Image Classification – COMP4423 Computer Vision

Xiaoyong Wei (魏驍勇)
x1wei@polyu.edu.hk

Department of Computing
電子計算學系

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Opening Minds • Shaping the Future
啟迪思維 • 成就未來

COMP4423 @ XIAOYONG

New Toy


Testing ... Rock
ures= 300
eat= 24.7445950
ures= 300
eat= 24.4998880
ures= 300
eat= 24.7053649
ures= 300
eat= 24.5307696
ures= 300
eat= 25.1461625
ures= 300
eat= 26.2269585
ures= 300
ures= 300
if len(results) == 0:
master*
Ln 138, Col 39
Spaces: 4
UTF-8
LF
Python
3.10.8 64-bit

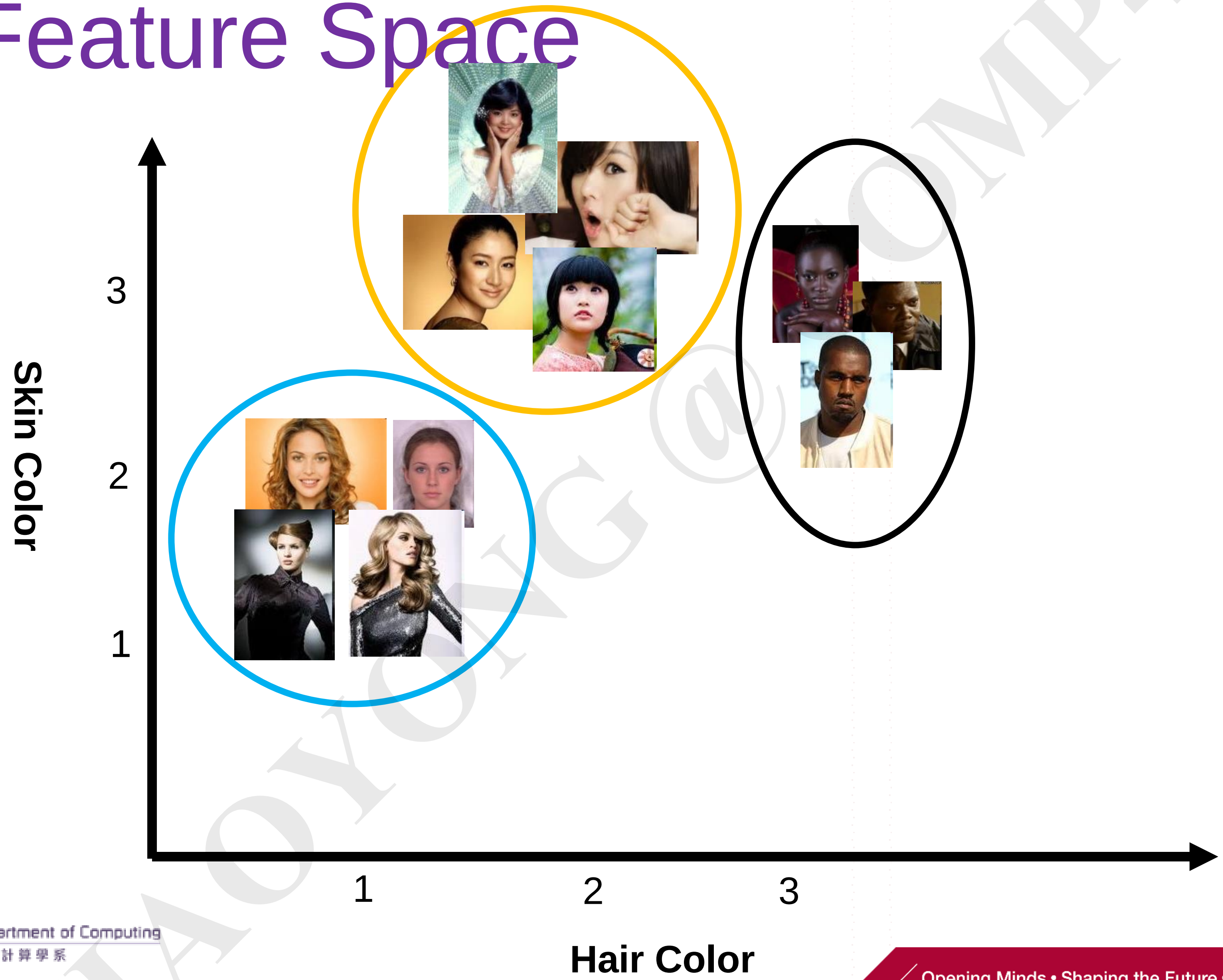
Outline

- >Classification
- >Supervised learning
- >K nearest neighbors (k-NN)
- >Bayesian classifiers
- >Support vector machines (SVM)

XIAOYONG @ COMP4

How do you group them?

Feature Space



XIAOYONG @ COMP4

Clustering is **unsupervised learning** which means we (human) don't have to tell the computers what each group looks like. It's data-driven without using human knowledge (supervision).

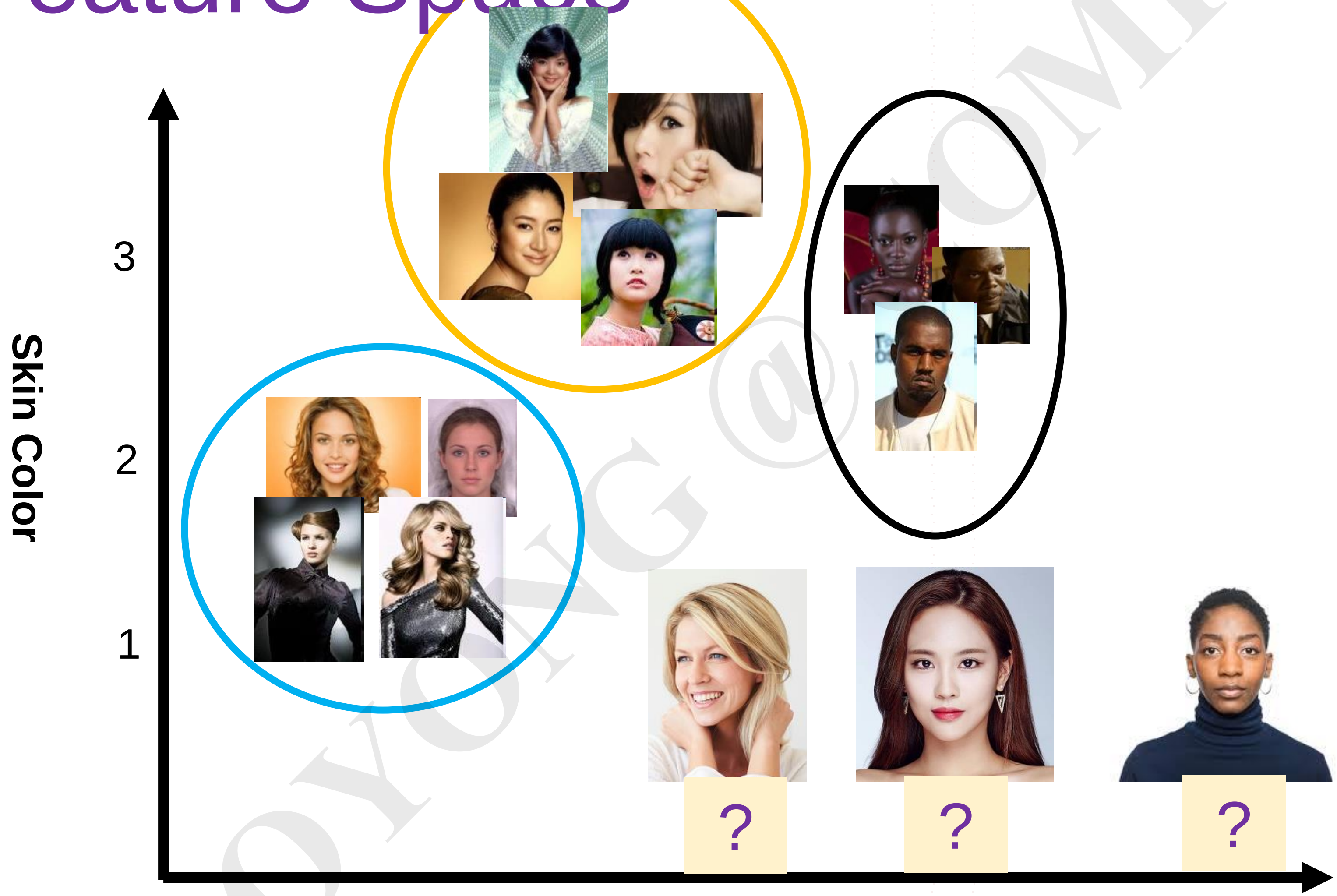
XIAOYONG @ COMP42

Sounds good?

