SUPPORT VECTOR MACHINE

- Nguyễn Hoàng Yến Như
- Nguyễn Trần Phúc Nghi
- Nguyễn Trần Phúc An
- Nguyễn Đức Anh Phúc
- Trịnh Thị Thanh Trúc
- KS. Cao Bá Kiệt
- KS. Quan Chí Khánh An
- KS. Lê Ngọc Huy
- CN. Bùi Cao Doanh
- CN. Nguyễn Trọng Thuận
- KS. Phan Vĩnh Long
- KS. Nguyễn Cường Phát
- ThS. Nguyễn Hoàng Ngân
- KS. Hồ Thái Ngọc
- ThS. Đỗ Văn Tiến
- ThS. Nguyễn Hoàn Mỹ
- ThS. Dương Phi Long
- ThS. Trương Quốc Dũng
- ThS. Nguyễn Thành Hiệp
- ThS. Nguyễn Võ Đăng Khoa
- ThS. Võ Duy Nguyên
- TS. Nguyễn Văn Tâm
- ThS. Trần Việt Thu Phương
- TS. Nguyễn Tấn Trần Minh Khang

Lịch sử SVM

- SVM được đề xuất bởi Vladimir N. Vapnik và các đồng nghiệp của ông vào năm 1963 tại Nga và sau đó trở nên phổ biến trong những năm 90 nhờ ứng dụng giải quyết các bài toán phi tuyến tính (nonlinear) bằng phương pháp Kernel Trick.

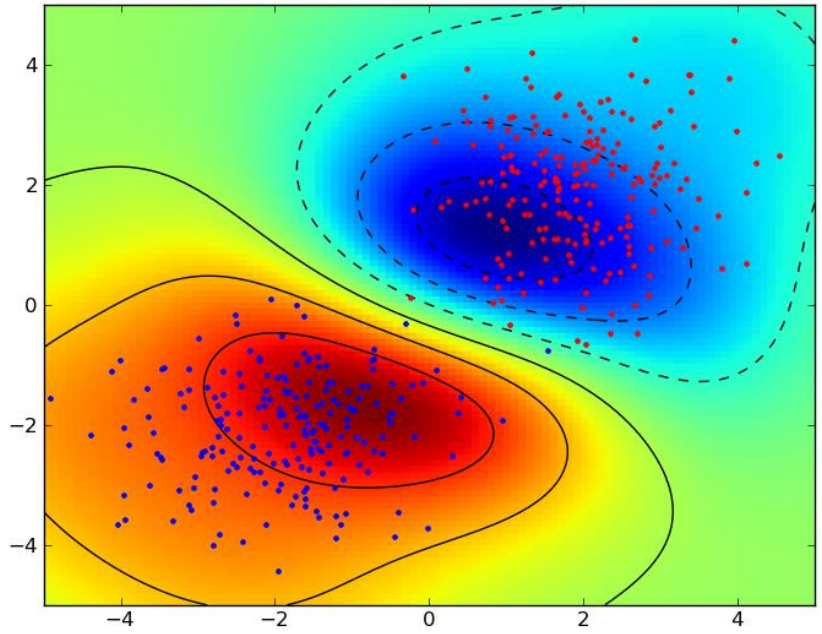
Lịch sử SVM

- SVM is related to statistical learning theory [3].
- SVM was first introduced in 1992 [1].
- SVM becomes popular because of its success in handwritten digit recognition [3].

- [1] B.E. Boser et al. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
- [2] L. Bottou et al. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.
- [3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

