

專案：6

# 圖片分類及超參數最佳化

講師：Alex 



# Outline

---

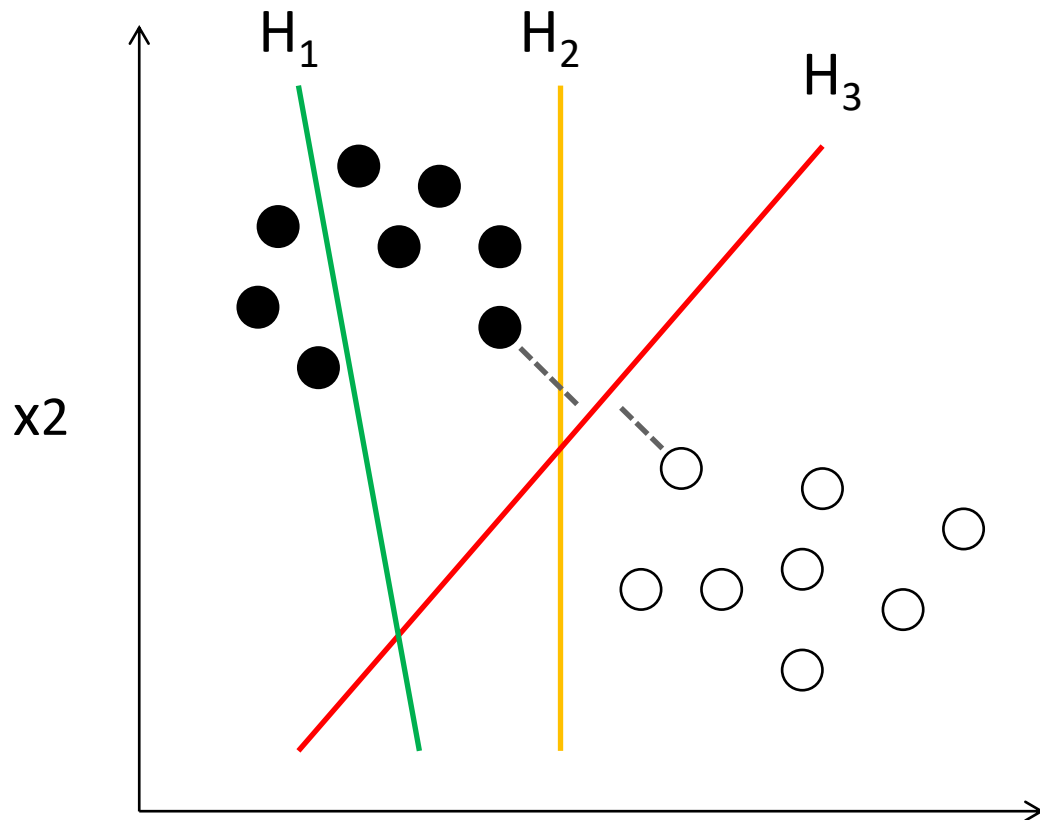
- The Introduction to Support Vector Machine
- Support Vector Machine Implementation 1 (6.1)
- HOG Feature and Multiclass Classification
- Support Vector Machine Implementation 2 (6.2)
- Loophole of Image Classification
- Support Vector Machine Implementation 3: Face Detection (6.3)
- Hyperparameter Optimization
- Optuna Basic Example (6.4)
- Face Classifier Optimization Example Using Optuna (6.5)
- Conclusion and Observation



# Support Vector Machine (支持向量機)

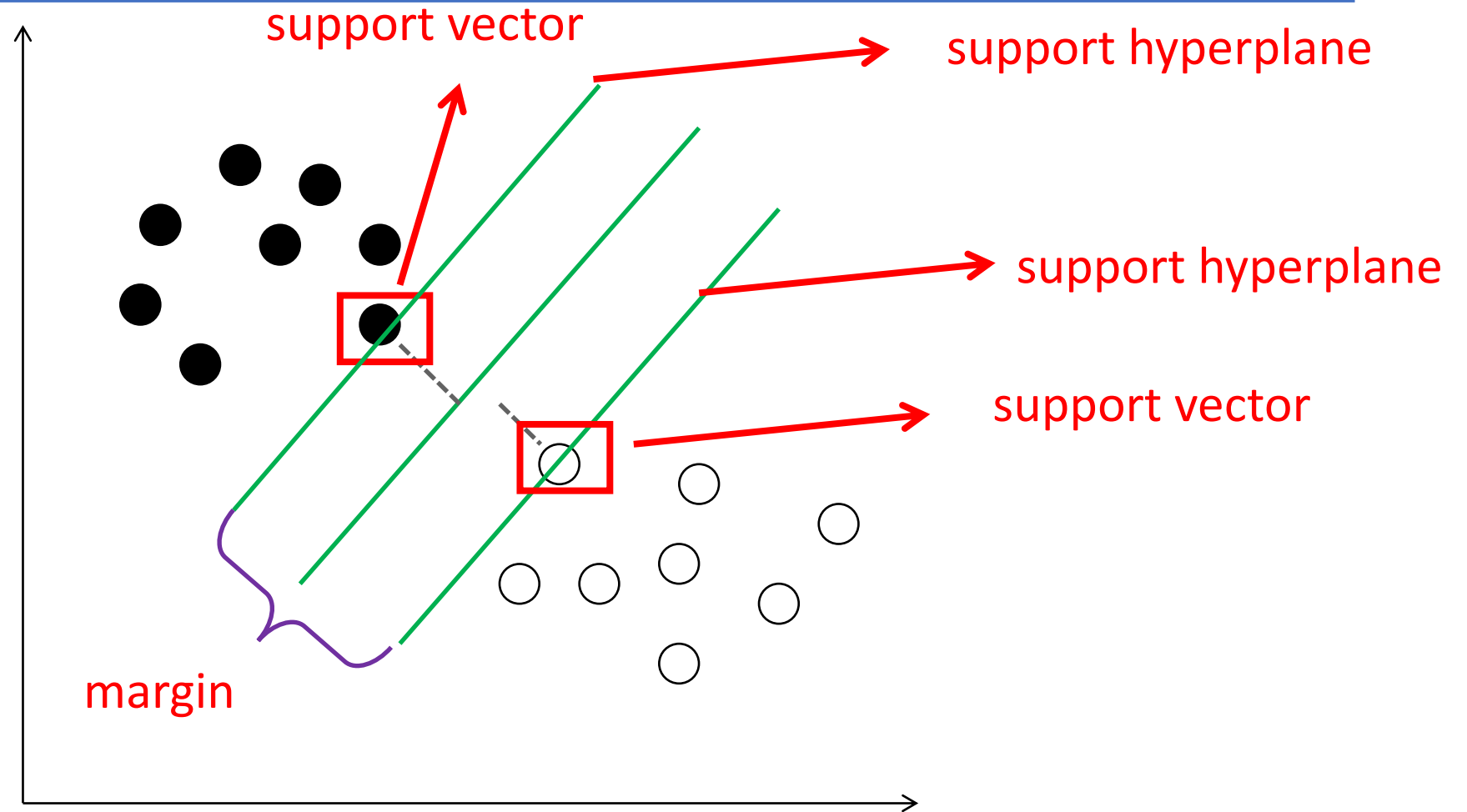
# What's Support Vector Machine

- Support Vector Machine (SVM) is a supervised machine learning algorithm.
- Linear SVM finds a hyperplane that separates data with maximum margin.



- $H_1$  does not separate the classes
- $H_2$  does, but only with a small margin
- $H_3$  separates them with the maximum margin