But …

XIAOYONG @COMP4

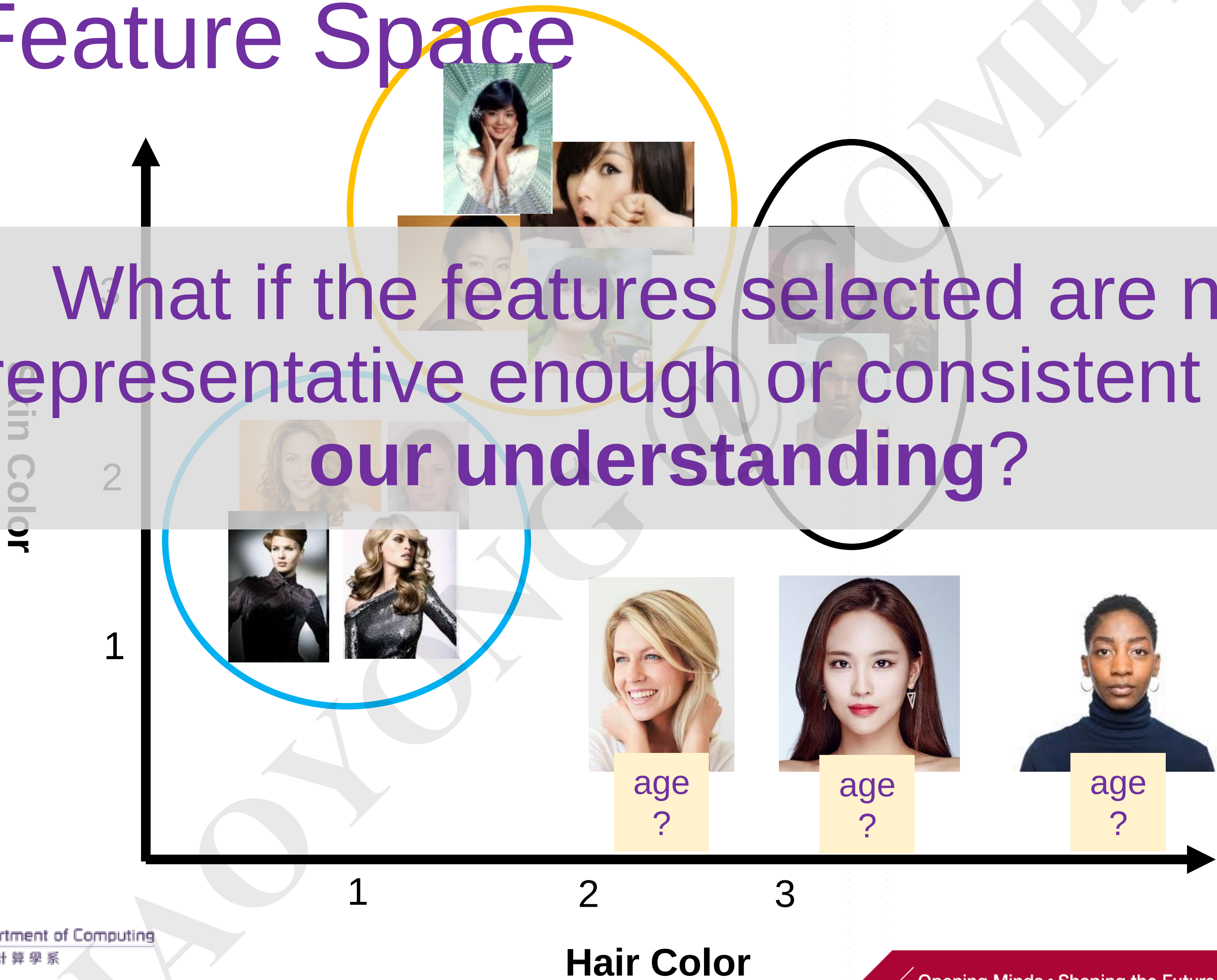
Feature Space



What if we have new examples unseen?

Feature Space

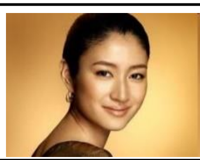
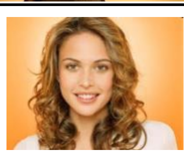
What if the features selected are not representative enough or consistent with **our understanding**?



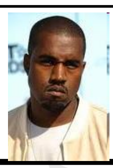
We can **tell** the computers about **our understanding** on the subjects by giving **labels**.

XIAOYONG@COMP4

Training Examples (Seen)

	Hair Color (H)	Skin Color (S)	Class Label (L)
	Yellow (2)	White (1)	W
	Black (3)	Yellow (2)	A
	Black (3)	Yellow (2)	A
	Yellow (2)	White (1)	W
	Black (3)	Black (3)	B

Testing Examples (Unseen)

	Hair Color (H)	Skin Color (S)	Class Label (L)
	2.2	0.8	?
	3.2	1.9	?
	3.1	2.2	?
	2.4	1.3	?
	3.1	2.9	?

XIAOYONG @ COMP4

Classification: to predict the labels of the testing (unseen) examples based on the knowledge learned from the training (seen) examples

XIAOYONG@COMP4

Classification: to predict the labels of the testing (unseen) examples based on the knowledge **learned** from the **training** (seen) examples

We are training the computers and the processing is called **training**.
The computers are learning. This is what the term “**machine learning**” is referring to.
Our participation in learning makes it
supervised learning.

Classification: to predict the labels of the testing (unseen) examples based on the **knowledge** learned from the training (seen) examples

The result is the machine's understanding of the knowledge.
We call it a **model** sometimes.

XIAOYONG@COMP4

Classification

Training Set

Samples	Features	Labels
1	[0.1, …]	1
2	[0.3, …]	-1
…	…	…

Validation Set

Samples	Features	Labels
1	[0.5, …]	-1
2	[0.9, …]	1
…	…	…

Model

Training | **Testing**

Testing Set

Samples	Features	Labels
1	[0.8, …]	?
2	[0.7, …]	-?
…	…	…

Samples	Features	Labels
1	[0.8, …]	-1
2	[0.7, …]	1
…	…	…

XIAOYONG @ COMP4

Then how?

XIAOYONG @ COMP42

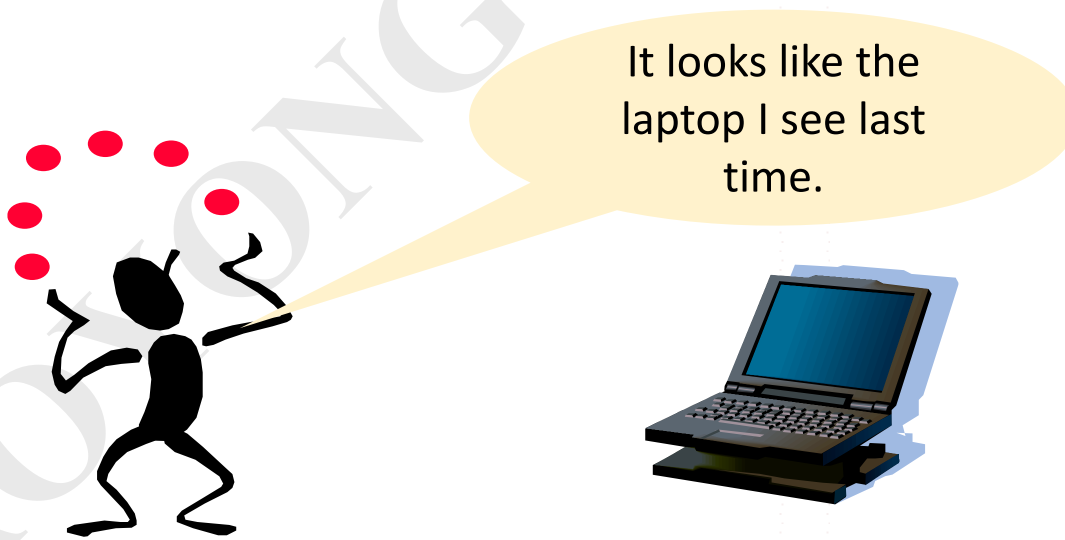
Instance-based Learning

PDA

Cellphone

Desktop PC

Laptop



How about we use image retrieval to find the most similar ones from the seen examples for reference?

