) = 0
There is person, ML detects person and car stops

True Negative (cost $C_{n,n}$) = 0
There is no person, car keeps going

penalties.

False Positive (cost $C_{n,p}$)
There is no person, ML detects person and car stops

False Negative (cost $C_{p,n}$)
There is person, ML fails to detect person and car keeps going

↑ higher penalty.

$C_{n,p}$? $C_{p,n}$ (>, <, or =)

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

- For unbalanced data…
  - Assume we have 1000 samples, of which 10 are *positive* and 990 are *negative*
  - Accuracy = 990/1000=0.99! Very high number!
  - Yet, half of the Class-1 are Classified to Class-2!

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	5 (TP)	5 (FN)
Class-2 (actual)	5 (FP)	985 (TN)

The goal is to highlight the problems of the results!

In this case, we shall

1) Use cost matrix, assign different costs for each entry
2) Use Precision and Recall! Precision = 0.5 and Recall = 0.5

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification

(True Positive Rate) TPR = TP/(TP+FN) Recall
(False Negative Rate) FNR = FN/(TP+FN)

(True Negative Rate) TNR = TN/(FP+TN)
(False Positive Rate) FPR = FP/(FP+TN)

TPR + FNR = 1 (100% of positive-class data)
TNR + FPR = 1 (100% of negative-class data)

only TP + FN (+ve class)
and FP + TN (-ve class)
must make up to 100%

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	TP	FN
N (actual)	FP	TN

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification

Prediction function $y = f(x)$

$f(-4) = -1.1$

$f(-3) = -0.5$

sample	N1	N2	P1	N3	P2	P3
input x	-4	-3	-2.5	-2	-1.5	-0.5
Prediction y	-1.1	-0.5	-0.1	0.2	0.6	0.9
Actual Label	-1	-1	1	-1	1	1

If threshold set to be y=0, $f(\cdot) < 0$ predict to be Negative

N3, P2, P3 will be taken as +1

P1, N2, N1 will be taken as -1

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$TP = 2$ P2 P3	$FN = 1$ P1
N (actual)	$FP = 1$ N3	$TN = 2$ N1, N2

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification

Prediction function y = f(x)

We can change the threshold!

Negative ← → Positive

sample	N1	N2	P1	N3	P2	P3
input x	-4	-3	-2.5	-2	-1.5	-0.5
Prediction y	-1.1	-0.5	-0.1	0.2	0.6	0.9
Actual Label	-1	-1	1	-1	1	1

If threshold set to be y=0.4,
P2, P3 will be taken as +1
N3, P1, N2, N1 will be taken as -1

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$TP = 2$ P2 P3	$FN = 1$ P1
N (actual)	$FP = 0$	$TN = 3$ N1 N2 N3

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification:

TP, FP, FN, TN will change wrt thresholds!

If threshold set to be y=0,
N3, P2, P3 will be taken as +1
P1, N2, N1 will be taken as -1

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$TP = 2$	$FN = 1$
N (actual)	$FP = 1$	$TN = 2$

If threshold set to be y=0.4,
P2, P3 will be taken as +1
N3, P1, N2, N1 will be taken as -1

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$TP = 2$	$FN = 1$
N (actual)	$FP = 0$	$TN = 3$

© Copyright EE, NUS. All Rights Reserved.

Evaluation Metrics

Classification

Confusion Matrix for Multicategory Classification

	$P_{\hat{1}}$ (predicted)	$P_{\hat{2}}$ (predicted)		$P_{\hat{C}}$ (predicted)
P_1 (actual)	$P_{1,\hat{1}}$	$P_{1,\hat{2}}$	...	$P_{1,\hat{C}}$
P_2 (actual)	$P_{2,\hat{1}}$	$P_{2,\hat{2}}$	...	$P_{2,\hat{C}}$
⋮	⋮	⋮	.....	⋮
P_C (actual)	$P_{C,\hat{1}}$	$P_{C,\hat{2}}$		$P_{C,\hat{C}}$

© Copyright EE, NUS. All Rights Reserved.

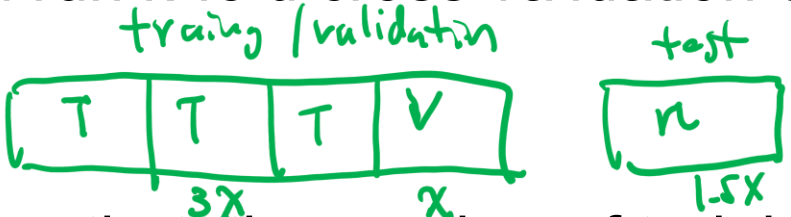
Other Issues

- Computational speed and memory consumptions are also important factors
  - Especially for mobile or edge devices
- Other factors
  - Parallelable, Modularity, Maintainability
- Not focus of this module

© Copyright EE, NUS. All Rights Reserved.

Practice Question

Suppose we have a dataset of 550 samples. We take out ***n*** samples as test set, and run k-fold cross validation on the remaining samples.



Within each fold, we know that, the number of training samples is **three times as large as** the number of validation samples, and **two times as large as** the number of test samples.

1. What is ***k***?
2. What is ***n***?

$k = 4$

$3X + X + 1.5X = 550$

$X = 100$

$n = 1.5X = 150$

© Copyright EE, NUS. All Rights Reserved.

EE2211 Introduction to Machine Learning

Lecture 11

Wang Xinchao
xinchao@nus.edu.sg

© Copyright EE, NUS. All Rights Reserved.

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Linear Algebra, Probability and Statistics
- Fundamental Machine Learning Algorithms I (Helen)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Helen)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

[Important] In the Final, no coding questions for Xinchao's part!

Despite you will see some in the tutorial, they won't be tested.

© Copyright EE, NUS. All Rights Reserved.

Outline

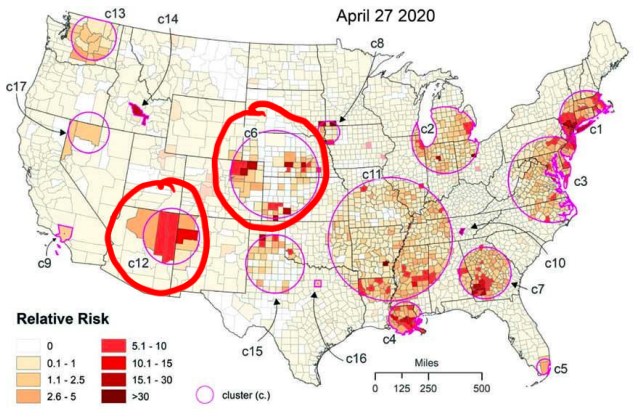
- Introduction of unsupervised learning
- K-means Clustering
  - The most popular clustering technique
- Fuzzy Clustering

Hard Clustering

Soft Clustering

© Copyright EE, NUS. All Rights Reserved.

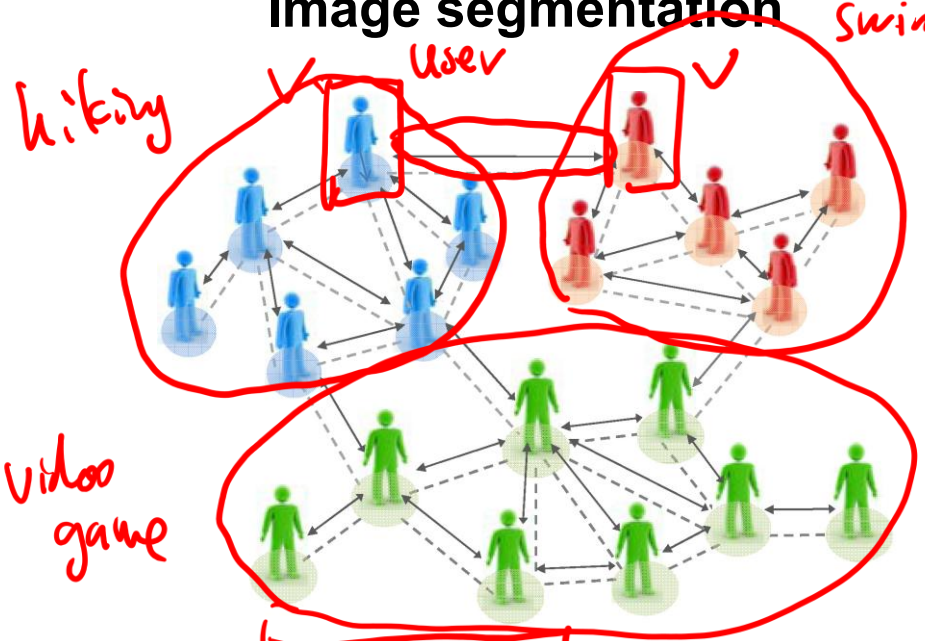
Unsupervised Learning



Discovering Covid clusters

Image segmentation

Business analysis



Community detection in social networks

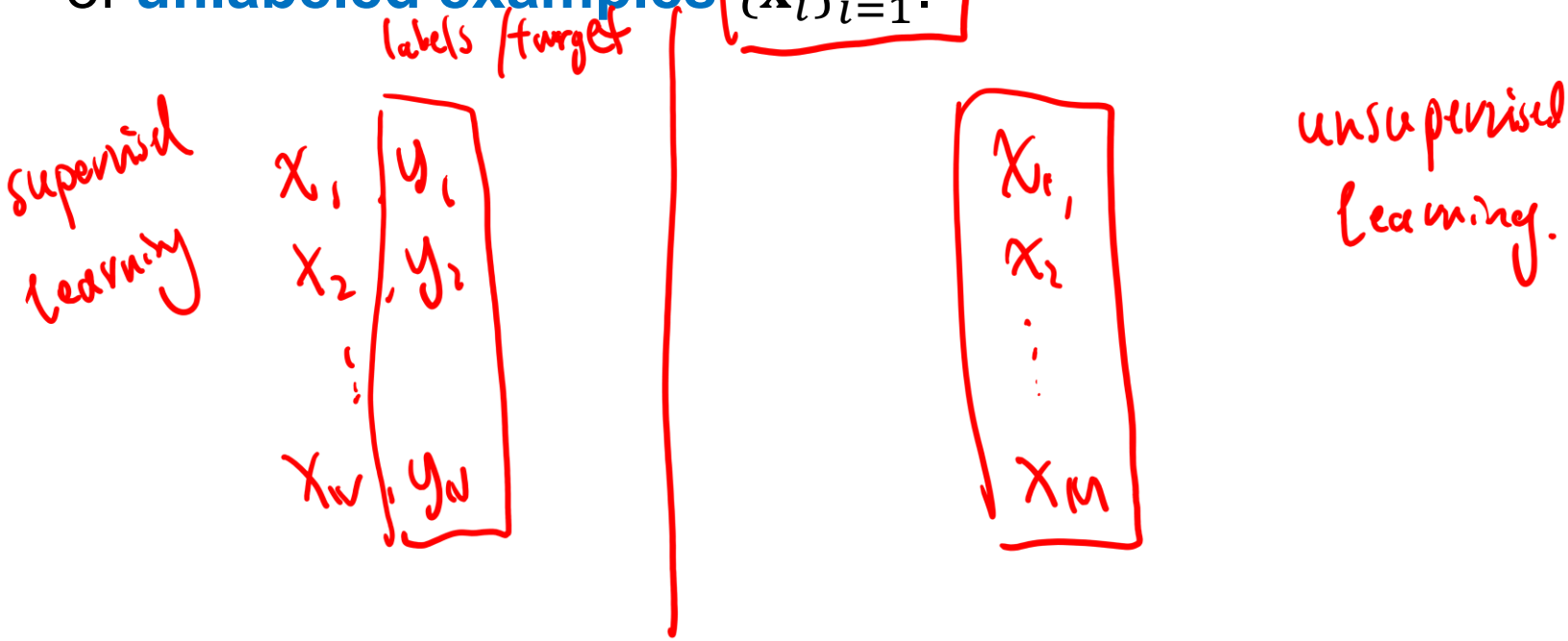
© Copyright EE, NUS. All Rights Reserved.

Unsupervised Learning

Introduction

Motivation: we do not always have labeled data.

In **unsupervised learning**, the dataset is a collection of **unlabeled examples** $\{\mathbf{x}_i\}_{i=1}^{M}$.



© Copyright EE, NUS. All Rights Reserved.

Unsupervised Learning

Introduction

Evaluation of unsupervised learning is hard:

- The absence of labels representing the desired behavior for your model means the absence of a solid reference point to judge the quality of your model.

© Copyright EE, NUS. All Rights Reserved.

Unsupervised Learning

Main Tasks/Approaches

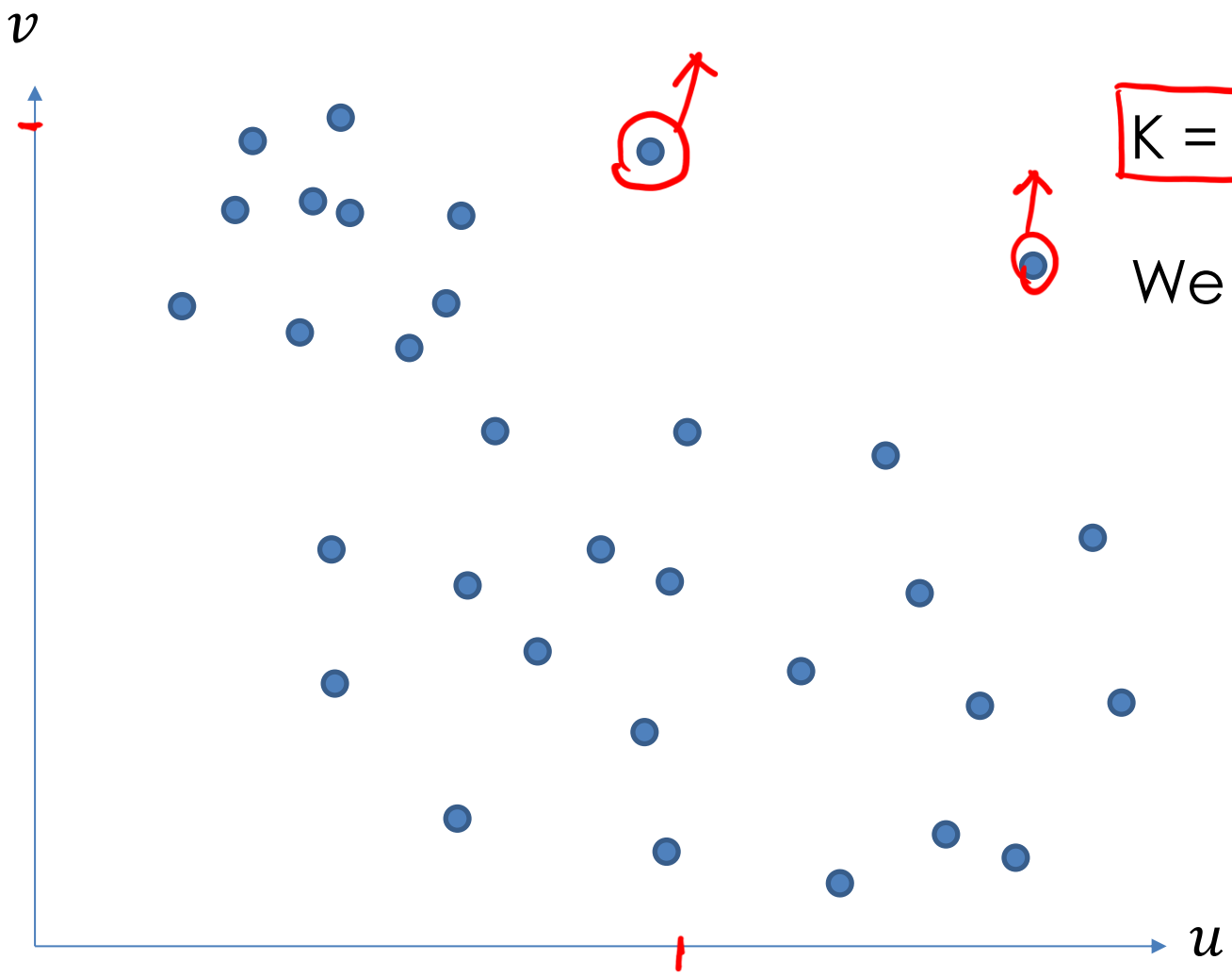
clustering is Not only one task of unsupervised learning.

- **Clustering**
  - ✓ Groups a set of objects in such a way that objects in the same group (called a **cluster**) are **more similar** (in some sense) to each other than to those in other groups (clusters).
- **Density Estimation**
  - ✓ Models the probability density function (pdf) of the unknown probability distribution from which the dataset has been drawn.
- **Component Analysis**
  - ✓ Breaks down the data from the perspective of signal analysis.
- **Unsupervised Neural Networks**
  - ✓ Autoencoder

PMF

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering (2D)

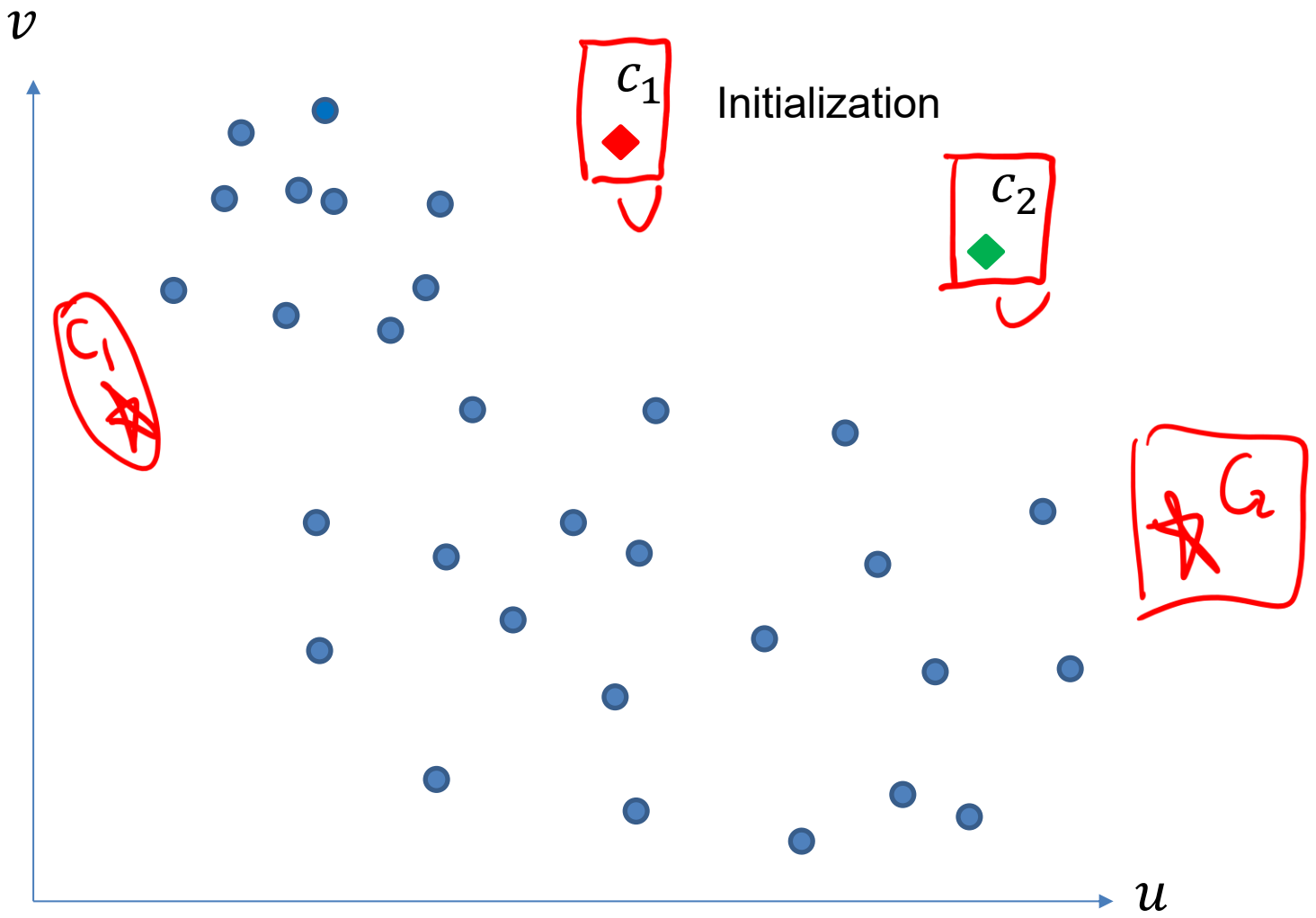


K = 2

We seek 2 clusters

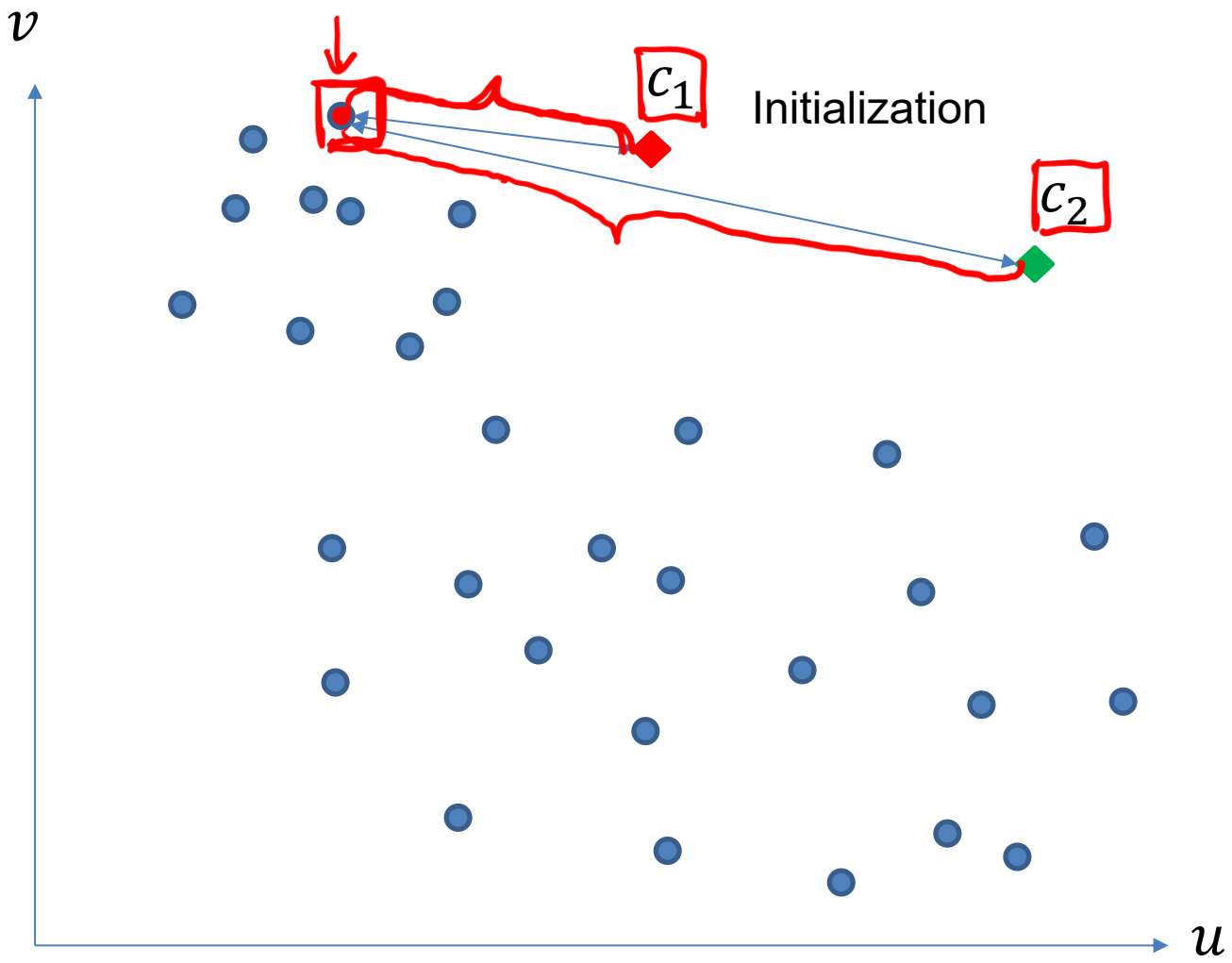
© Copyright EE, NUS. All Rights Reserved.