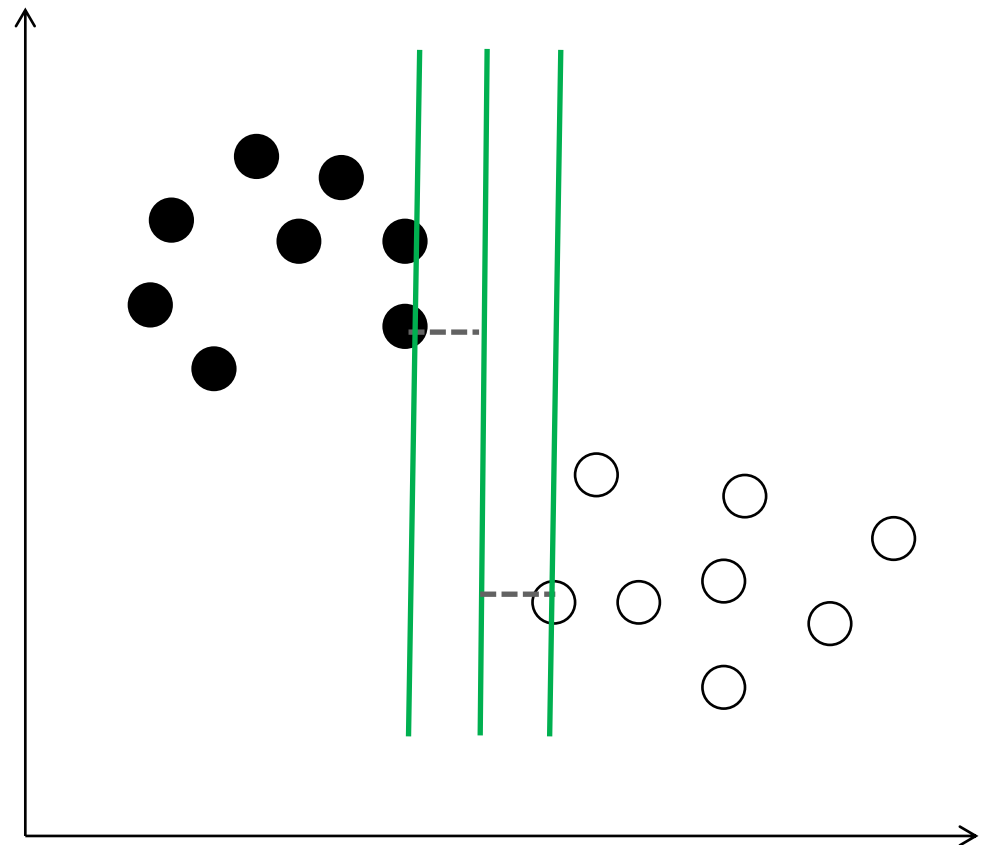
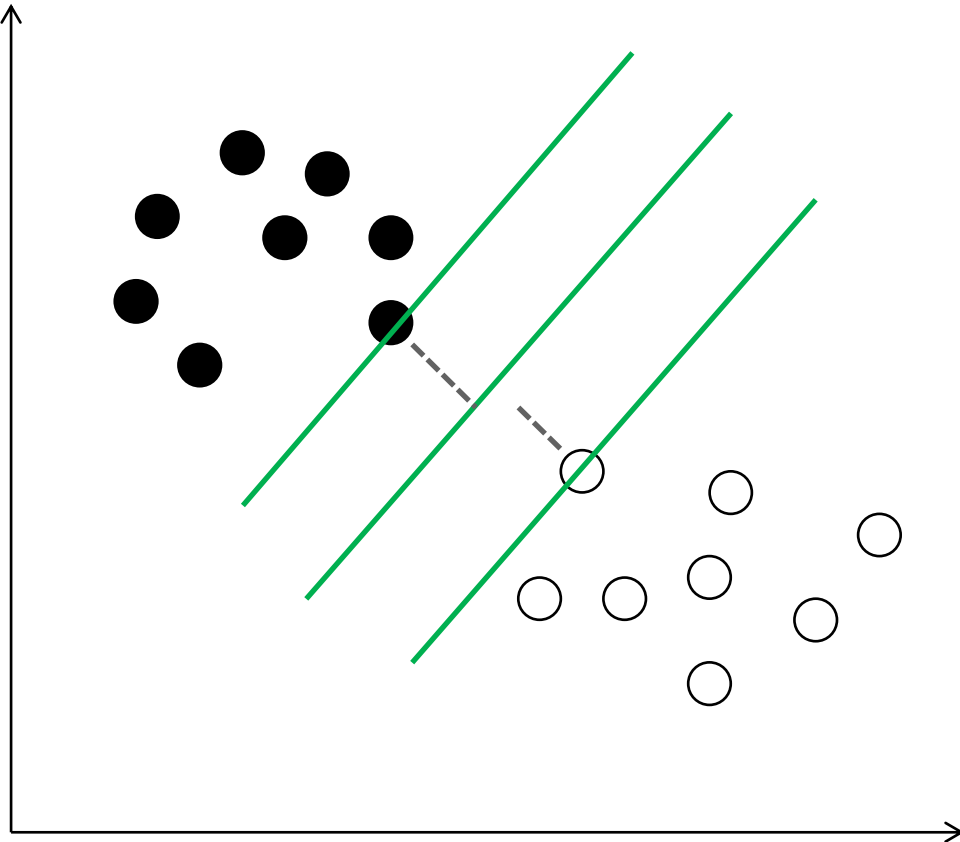
# What's Support Vector



**support vectors are points that affect hyperplane with maximum margin**

# The goal of SVM

- **Linear SVM** find a hyperplane that separate data with maximum margin



# How to calculate margin/distance?

---

- What's margin/distance between hyperplane  $2x - y + 2z = -5$  and point  $(2, 0, 0)$

change all stuffs on one side

$$2x - y + 2z + 5 = 0$$

calculate distance

$$\frac{|2 * 2 - 0 - 2 * 0 + 5|}{\sqrt{2^2 + (-1)^2 + 2^2}} = 3$$

# How to calculate margin/distance?

---

- Any Hyperplane in N-dimension can model

$$w^T x - b = 0$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$



# Example

---

- What's distance between the following 5-D hyperplane and point (2, 3, 4, 1, 1)

$$H: w^T x - b = 0 \text{ where } w = \begin{bmatrix} 1 \\ -1 \\ 2 \\ 3 \\ 1 \end{bmatrix} \text{ and } b = -5$$

# Example

---

change all stuffs on one side  $w^T x - b = 0$

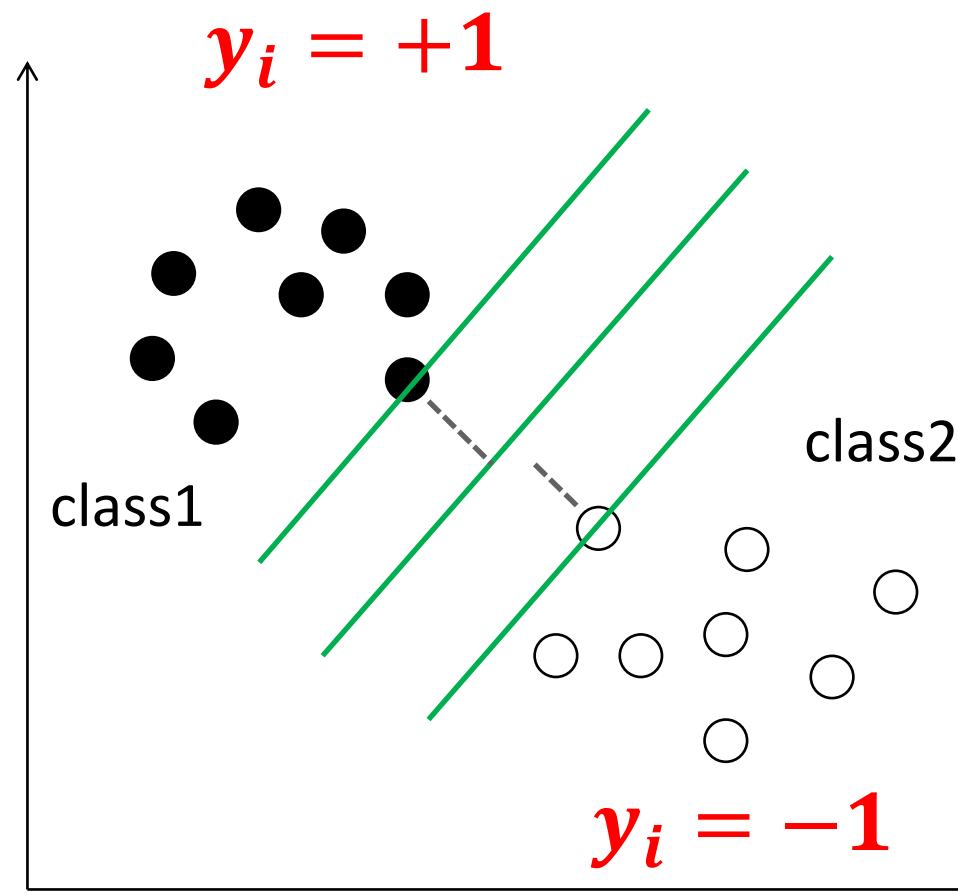
$$w = \begin{bmatrix} 1 \\ -1 \\ 2 \\ 3 \\ 1 \end{bmatrix} \text{ and } b = -5$$

calculate distance  $\frac{|1 * 2 + (-1) * 3 + 2 * 4 + 3 * 1 + 1 * 1 + 5|}{\sqrt{1^2 + (-1)^2 + 2^2 + 3^2 + 1^2}} = 4$