XIAOYONG@COMP4422

kNN Classifiers

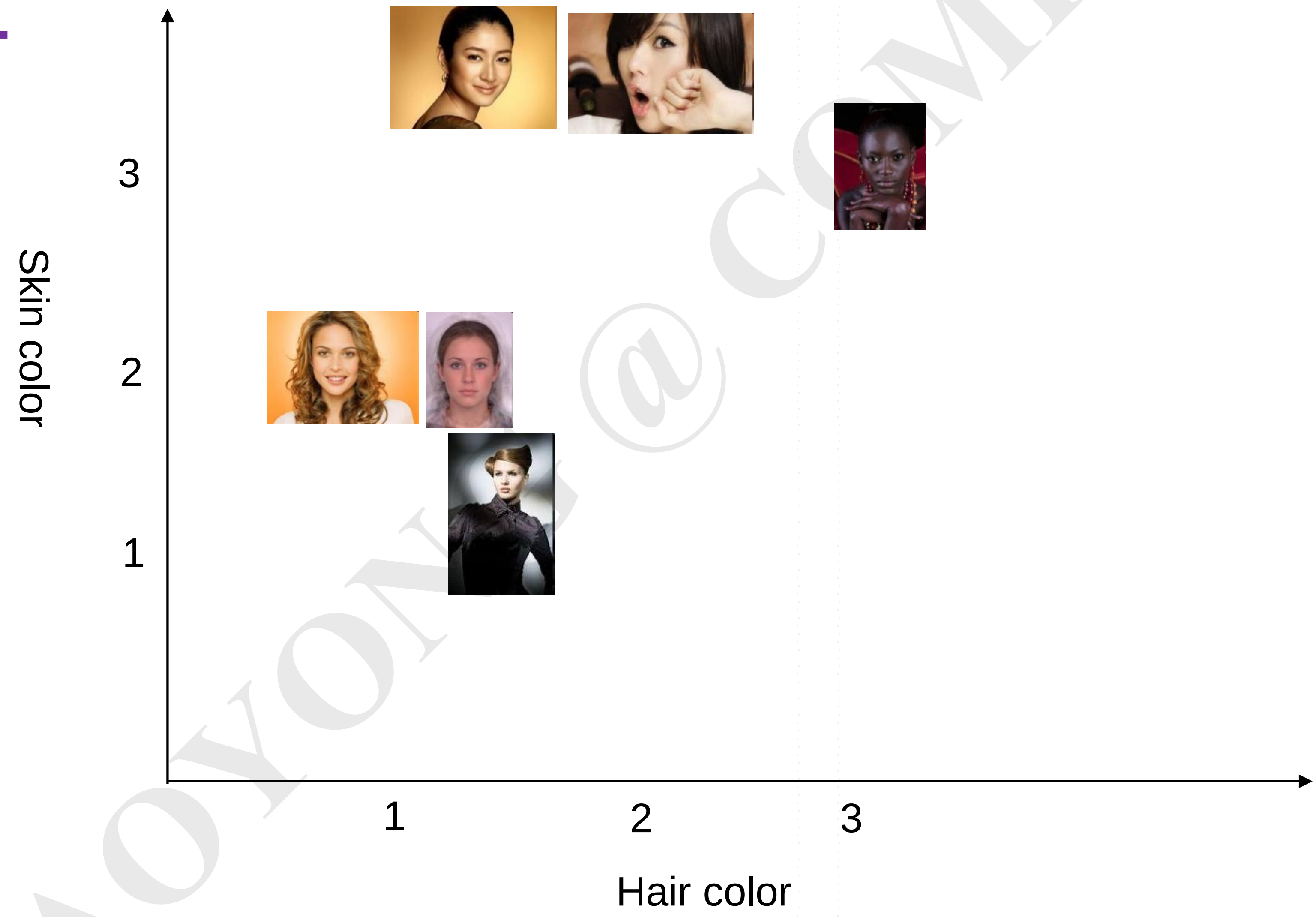
XIAOYONG @ COMP4

kNN Classifier

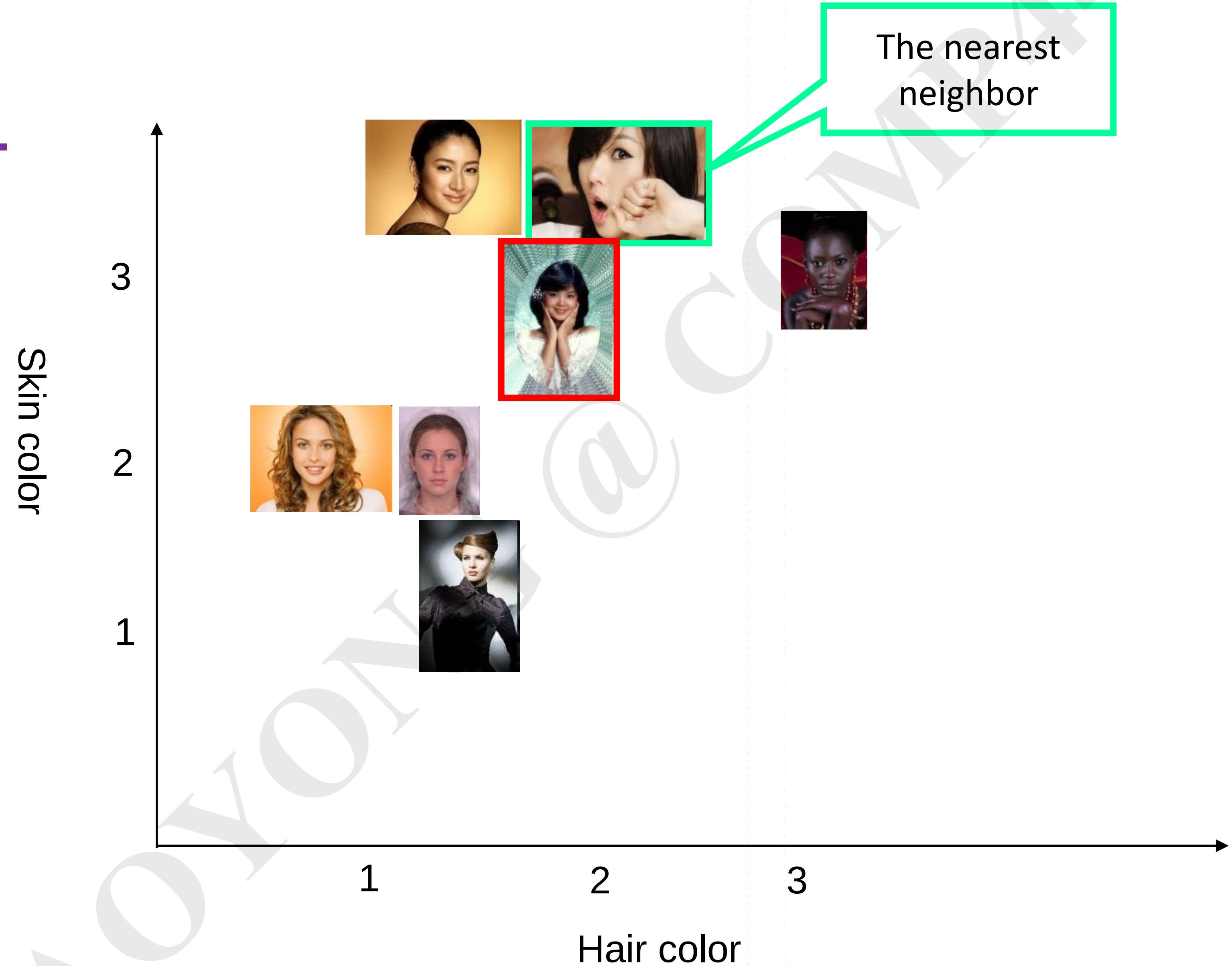
- NN: nearest neighbors
- k: number of nearest neighbors
- Idea
  - When k=1: assign the unseen with the label of its nearest neighbor
  - 近朱者赤，近墨者黑(If you lie down with dogs, you will get up with fleas)