Lịch sử SVM

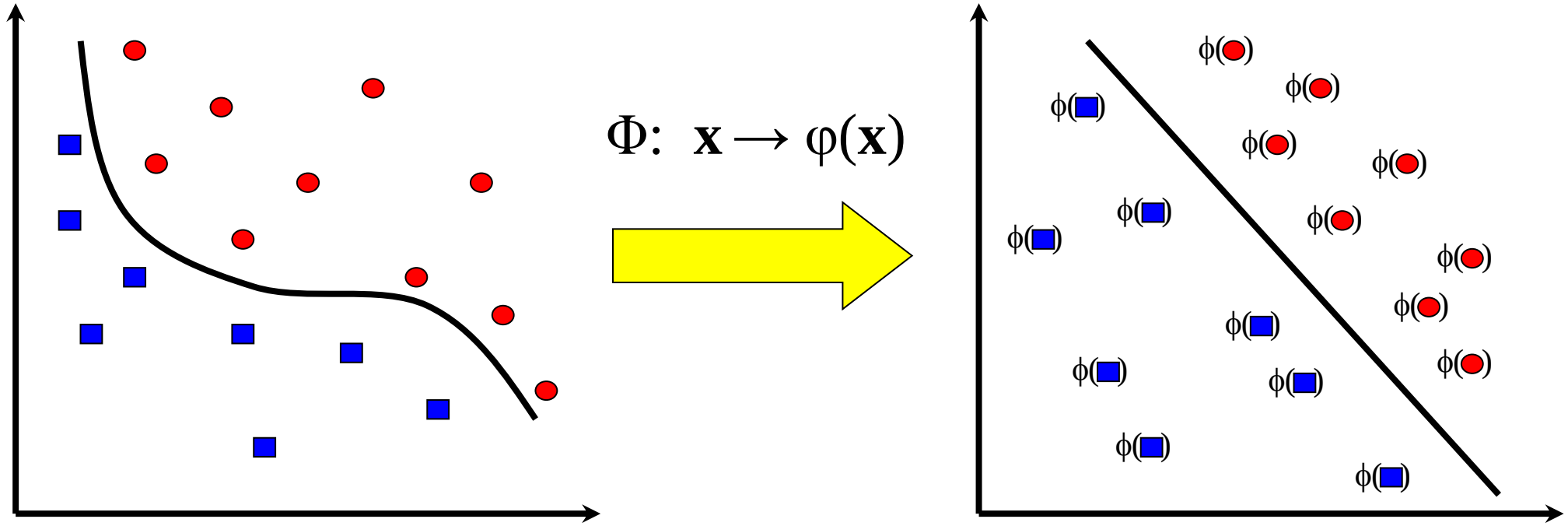
- Empirically good performance: successful applications in many fields (bioinformatics, text, image recognition, . . .).
- A large and diverse community work on them: from machine learning, optimization, statistics, neural networks, functional analysis, etc.