# SVM

$$\{x_i, y_i\}, i = 1, \dots, n$$
$$x_i \in R^d, y^i \in \{+1, -1\}$$

↑  
label



# SVM

$$f(x) = w^T x - b = 0$$

$$\text{if } f(x_i) = w^T x_i - b < 0$$



Class 1



$$y_i = -1$$

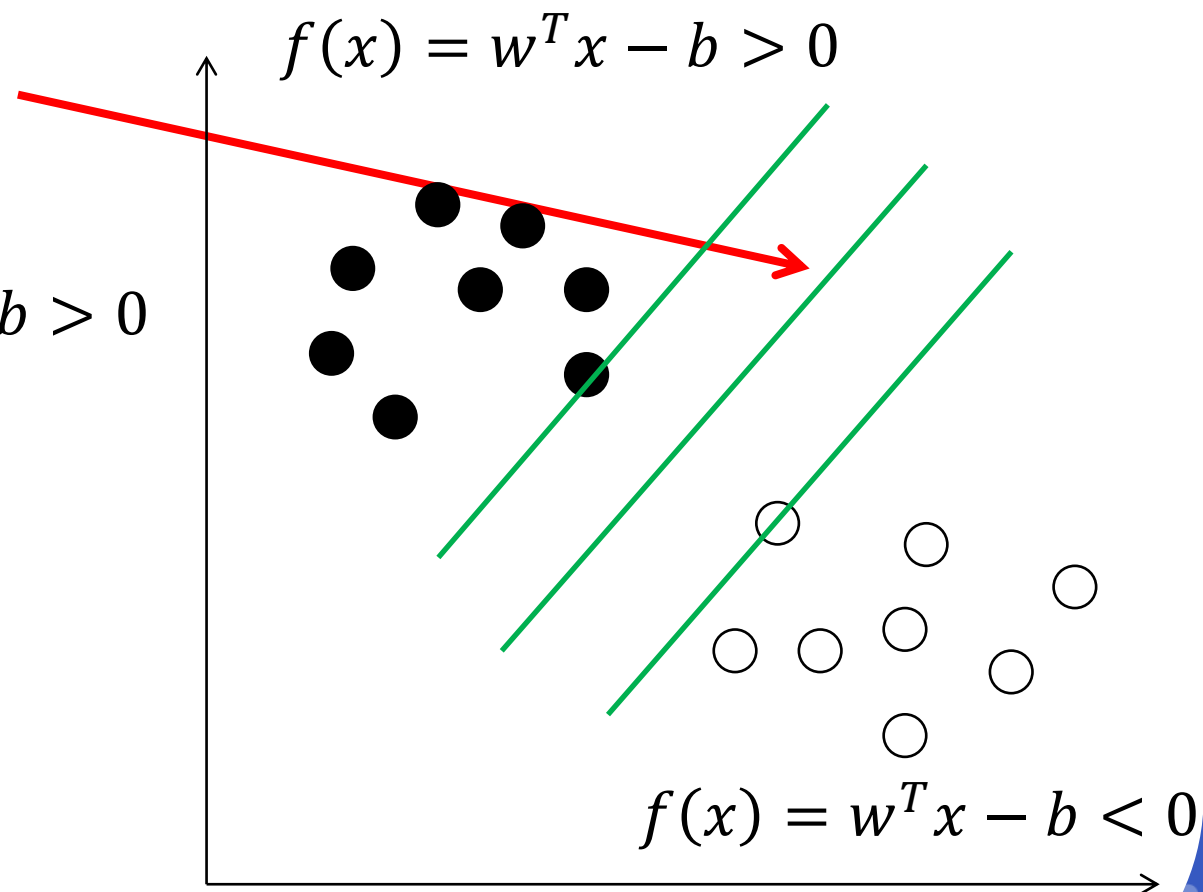
$$\text{if } f(x_i) = w^T x_i - b > 0$$



Class 2



$$y_i = +1$$



# SVM

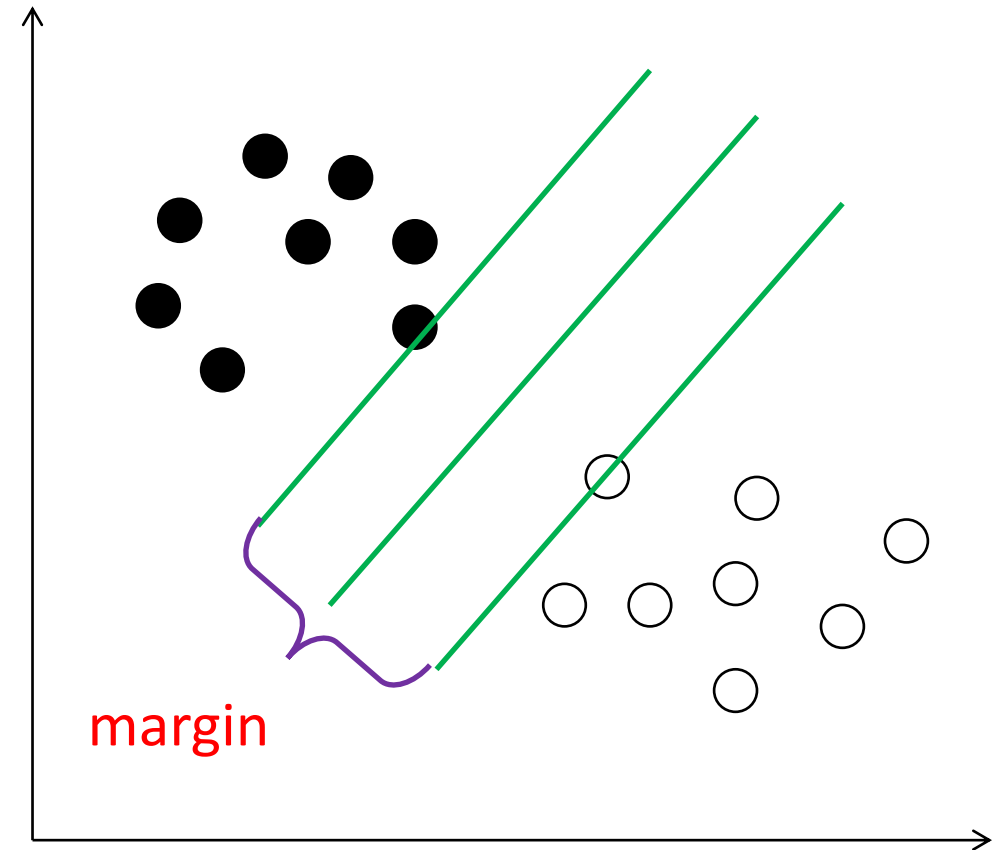
**Goal(what we want):**

$$\text{margin} = \frac{|w^T x_{black} - b|}{\|w\|} + \frac{|w^T x_{white} - b|}{\|w\|} = \frac{2}{\|w\|}$$

**Note :**

$$f(x_{black}) = w^T x_{black} - b = 1$$

$$f(x_{white}) = w^T x_{white} - b = -1$$



# SVM

## Constrain:

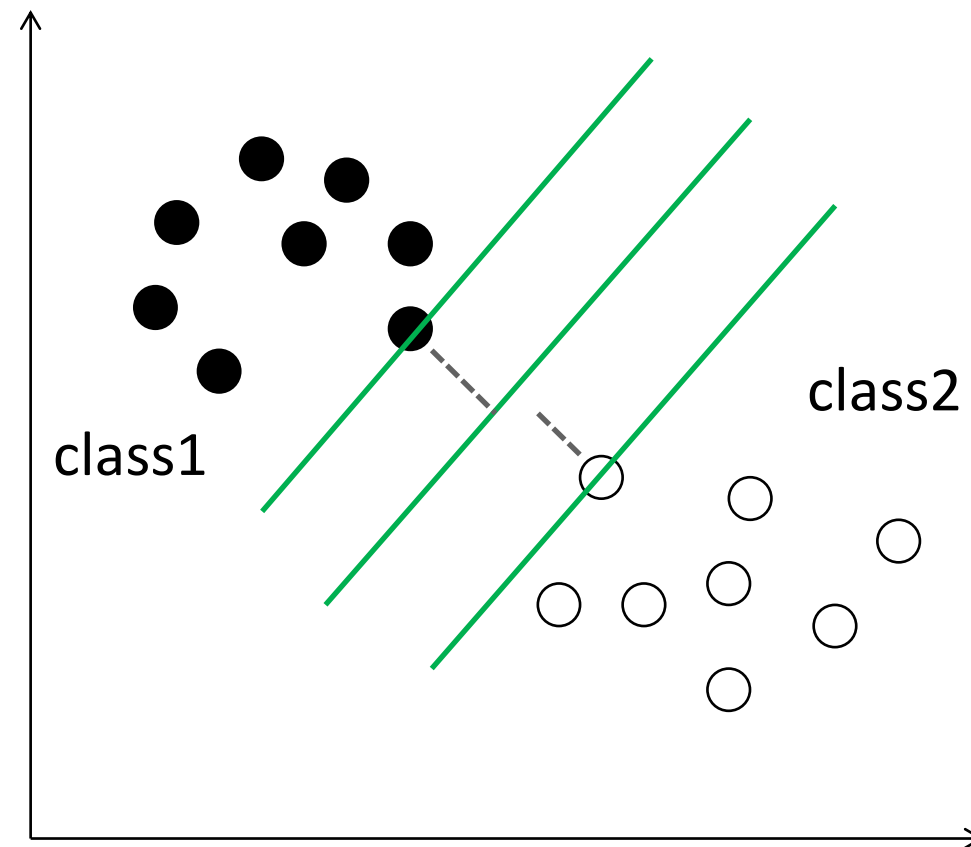
$$w^T x_i - b \leq -1 \quad \forall y_i = -1$$

$$w^T x_i - b \geq +1 \quad \forall y_i = +1$$



combine

$$y_i(w^T x_i - b) - 1 \geq 0$$



# SVM

$$\max \frac{2}{\|w\|}$$

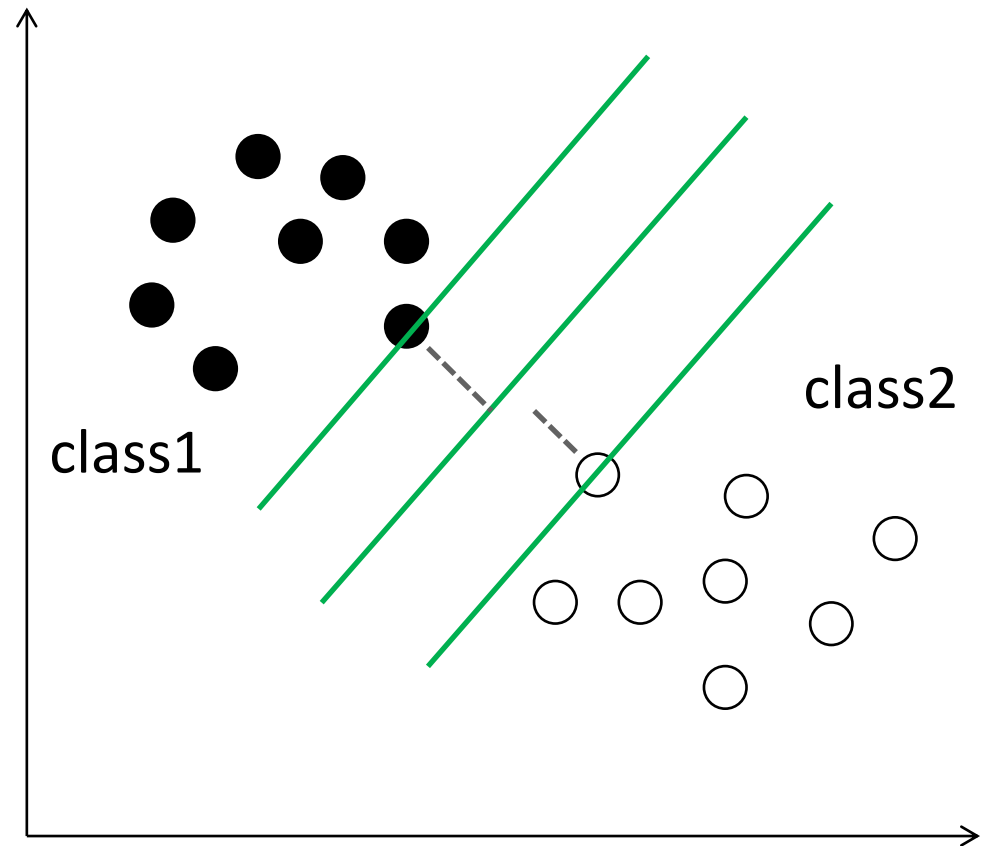
$$\text{subject to } y_i(w^T x_i - b) - 1 \geq 0 \quad \forall i$$