K-means Clustering



© Copyright EE, NUS. All Rights Reserved.

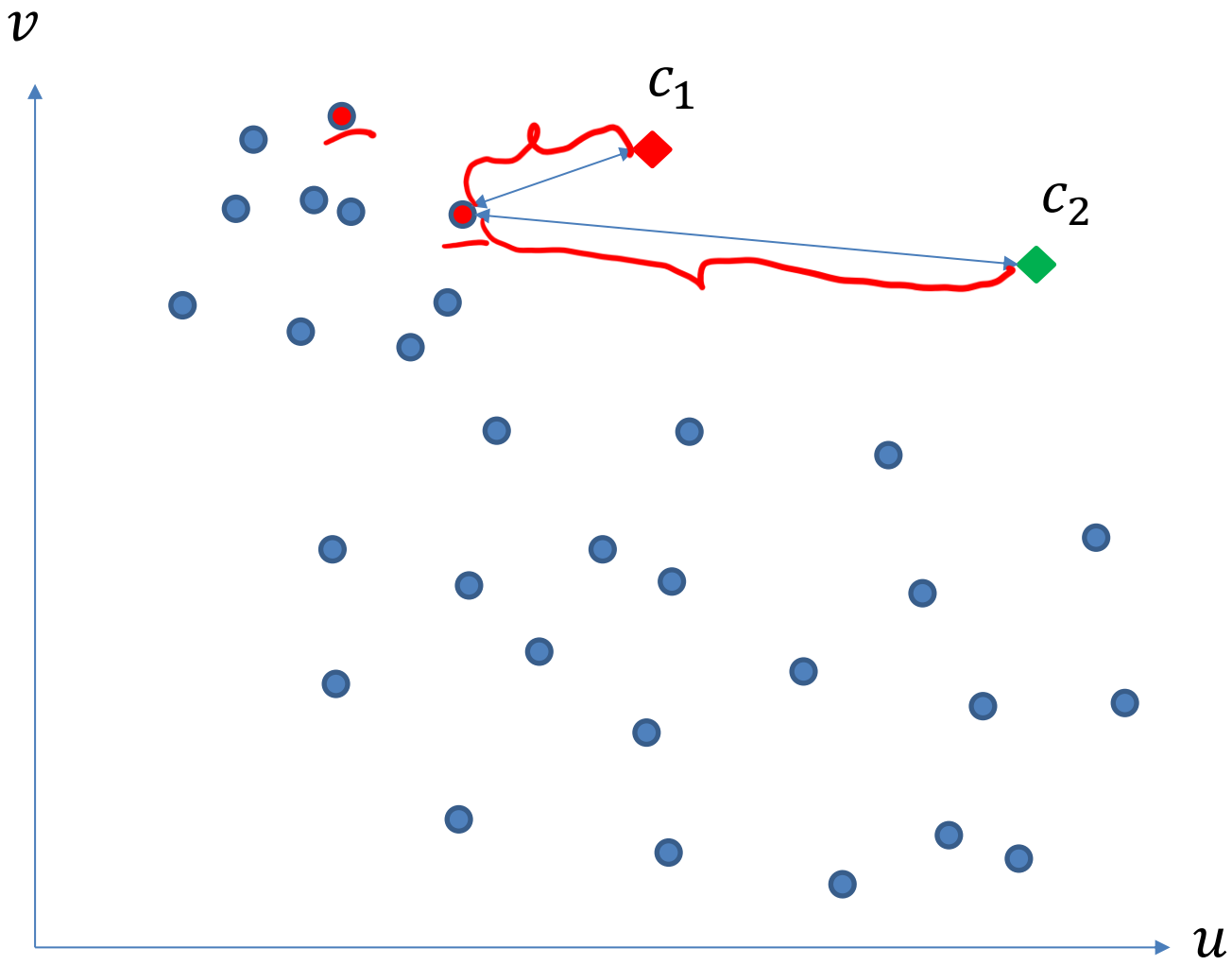
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

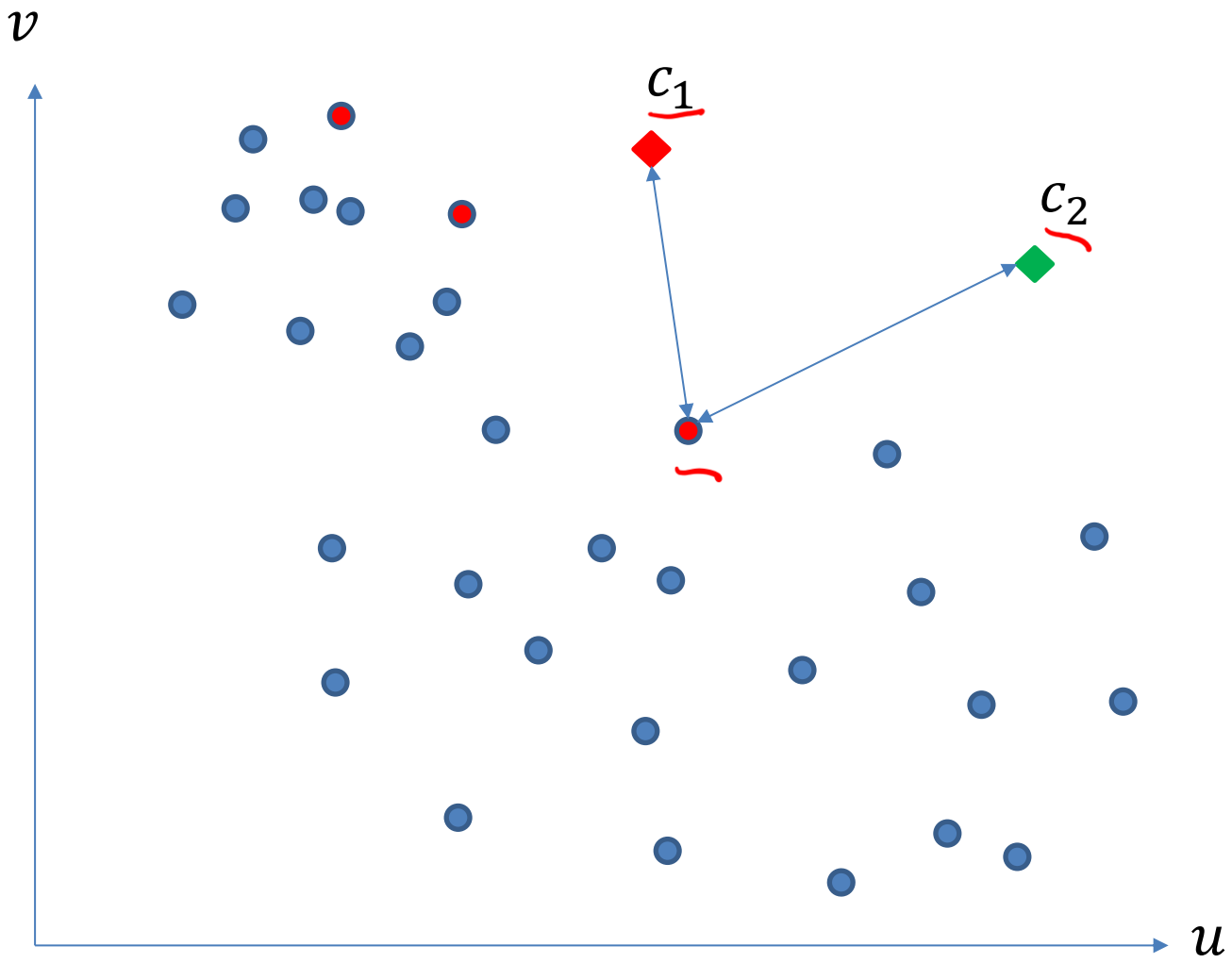
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

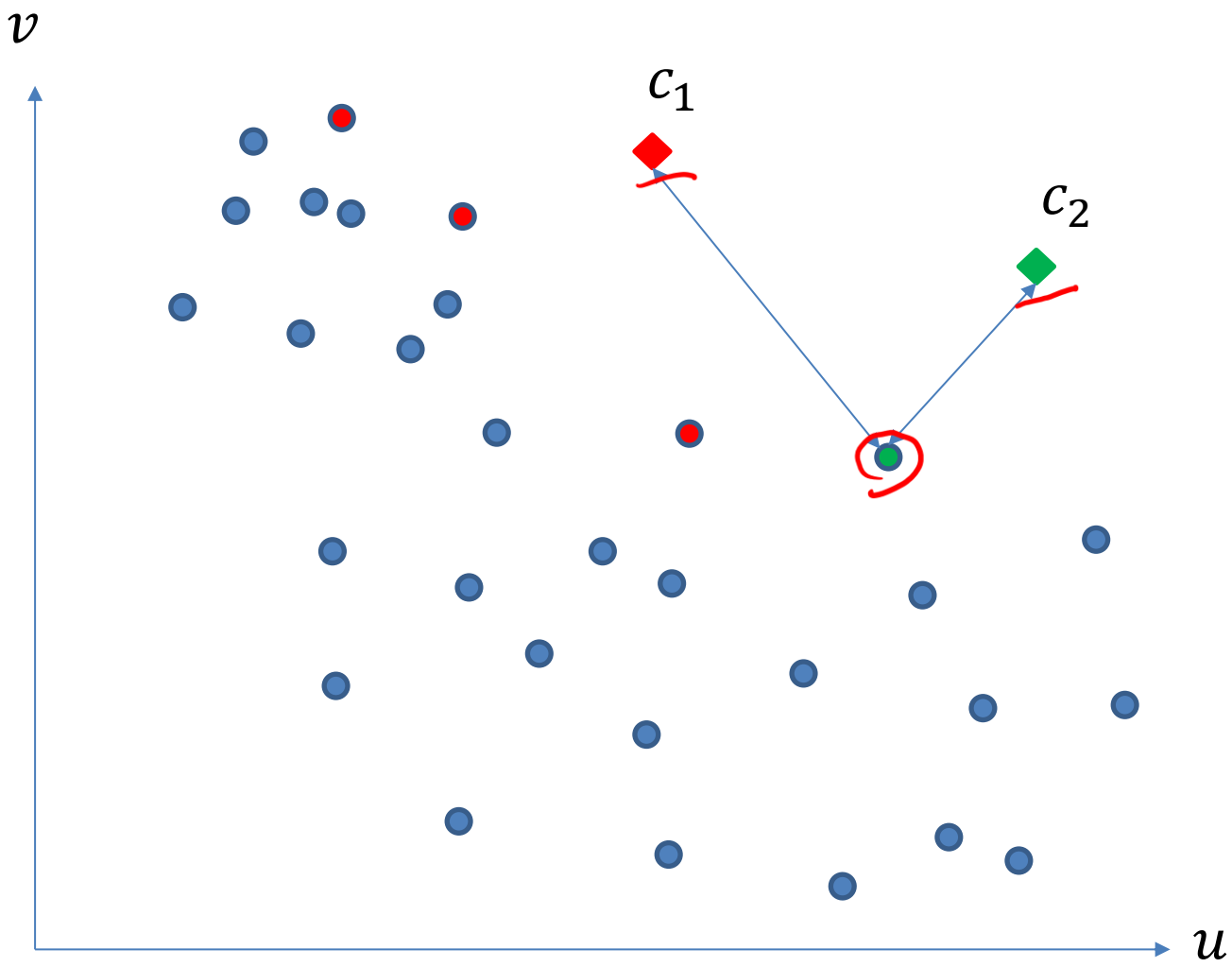
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

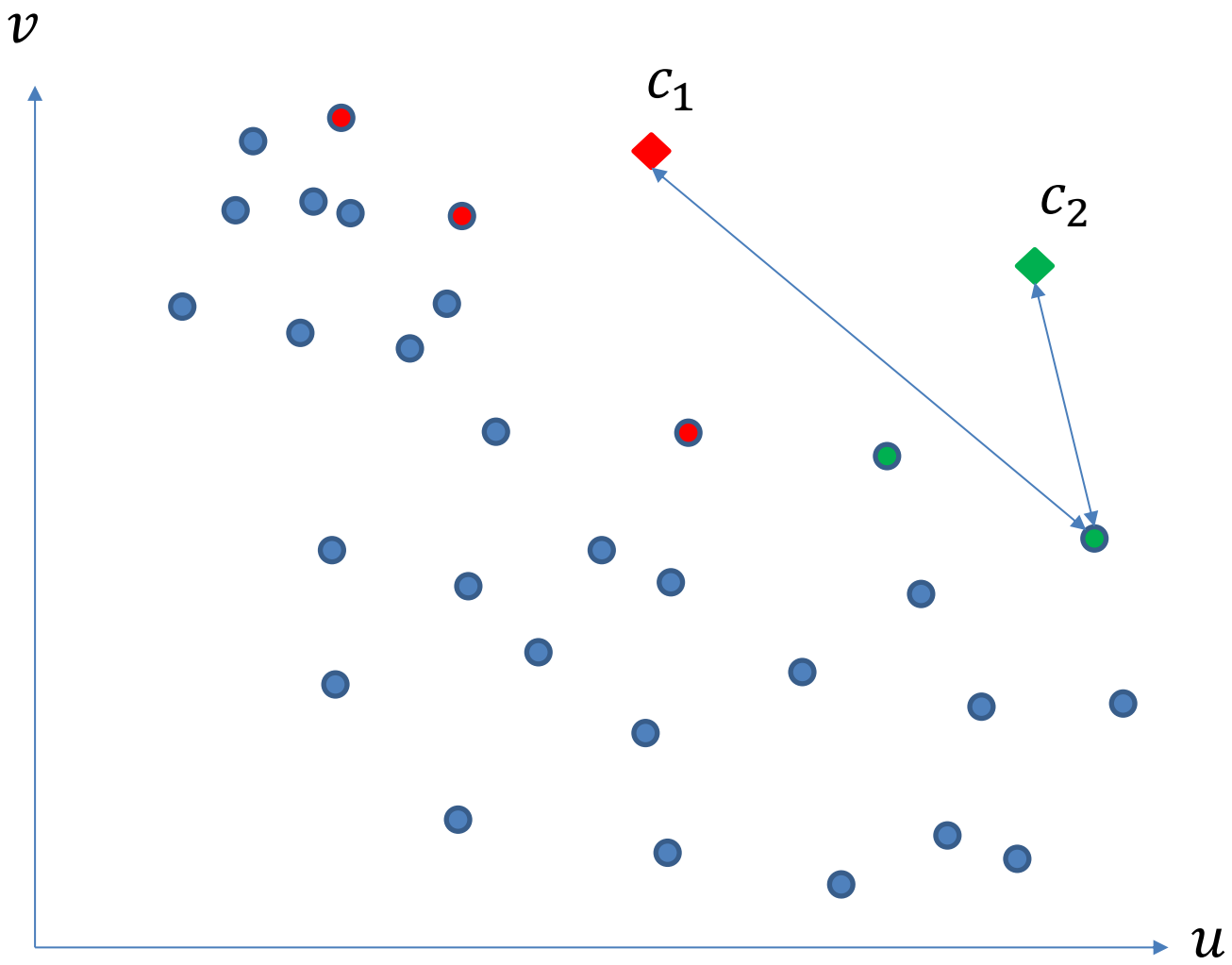
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

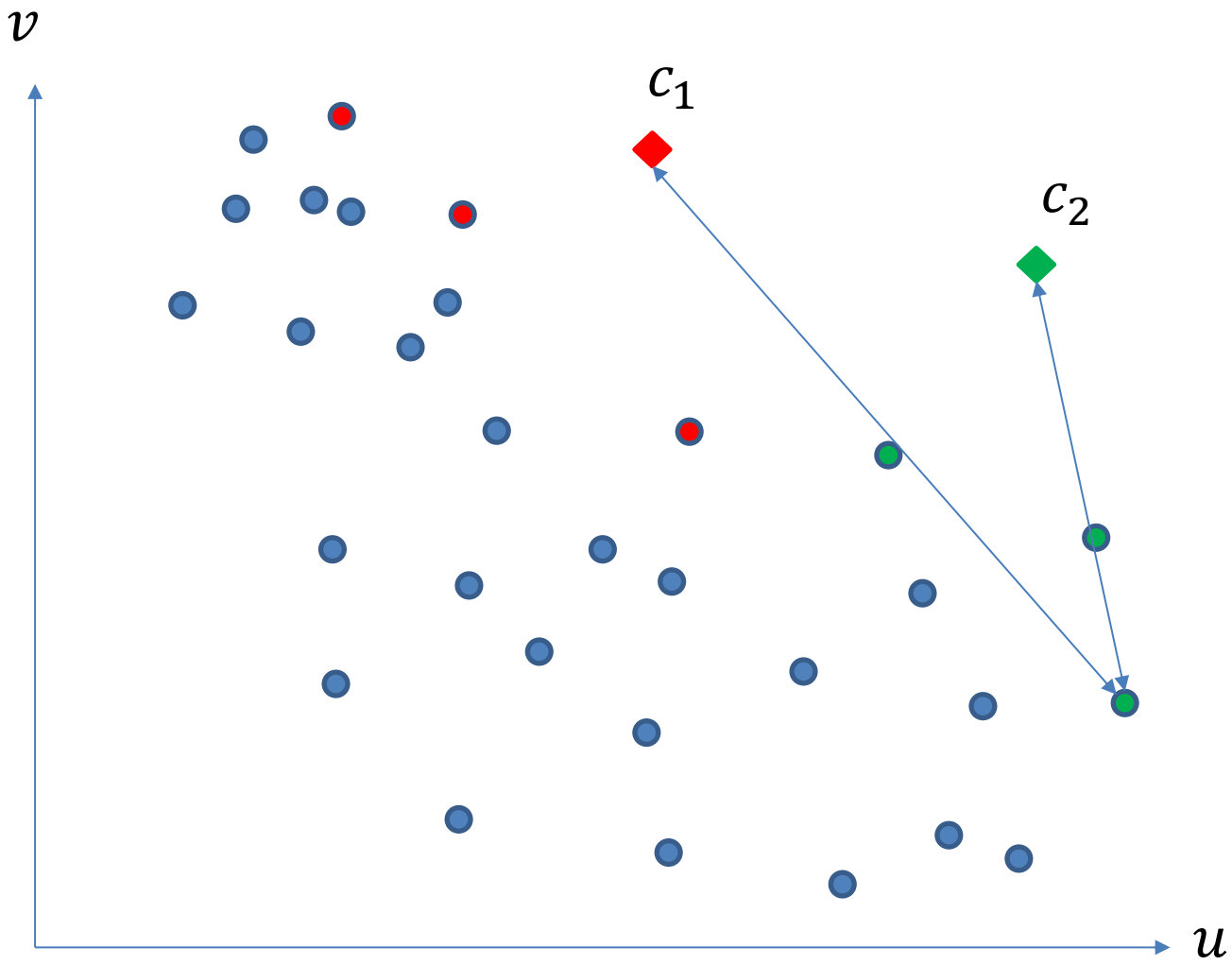
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

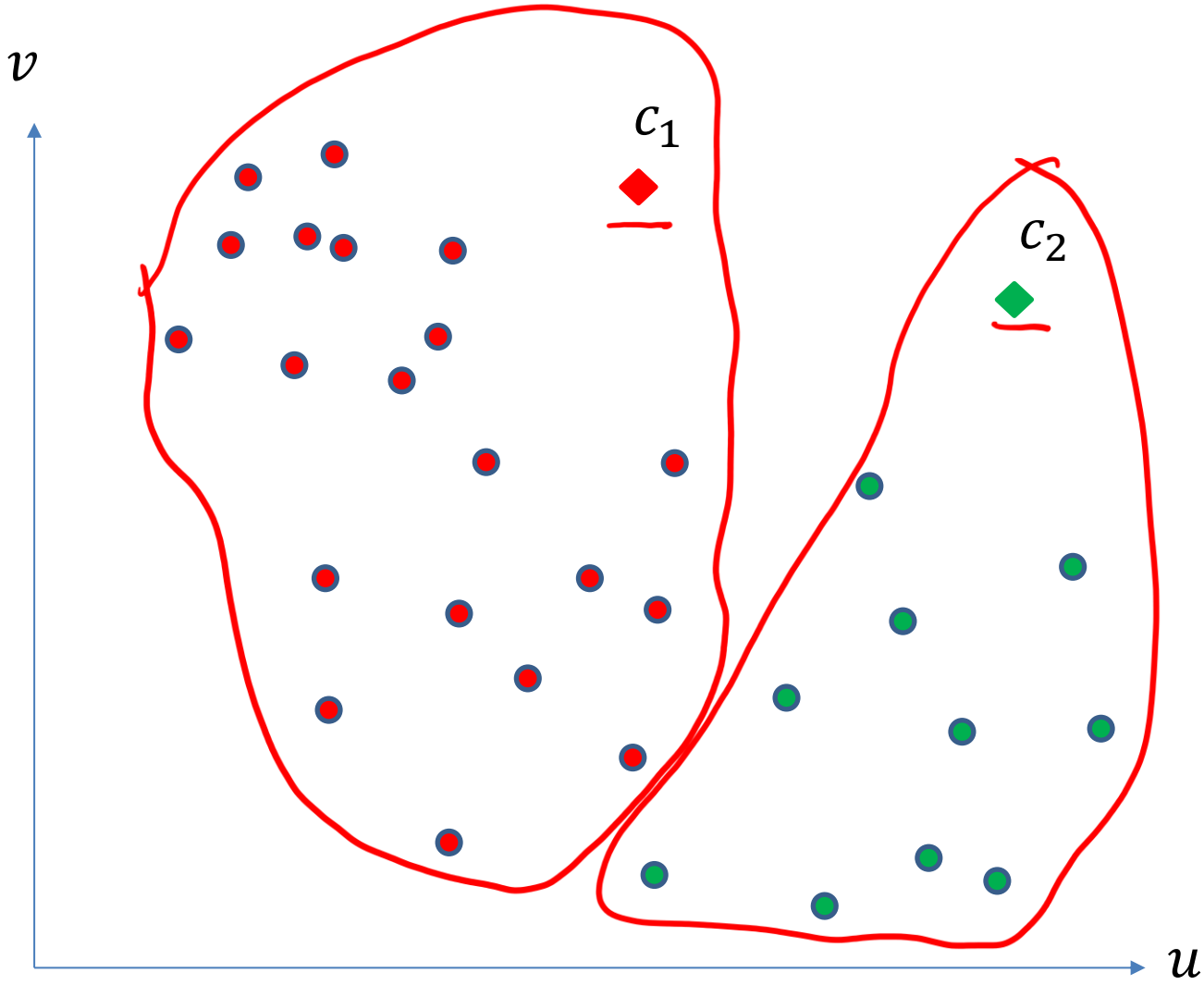
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