XIAOYONG@COMP4…

k=1



k=1

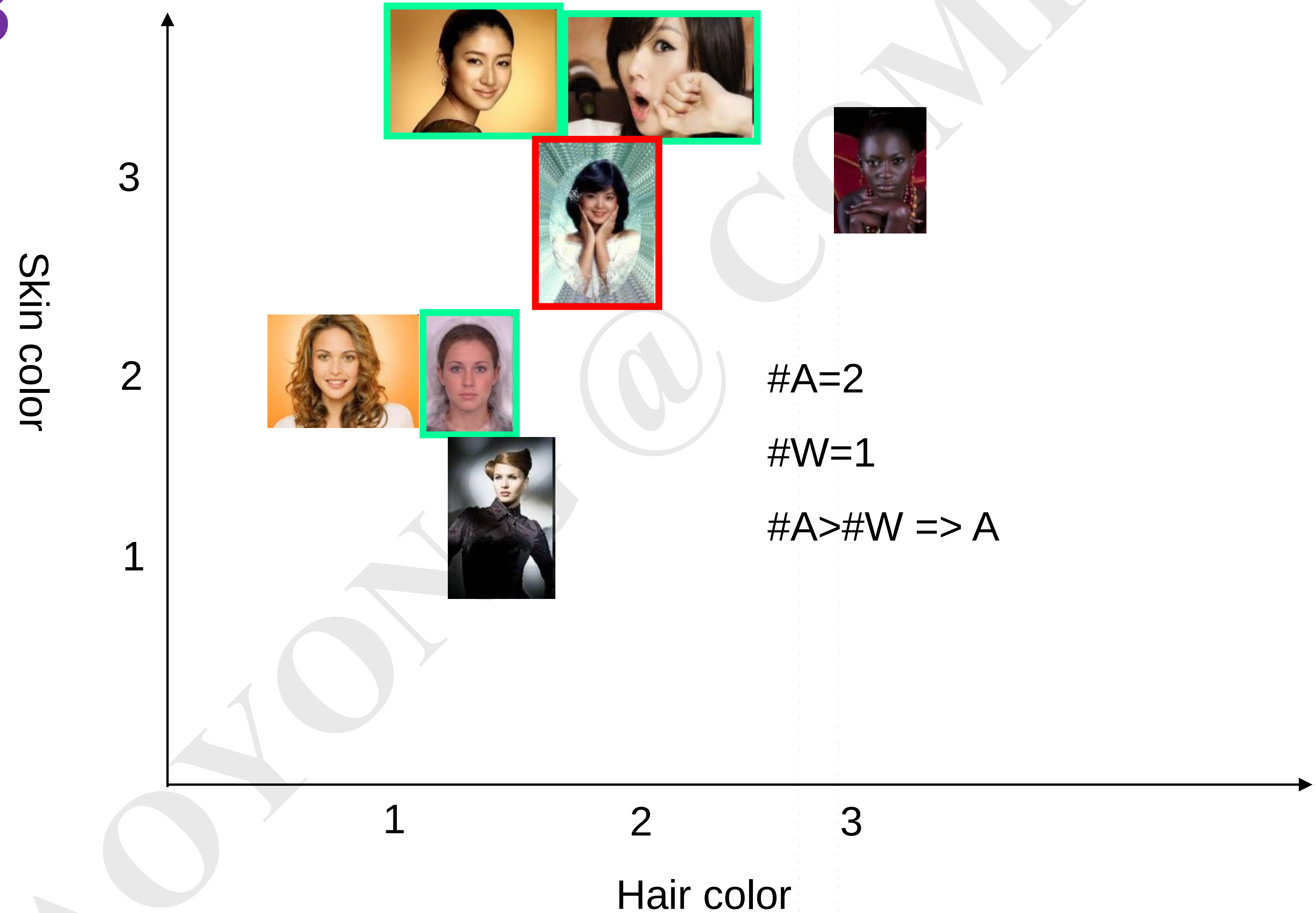


kNN Classifier

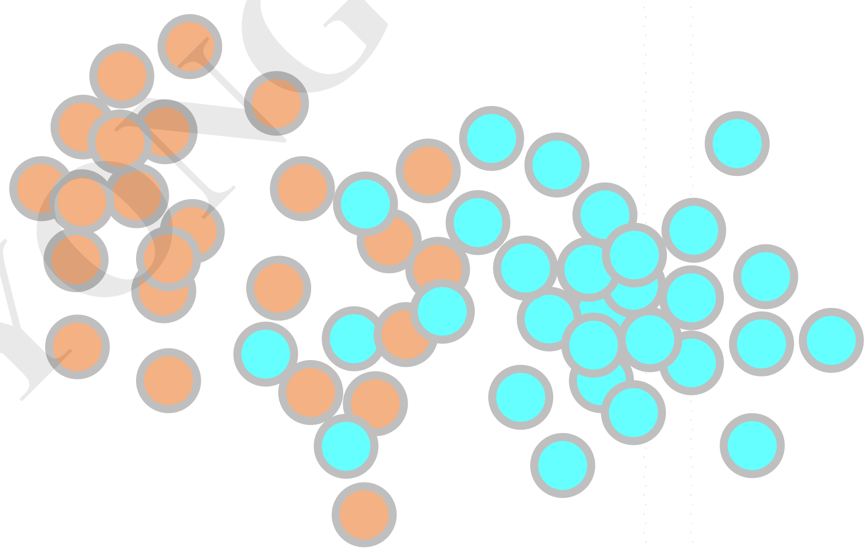
- NN: nearest neighbors
- k: number of nearest neighbors
- Idea
  - k=1: assign the unseen with the label of its nearest neighbor
  - **k>1: assign the dominating label among these of the k nearest neighbors**

XIAOYONG@COMP4

k=3

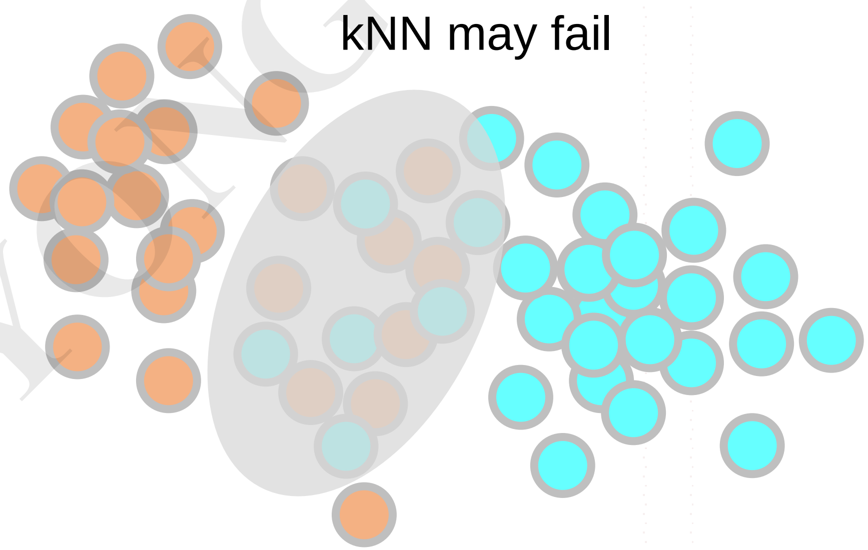


It's straightforward. But so far, we picked the simplest case (classes are well separated) for illustration purpose.
In a more general sense, this is what we're going to have.



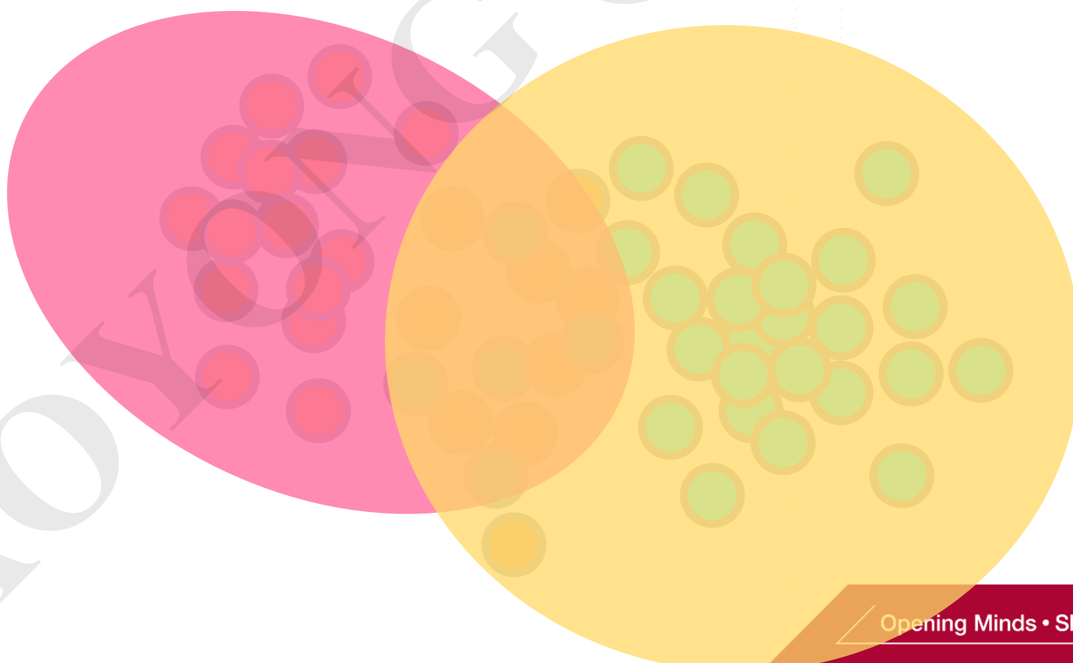
XIAOYONG @ COMP4

It's straightforward. But so far, we picked the simplest case (classes are well separated) for illustration purpose. In a more general sense, this is what we're going to have.



XIAOYONG @ COMP42

Can we teach the computer to draw a "circle" for each of the class and evaluate the membership of an example by measuring how much it falls into "circles"?

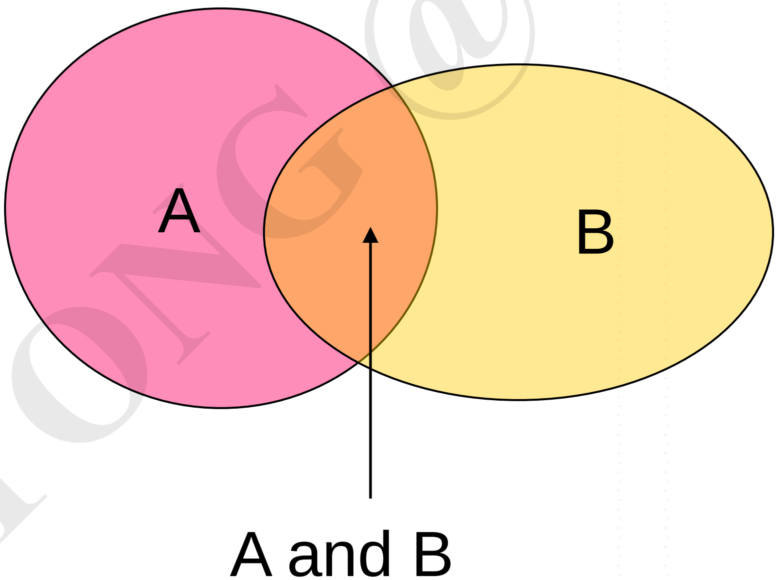


Bayesian Classifiers

XIAOYONG @ COMP4

Bayesian Classifiers

>Classes A and B as two sets

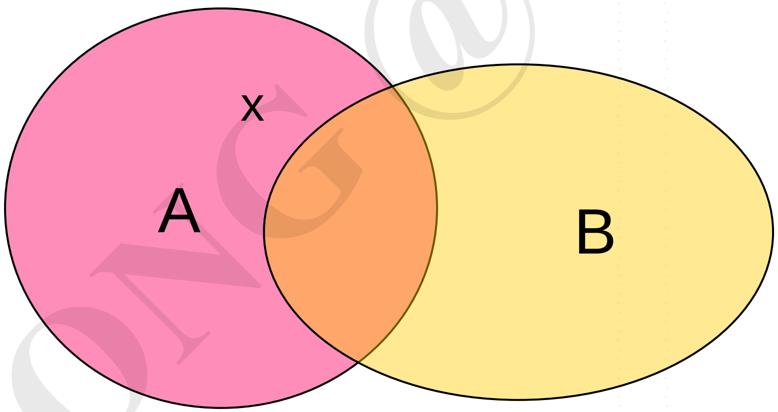


XIAOYONG @ COMP42

Bayesian Classifiers

>Classes A and B as two sets

- P(A|x): the probability of A is observed when seeing an x
- P(B|x): the probability of B is observed when seeing an x