$$\min \frac{1}{2} \|w\|^2$$

$$\text{subject to } y_i(w^T x_i - b) \geq 1 \quad \forall i$$

What SVM solve in math



# Hard Cost V.S. Soft Cost

## Hard Cost

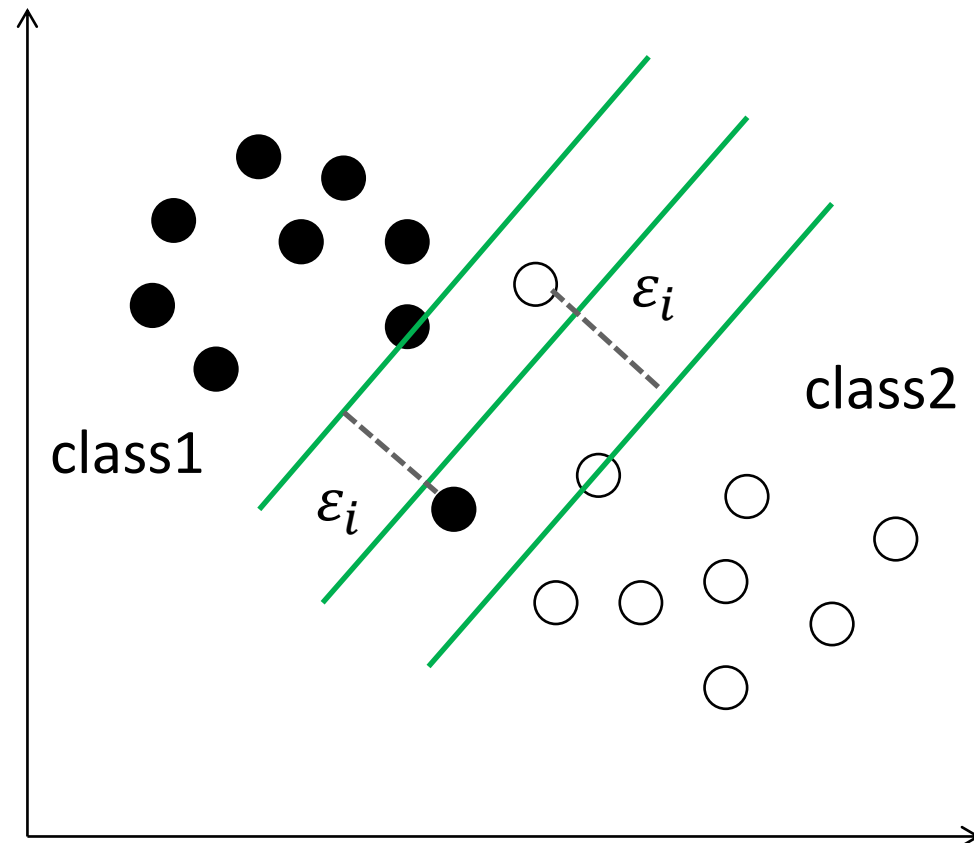
$$\min \frac{1}{2} \|w\|^2$$

$$\text{subject to } y_i(w^T x_i - b) \geq 1 \quad \forall i$$

## Soft Cost

$$\min \frac{1}{2} \|w\|^2 + C \sum \varepsilon_i$$

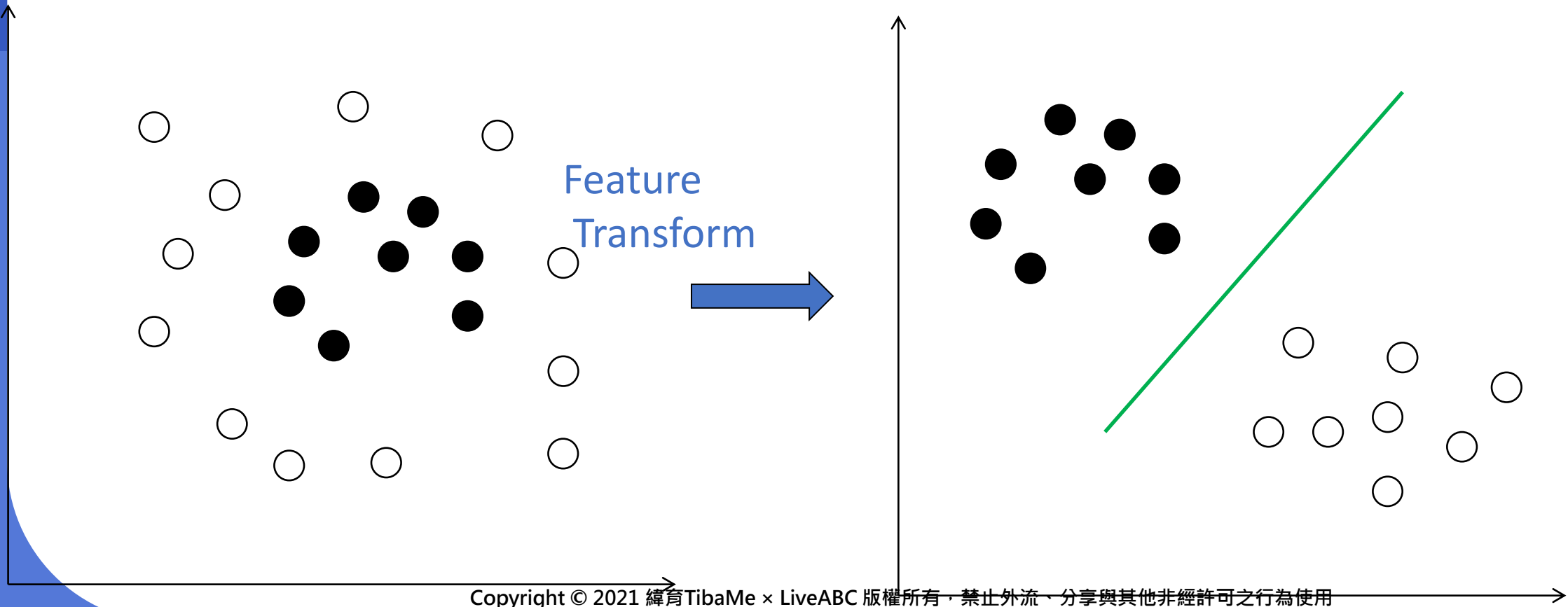
$$\begin{aligned} \text{subject to } & y_i(w^T x_i - b) \geq 1 - \varepsilon_i \\ & \varepsilon_i \geq 0 \quad \forall i \end{aligned}$$





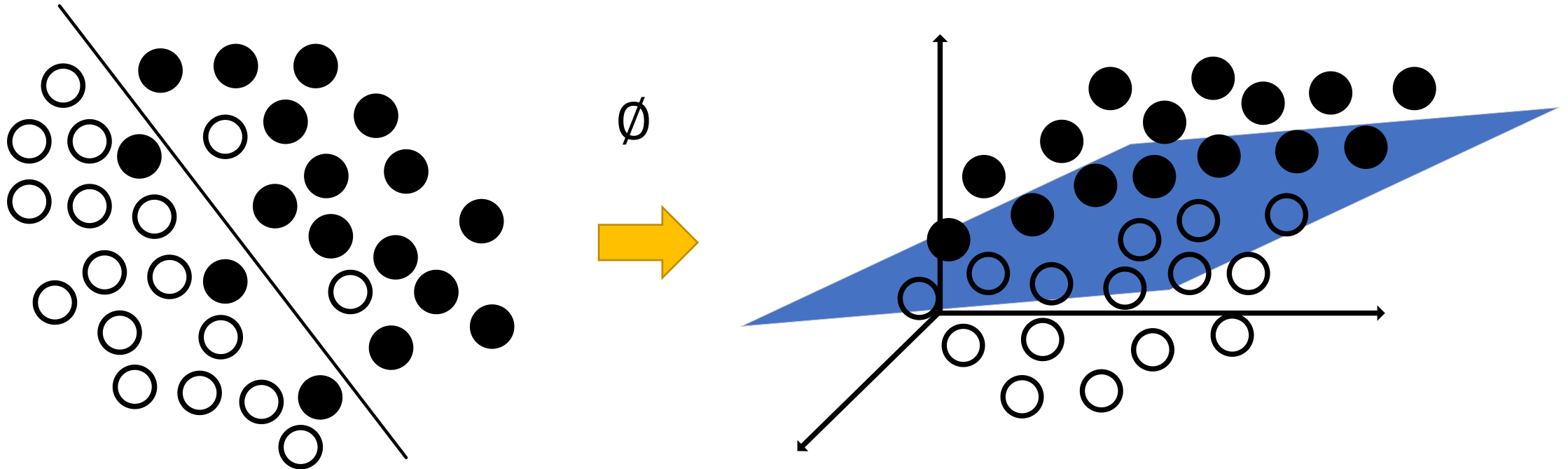
# Linear VS nonlinear problems

- with non-linearly separable data, usually we choose
  - hard-margin SVM + feature transformations
  - use soft-margin SVM directly (In practical, soft-margin SVM usually gets good results)
- However, soft-margin linear SVM can't deal with the left example well



# SVM Kernel Trick

- Usually, data can't be linearly separable
  - map data to higher dimension
  - <https://www.youtube.com/watch?v=3liCbRZPrZA>



# SVM Kernel Trick

---

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\boxed{\Phi(\mathbf{x})^\top \Phi(\mathbf{z})} = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix}$$



kernel

$$\begin{aligned} &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (\mathbf{x}^\top \mathbf{z})^2 \end{aligned}$$