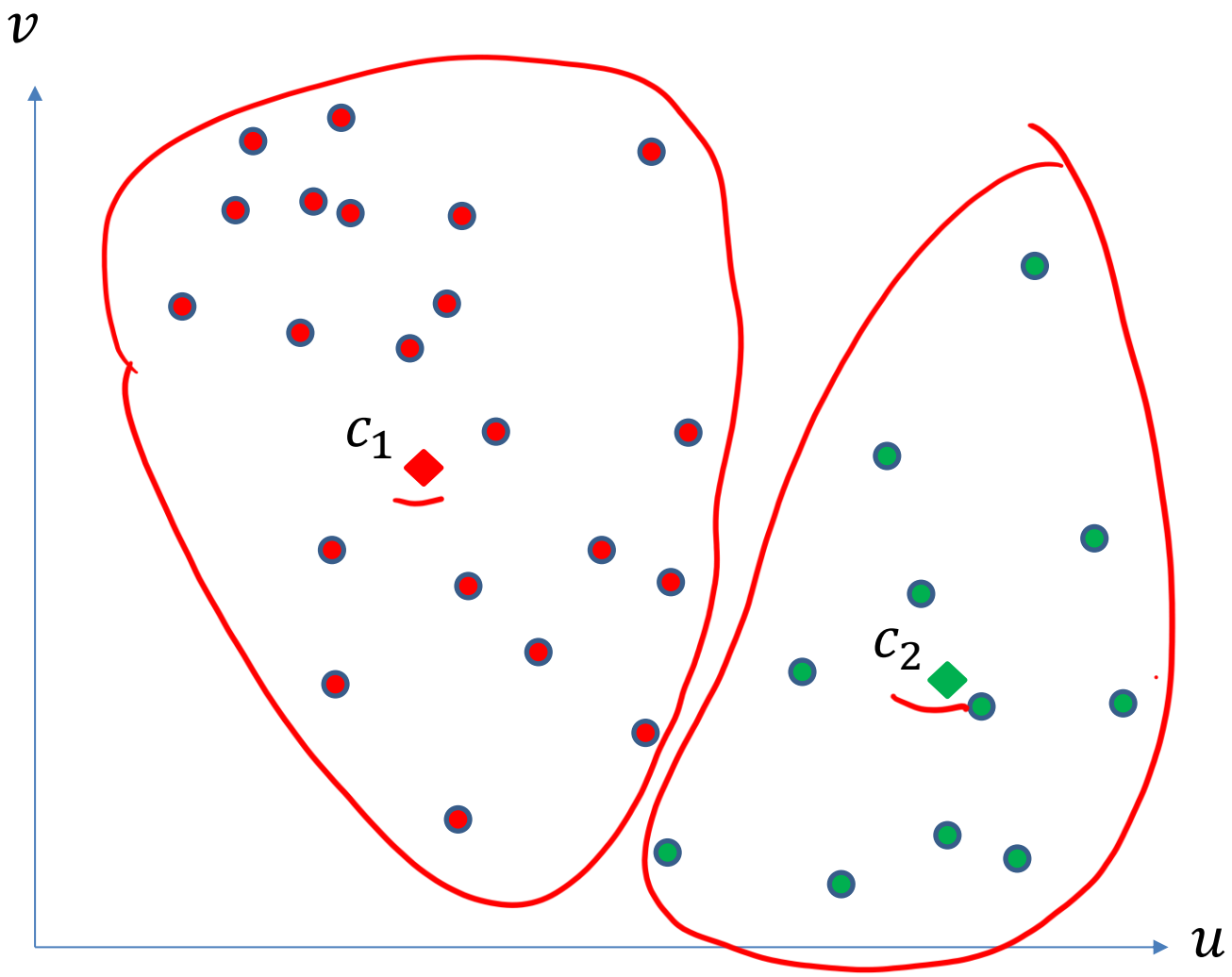
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

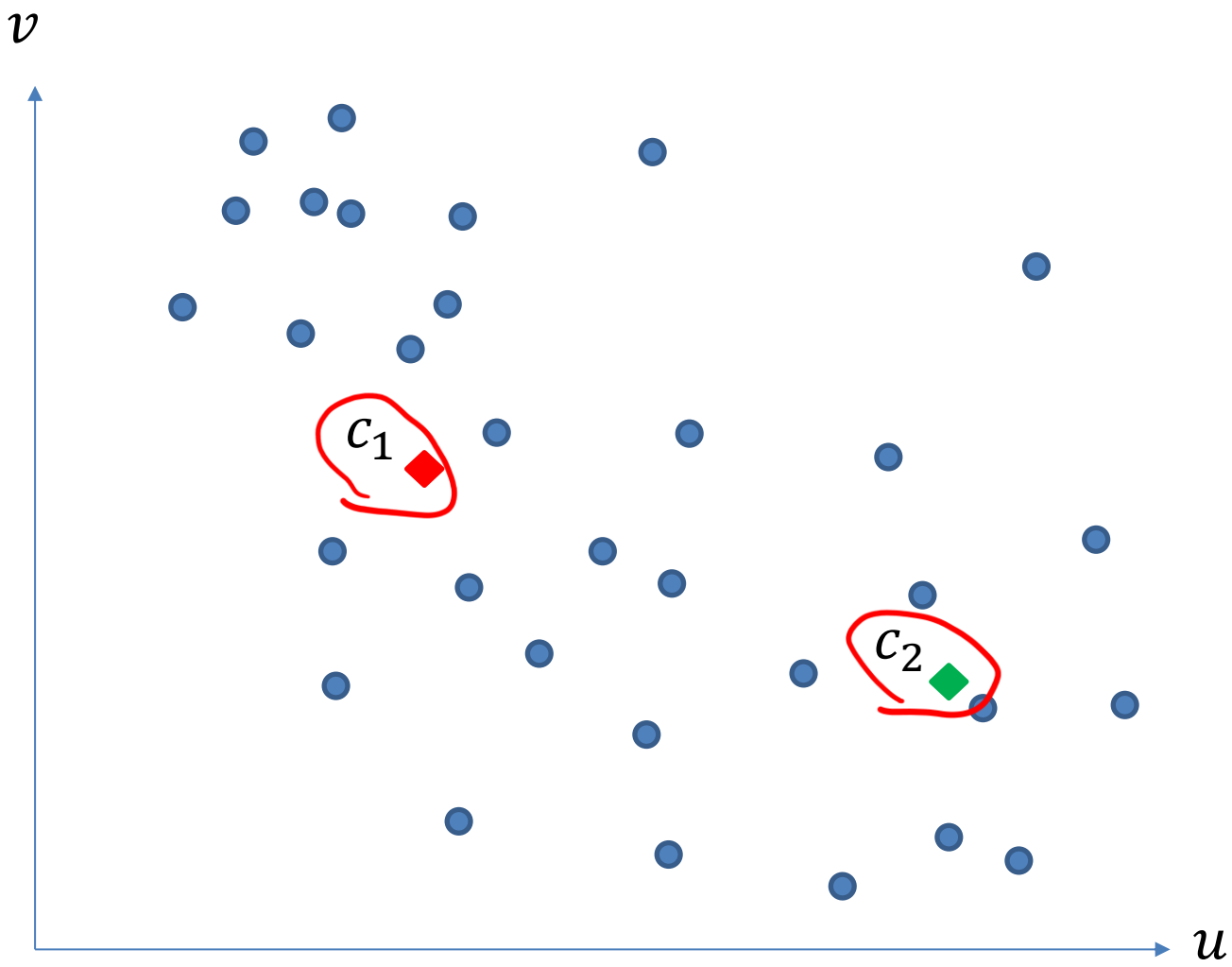
K-means Clustering



Centroid Update

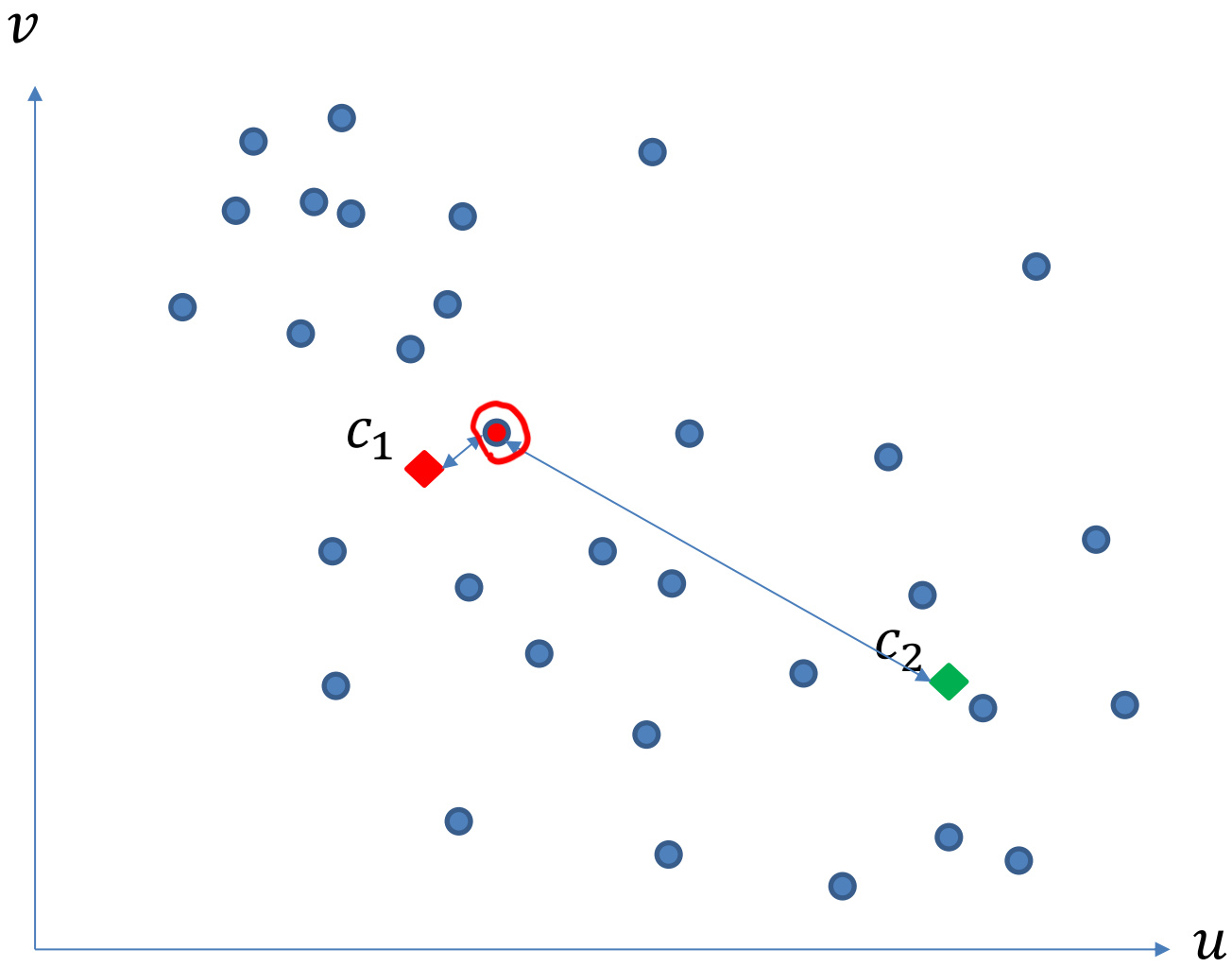
© Copyright EE, NUS. All Rights Reserved.

K-means Clustering



© Copyright EE, NUS. All Rights Reserved.

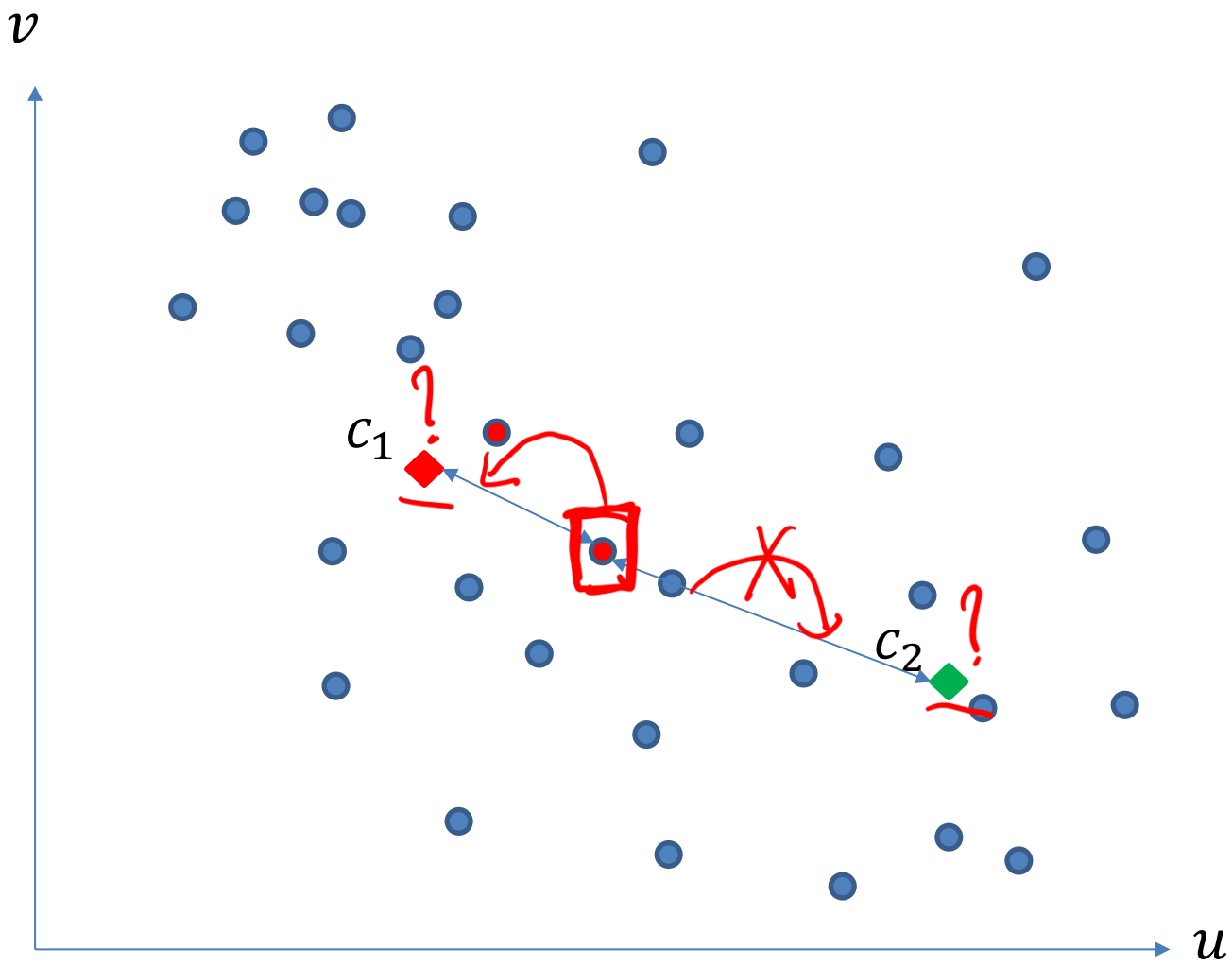
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

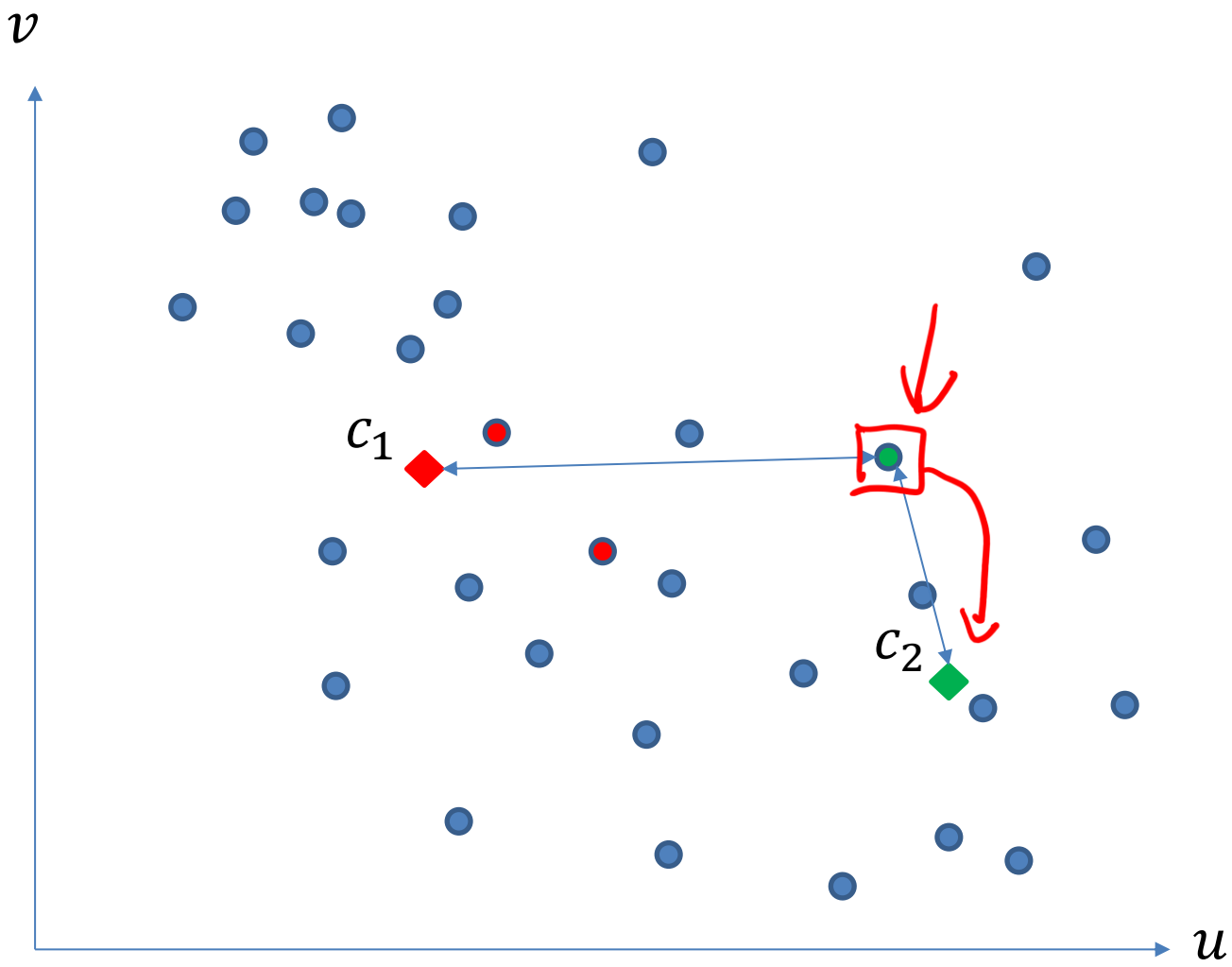
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

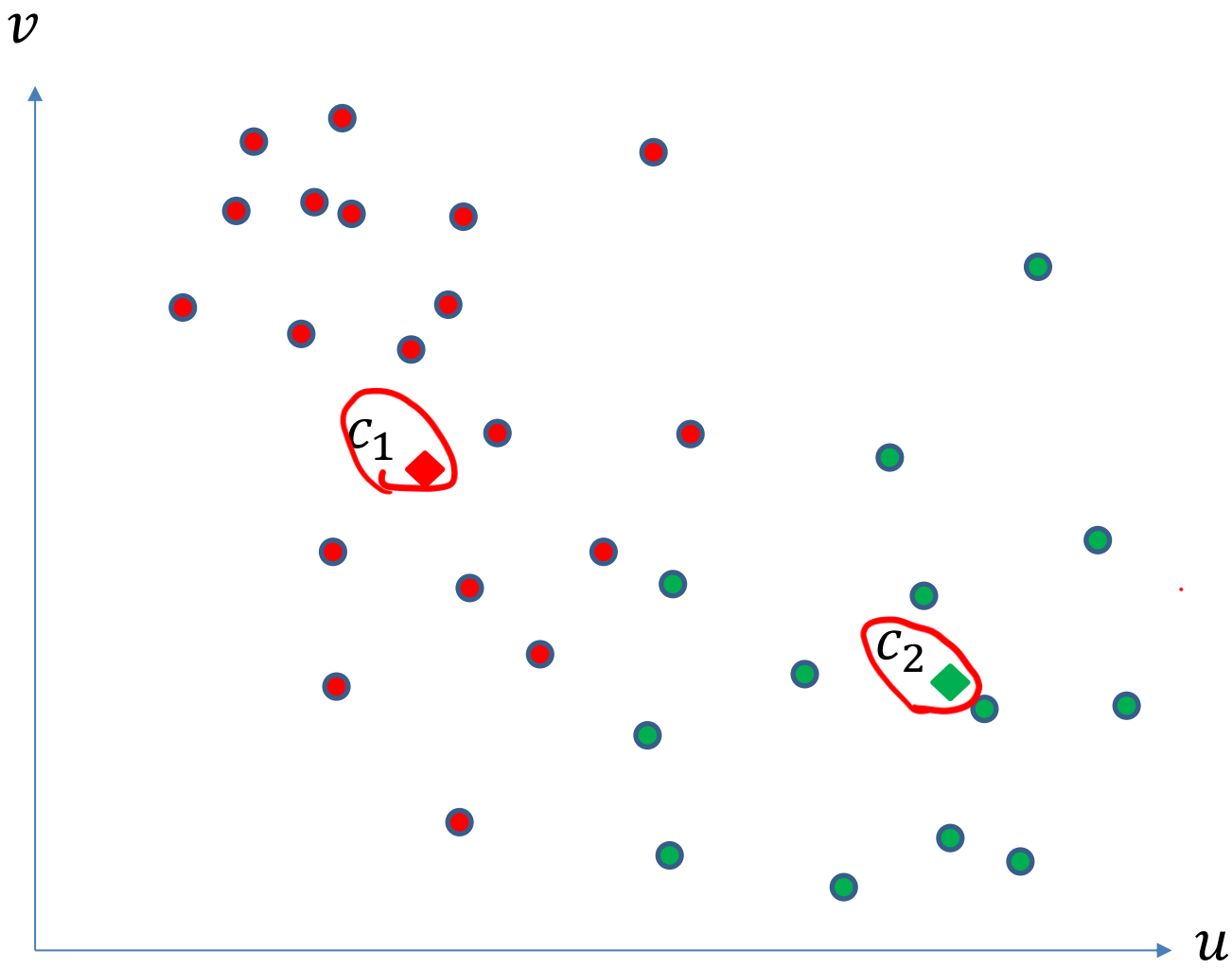
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

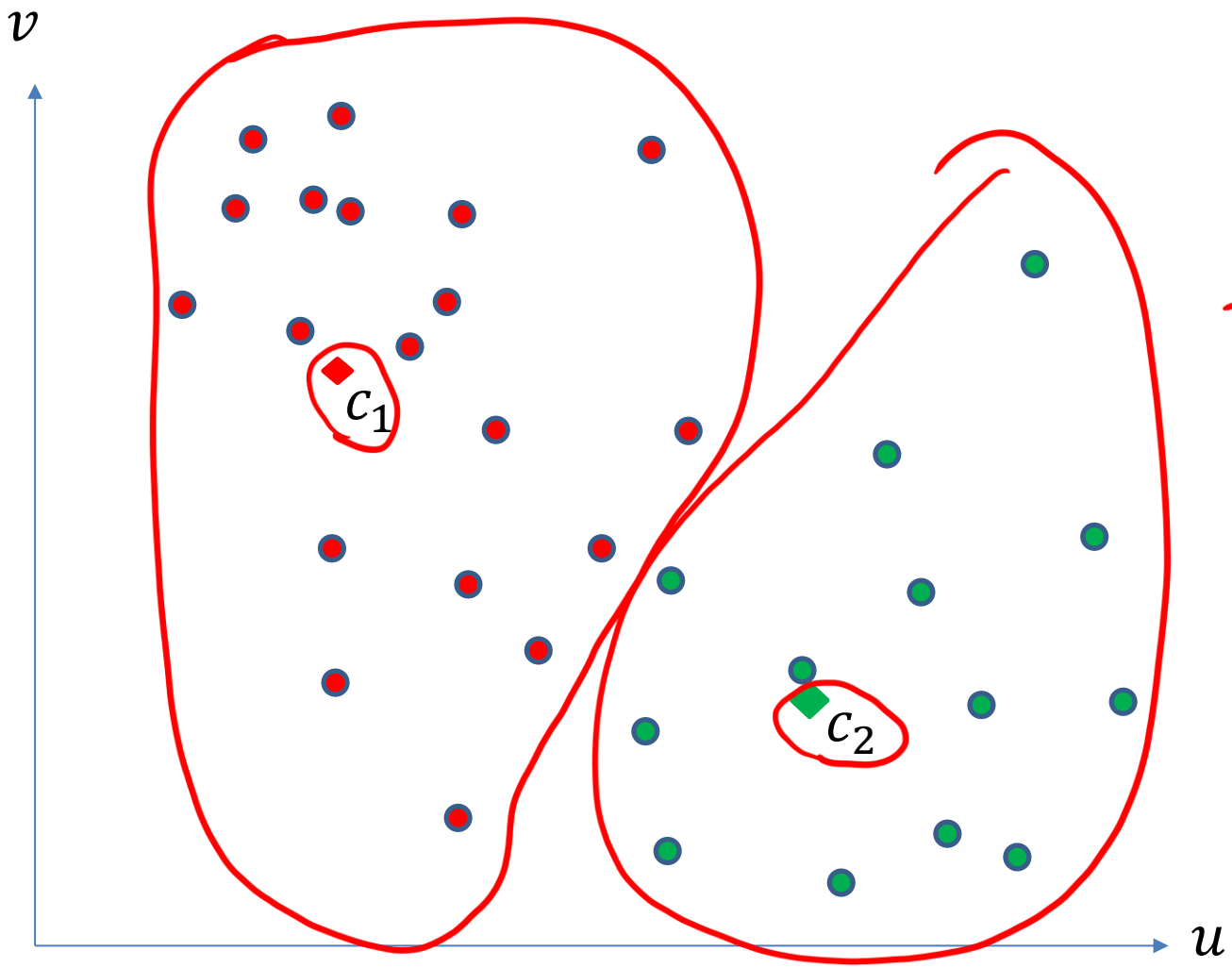
K-means Clustering



Assignment

© Copyright EE, NUS. All Rights Reserved.