XIAOYONG @ COMP4

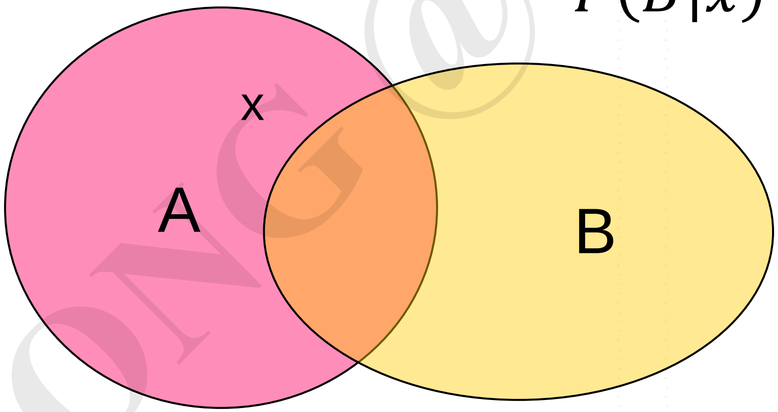
Bayesian Classifiers

>Classes A and B as two sets

- P(A|x) ∝ P(x|A)P(A)
- P(B|x) ∝ P(x|B)P(B)

$$P(A|x) = \frac{P(x|A)P(A)}{P(x)}$$

$$P(B|x) = \frac{P(x|B)P(B)}{P(x)}$$

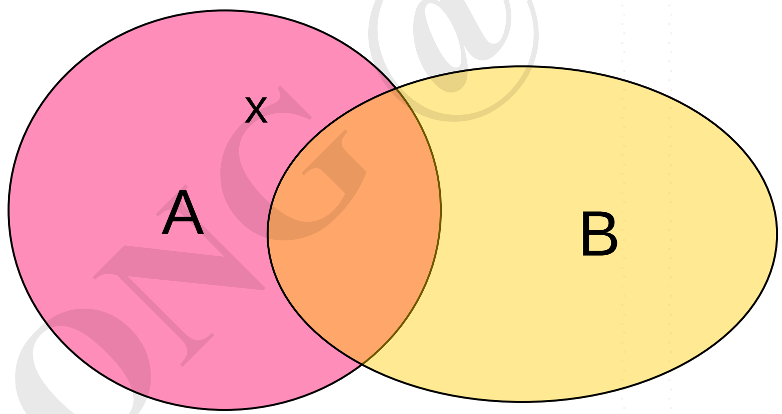


https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

Bayesian Classifiers

>Classes A and B as two sets

- $P(A|x) \propto P(x|A)P(A)$
- $P(B|x) \propto P(x|B)P(B)$



$P(A)=\#A/(\#A+\#B)$ $P(B)=\#B/(\#A+\#B)$

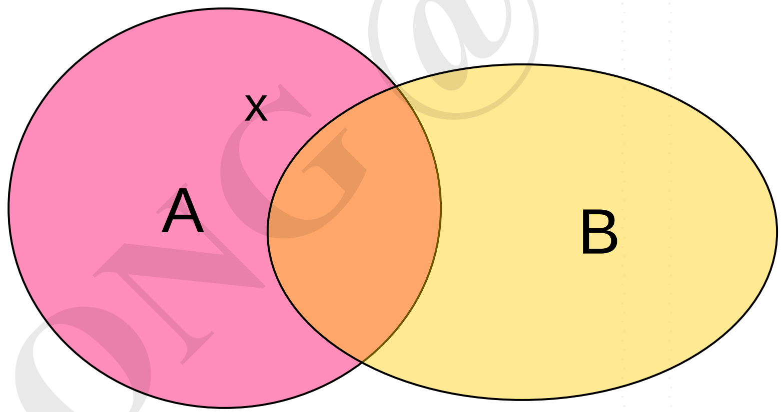
The Prior Probability

XIAO YONG @ COMP44

Bayesian Classifiers

>Classes A and B as two sets

- $P(A|x) \propto P(x|A)P(A)$
- $P(B|x) \propto P(x|B)P(B)$



P(x|A)=#x/#A

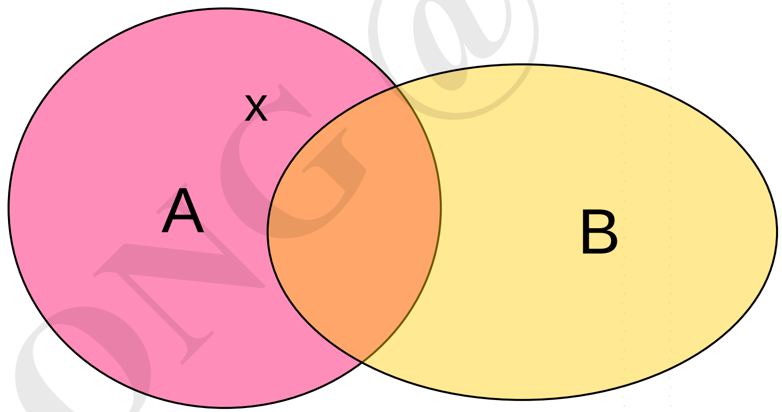
P(x|B)=#x/#B

The Conditional Probability (Likelihood)

Bayesian Classifiers

>Classes A and B as two sets

- $P(A|x) \propto P(x|A)P(A)$
- $P(B|x) \propto P(x|B)P(B)$



Decision function: x is $\begin{cases} A & \text{if } P(A|x)>P(B|x), \\ B & \text{otherwise} \end{cases}$

Naive Bayesian

> Advantages:

- Fast
- Extendable to multi-class problems
- Requires less training examples
- Works well for categorical data

> Disadvantages:

- **Features are assumed to be independent to each** other (not true in real-world applications)
- Zero-frequency problem

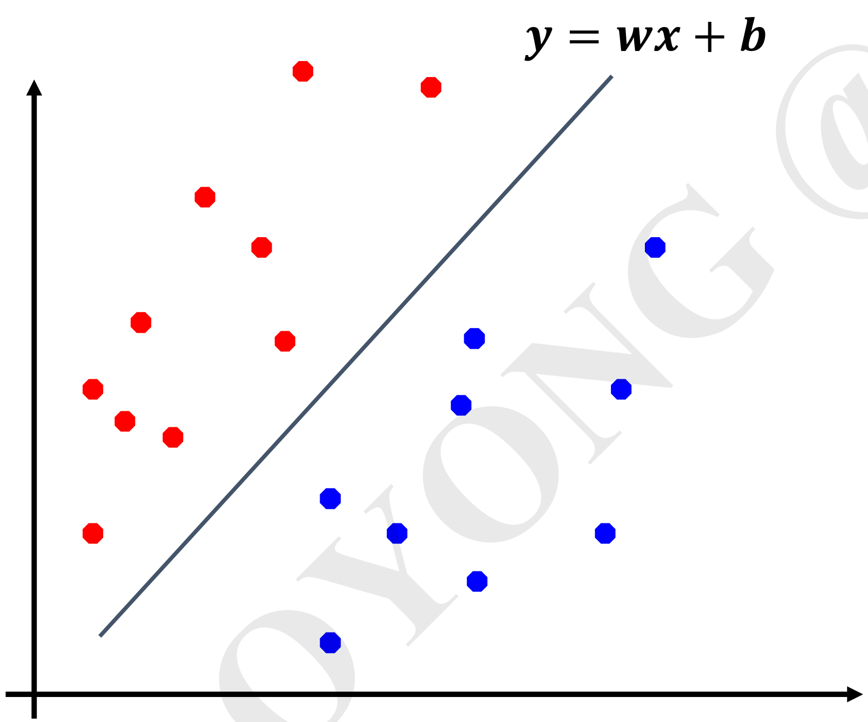
XIAOYONG @ COMP4

Support Vector Machines

XIAOYONG @ COMP4

Linear Separators

> Binary classification can be viewed as the task of separating classes in feature space:



$$-y + wx + b = 0$$

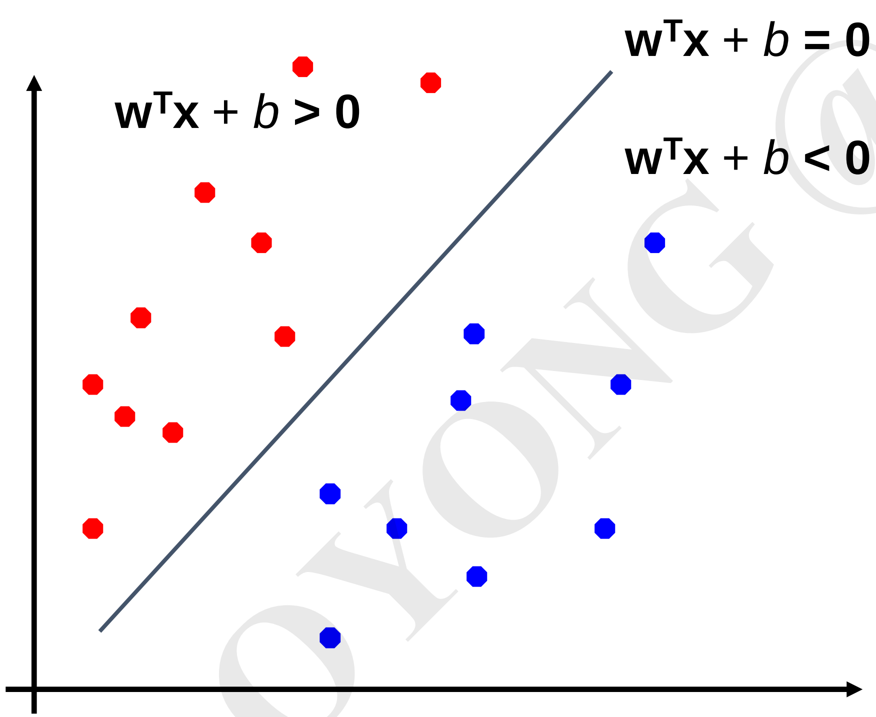
$$\begin{bmatrix} w \\ -1 \end{bmatrix} [x \quad y] + b = 0$$

$$\mathbf{w}^T\mathbf{x} + b = 0$$

XIAOYONG @ COMP42

Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:

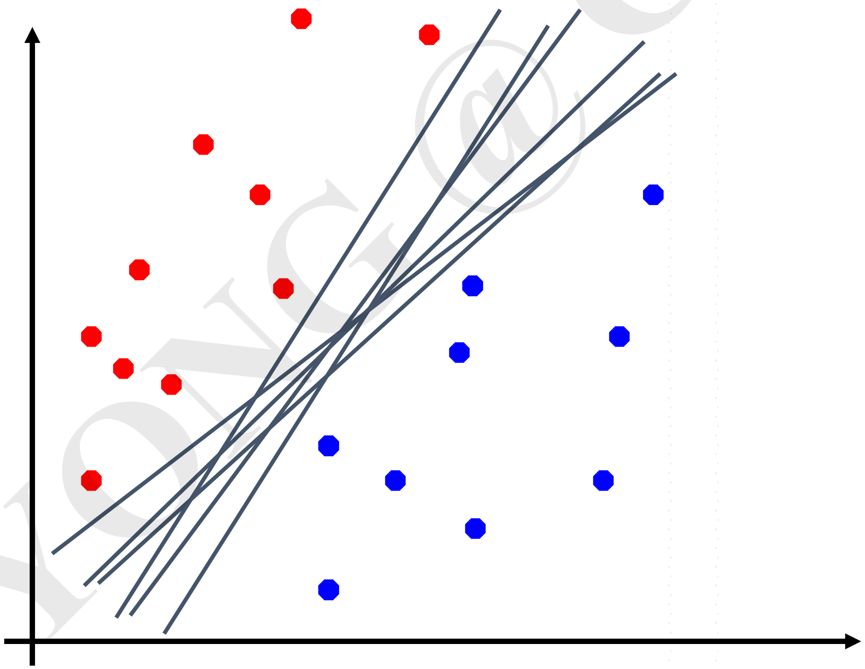


$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

XIAOYONG @ COMP4

Linear Separators

>Binary classification can be viewed as the task of separating classes in feature space:

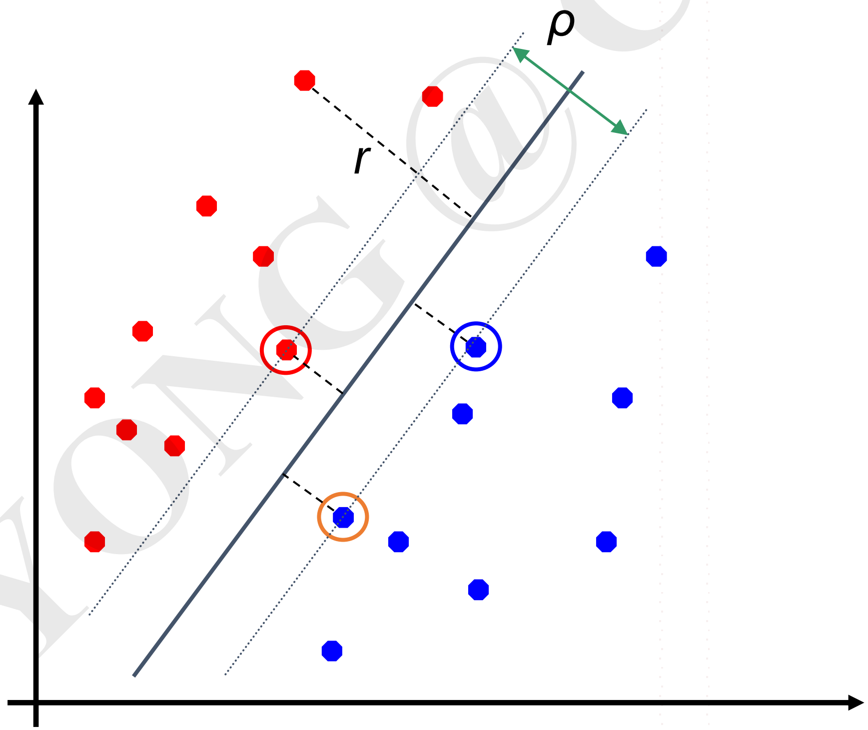


>Which one is the best?

XIAOYONG @ COMP4

Margin

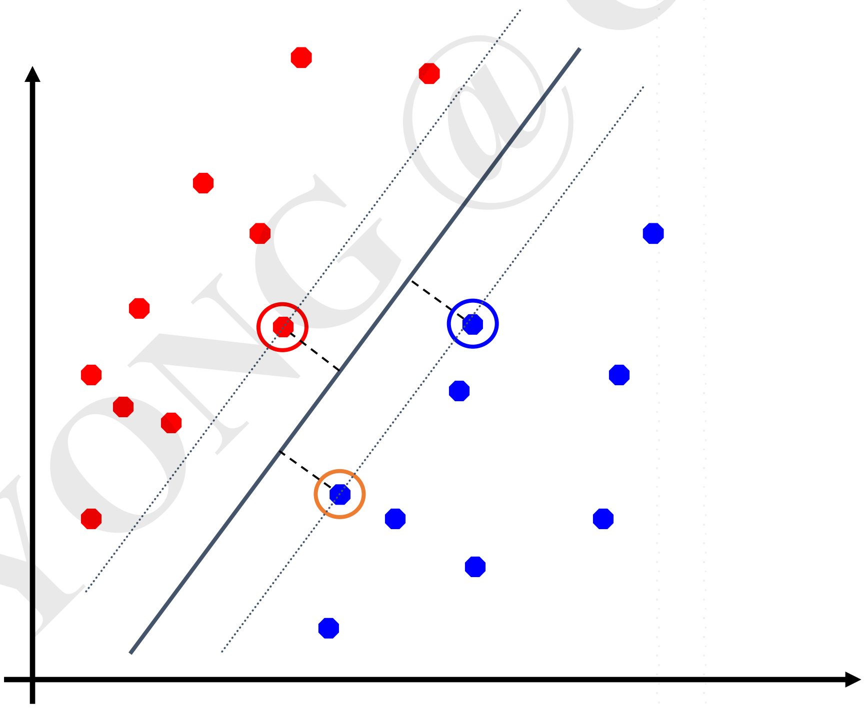
- Distance from example $\mathbf{x}_i$ to the separator is $r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



XIAOYONG @ COMP42

Maximum Margin Classification

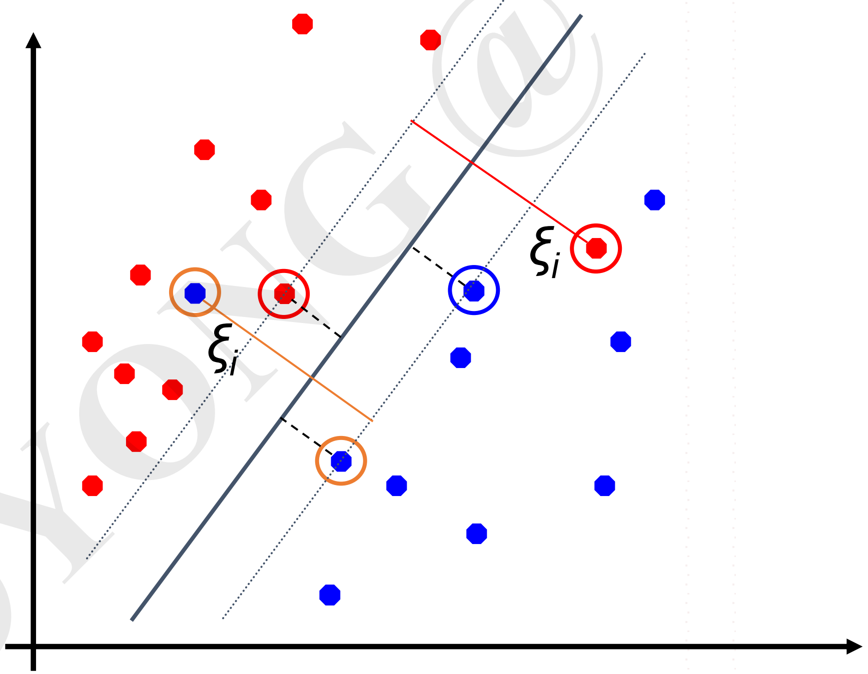
- Maximizing the margin is good according to intuition and PAC theory (Probably Approximately Correct).
- Implies that only support vectors matter; other training examples are ignorable.