# SVM Kernel Trick

$$\min \frac{1}{2} \|w\|^2$$

$$\text{subject to } y_i(w^T x_i - b) \geq 1 \quad \forall i$$

primal problem



Lagrange Multiplier

$$\max \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (x_i)^T x_j$$

$$\text{subject to } \alpha_i \geq 0 \quad \forall i$$


$$\sum_{i=1}^m \alpha_i y_i = 0$$

dual problem (Only  $\alpha$  is unknown)

Solving SVM: SMO (Sequential Minimal Optimization)

# SVM Kernel Trick

$$\max \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \boxed{(x_i)^T x_j}$$



$$\phi(x_i)^T \phi(x_j)$$

subject to  $\alpha_i \geq 0 \forall i$

$$\sum_{i=1}^m \alpha_i y_i = 0$$

dual problem

# Common Kernels in SVM

---

- linear  $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$
- polynomial  $k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$
- Gaussian or radial basis  $k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$
- sigmoid  $k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$



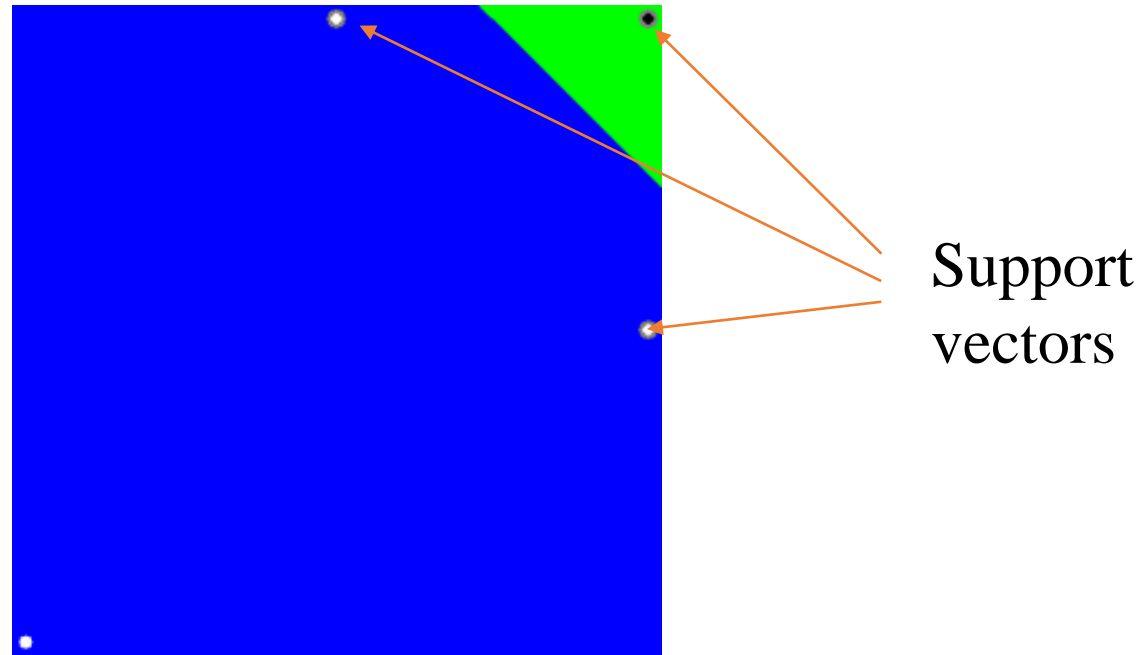
## 6.1 Support Vector Machine Implementation 1 (支持向量機實作一)

# 6.1 SVM implementation 1: case 1

---

- Please go to the following link for trying this python example:

<https://tinyurl.com/ulmrgnj>

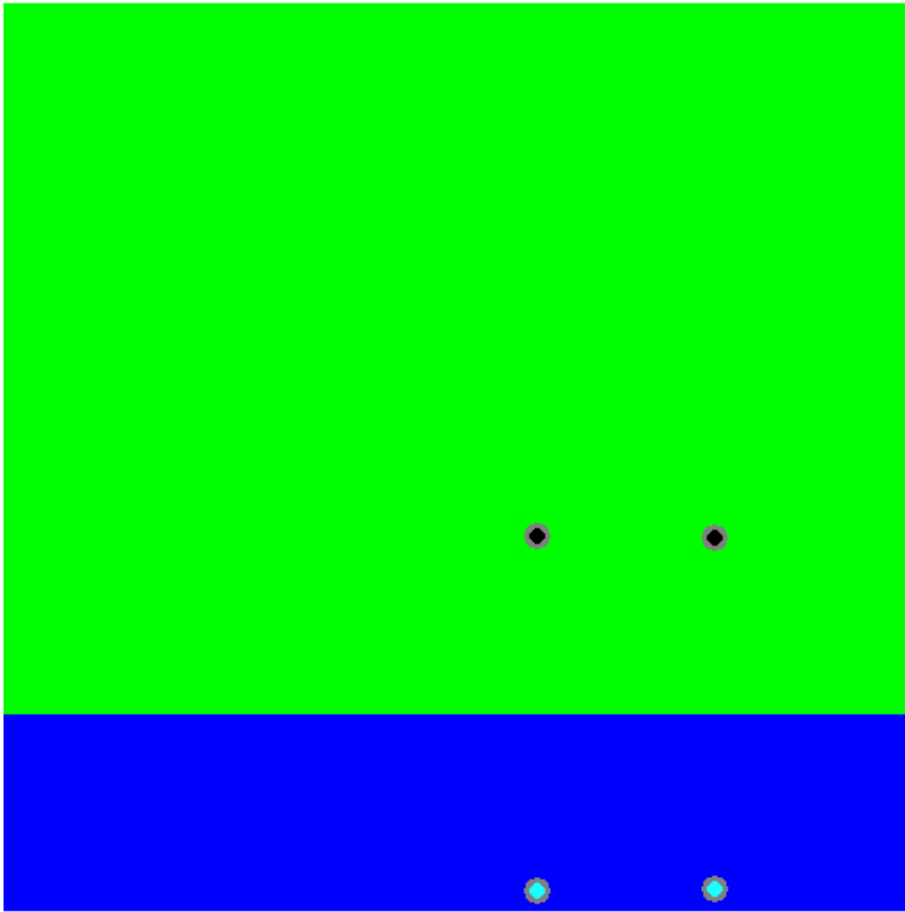




## SVM implementation 1: case 2

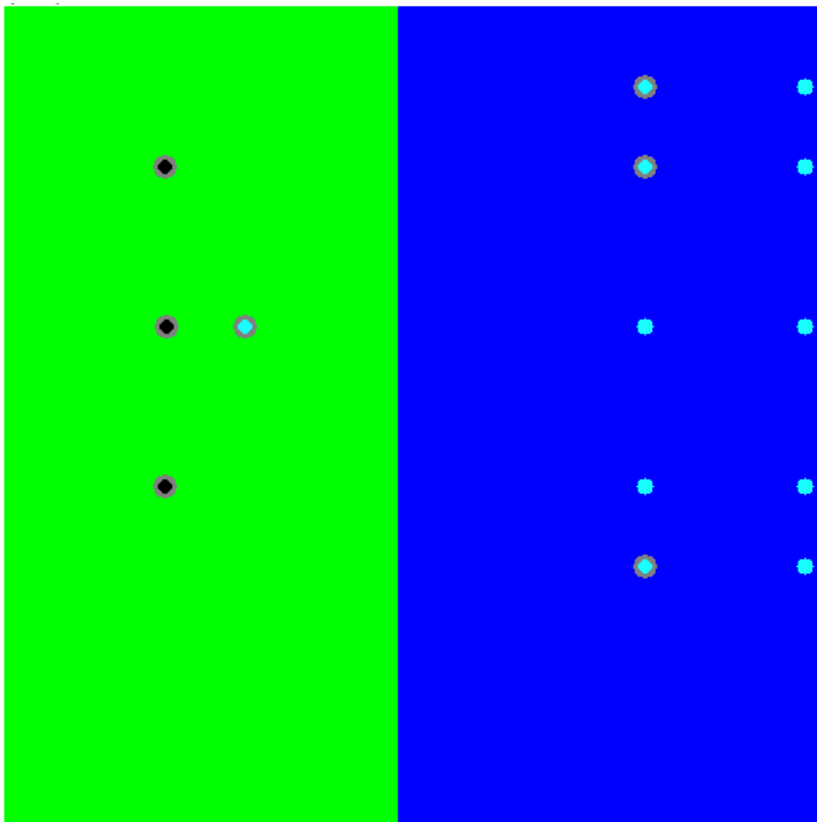
---

- This case would generate 4 support vectors.

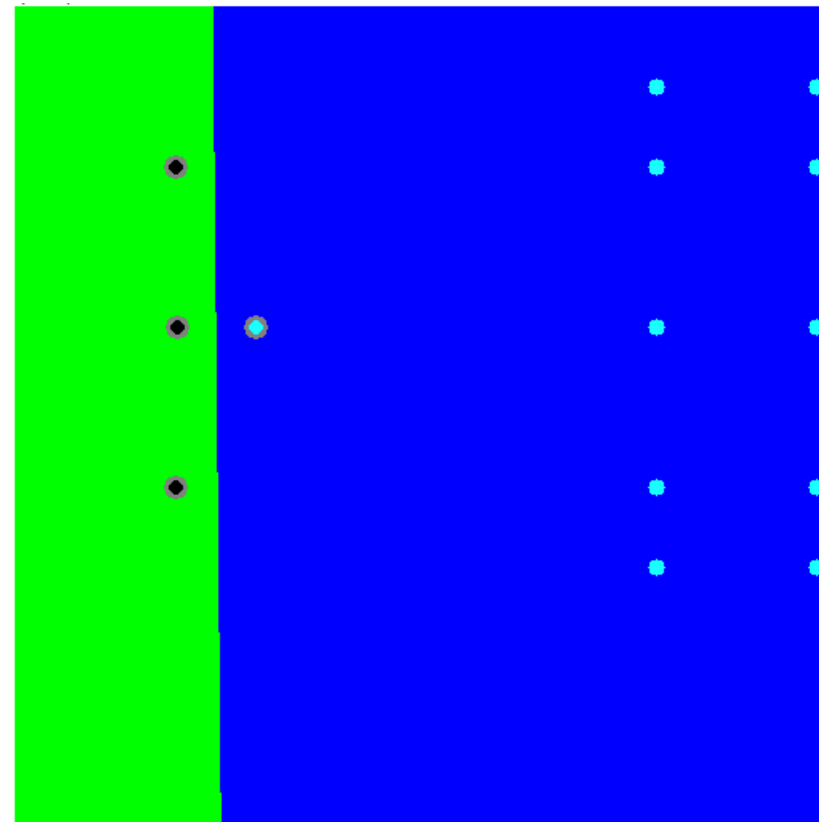


# SVM implementation 1: case 3

- Can you explain why the following results are different in terms of different  $C$  values in the soft cost case?