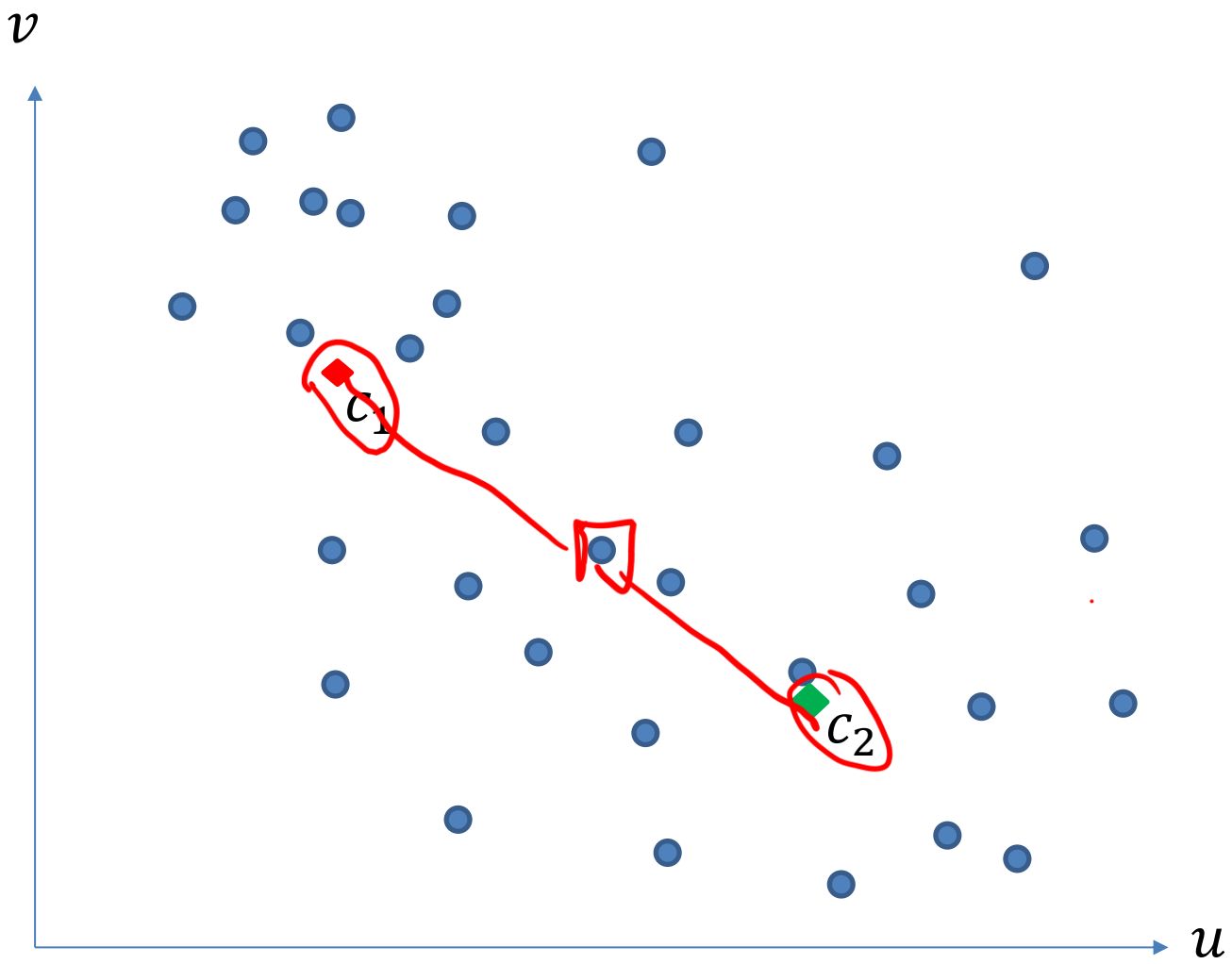
K-means Clustering



Centroid Update

© Copyright EE, NUS. All Rights Reserved.

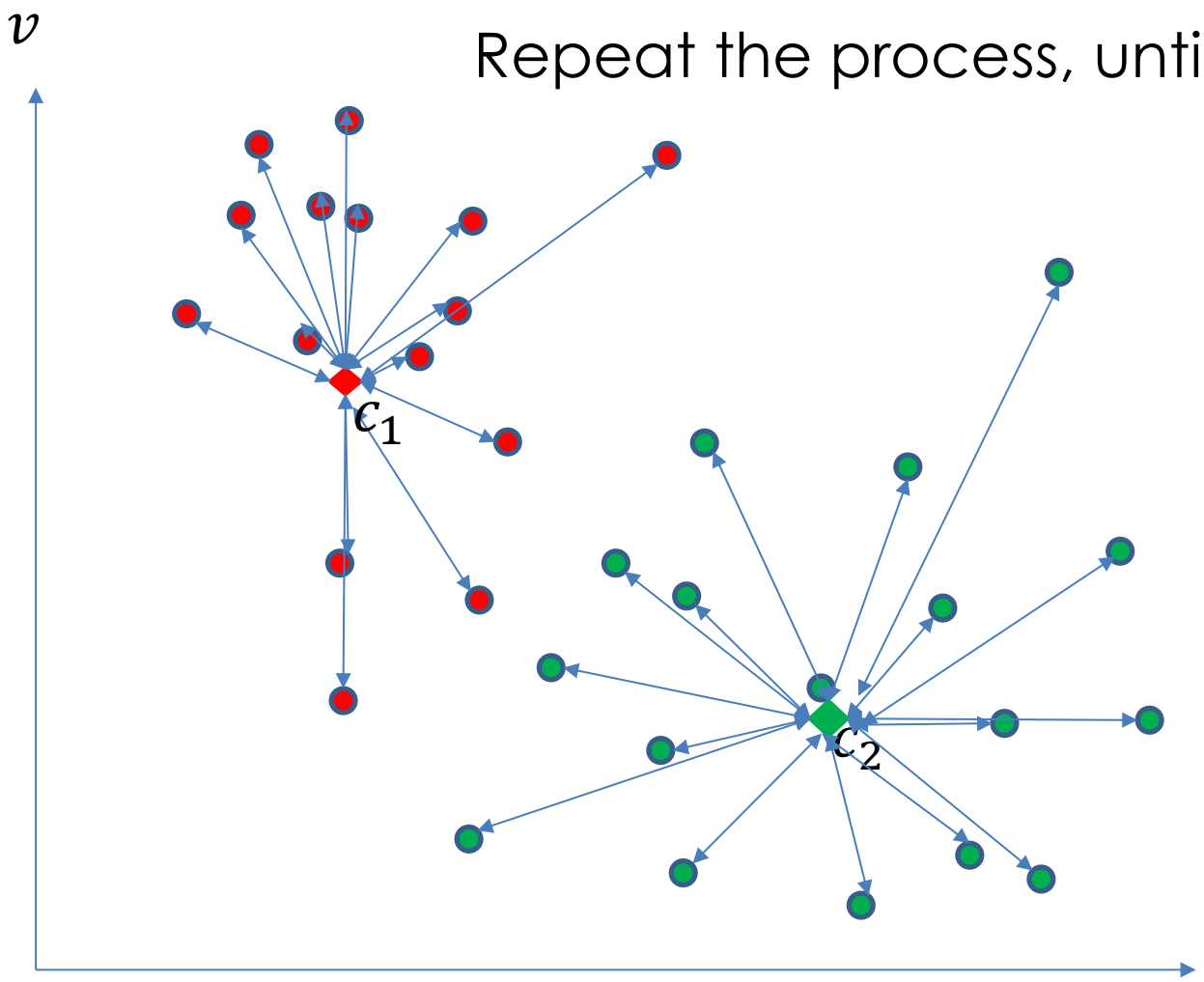
K-means Clustering



© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

Repeat the process, until convergence



Assignment

Center Update.

1) assignment keeps the same

2) center doesn't change.

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

Basic/Naïve K-means Clustering

Looping between Assignment and Centroid Update

1. First, we choose K — the number of clusters. Then we randomly select K feature vectors, called **centroids**, to the feature space.
2. Next, compute the distance from each example **x** to each centroid **c** using some metric, like the Euclidean distance. Then we assign the closest centroid to each example (like if we labeled each example with a centroid id as the label).
3. For each centroid, we calculate the average feature vector of the examples labeled with it. These average feature vectors become the new locations of the centroids.
4. We recompute the distance from each example to each centroid, modify the assignment and repeat the procedure until **the assignments don't change after the centroid locations are recomputed.**
5. Finally, we conclude the clustering with a list of assignments of centroids IDs to the examples.

© Copyright EE, NUS. All Rights Reserved.

```python
# Define the k-means function
def kmeans_step(data, k, centroids):

    # Assign each data point to the closest centroid
    distances = np.sqrt(((data - centroids[:, np.newaxis])**2).sum(axis=2))
    labels = np.argmin(distances, axis=0)

    # Update centroids to be the mean of the data points assigned to them
    new_centroids = np.zeros_like(centroids)
    for j in range(k):
        new_centroids[j] = np.mean(data[labels == j], axis=0)

    # End if centroids no longer change
    if np.linalg.norm(new_centroids - centroids) < tolerance:
        print("End Clustering, Centroids no change.")
        # Return the original centroids and labels, and set end to True
        return centroids, labels, True
    else:
        # Return the centroids and labels, and set end to False
        return new_centroids, labels, False
```

- See Python code:
  - lec11.ipynb
- See live demo at:
  - lec11_kmeans.html

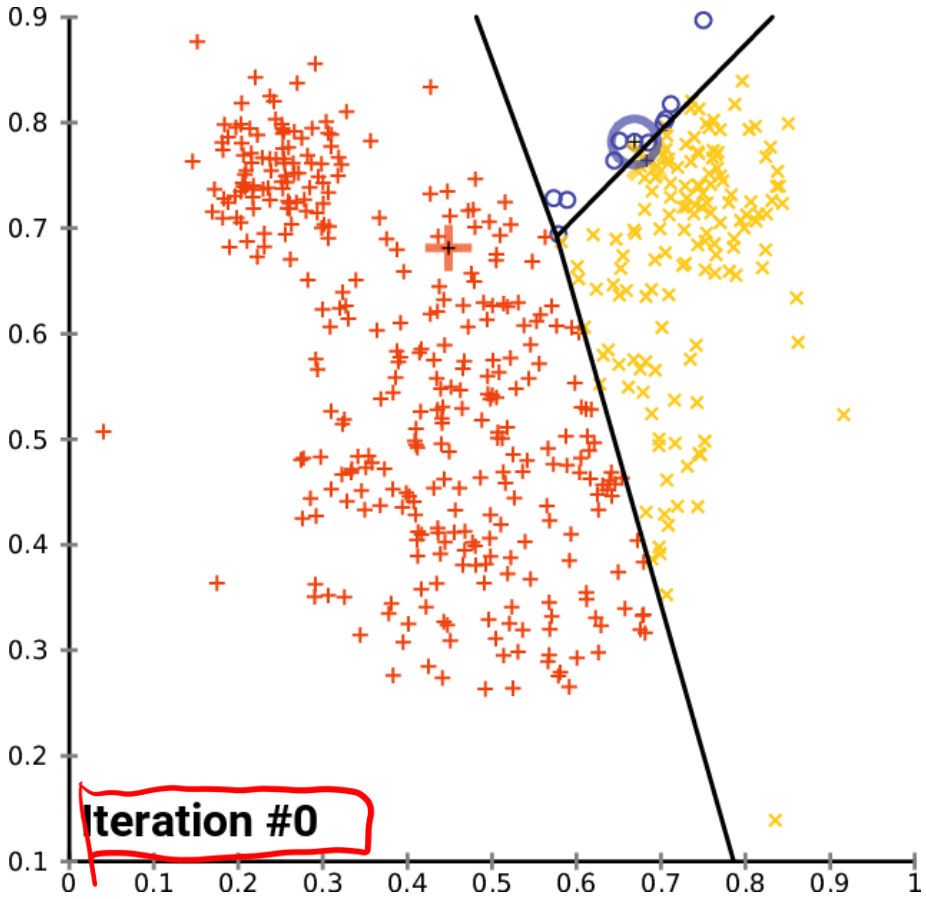
All available in Canvas
Files\For Students\Lecture Notes

In the Final, no coding questions for Xinchao's part!

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

K=3



https://en.wikipedia.org/wiki/K-means_clustering

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

Optimization Objective Function (within-cluster variance)

binary

m: # of samples; i: index of samples
K: # of clusters; k: index of clusters

Minimize J

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|\mathbf{x}_i - \mathbf{c}_k\|^2 \quad (1)$$

$w_{ik} \in \{0, 1\}$: i-th sample is assigned to the k-th group.

$\mathbf{x}_i$: i-th sample

$\mathbf{c}_k$: center of the k-th group.

The term w_{ik} is equal to 1 for data point $\mathbf{x}_i$ if the data point belongs to cluster S_k, else $w_{ik} = 0$.

Note: The optimization objective function was called $C(\mathbf{w})$ in Lecture 8. Here, we use J (with parameters w_{ik} and $\mathbf{c}_k$) so that it is differentiated from the centroids $\mathbf{c}_k$.

Ref: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
https://en.wikipedia.org/wiki/K-means_clustering

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

Optimization Objective Function (within-cluster variance)

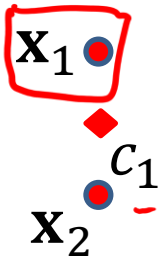
Minimize J

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|\mathbf{x}_i - \mathbf{c}_k\|^2 \quad (1)$$

given

$w_{11} = ?$ 1

$w_{12} = ?$ 0



$w_{41} = ?$ 0

$w_{42} = ?$

The term w_{ik} is equal to 1 for data point $\mathbf{x}_i$ if the data point belongs to cluster S_k, else $w_{ik} = 0$.

Note: The optimization objective function was called $C(\mathbf{w})$ in Lecture 8. Here, we use J (with parameters w_{ik} and $\mathbf{c}_k$) so that it is differentiated from the centroids $\mathbf{c}_k$.

Ref: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
https://en.wikipedia.org/wiki/K-means_clustering

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

Naïve K-means Algorithm

Computing distances to all centroids

1. **Assignment Step (fix $\mathbf{c}$ and update w):**

$$\mathbf{x}_i \in S_k \ (w_{ik} = 1) \text{ if } \|\mathbf{x}_i - \mathbf{c}_k\|^2 < \|\mathbf{x}_i - \mathbf{c}_j\|^2 \ (\text{else } w_{ik} = 0),$$
$$i = 1, \cdots, m; \ \ j, k = 1, \ldots, K.$$

2. **Update Step (fix w and update $\mathbf{c}$):**

$$\frac{\partial J}{\partial \mathbf{c}_k} = -2 \sum_{i=1}^{m} w_{ik}(\mathbf{x}_i - \mathbf{c}_k) = 0 \Rightarrow \mathbf{c}_k = \frac{\sum_{i=1}^{m} w_{ik}\mathbf{x}_i}{\sum_{i=1}^{m} w_{ik}}$$

Solving an optimization, i.e., selling derivative to 0

Note: $\|\mathbf{x} - \mathbf{c}\| = \sqrt{\sum_{d=1}^{D}(x_d - c_d)^2}$ is called the Euclidean distance.

where $\mathbf{x} = (x_1, x_2, \ldots, x_D), \mathbf{c} = (c_1, c_2, \ldots, c_D)$

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

1. **Assignment Step (fix $\mathbf{c}$ and update w):**

$$\mathbf{x}_i \in S_k \ (w_{ik} = 1) \text{ if } \|\mathbf{x}_i - \mathbf{c}_k\|^2 < \|\mathbf{x}_i - \mathbf{c}_j\|^2 (\text{else } w_{ik} = 0), \quad i = 1, \cdots, m; \ j, k = 1, \ldots, K.$$

2. **Update Step (fix w and update $\mathbf{c}$):**

$$\frac{\partial J}{\partial \mathbf{c}_k} = -2 \sum_{i=1}^{m} w_{ik}(\mathbf{x}_i - \mathbf{c}_k) = 0 \Rightarrow \mathbf{c}_k = \frac{\sum_{i=1}^{m} w_{ik}\mathbf{x}_i}{\sum_{i=1}^{m} w_{ik}}$$

By repeating this two steps, the total loss $J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|\mathbf{x}_i - \mathbf{c}_k\|^2$, is **guaranteed to *NOT increase (i.e., remain the same or decrease)*** until convergence.

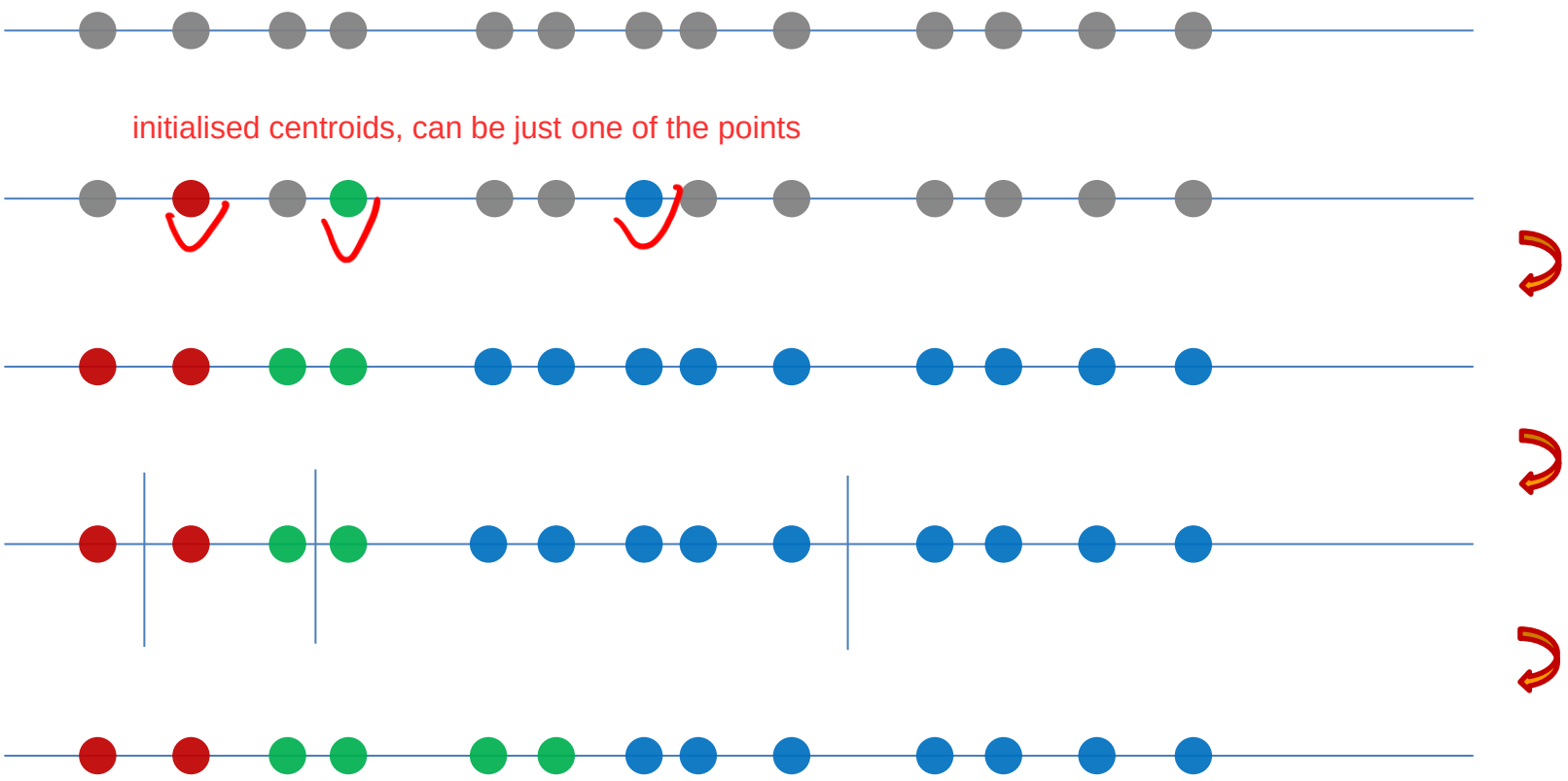
Why?

At Step 2: we compute the new mean, by solving an optimization, i.e., compute the derivative and set to zero, and solve $\mathbf{c}_k$.
This means that, the new $\mathbf{c}_k$ is guaranteed to give a smaller J value.