XIAOYONG @ COMP4

Soft Margin Classification

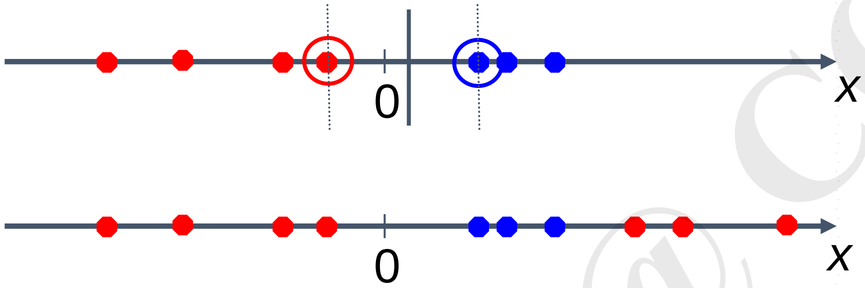
- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



XIAOYONG@COMP4432

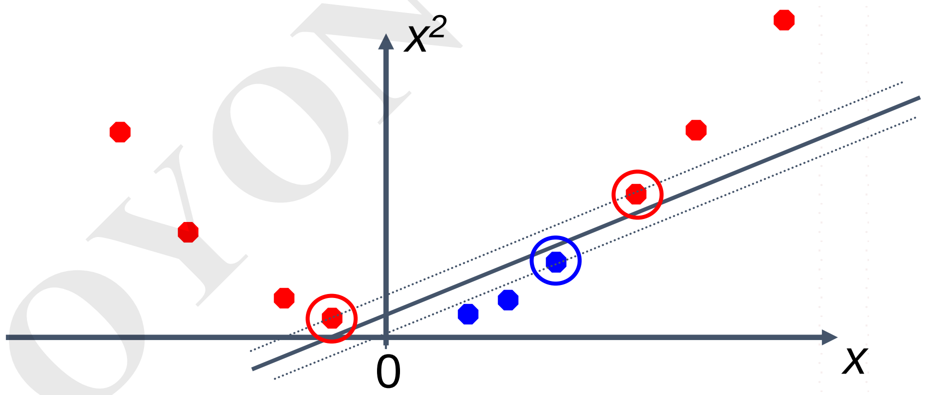
Non-linear SVMs

> Datasets that are linearly separable with some noise work out great:



> But what are we going to do if the dataset is just too hard?

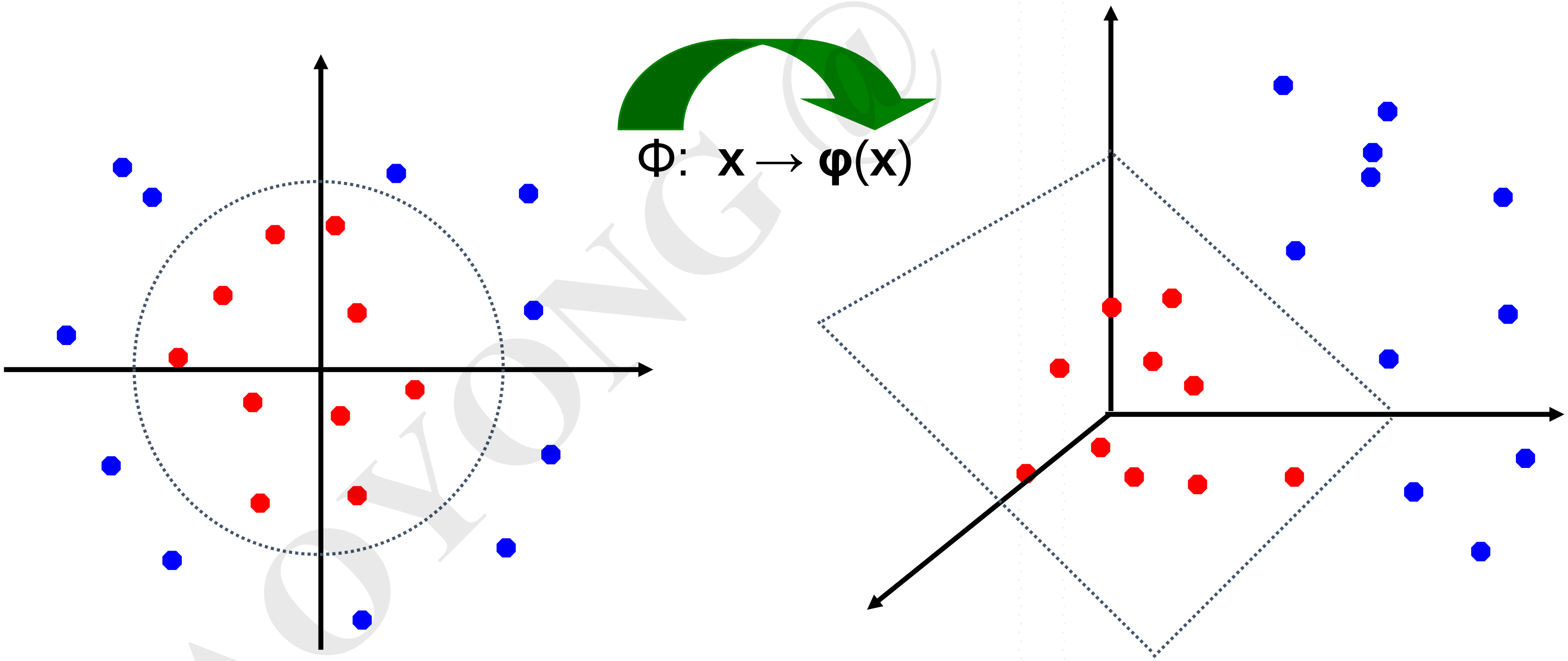
> How about… mapping data to a higher-dimensional space:



XIAOYONG @ COMP4432

Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



XIAOYONG @ COMP4

The "Kernel Trick"

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$$

- A *kernel function* is a function that is equivalent to an inner product in some feature space.
- Example:

2-dimensional vectors $\mathbf{x}=[x_1 \ \ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$$

$$= [1 \ \ x_{i1}^2 \ \ \sqrt{2}\, x_{i1}x_{i2} \ \ x_{i2}^2 \ \ \sqrt{2}x_{i1} \ \ \sqrt{2}x_{i2}]^T [1 \ \ x_{j1}^2 \ \ \sqrt{2}\, x_{j1}x_{j2} \ \ x_{j2}^2 \ \ \sqrt{2}x_{j1} \ \ \sqrt{2}x_{j2}] =$$

$$= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j), \quad \text{where } \boldsymbol{\varphi}(\mathbf{x}) = [1 \ \ x_1^2 \ \ \sqrt{2}\, x_1x_2 \ \ x_2^2 \ \ \sqrt{2}x_1 \ \ \sqrt{2}x_2]$$

- Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

XIAOYONG @ COMP4

Examples of Kernel Functions

> Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

> Polynomial of power *p*: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j)^p$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

> Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

> Higher-dimensional space still has *intrinsic* dimensionality *d* (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

XIAO YONG @ COMP4

Classification

Training Set

Samples	Features	Labels
1	[0.1, …]	1
2	[0.3, …]	-1
…	…	…

Validation Set

Samples	Features	Labels
1	[0.5, …]	-1
2	[0.9, …]	1
…	…	…

Model

Testing Set

Samples	Features	Labels
1	[0.8, …]	?
2	[0.7, …]	-?
…	…	…

Samples	Features	Labels
1	[0.8, …]	-1
2	[0.7, …]	1
…	…	…

Training **Testing**

XIAO YONG @ COMP4

The New Toy

XIAOYONG @ COMP4422

New Toy


Testing ... Rock
ures= 300
eat= 24.7445950
ures= 300
eat= 24.4998880
ures= 300
eat= 24.7053649
ures= 300
eat= 24.5307696
ures= 300
eat= 25.1461625
ures= 300
eat= 26.2269585
ures= 300
eat= 25.9468358
ures= 300
if len(results) == 0:
master*
Ln 138, Col 39 Spaces: 4 UTF-8 LF Python 3.10.8 64-bit

Feature Extraction

XIAOYONG @ COMP4

```python
import cv2
import requests
from requests_toolbelt.multipart.encoder import MultipartEncoder
import numpy as np, random

cap = cv2.VideoCapture(0)
WindName = "Toy Program @ COMP 4423"
cv2.namedWindow(WindName)
cv2.resizeWindow(WindName, 1024, 768)

#polysmart_svr_url = 'http://127.0.0.1:8000/'
polysmart_svr_url = 'http://158.132.255.32:8088/'

polysmart_facerecg_svr = polysmart_svr_url+'handdetect/'

def detect(pic):
    _, im_buf = cv2.imencode(".jpg", pic)
    byte_im = im_buf.tobytes()

    data = MultipartEncoder(fields={'file': ('img.jpg', byte_im)})
    response = requests.post(polysmart_facerecg_svr, data=data, headers={
        'Content-Type': data.content_type})
# print((response.status_code,response.json()))
    retJson = response.json()
    return retJson['results'] if retJson['code'] >= 0 else []

connections = [[4, 3, 2, 1, 0],# thumb
               [8, 7, 6, 5],# index
               [12, 11, 10, 9],# middle
               [16, 15, 14, 13],# ring
               [20, 19, 18, 17, 0], #pinky
               [3, 5, 9, 13, 17]# palm
               ]