$C=0.00001$



$C=1$



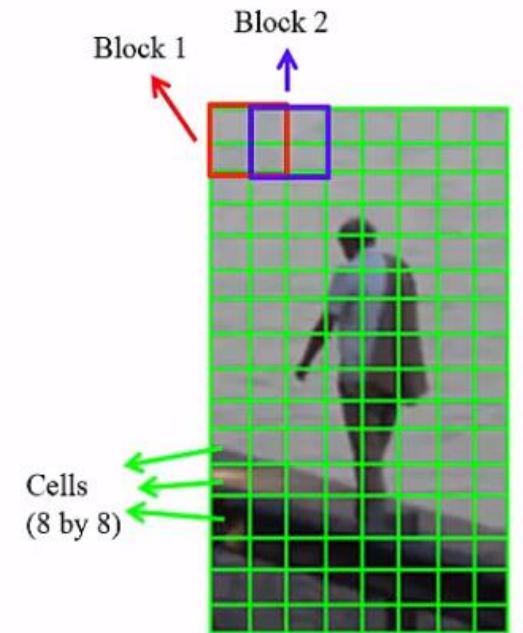
# HOG Feature and Multiclass Classification

(方向梯度直方圖與多類別分類)

# HOG steps

- HOG feature extraction

- Computer centered horizontal and vertical gradients with no smoothing
- Compute gradient orientation and magnitudes
  - For color image, pick the color channel with the highest gradient magnitude for each pixel
- For a 64x128 image,
  - divide the image into 16x16 blocks of 50% overlap
    - $7 \times 15 = 105$  blocks in total
- Each block should consist 2x2 cells with size 8x8
- Quantize the gradient orientation into 9 bins
  - The vote is the gradient magnitude
  - Interpolate votes between neighboring bin center
  - The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.
  - Concatenate histograms (Feature Dimension:  $105 \times 4 \times 9 = 3780$ )



# Computing Gradients

- Centered: 
$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2h}$$

- Filter masks in x and y directions

- Centered:

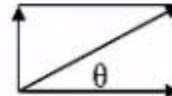
-1	0	1
----	---	---

-1
0
1

- Gradient

- Magnitude:

$$s = \sqrt{s_x^2 + s_y^2}$$

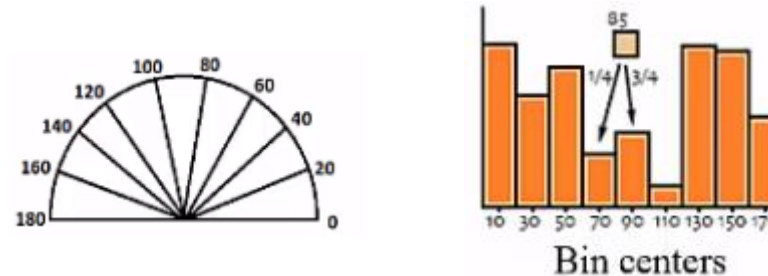


- Orientation:

$$\theta = \arctan\left(\frac{s_y}{s_x}\right)$$

# Votes

- Each block consists of 2x2 cells with size 8x8
- Quantize the gradient orientation into 9 bins (0-180)
  - The vote is the gradient magnitude
  - Interpolate votes linearly between neighboring bin centers.
    - Example: if  $\theta=85$  degrees.
      - Distance to the bin center Bin-70 and Bin90 are 15 and 5 degrees, respectively
      - Hence, ratios are  $5/20=1/4$ ,  $15/20=3/4$
  - The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block



# How to do multi-class classification using multiple binary classifiers?

- Multicategory case
  - One-against-the-rest
    - Use  $M$  two-class linear discriminants
      - Assign  $\mathbf{x}$  to  $\omega_i$  or not  $\omega_i$
    - Class imbalance problem
      - #negative  $\gg$  #positive
  - Pairwise separation:
    - Use  $M(M-1)/2$  linear discriminants
      - For every pair of classes
      - Decision is made by majority vote
    - Could lead to nonlinear separation of classes

Both approaches lead to undefined regions in the feature space

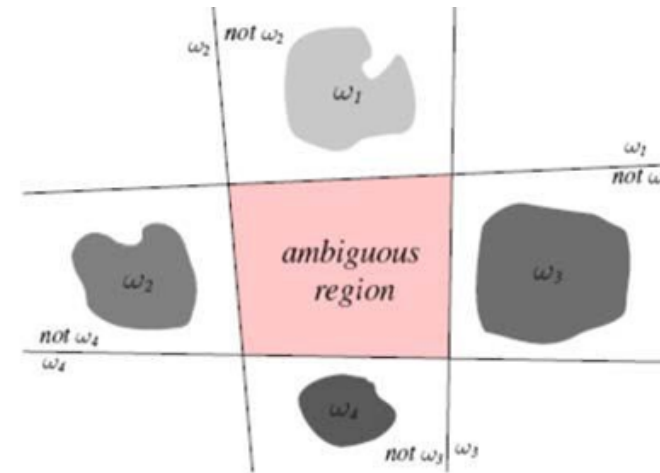
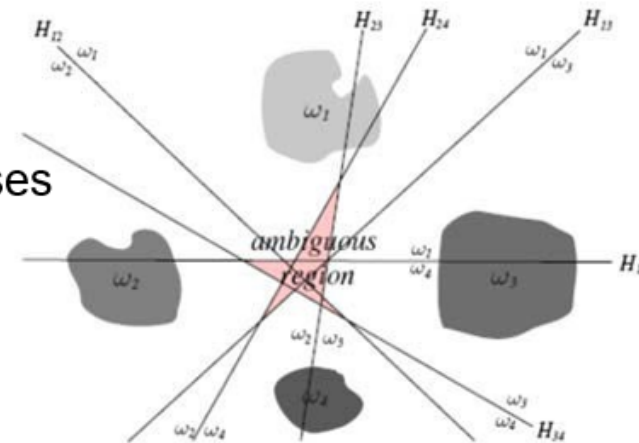


Fig. 5.3 [Duda 01]





## 6.2 Support Vector Machine Implementation 2: Heavy makeup and non-heavy make up face classification (支持向量機實作二:濃妝與非濃妝人臉分類)



# Image classification dataset:

## Sample Images



Ref: <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

Copyright © 2021 緯育TibaMe × LiveABC 版權所有，禁止外流、分享與其他非經許可之行為使用

Let's do two-class classification using HOG + SVM

---



Heavy makeup



Non-Heavy makeup

# Dataset size

---

- Training set: 200 imgs for heavy-makeup case; 200 imgs for non-heavy-makeup case;
- Validation set: 40 imgs for heavy-makeup case; 40 imgs for non-heavy-makeup case;

## 6.2 HOG+SVM face classification

---

- Please go to the following link for trying this python example:
- <https://tinyurl.com/tfeyj56w>
- How good is HOG+ (linear) SVM in terms of accuracy in this case?  
0.82
- Could deep learning approach achieve higher accuracy?
  - Yes, at least 0.85 using Alexnet



# Loophole of Image Classification



# Two-Class Classification is actually not that easy!

---



<https://medium.com/@tyreeostevenson/teaching-a-computer-to-classify-anime-8c77bc89b881>

Copyright © 2021 緯育TibaMe × LiveABC 版權所有，禁止外流、分享與其他非經許可之行為使用

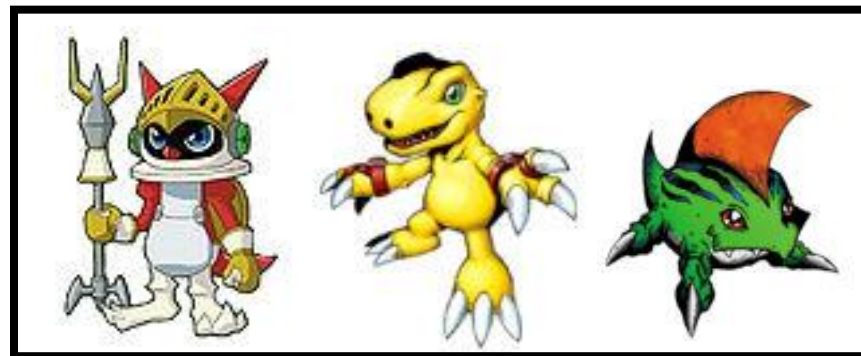
# Task

Pokémon images: <https://www.Kaggle.com/kvpratama/pokemon-images-dataset/data>

Digimon images: <https://github.com/DeathReaper0965/Digimon-Generator-GAN>



Pokémon



Digimon

Testing  
Images:



# Experimental Results

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same', input_shape=(120,120,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024))
model.add(Activation('relu'))
model.add(Dense(2))
model.add(Activation('softmax'))
```

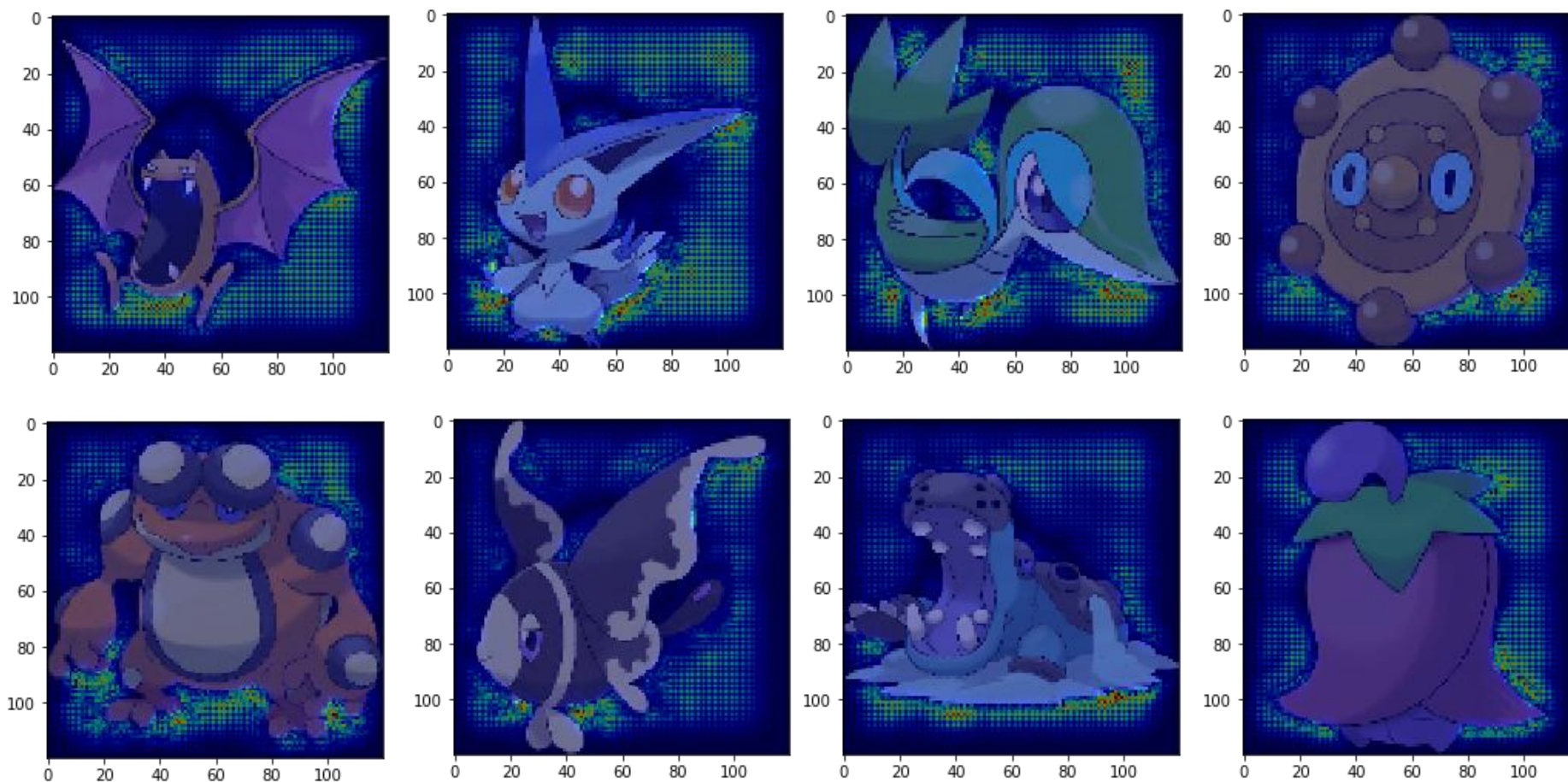
Training Accuracy: 98.9%

Testing Accuracy: 98.4%

太神啦!!!!!!



# Saliency Map



# What Happened?

- All the images of Pokémon are PNG, while most images of Digimon are JPEG.



PNG 檔透明背景



讀檔後背景是黑的!

Machine discriminates Pokémon and Digimon based on Background color.



## 6.3 Support Vector Machine Implementation 3: Face Detection (支持向量機實作三:人臉偵測)

# From Image classification to object detection



•  
•



Sliding window



HOG feature  
extraction



SVM  
classification

## 6.3 Implementation: face detection using HOG+SVM

---

- Training dataset
  - Positive: Labeled faces in the wild (13233 images)
  - Negative: random cropped images from 'camera', 'text', 'coins', 'moon', 'page', 'clock', 'immunohistochemistry', 'chelsea', 'coffee', 'hubble\_deep\_field' in “**The USC-SIPI Image Database**”
- Testing image:
  - Test image: astronaut Eileen Collins in **The USC-SIPI Image Database**”
- Example:
  - <https://tinyurl.com/y3odsdro>





Positive training images





Negative training images

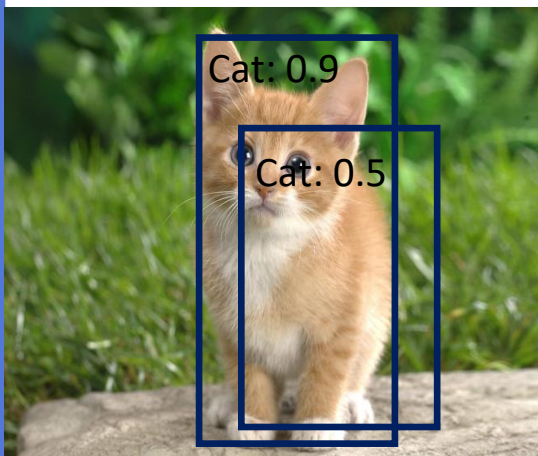


# Non-Maximal Suppression

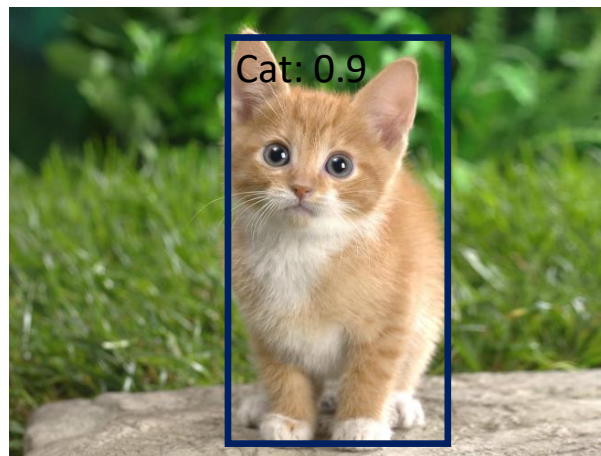
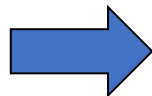
- Boxes with lower confidence and  $IOU < 0.5$  with the one with higher confidence should be eliminated.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


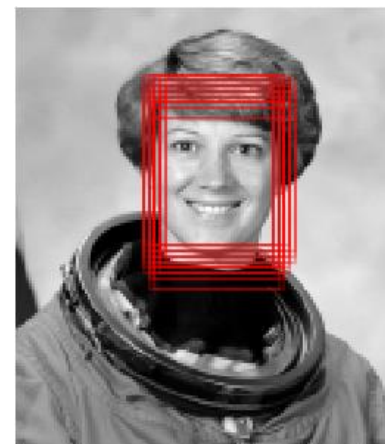
<http://blog.csdn.net/lanxunhui>



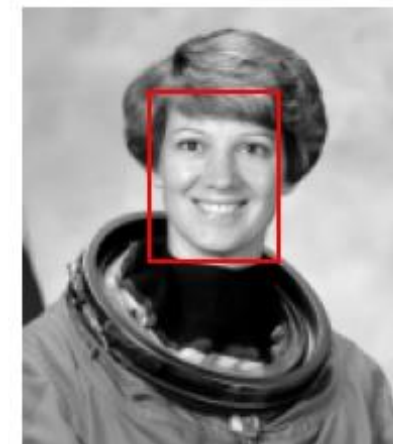
Before NMS



After NMS



Before NMS



After NMS

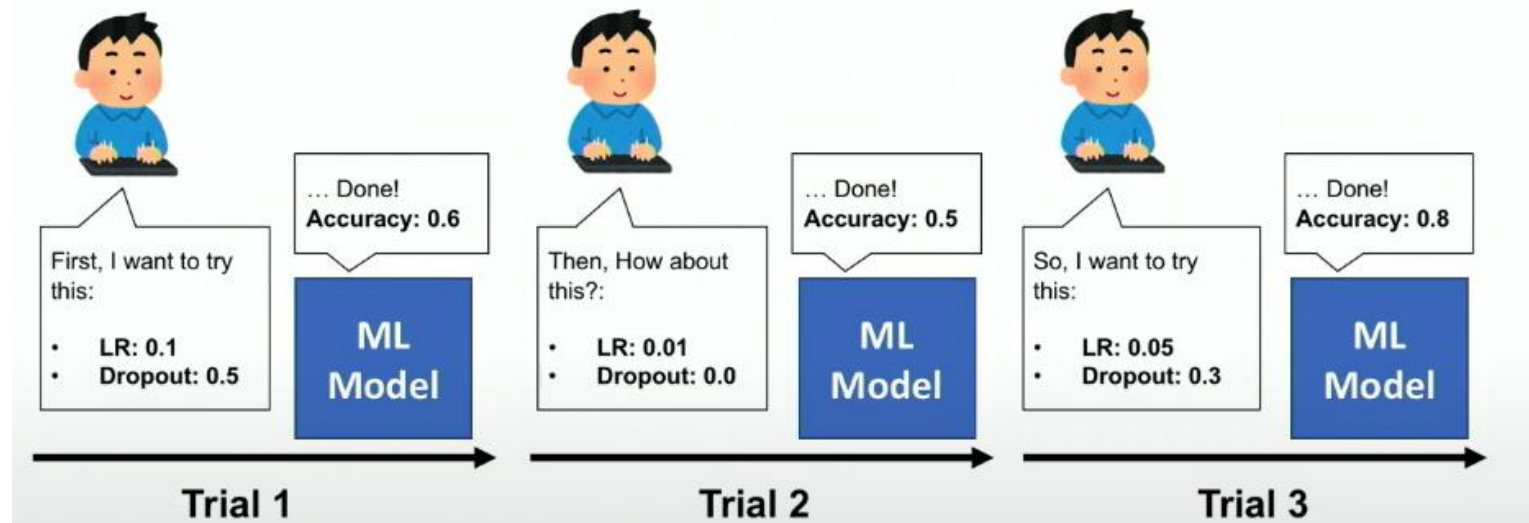


# Hyperparameter Optimization (超參數最佳化)



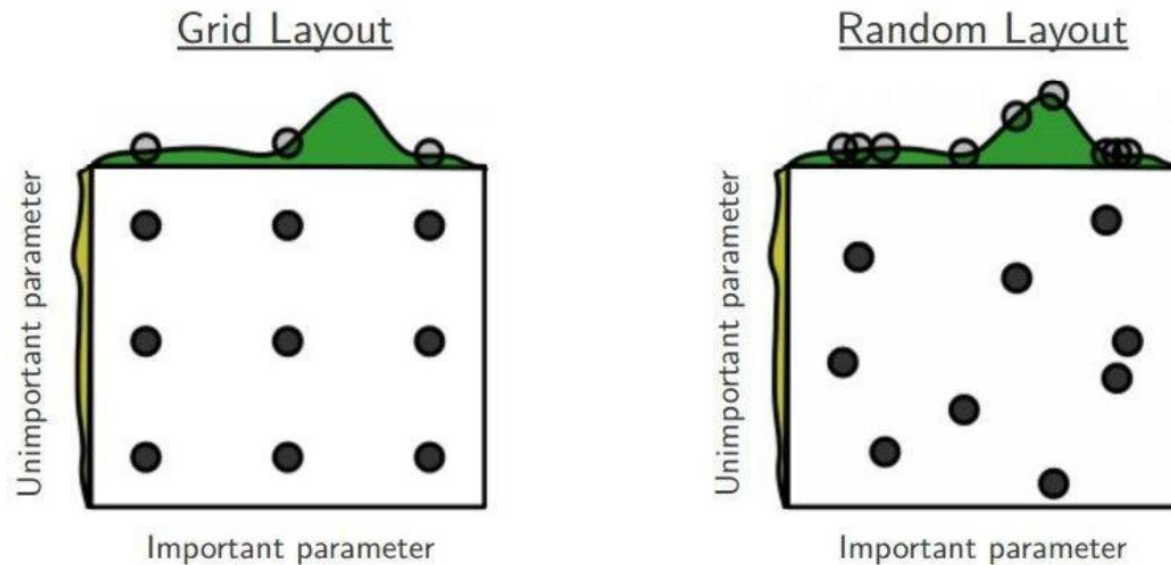
# Hyperparameters matter!

- Many computer vision algorithms depend on configuration settings that are typically hand-tuned in the course of evaluating the algorithm for a particular data set.
- While such parameter tuning is often presented as being incidental to the algorithm, correctly setting these parameter choices is frequently critical to realizing a method's full potential.
- Hyperparameter often must be re-tuned when the algorithm is applied to a new problem domain, and the tuning process itself often depends on personal experience and intuition in ways that are hard to quantify or describe.
- Since the performance of a given technique depends on both the fundamental quality of the algorithm and the details of its tuning, it is sometimes difficult to know whether a given technique is genuinely better, or simply better tuned.



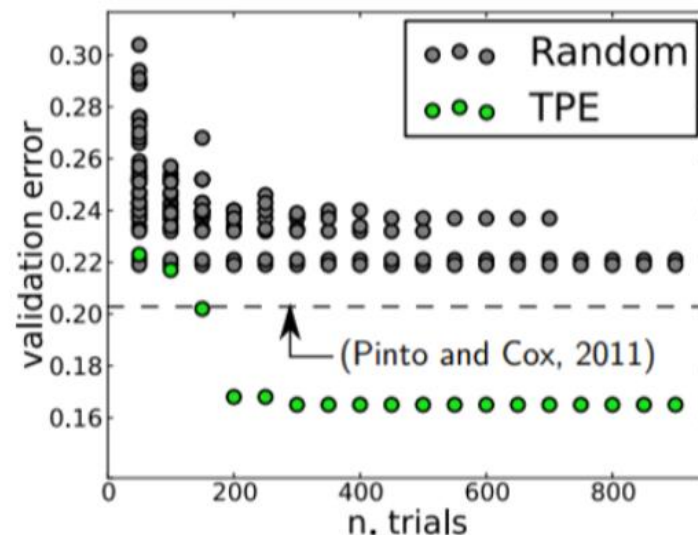
# Grid Search vs Random Search

- Heuristic hand-tuning is more likely to be grid search and it was proven less effective than random search.
- An automatic random search usually could lead us to better results because