At Step 1: we only change the assignment, if the distance to the new centroid is smaller! In other words, we either remain in the old group, or change to a new group that is closer (i.e., gives a smaller J)

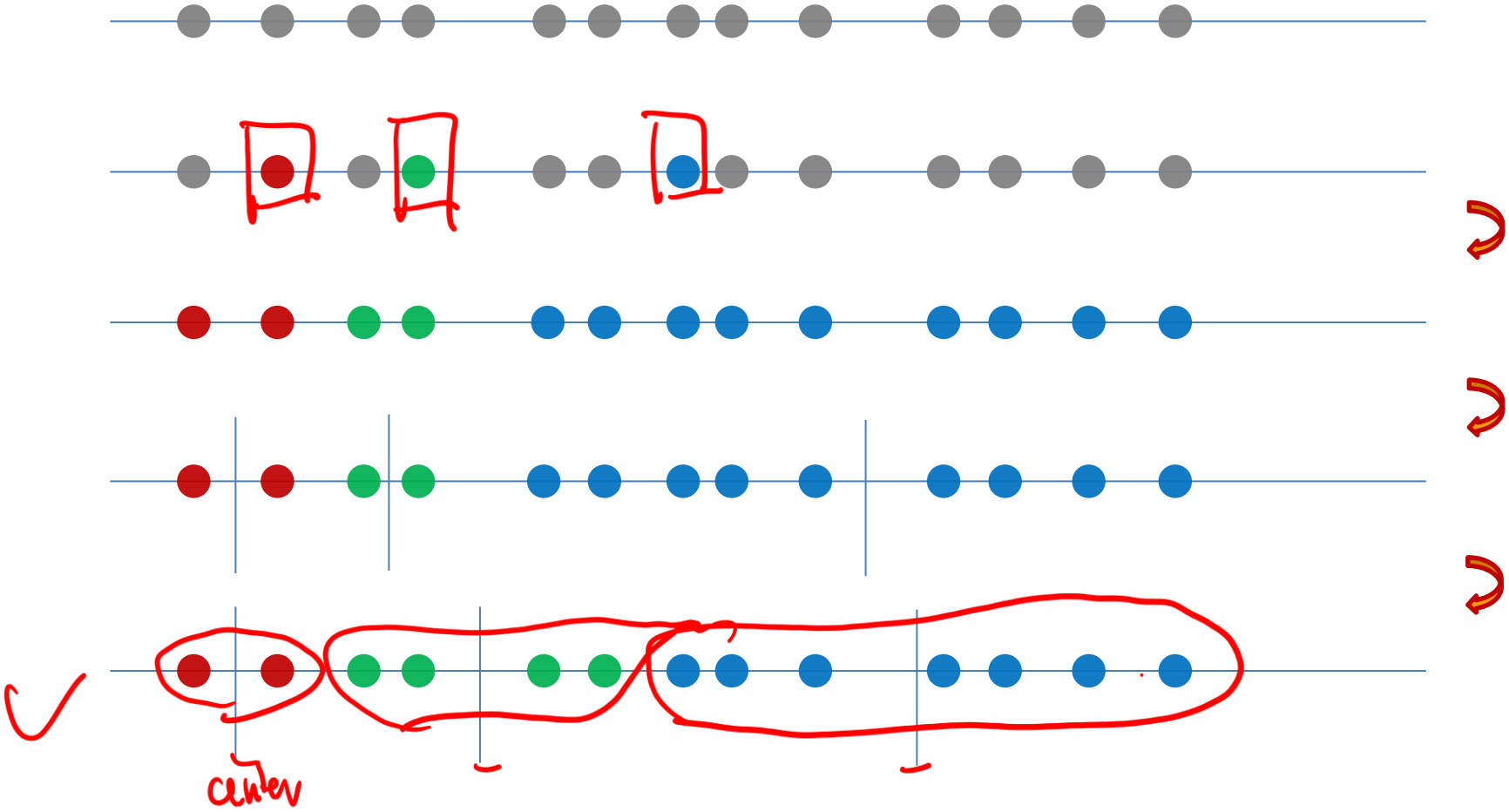
© Copyright EE, NUS. All Rights Reserved.

K-means Clustering (1 D)



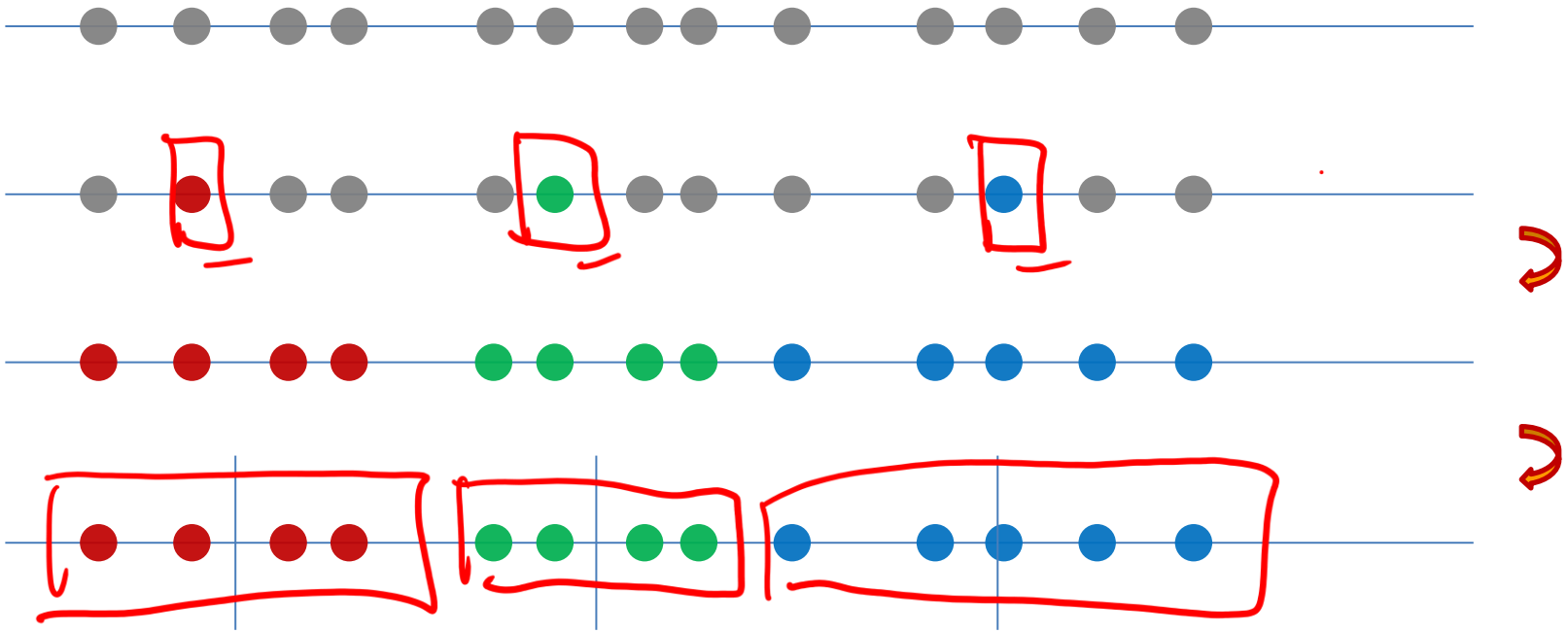
© Copyright EE, NUS. All Rights Reserved.

K-means Clustering (1 D)



© Copyright EE, NUS. All Rights Reserved.

K-means Clustering (1 D)

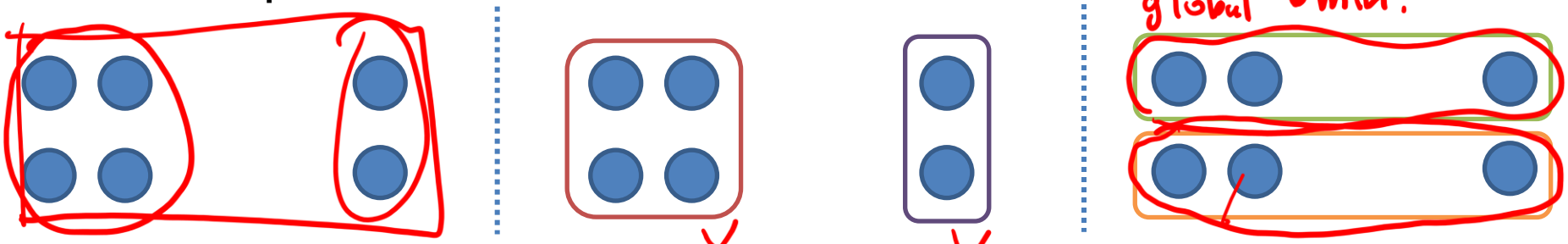


Different initializations give different clusters!

© Copyright EE, NUS. All Rights Reserved.

K-means Clustering

- Unfortunately, k-means is not guaranteed to find a global minimum, it finds only local minimum.
- Example:



Samples | Optimal J | K-means

- Finding the optimal J is NP-hard*
- In practice, k-means clustering usually performs well
- It can be very efficient, and its solution can be used as a starting point for other clustering algorithms

*https://en.wikipedia.org/wiki/NP-hardness

© Copyright EE, NUS. All Rights Reserved.

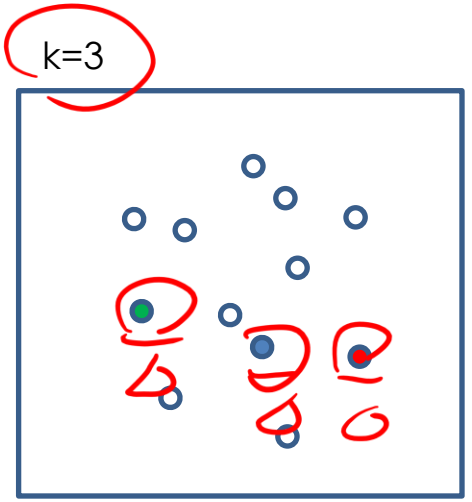
K-means Clustering

- **Initialization**

Forgy method:

– Randomly chooses *k* observations from the dataset and uses these as the initial means.

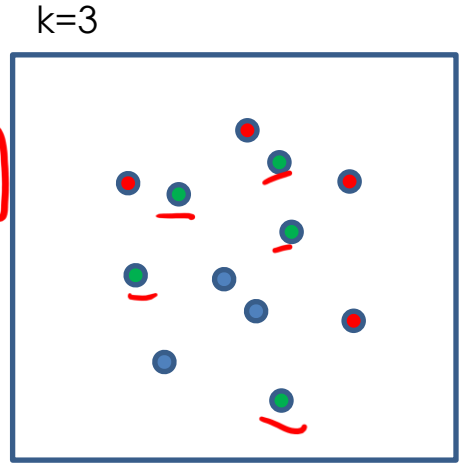
Initialization by centroid



Random partition:

– First randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points

Initialization by grouping



Ref: https://en.wikipedia.org/wiki/K-means_clustering#Standard_algorithm_(naive_k-means)

© Copyright EE, NUS. All Rights Reserved.

Hard vs Soft Clustering

Hard clustering:

Each data point can belong only one cluster, e.g. K-means

- For example, an apple can be red OR green (hard clustering)

this is discrete

Soft clustering (also known as **Fuzzy clustering**):

Each data point can belong to more than one cluster.

- For example, an apple can be red AND green (fuzzy clustering)
- Here, the apple can be red to a certain degree as well as green to a certain degree.
- Instead of the apple belonging to green [green = 1] and not red [red = 0], the apple can belong to green [green = 0.3] and red [red = 0.5]. These value are normalized between 0 and 1; however, they do not represent probabilities, so the two values **do not need to add up to 1**.

the weight wik is no longer discrete, it tells the degree of a point belonging to a cluster

fuzzy is probability

Ref: https://en.wikipedia.org/wiki/Fuzzy_clustering

© Copyright EE, NUS. All Rights Reserved.

Hard vs Soft Clustering

Objective Function for Fuzzy C-means

Minimize J

$$J = \sum_{i=1}^{m}\sum_{k=1}^{C} (w_{ik})^{r} \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

where $$w_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{\|\mathbf{x}_i - \mathbf{c}_k\|}{\|\mathbf{x}_i - \mathbf{c}_j\|}\right)^{\frac{2}{r-1}}}$$

No need to memorize the equation!

not binary.

Each element, $w_{ik} \in [0,1]$, tells the degree to which element, $\mathbf{x}_i$, belongs to cluster $\mathbf{c}_k$.

The fuzzifier $r > 1$ determines the level of cluster fuzziness; usually $1.25 \leq r \leq 2$.

© Copyright EE, NUS. All Rights Reserved.

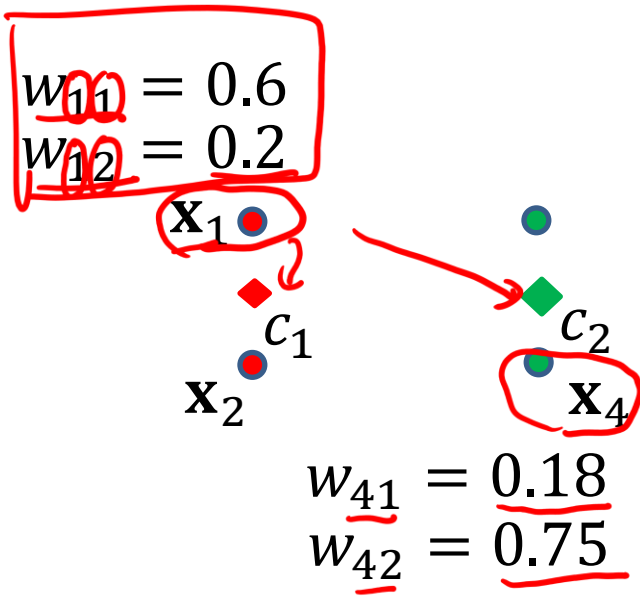
Hard vs Soft Clustering

Objective Function for Fuzzy C-means

Minimize J

$$J = \sum_{i=1}^{m}\sum_{k=1}^{c}(w_{ik})^r \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

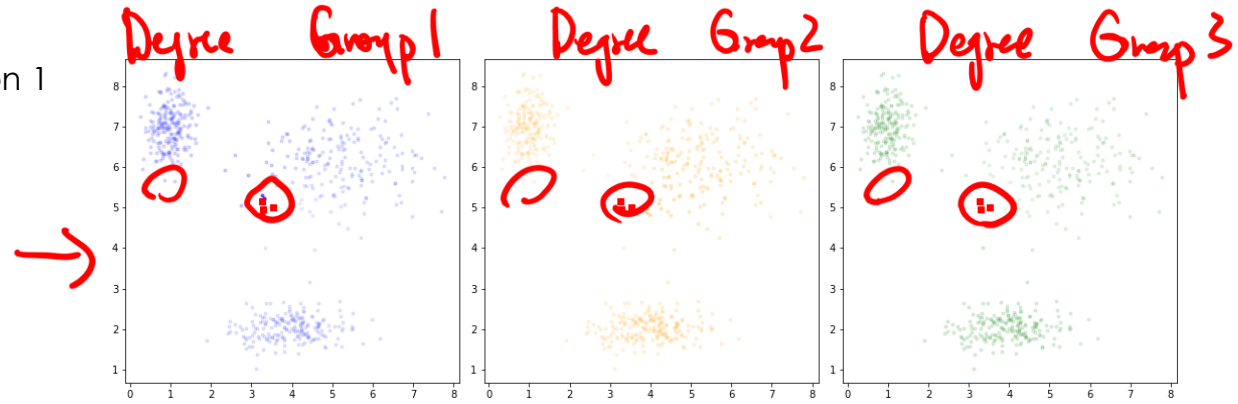
where $w_{ik} = \dfrac{1}{\sum_{j=1}^{c}\left(\dfrac{\|\mathbf{x}_i - \mathbf{c}_k\|}{\|\mathbf{x}_i - \mathbf{c}_j\|}\right)^{\frac{2}{r-1}}}$



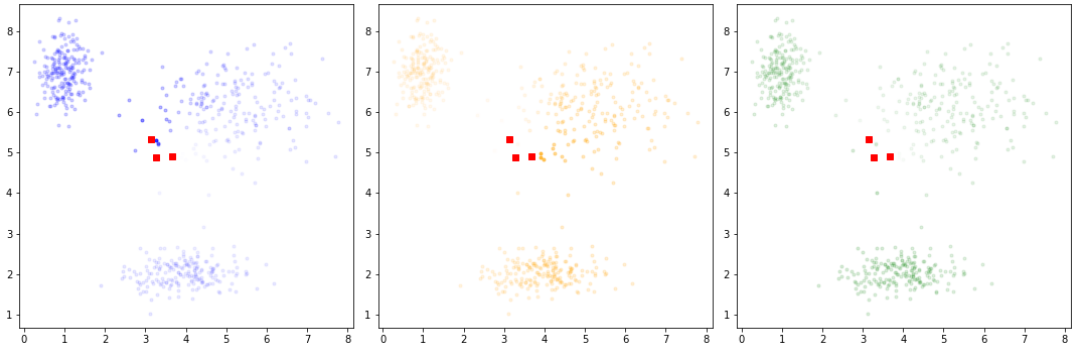
Each element, $w_{ik} \in [0,1]$, tells the degree to which element, $\mathbf{x}_i$, belongs to cluster $\mathbf{c}_k$.
The fuzzifier $r > 1$ determines the level of cluster fuzziness; usually $1.25 \leq r \leq 2$.

© Copyright EE, NUS. All Rights Reserved.

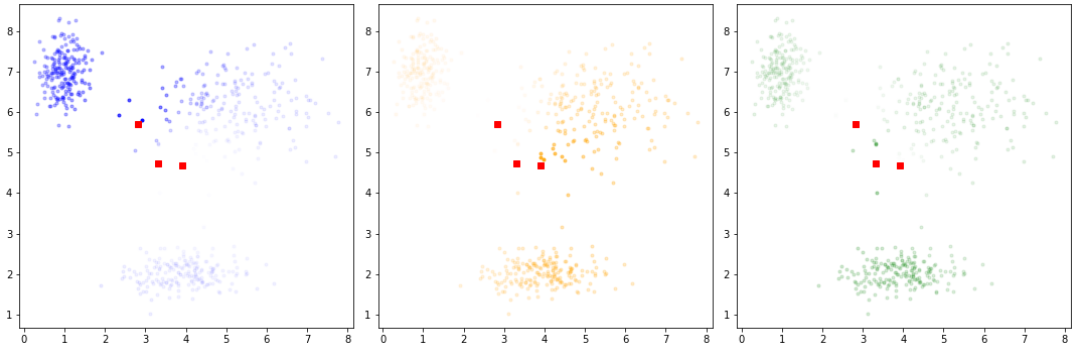
Iteration 1



Iteration 2



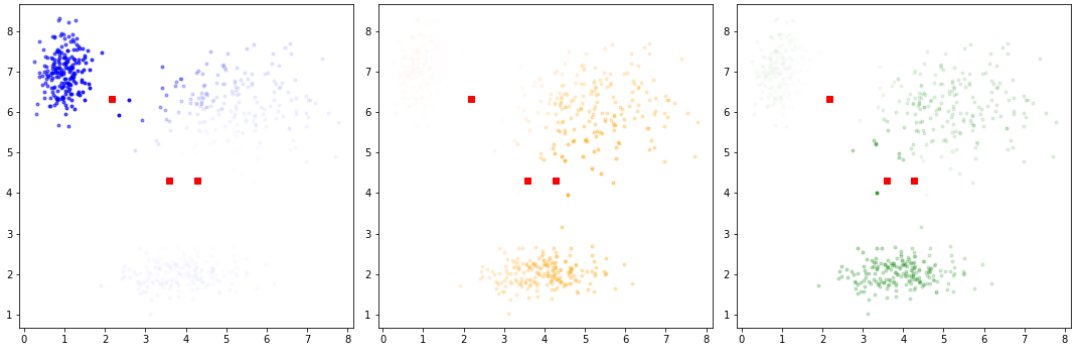
Iteration 3



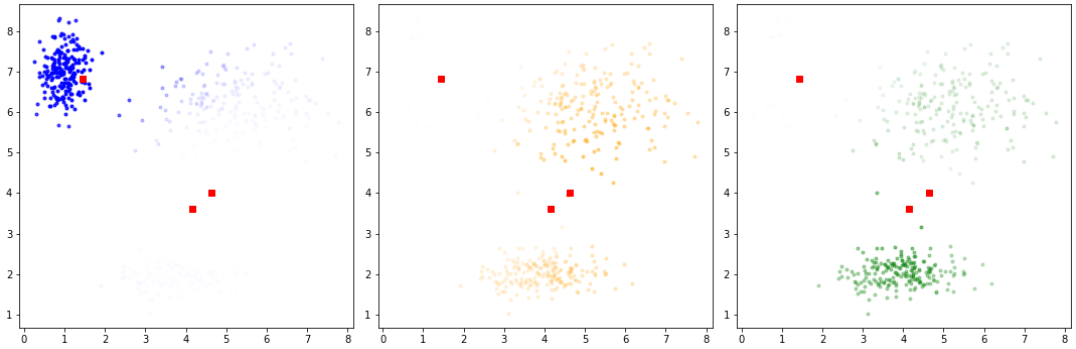
Visualization of Fuzzy C-means Iterations

© Copyright EE, NUS. All Rights Reserved.

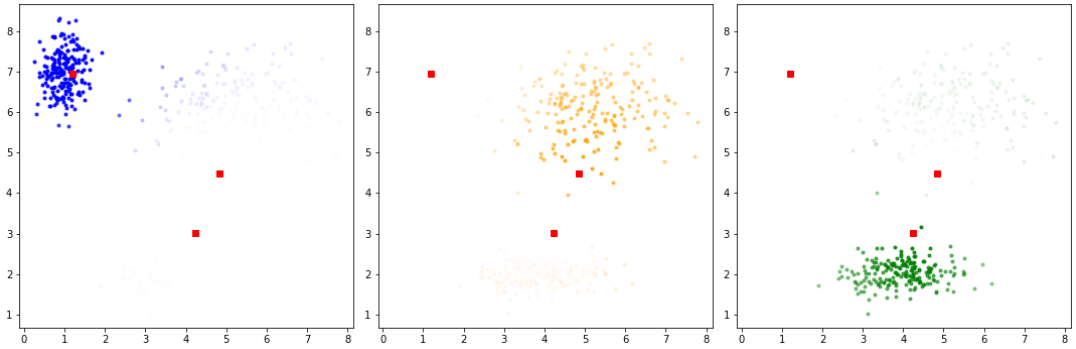
Iteration 4



Iteration 5



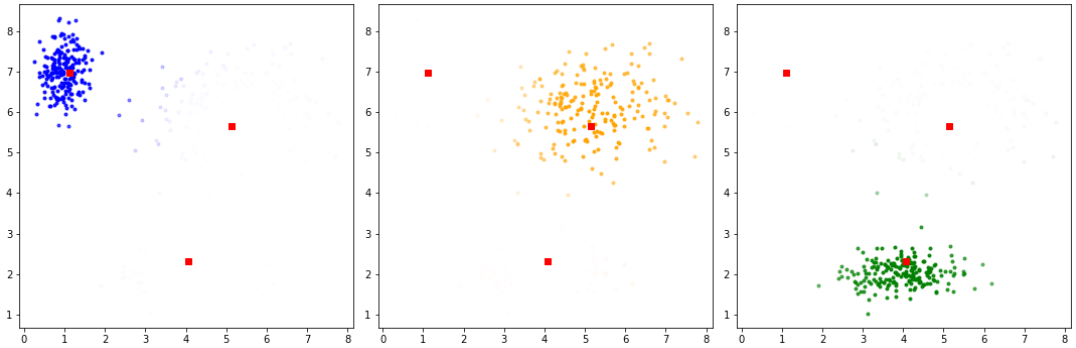
Iteration 6



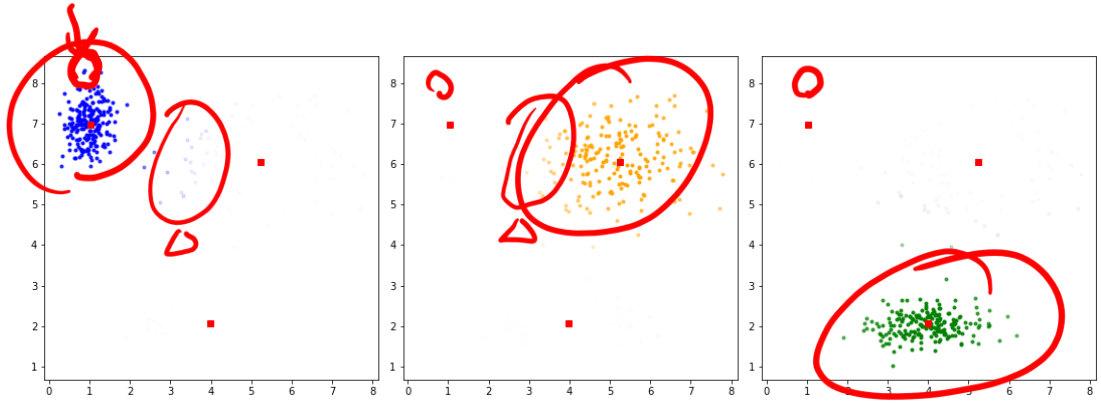
Visualization of Fuzzy C-means Iterations

© Copyright EE, NUS. All Rights Reserved.