```

```python
def draw_landmarks(image, landmarks):
    h, w, c = image.shape
    # print([h, w, c])
    id2cords = {}
    for lm in landmarks:
        idx, ftx, fty = lm['idx'], int(lm['x']*w), int(lm['y']*h)
        id2cords[idx] = [ftx, fty]
    for line in connections:
        pts = [[id2cords[idx][0], id2cords[idx][1]] for idx in line]
        pts = np.array(pts, np.int32)
        pts = pts.reshape((-1, 1, 2))
        image = cv2.polylines(image, [pts], False, (0, 128, 128), 4)
    for idx in id2cords:
        image = cv2.circle(
            image, (id2cords[idx][0], id2cords[idx][1]), 10, (224, 224, 0), 5)
    image = cv2.circle(
        image, (id2cords[8][0], id2cords[8][1]), 15, (0, 0, 128), 5)
    return image, id2cords

def extrac_feature(id2cords):
    feat=[]
    for id in range(21):
        a=np.array(id2cords[id])
        for tag in range(id+1,21):
            b=np.array(id2cords[tag])
            dist=np.linalg.norm(a-b)/800 # normalize the distane in the range of [0,1] by assuming the 800 is the maximum dist possible
            feat.append(dist)
    print('sum feat=',sum(feat))
    return feat
```

XIAO YONG @ COMP4423

```python
feat_x,feat_y=[],[]
while True:
    success, image = cap.read()
    if not success:
        continue
    image = cv2.flip(image, 1)
    imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    results = detect(imageRGB)
    id2cords = {}
    #print(['len=', len(results)])
    if len(results) == 0:
        print("Nothing detected ...")
    else:
        image, id2cords = draw_landmarks(image, results[0]['landmarks'])

    cv2.imshow(WindName, image)

    key=cv2.waitKey(1) & 0xFF
    if key == ord('q') or key==27:
        break

    if key == ord('p') and not id2cords =={}:
        # caputure a sample for the class 'paper'
        feat=extrac_feature(id2cords)
        feat_x.append(feat)
        feat_y.append(1)
    if key == ord('r') and not id2cords =={}:
        # caputure a sample for the class 'rock'
        feat=extrac_feature(id2cords)
        feat_x.append(feat)
        feat_y.append(2)
    if key == ord('x') and not id2cords =={}:
        # caputure a sample for the class 'rock'
        feat=extrac_feature(id2cords)
        feat_x.append(feat)
```

Training & Testing

XIAOYONG @ COMP4

```python
feat_x,feat_y=np.load('feat_x.npy'),np.load('feat_y.npy')

######################### run cross-validation #########################
from sklearn import svm
from sklearn.model_selection import cross_val_score
model = svm.SVC(kernel='rbf')
scores = cross_val_score(model, feat_x, feat_y, cv=10)
print('cross validation scores: ',scores)
#print(feat_x.shape)

######################### visualize the data #########################
from sklearn.manifold import TSNE
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(rc={'figure.figsize':(10,8)})
palette = sns.color_palette("bright", 3)
tsne = TSNE()
X_embedded = tsne.fit_transform(feat_x)
label2str={1:'Paper',2:'Rock',3:'Scissor'}
markers=[label2str[feat_y[i]] for i in range(len(feat_y))]
sns.scatterplot(x=X_embedded[:,0], y=X_embedded[:,1], markers=markers, hue=markers, legend='full', palette=palette)
plt.show()

######################### fit the model #########################
model.fit(feat_x,feat_y)
```

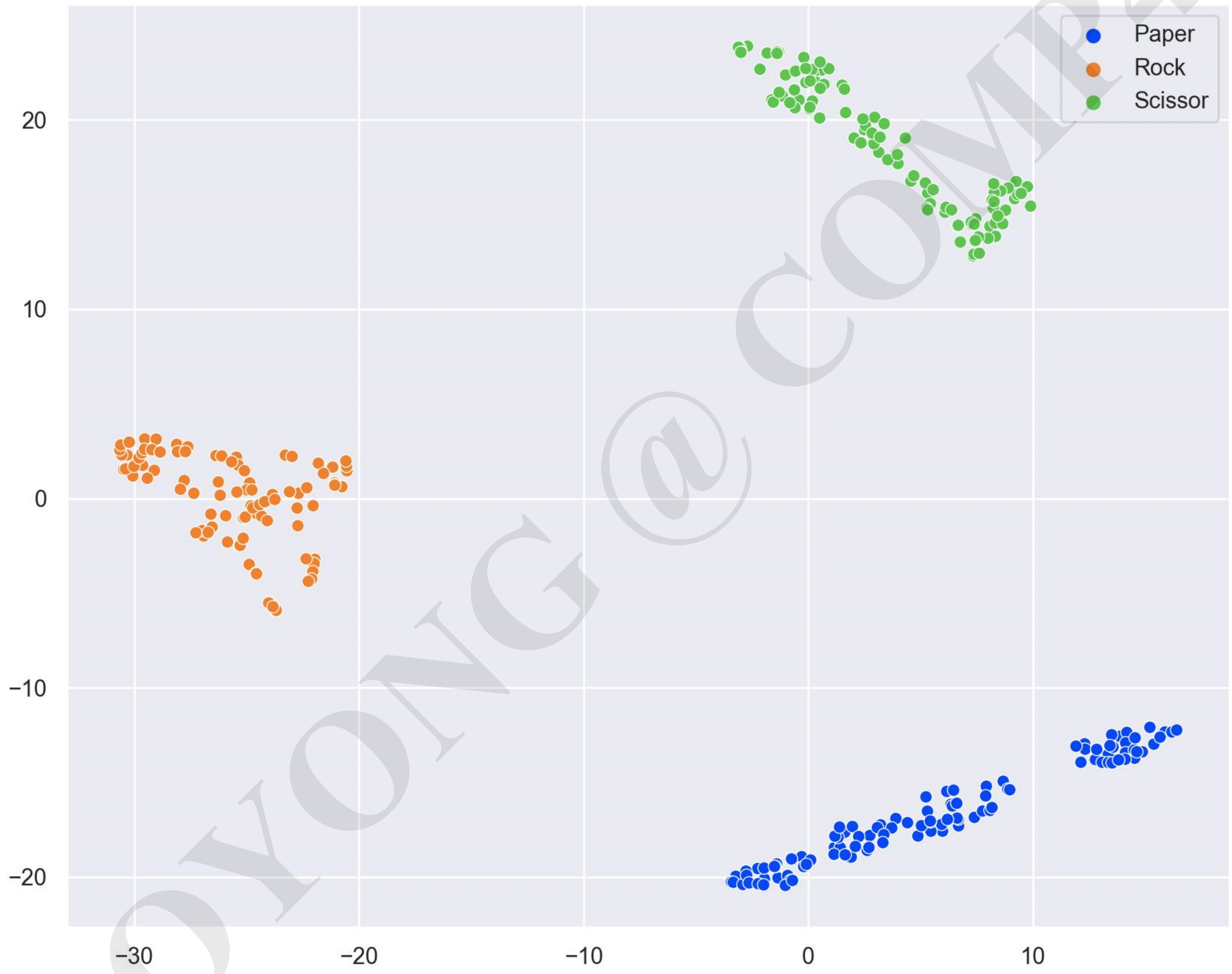

XIAO YONG @ COMP4423

```python
# show status text
if status_id>0: # in game mode
    dur=time.time()-game_start_time
    status_id=next((i for i in range(len(status_check_points)) if dur < status_check_points[i]),-1)
    if status_id==-1:
        status_id, game_start_time, votes = 1, time.time(), {} # restart a game
overlay=cv2.rectangle(overlay, (125,40),(1500,120),color=(213, 231, 242),thickness=-1)
image=cv2.addWeighted(overlay, 0.5, image, 0.5, 0)
image=cv2.putText(image,status_texts[status_id],(150,100),cv2.FONT_HERSHEY_SIMPLEX,1,(70, 62, 57),2)

id2cords = {}
if status_id in [0,4]:
    results = detect(imageRGB)

    if len(results) == 0:
        print("Nothing detected ...")
    else:
        image, id2cords = draw_landmarks(image, results[0]['landmarks'])
        feat=np.array([extrac_feature(id2cords)])
        print(feat.shape)
        label=model.predict(np.array(feat))[0]
        print('label=',label)
        votes[label]=votes[label]+1 if label in votes else 1
        if status_id==0:
            image=cv2.putText(image,label2str[label],(250,120),cv2.FONT_HERSHEY_TRIPLEX,5,(134, 152, 109),3)
        if status_id==4:
            computer_move=random.randint(1,3)

```



XIAOYONG @ COMP4

The Hong Kong Polytechnic University
香港理工大學
Department of Computing
電子計算學系

Thank you!