  - it is automatic, efficient, and effective.
  - it provides you better time management (waking up at midnight to hit enter for the next sets of hyperparameters is not required).
- However, grid search and random search are completely uninformed by past evaluations, and as a result, often spend a significant amount of time evaluating “bad” hyperparameters.



# Random Search vs TPE

- Bayesian Optimization (using the Tree Parzen Estimator or TPE) builds a probability model of the objective function and uses it to select the most promising hyperparameters to evaluate in the true objective function.
- In other words, when a region of the space turns out to be good, it should be explored more. Such techniques take care of the “zooming” process for you and lead to much better solutions in much less time.
- In the following experiment (SVM on LFW dataset), compared to random search, TPE gradually achieves lower test set performance with fewer trial and thus could lead to a smaller number of trials.



Bergstra et al. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, ICML, 2013.

# Hyperparameter libraries

- If you don't have a lot of resources, use Scikit-Optimize.
- If you want to get SOTA and don't care about API/Docs, use HpBandSter.
- If you want good docs/api/parallelization, use Optuna.

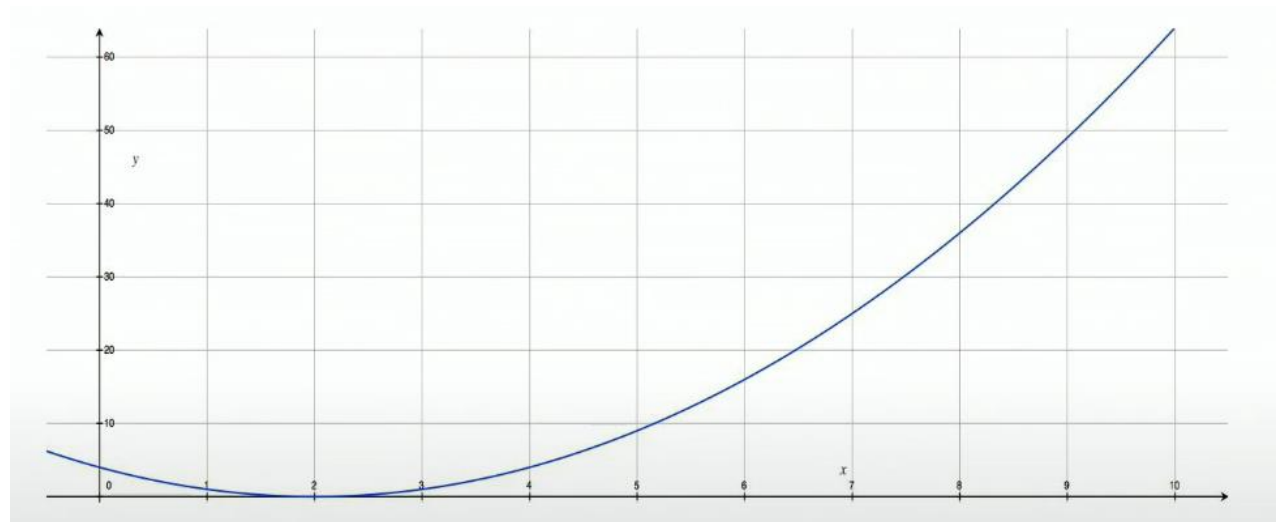
	Scikit-Optimize	Optuna	HpBandSter	Hyperopt
API/ease of use	Great	Great	Difficult	Good
Documentation	Great	Great	OK	Bad
Speed/Parallelization	Fast if sequential/None	Great	Good	OK
Visualizations	Amazing	None	Very lib-specific	Some
Experimental results	Good	Good	Great	OK



## 6.4 Optuna Basic Example (Optuna 基本實作範例)

## 6.4 implementation: minimize $(x-2)^2$

- Basic steps to set up Optuna
  - Define the objective function that calculates the minimization/maximization target.
  - Inside the objective function, set the hyperparameters to be optimized with suggest methods.
  - Instantiate the study object.
  - Start the optimization with study.optimize, specifying the number of trials with n\_trials.
- <https://tinyurl.com/5ehdw3ep>





## 6.5 Face Classifier Optimization Example Using Optuna (Optuna最佳化人臉分類器實作)

## 6.5 implementation: Advanced example: HOG+SVM face classification:

---

- Please go to the following link for trying this python example:
- <https://tinyurl.com/np5w3bwf>
- Baseline: linear SVM: 0.82
- Hyperparameters to be optimized:
  - Polynomial-kernel SVM: 0.855 by d (4.52) and C (62.54)



# Conclusion & Observation

---

- Before CNNs started to dominate, Support Vector Machines (SVMs) were the state-of-the-art.
- SVM using soft cost is quite common in all kinds of applications.
- Application of SVM on images would usually involve feature extraction for reducing
  - (1) the computation resources.
  - (2) amount of redundant information.
- Don't manually tune hyperparameters; try automatic optimization tool in every project related to machine learning and deep learning models.