Iteration 7



Iteration 8



Visualization of Fuzzy C-means Iterations

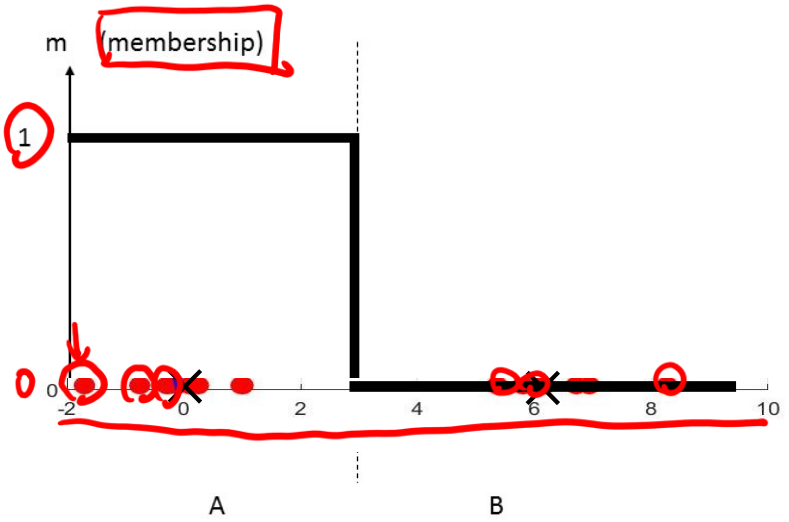
© Copyright EE, NUS. All Rights Reserved.

Hard vs Soft Clustering

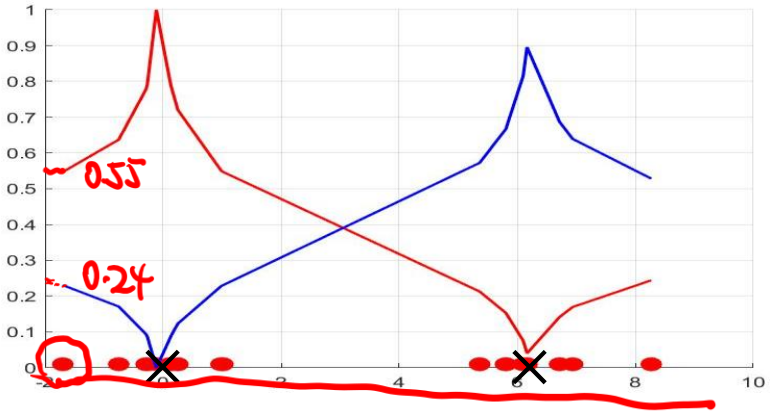
Naïve K-means versus Fuzzy C-means

Naïve K-means: $w_{ik} \in \{0,1\}$

1D sample



Fuzzy C-means: $w_{ik} \in [0,1]$



Ref: https://en.wikipedia.org/wiki/Fuzzy_clustering

© Copyright EE, NUS. All Rights Reserved.

Summary

- Introduction of unsupervised learning
- K-means Clustering
  - The most popular clustering technique
- Fuzzy Clustering

© Copyright EE, NUS. All Rights Reserved.

Practice Question

We have a collection of 9 foreign coins. We measure their radius in millimeters and summarize them as follows.

Coin ID	01	02	03	04	05	06	07	08	09
Radius (mm)	10	11	12	15	16	17	20	21	22

We'd like to group the coins into three groups according to their radius.

Assume we pick coin 01 as the initial centroid for Group A, coin 04 for Group B, and coin 07 for Group C. We would like to assign the coins to the three groups using Euclidean distance. Before updating the new centroid, we will have _BLANK1_ coins in Group A (please enter an integer here).

3

01, 02, 03

© Copyright EE, NUS. All Rights Reserved.

EE2211 Introduction to Machine Learning

Lecture 12

Wang Xinchao
xinchao@nus.edu.sg

© Copyright EE, NUS. All Rights Reserved.

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Linear Algebra, Probability and Statistics
- Fundamental Machine Learning Algorithms I (Helen)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Helen)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

[Important] In the Final, no coding questions for Xinchao's part!

Despite you will see some in the tutorial, they won't be tested.

© Copyright EE, NUS. All Rights Reserved.

About this week's lecture…

- Neural Network (NN) is a very big topic
  - In NUS we have multiple full-semester modules to discuss NN
    - EE4305 Fuzzy/Neural Systems for Intelligent Robotics
    - EE5934/EE6934 Deep Learning
    - EE4802/IE4213 Learning from Data
    - …
  - In EE2211, we only give a very gentle introduction
- Understanding at conceptual level is sufficient
  - **In final exam, we have at 1-2 True/False and 1-2 MCQ about NN.**
  - **You can refer to the past paper**
- You will do some computation in tutorial, but final exam will be much simpler than the questions in tutorial

© Copyright EE, NUS. All Rights Reserved.

Outline

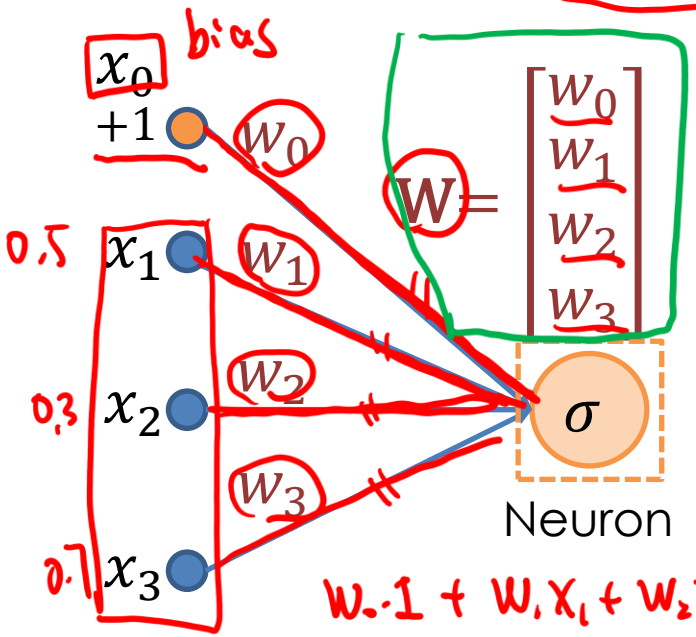
- Introduction to Neural Networks
  - Perceptron
  - Activation Functions
  - Multi-layer Perceptron
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

a special type of NN.

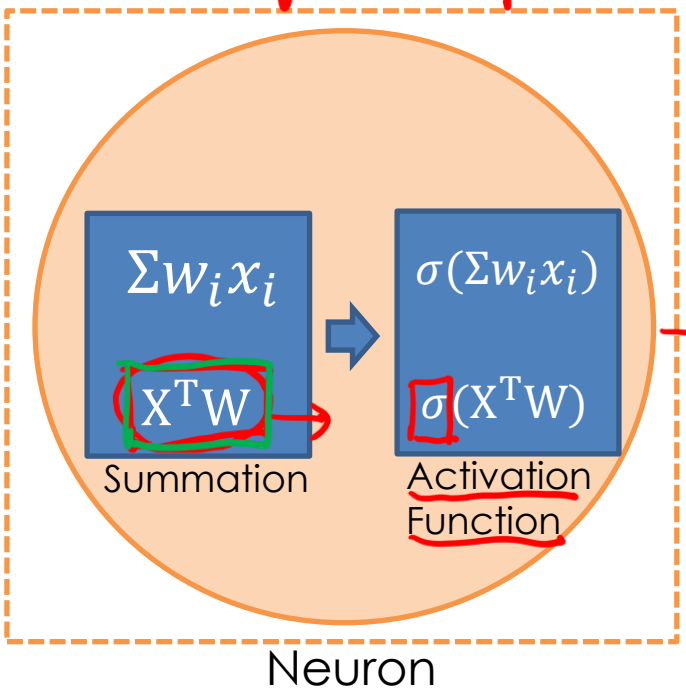
© Copyright EE, NUS. All Rights Reserved.

Perceptron

is the building block of NN.



$w_0 \cdot 1 + w_1 x_1 + w_2 x_2 + w_3 x_3$



→ scaler

$$\mathbf{X} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Output of Neuron: $\sigma(X^T W)$ or $\sigma(\Sigma w_i x_i)$

Activation Function: non-linear function to introduce non-linearity into the neural networks!

Goal of training: to learn W!

parameters of Perceptron / NN.

© Copyright EE, NUS. All Rights Reserved.

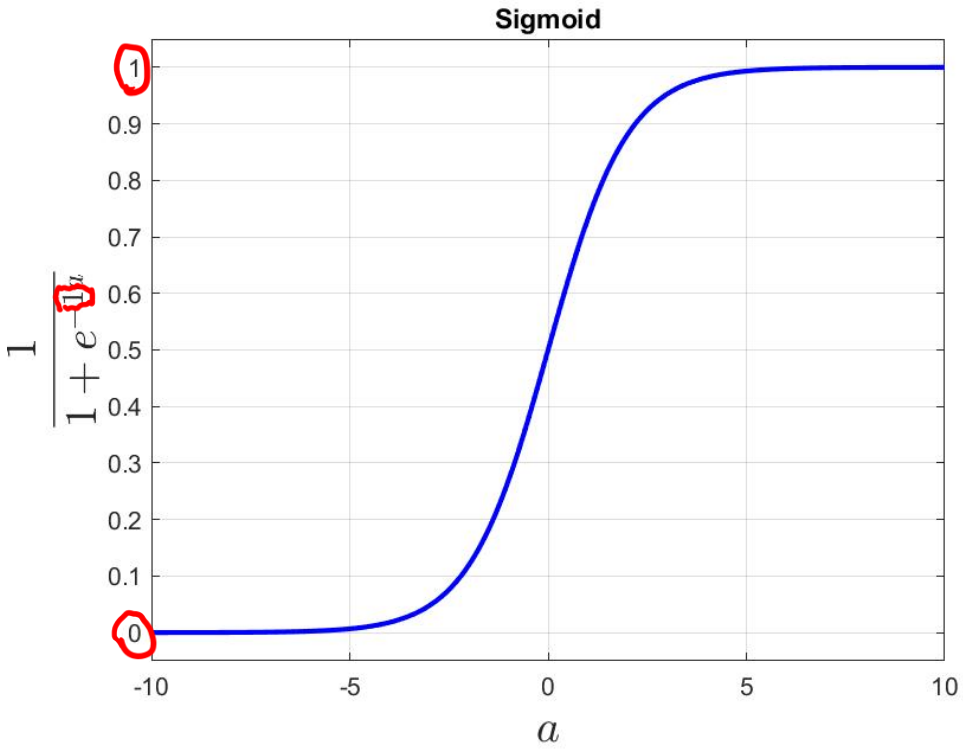
Activation Functions

First

Sigmoid Activation Function

$$\sigma(a) = \frac{1}{1 + e^{-\beta a}},$$

asymptote on both ends



© Copyright EE, NUS. All Rights Reserved.

Second

Activation Functions

ReLU Activation Function

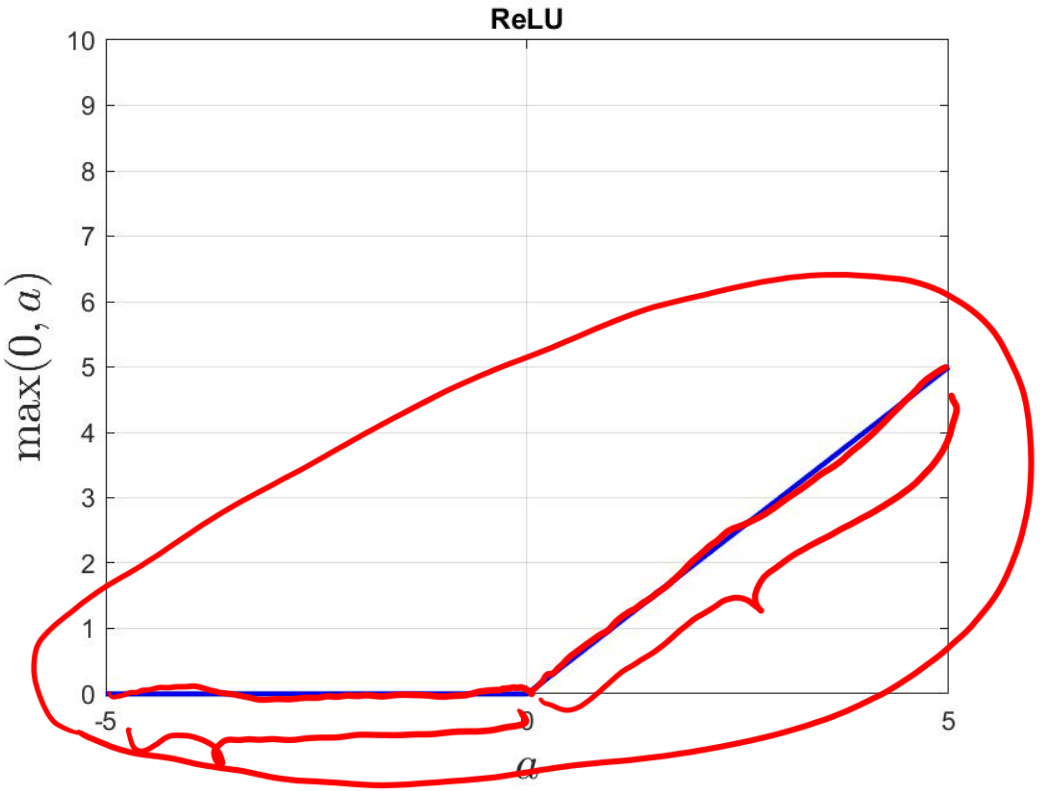
max(0,a) means if a < 0, return 0, else return a

$$\sigma(a) = \max(0, a)$$

Rectified Linear Unit (ReLU)

a = 0.5, σ(a) = 0.5

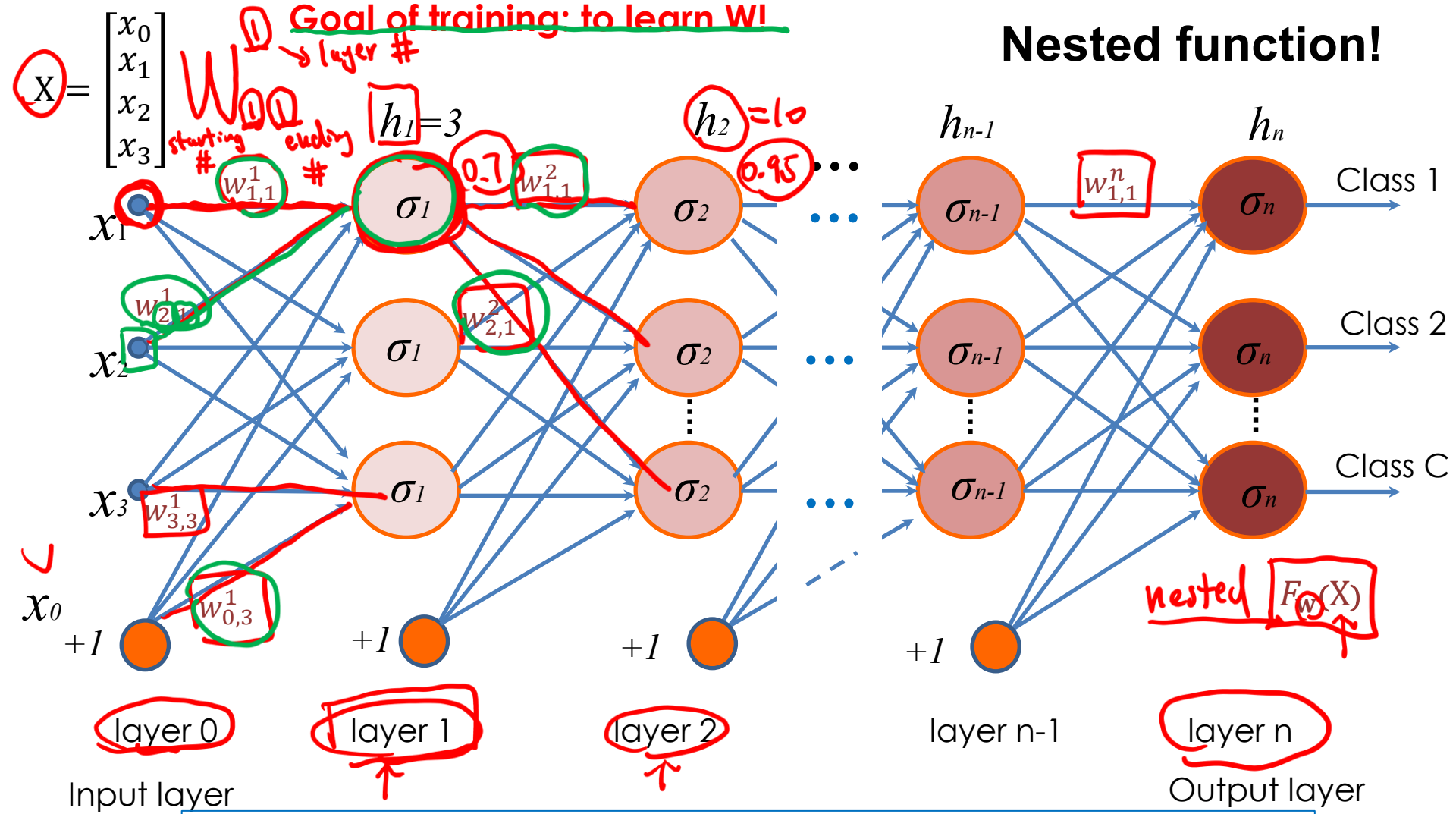
a = -2, σ(a) = 0



© Copyright EE, NUS. All Rights Reserved.

dimention of weights vector: W at layer l = (d+1) x Dl = (no. inputs from prev layer+ 1 bias) x no of neurons in curr layer
row no. = input layer no where input[0] is row[0], and each col is for each neuron in curr layer

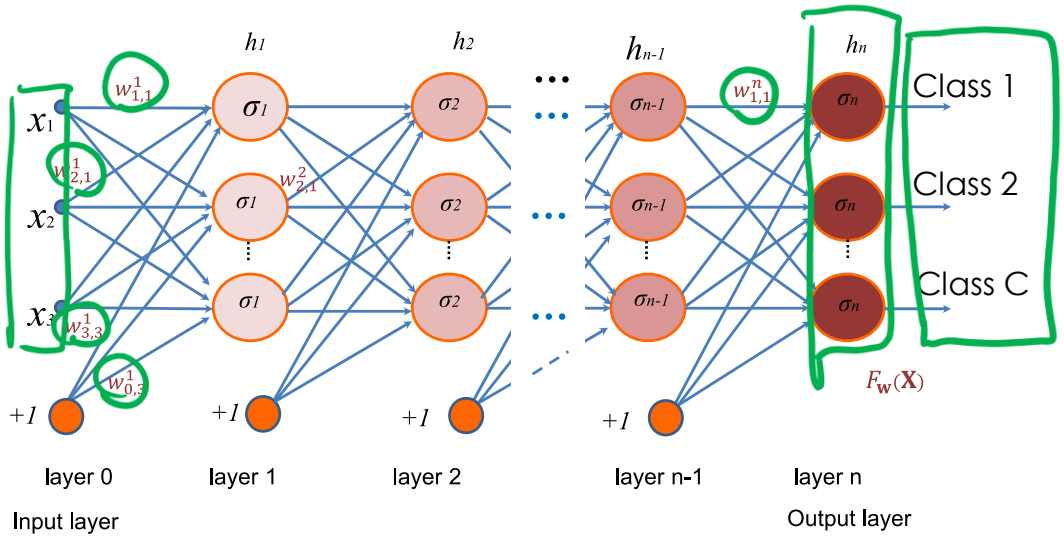
Multilayer Perceptron (Neural Network)



Note: h_n denotes the number of neurons in layer n.

© Copyright EE, NUS. All Rights Reserved.

Things to Note



1. The number of neurons in different layers may differ, i.e., h_1 don't have to be equal to h_2.
2. For **classification** task, the number of neurons in the last layer equals to the number of classes.
3. We can treat the whole network as a function $F_{\mathbf{w}}(\mathbf{X})$, where $\mathbf{w}$ is to be learned.

MLP is a feedforward NN, which is a NN where info flows in a single path, starting at input layer, passing through hidden layers and ending at the output layer

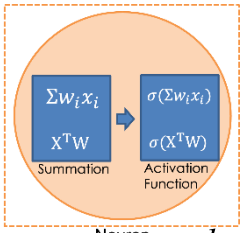
© Copyright EE, NUS. All Rights Reserved.

Multilayer Perceptron (Neural Network)

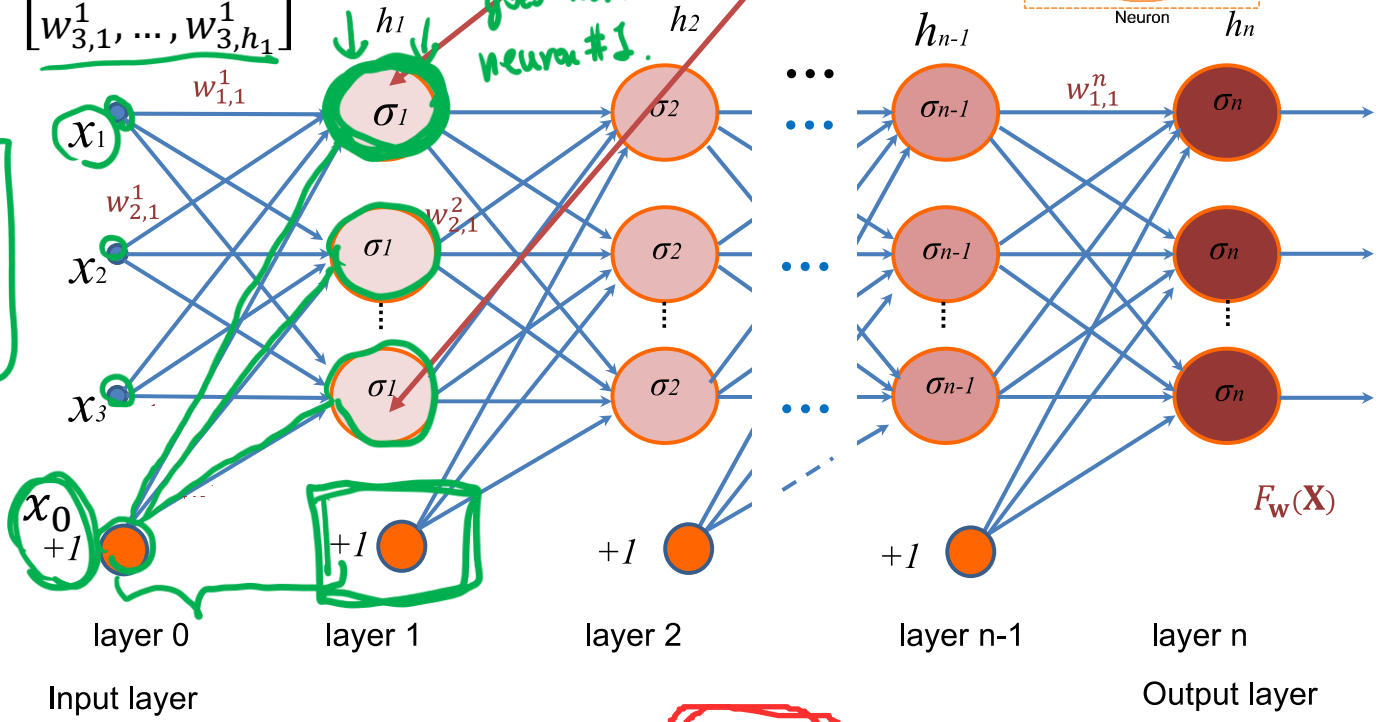
$$W^1 = \begin{bmatrix} w^1_{0,1}, w^1_{0,2}, w^1_{0,h_1} \\ w^1_{1,1}, \dots, w^1_{1,h_1} \\ w^1_{2,1}, \dots, w^1_{2,h_1} \\ w^1_{3,1}, \dots, w^1_{3,h_1} \end{bmatrix}$$

$$X^T W^1 = [\Sigma w^1_{i,1} x_i, \dots, \Sigma w^1_{i,h_1} x_i]$$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



everything goes into neuron #1.



A **neural network** is essentially a **nested function**.

$$F_W(X) = \sigma([1, \dots \sigma([1, \sigma(X^T W^1)] W^2) \dots] W^n)$$

© Copyright EE, NUS. All Rights Reserved.

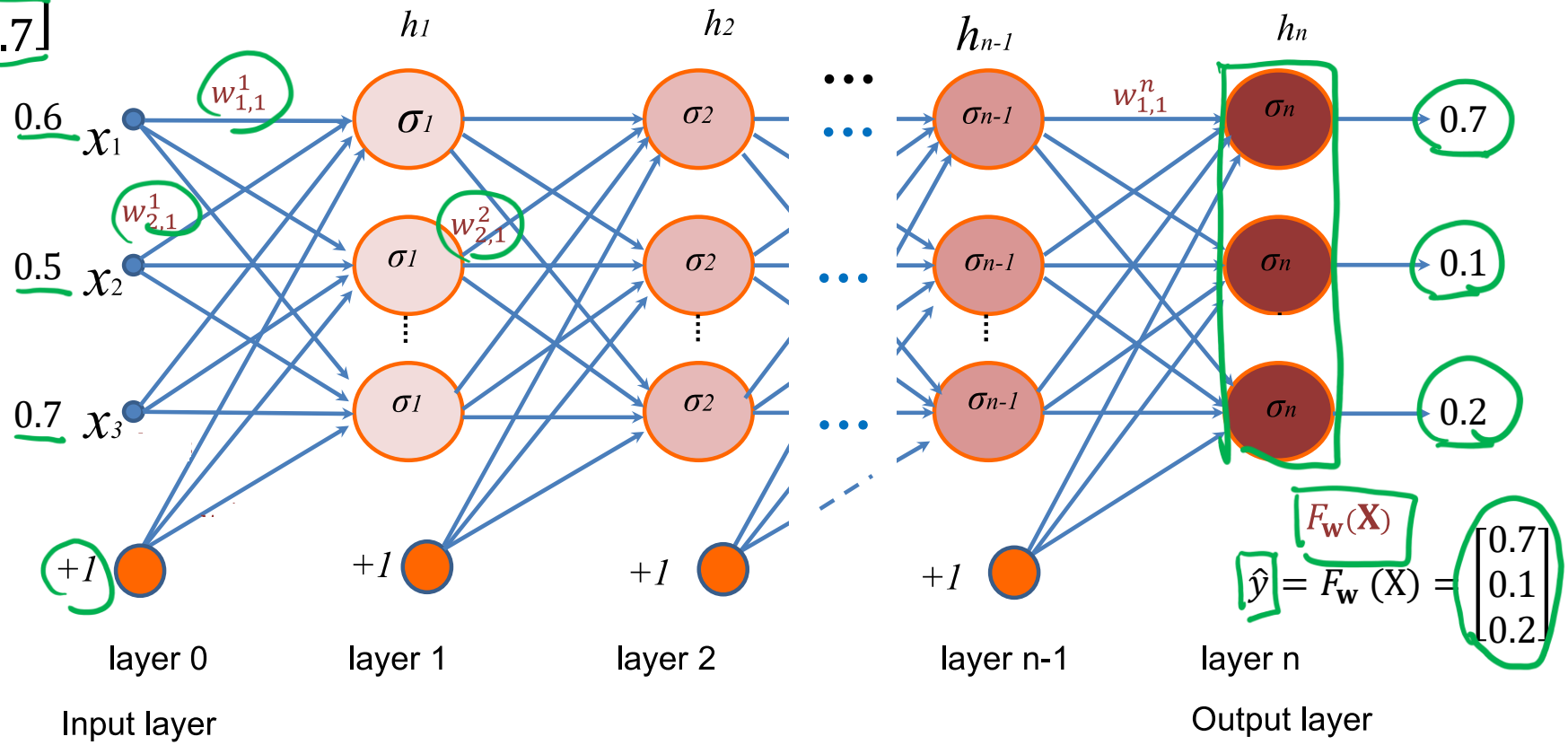
Outline

- Introduction to Neural Networks
  - Multi-layer perceptron
  - Activation Functions
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

© Copyright EE, NUS. All Rights Reserved.

Goal of Neural Network Training: to Learn W

$$X = \begin{bmatrix} 0.6 \\ 0.5 \\ 0.7 \end{bmatrix}$$



$$\hat{y} = F_{\mathbf{W}}(\mathbf{X}) = \begin{bmatrix} 0.7 \\ 0.1 \\ 0.2 \end{bmatrix}$$

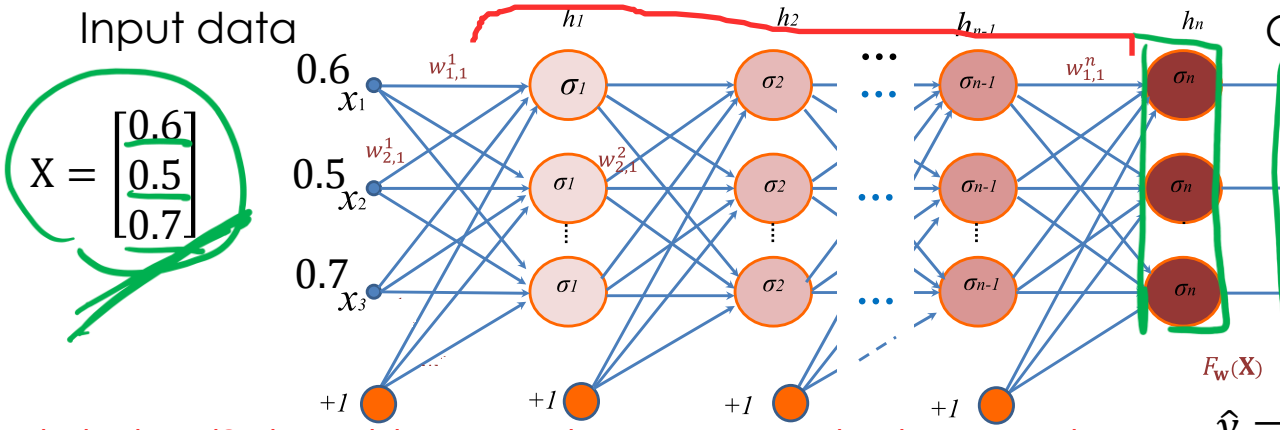
Specifically, **W** is learned through

1. Random initialization
2. **Backpropagation**

© Copyright EE, NUS. All Rights Reserved.

Neural Network Training: Backpropagation

Assume we train a NN for 3-class classification



probability.

One-hot Class Labels

0
0
1

$\hat{y} = F_{\mathbf{w}}(\mathrm{X}) = \begin{bmatrix} 0.7 \\ 0.1 \\ 0.2 \end{bmatrix}$ prediction.

$y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ gt / target.

why backward? when training, we need to use y_expected and x_expected. the true reference point is at the output (last layer) which is our y_expected. x_expected is from previous layer. and x_expected now is previous layer's y_output, Hence, train w by propagating backwards.

1. Forward: (weights are fixed)
To compute network responses
To compute the errors at each output

2. Backward: (weights are updated)
To pass back the error from the output to the hidden layers
To update all weights to optimize the network

single sample

A loss function for a single sample:

$\min_{\mathbf{w}} \sum_{i=1}^{C} (\hat{y}_i - y_i)^2$

or

$\min_{\mathbf{w}} ||\hat{y} - y||^2$

Update W!

© Copyright EE, NUS. All Rights Reserved.

Neural Network Training: Backpropagation

- Recall that the parameters W are randomly initialized.
- We use **Backpropagation** to update W.
- In essence, **Backpropagation** is gradient descent!
- Assume we have N samples, each sample denoted by X^j and the output of NN by $\hat{y}^j$, loss function is then

$$J = \sum_{j=1}^{N} \left\| \hat{y}^j - y^j \right\|^2, \quad \min_{\mathbf{w}} J$$

Recall gradient descent in Lec 8: $\mathrm{w} \leftarrow \mathrm{w} - \eta \nabla_{\mathrm{w}} J$

- We would therefore like to compute $\nabla_{\mathrm{w}} J$!
  - J is a function of $\hat{y}$, and $\hat{y}$ is a function of $\mathbf{w}$, i.e., $\hat{y} = F_{\mathbf{w}}(\mathrm{X})$
  - Use gradient descent and chain rule!

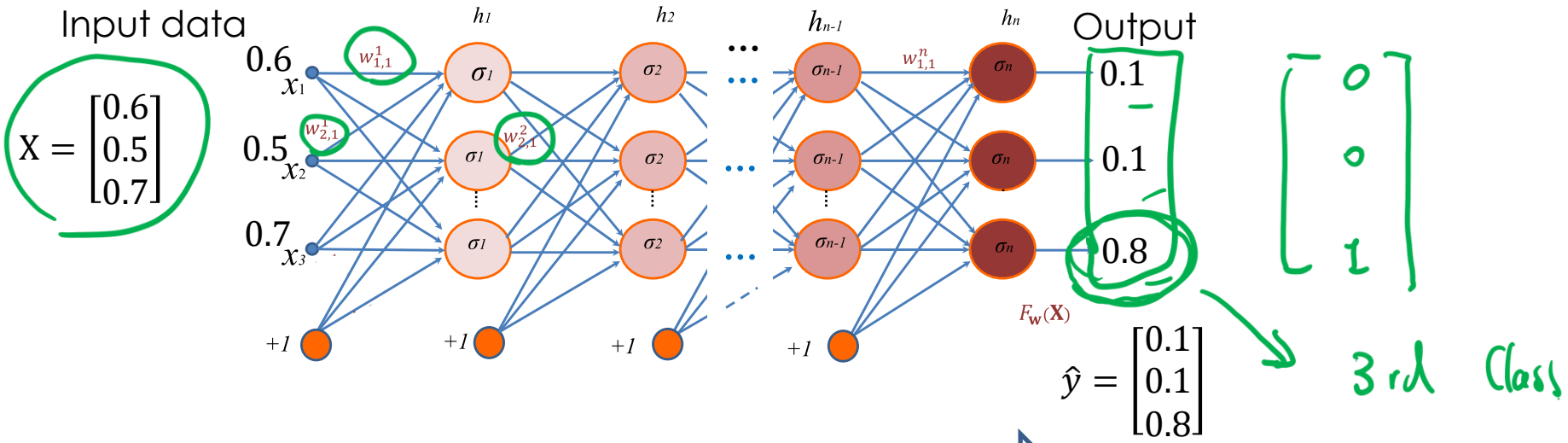
Being aware of the basic concept is sufficient for exam. No calculation needed.

© Copyright EE, NUS. All Rights Reserved.

Neural Network Testing

after Ws are learned (and fixed)

Once all network is trained and parameters are updated, we may conduct testing.



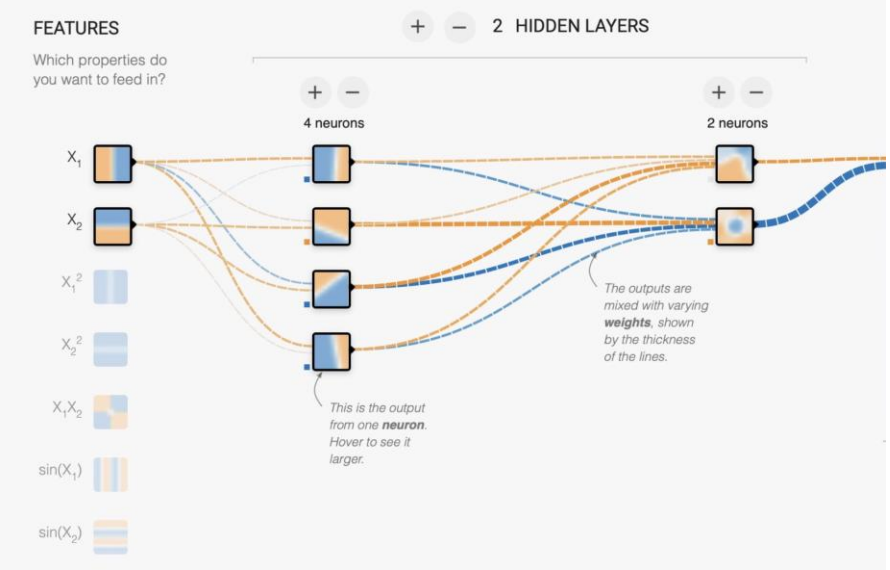
1. Forward: (weights are fixed)
To estimate compute network responses
To predict the output labels given novel inputs

© Copyright EE, NUS. All Rights Reserved.

Python Demo

```
[ ]  from sklearn.neural_network import MLPClassifier
     # Train the neural network classifier with 3 hidden layers of 200 neurons each
     clf = MLPClassifier(hidden_layer_sizes=(200, 200, 200), activation="relu", learning_rate="invscaling", verbose=True)
     clf.fit(X, y)
     plot_clf(clf, X, y)
```

- Python Code (A simple multi-layer neural network classifier)
  - lec12.ipynb
- Demo
  - https://playground.tensorflow.org/



All available in Canvas
Files\For Students\Lecture Notes

In the Final, no coding questions for Xinchao's part!

© Copyright EE, NUS. All Rights Reserved.

Supplementary materials (Not required for exam)

1) https://www.youtube.com/watch?v=tIeHLnjs5U8

This video series includes animations that explain backpropagation calculus.

2) https://www.youtube.com/playlist?list=PLQVvvaa0QuDcjD5BAw2DxE6OF2tius3V3

This video series includes hands-on coding examples in Python.

© Copyright EE, NUS. All Rights Reserved.

Outline

- Introduction to Neural Networks
  - Multi-layer perceptron
  - Activation Functions
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)

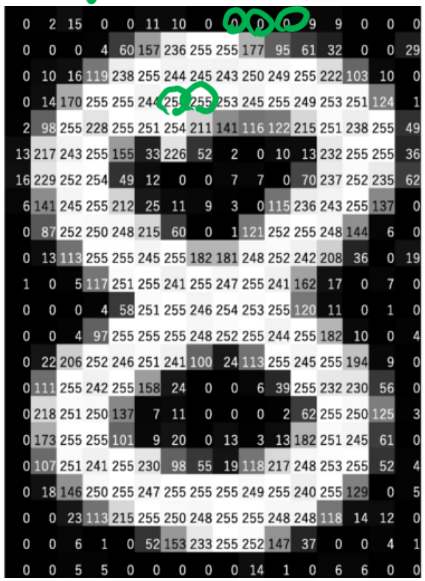
- A convolutional neural network (CNN) is a special type of neural network that significantly reduces the number of parameters in a deep neural network.
- Very popular in image-related applications
- Each image is stored as a matrix in a computer

parameters of networks are always learn by backpropagation

deep means have multiple layers of neurons

image

brighter, higher value. | darker, smaller values



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

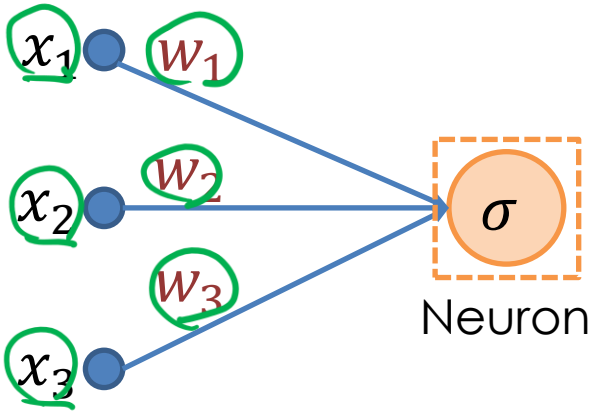
0 ~ 255

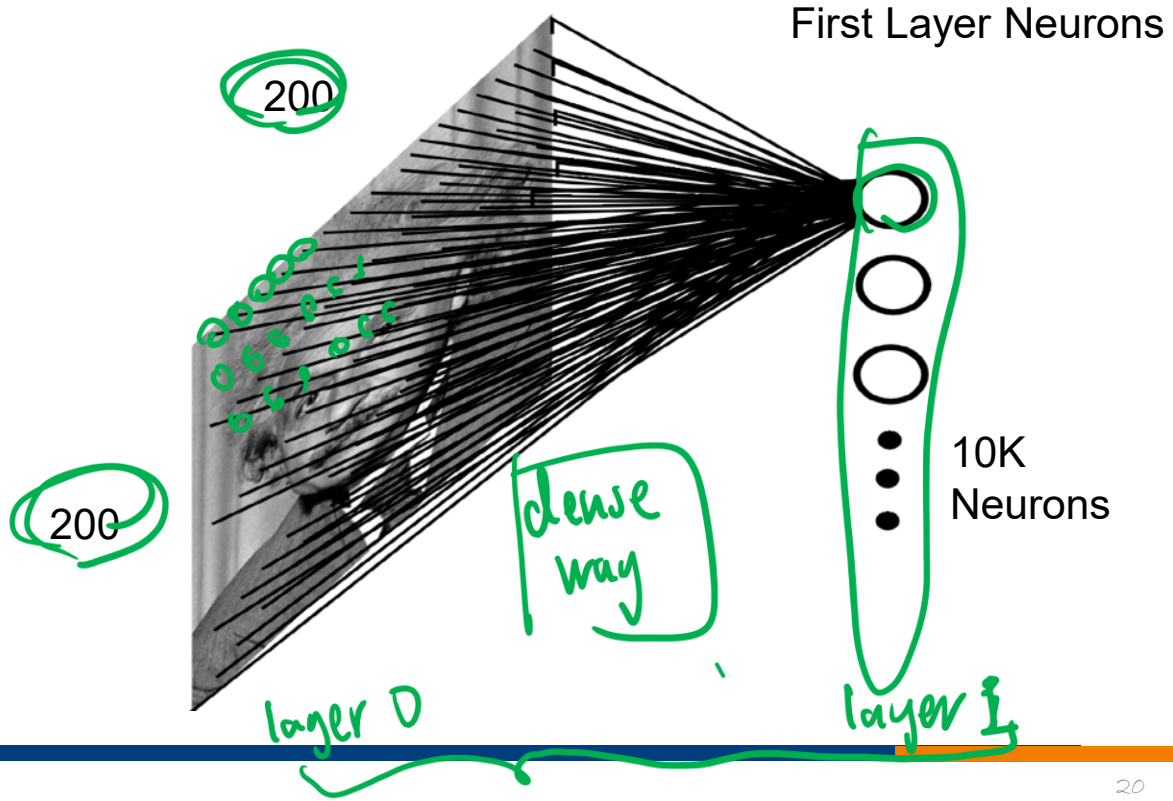
https://medium.com/lifeandtech/convert-csv-file-to-images-309b6fdb8c49

© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)

- If we model all matrix entries as inputs all at once
  - Assume we have an image/matrix size of 200x200
  - Assume we have 10K neuros in the first layer
  - We already have 200x200x10K=400 Million parameters to learn!

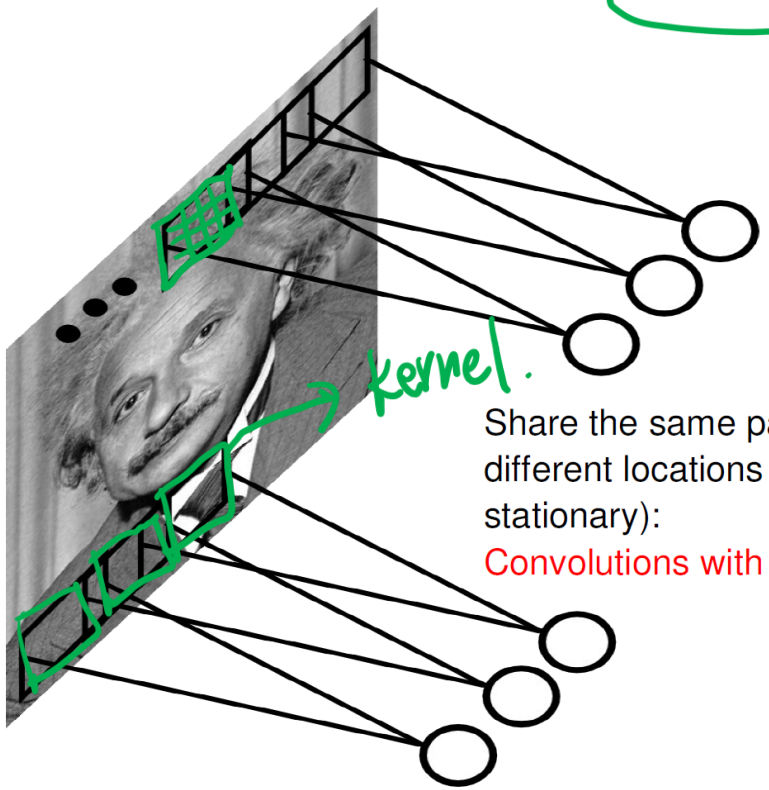




© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)

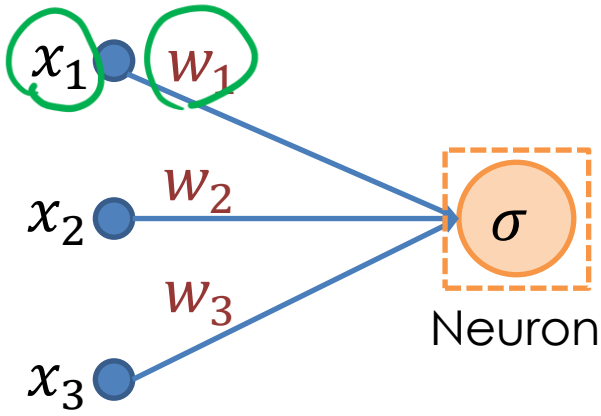
- Hence, we introduce CNN to reduce the number of parameters.
- Works in **a sliding-window manner!**



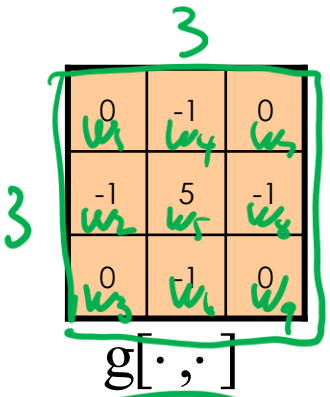
Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)

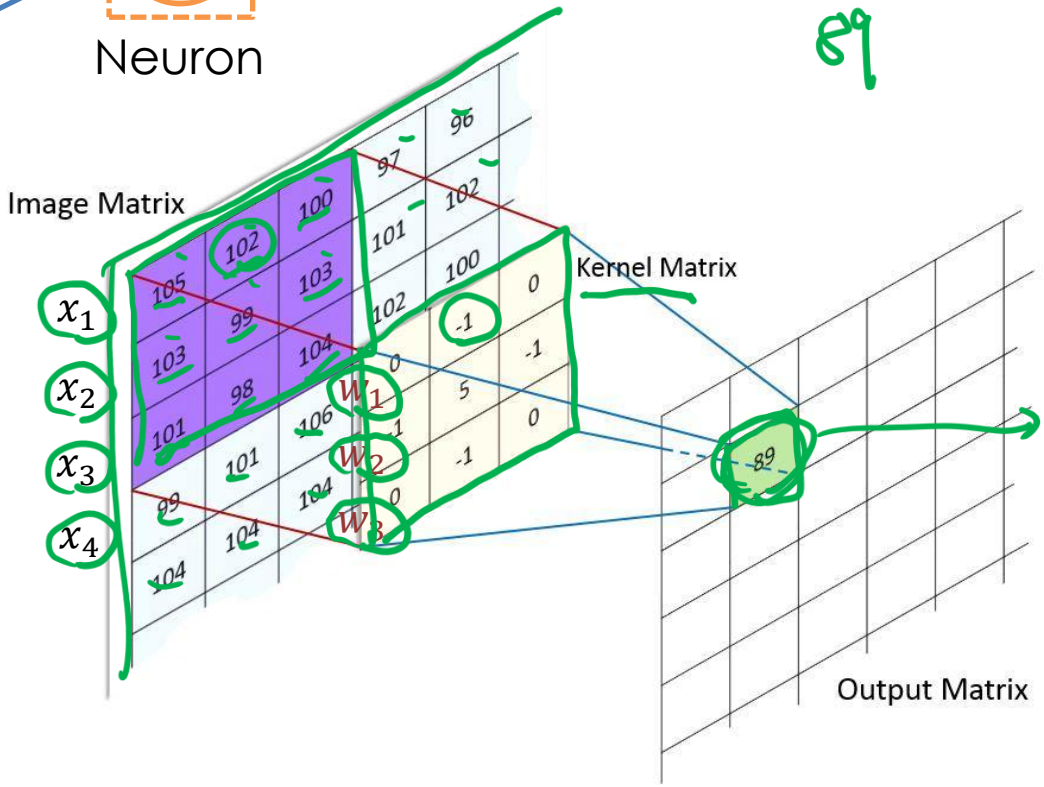


$105 \cdot 0 + 102 \cdot (-1) + 100 \cdot 0 +$
$103 \cdot (-1) + 99 \cdot 5 + 103 \cdot (-1) +$
$101 \cdot (0) + 98 \cdot (-1) + 104 \cdot 0 .$
$= 89$



Kernels to be learned

convolve

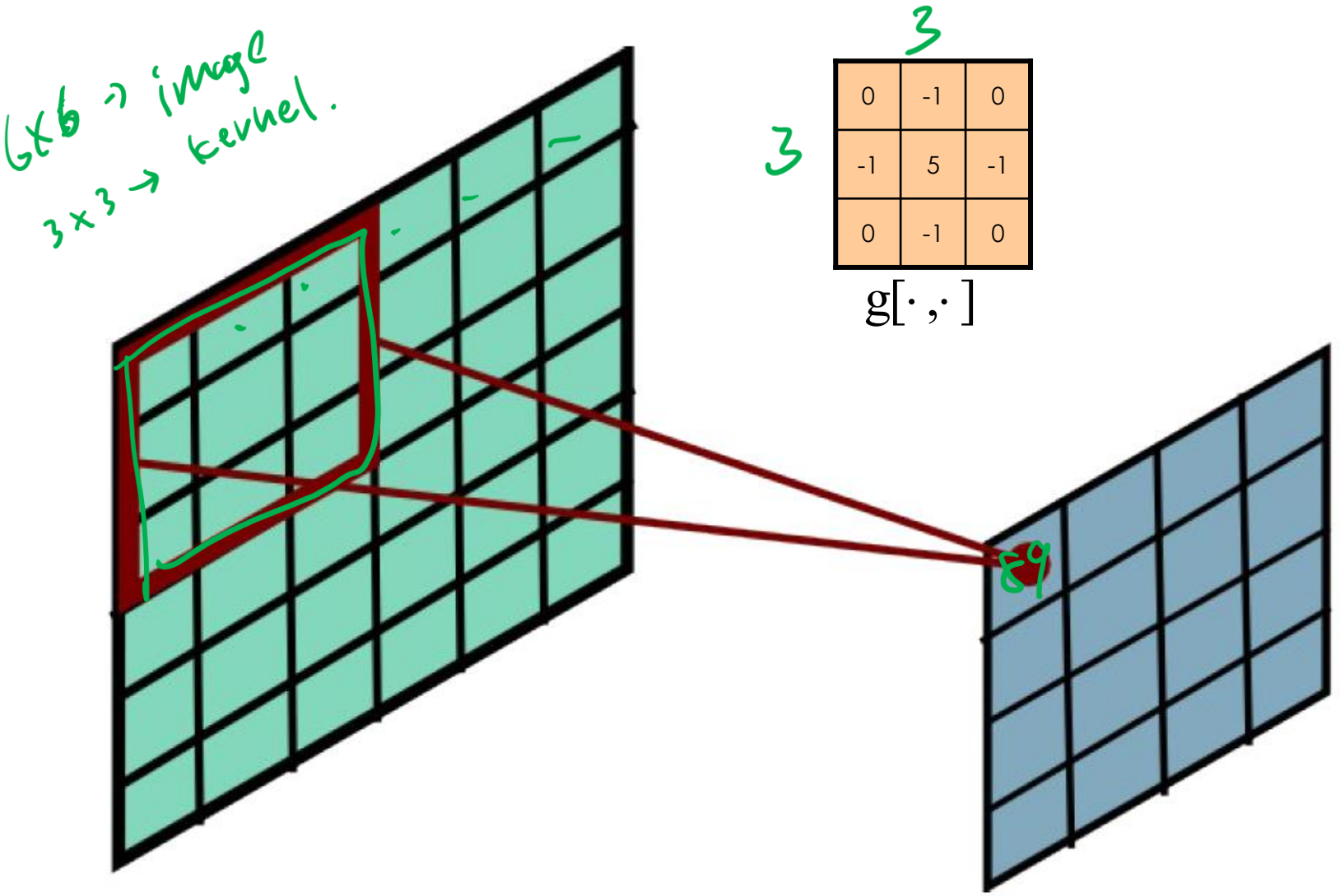


center of 3x3 window.

Image source: https://brilliant.org/wiki/convolutional-neural-network/

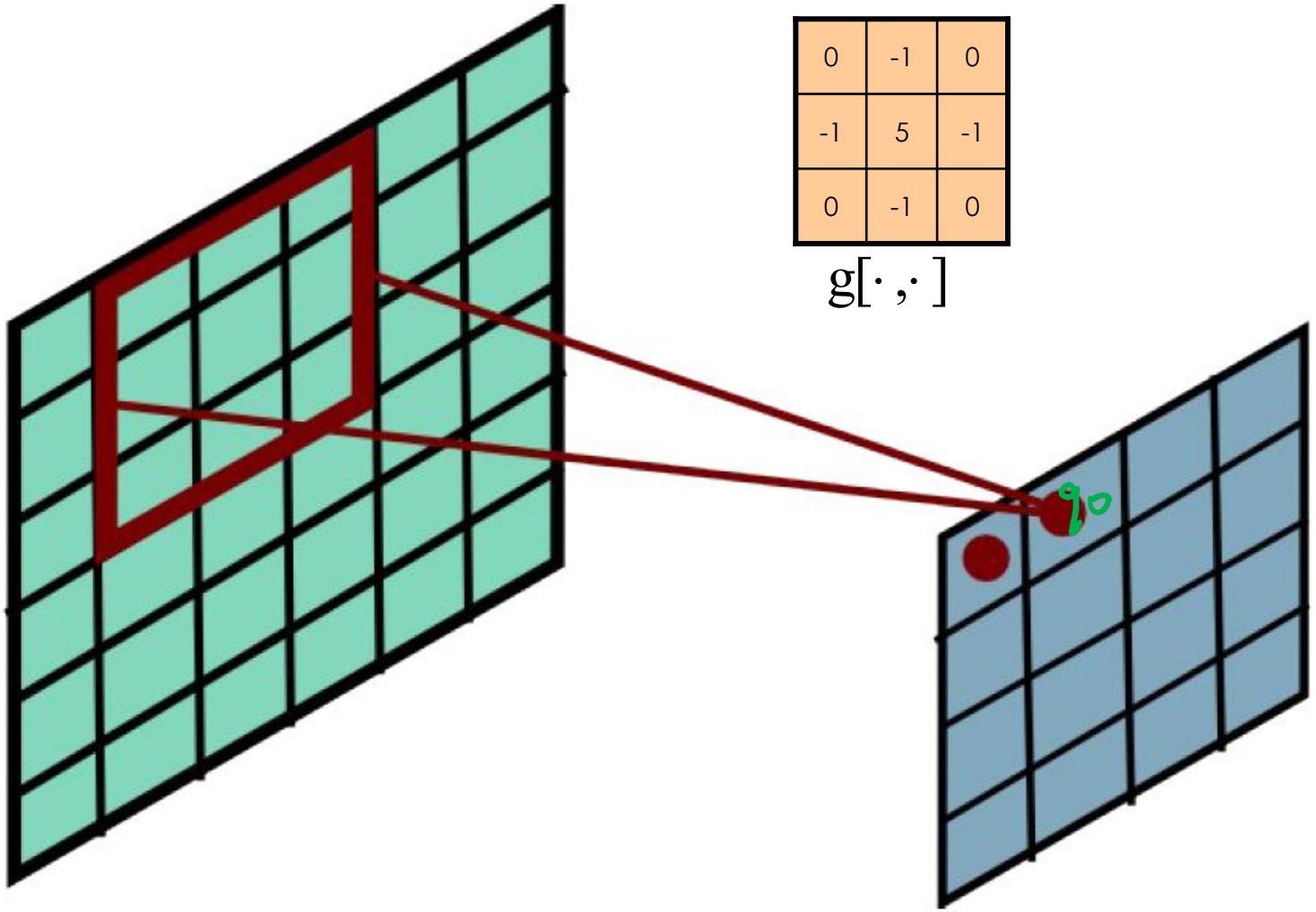
© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)



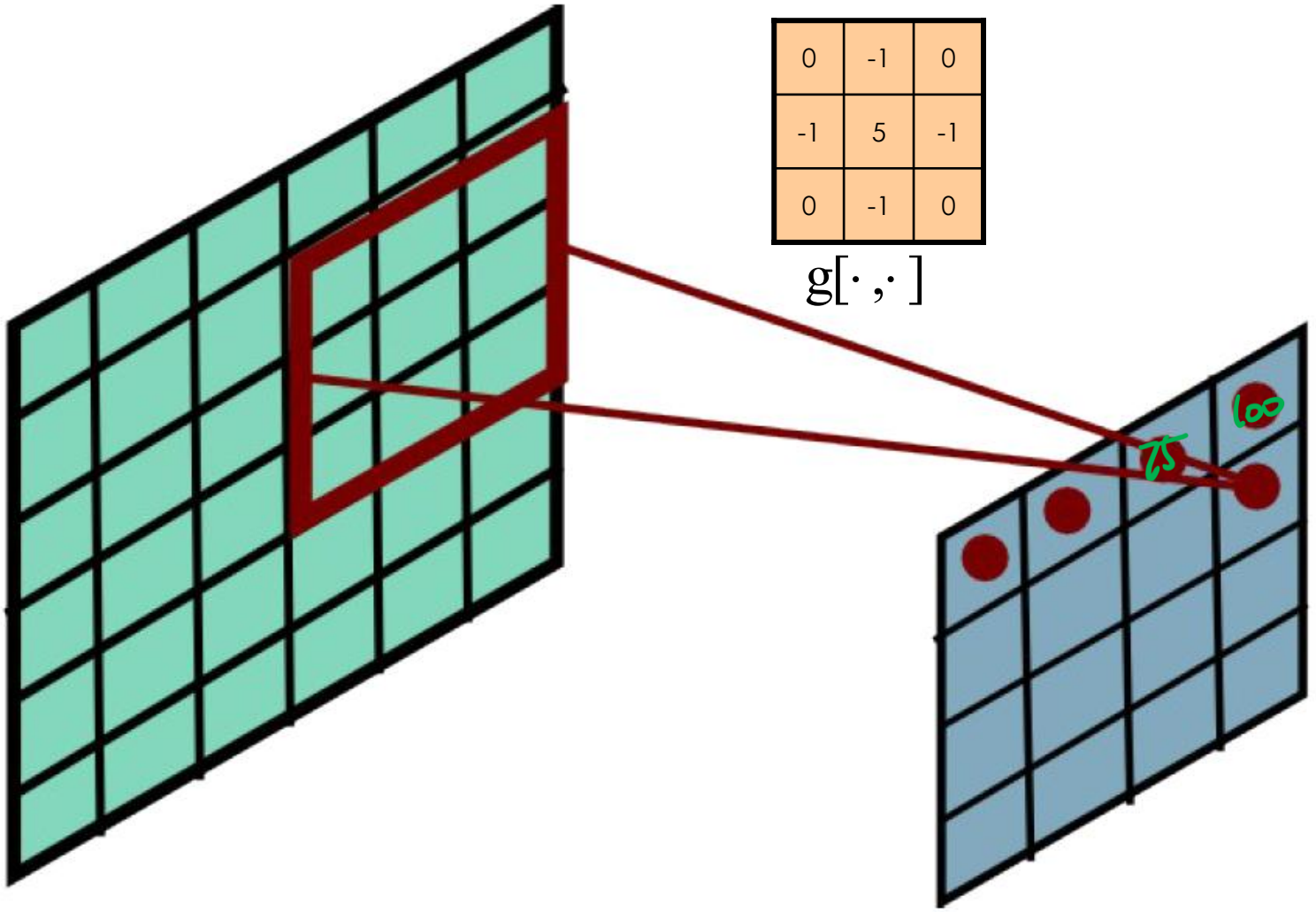
© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)



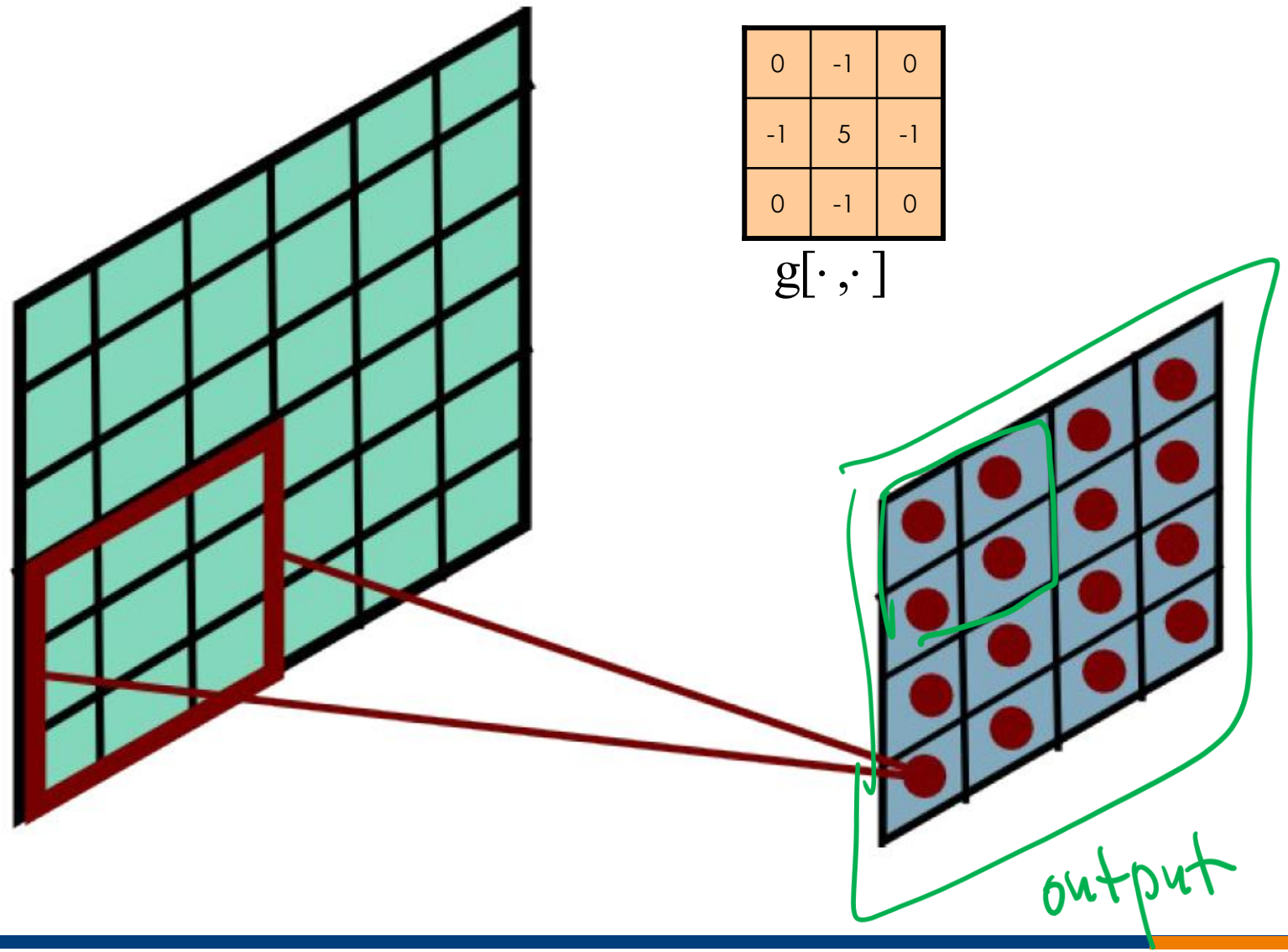
© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)



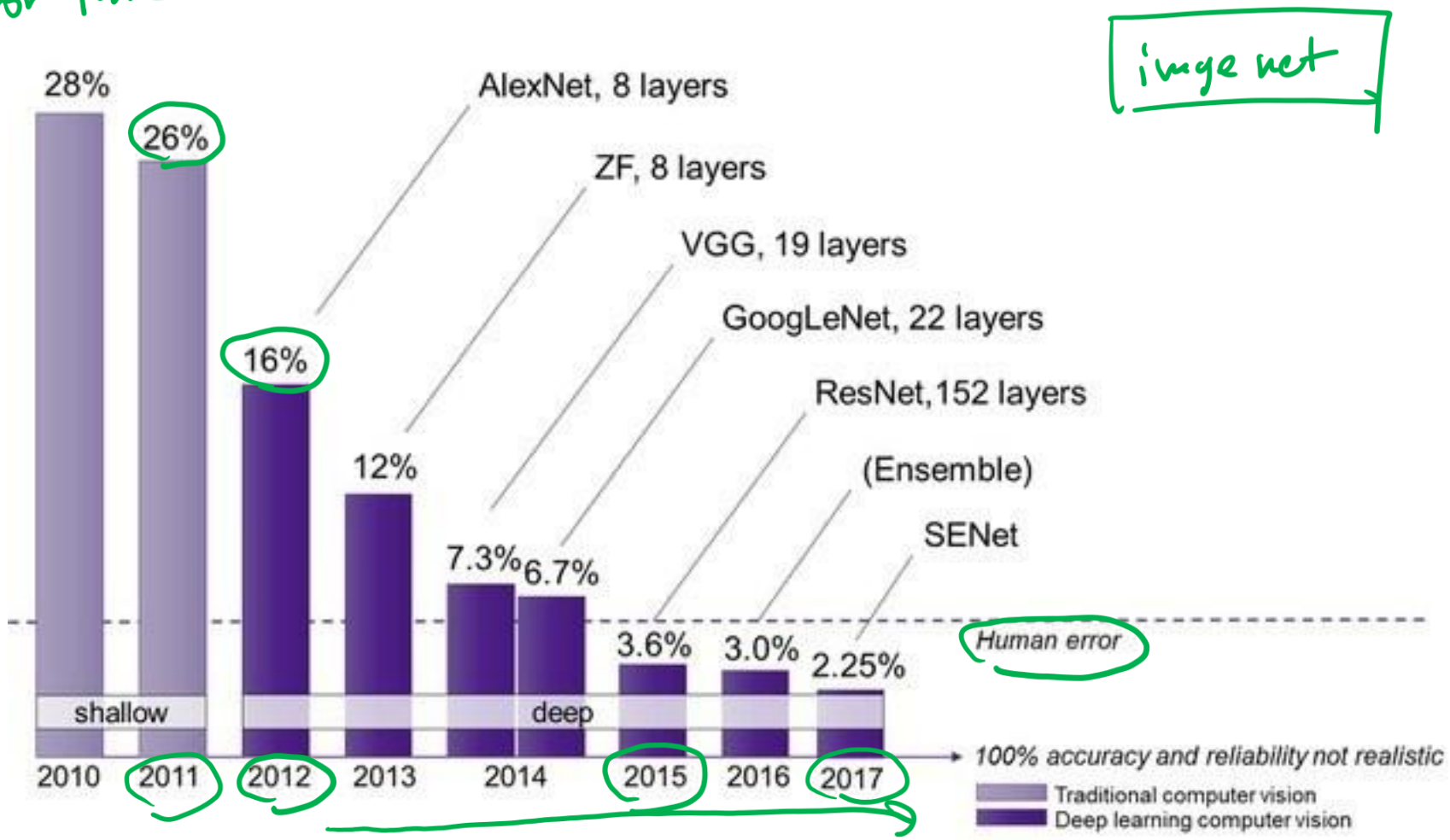
© Copyright EE, NUS. All Rights Reserved.

Convolutional Neural Network (CNN)



© Copyright EE, NUS. All Rights Reserved.

Neural Networks are Effective



© Copyright EE, NUS. All Rights Reserved.

Summary

- Introduction to Neural Networks
  - Multi-layer perceptron
  - Activation Functions
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

© Copyright EE, NUS. All Rights Reserved.

Practice Question

- Which of the following statements is correct?
  - A) Sigmoid serves as a linear activation function in neural networks.
  - B) Compared to dense connection (connecting each node in previous layer to the current layer), convolutional neural networks (CNNs) yield more computational complexity.
  - C) When training neural networks, its network weights W are fixed.
  - D) None of the others is correct.

© Copyright EE, NUS. All Rights Reserved.