General idea

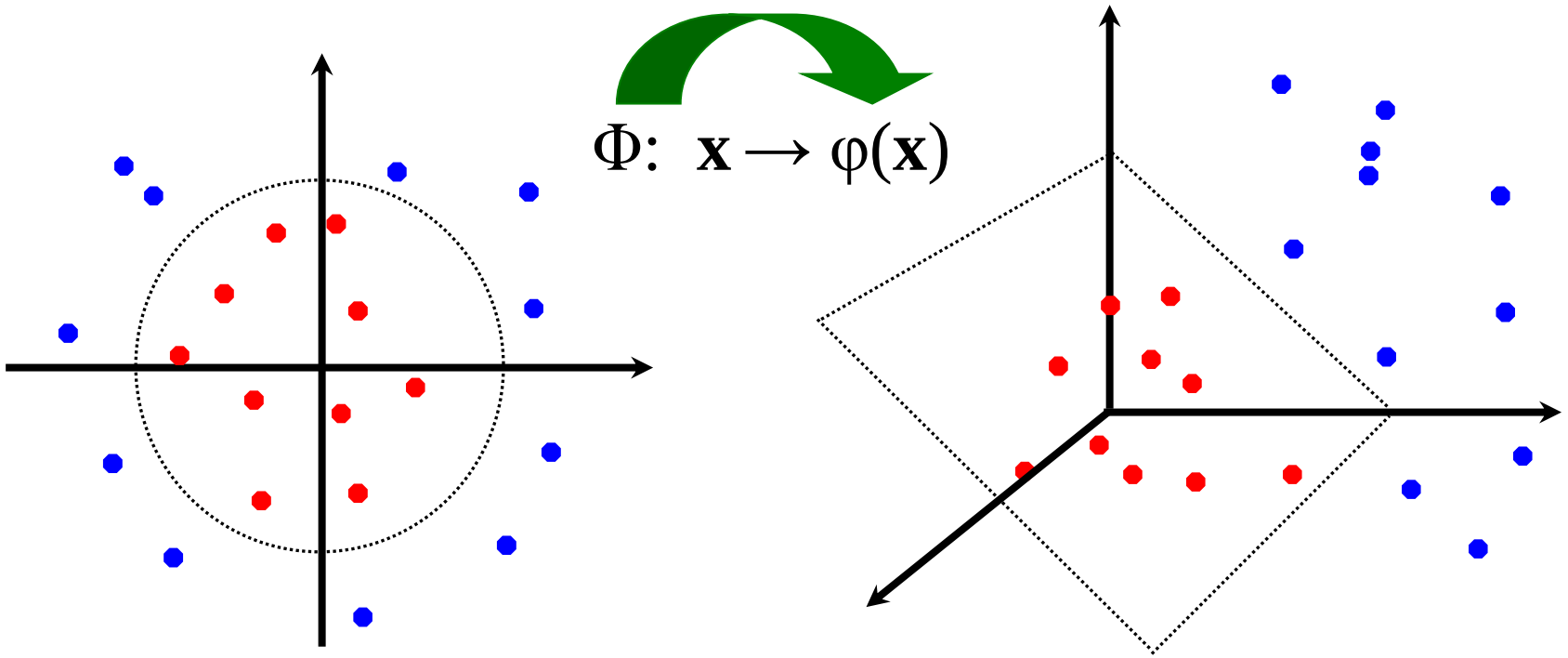
GENERAL IDEA

General idea

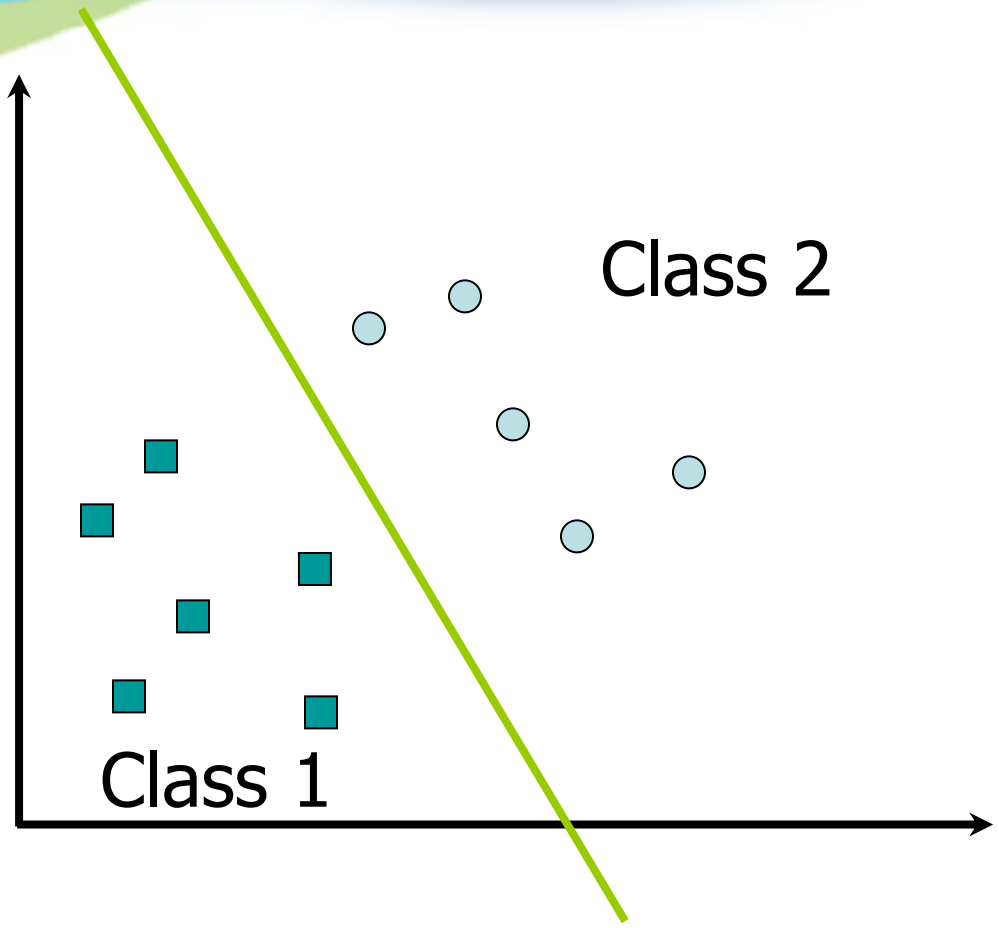


General idea

– The original input space can always be mapped to some higher-dimensional feature space where the training set is separable:

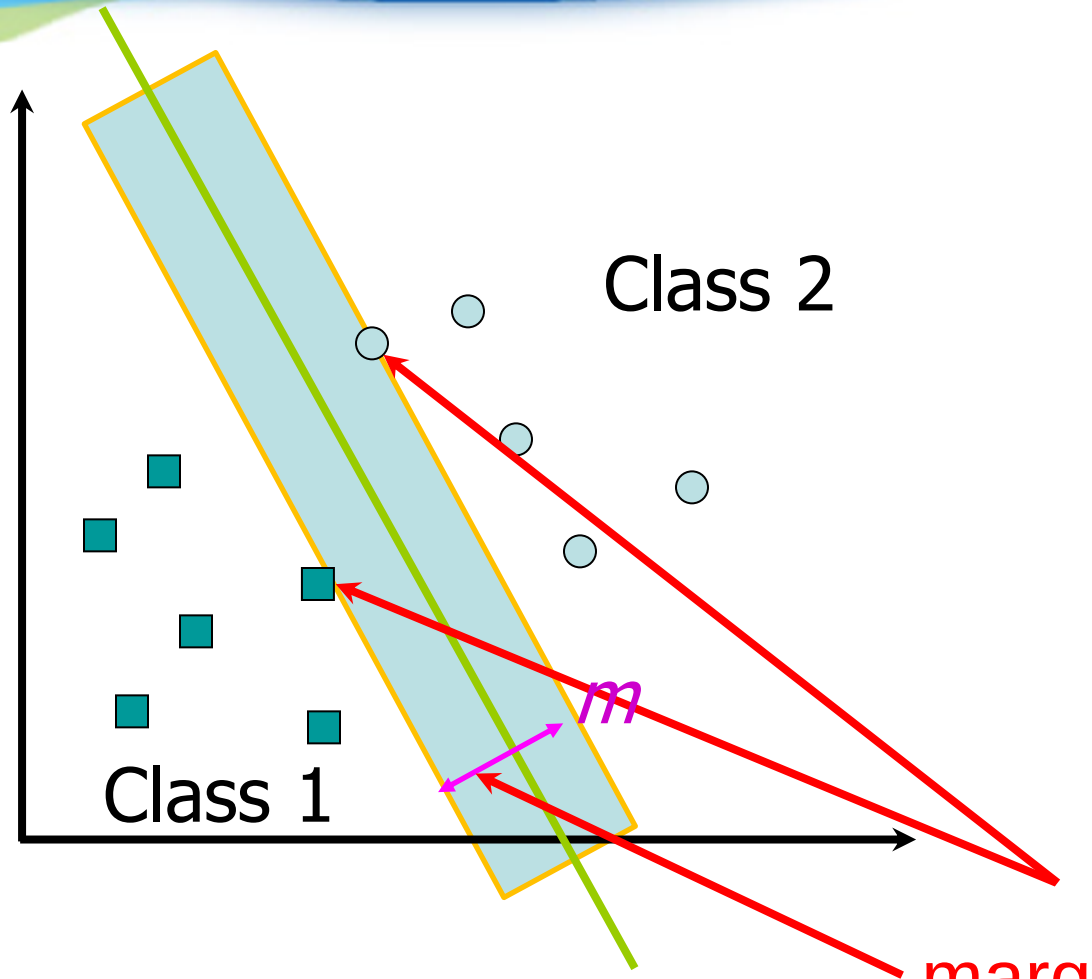


General idea



- Dùng để làm gì? Support vector machine (svm) xây dựng (learn) một siêu phẳng (hyperplane) để phân lớp (classify) tập dữ liệu thành 2 lớp riêng biệt.
- Siêu phẳng hiểu một cách đơn giản nhất là một đường thẳng.

General idea

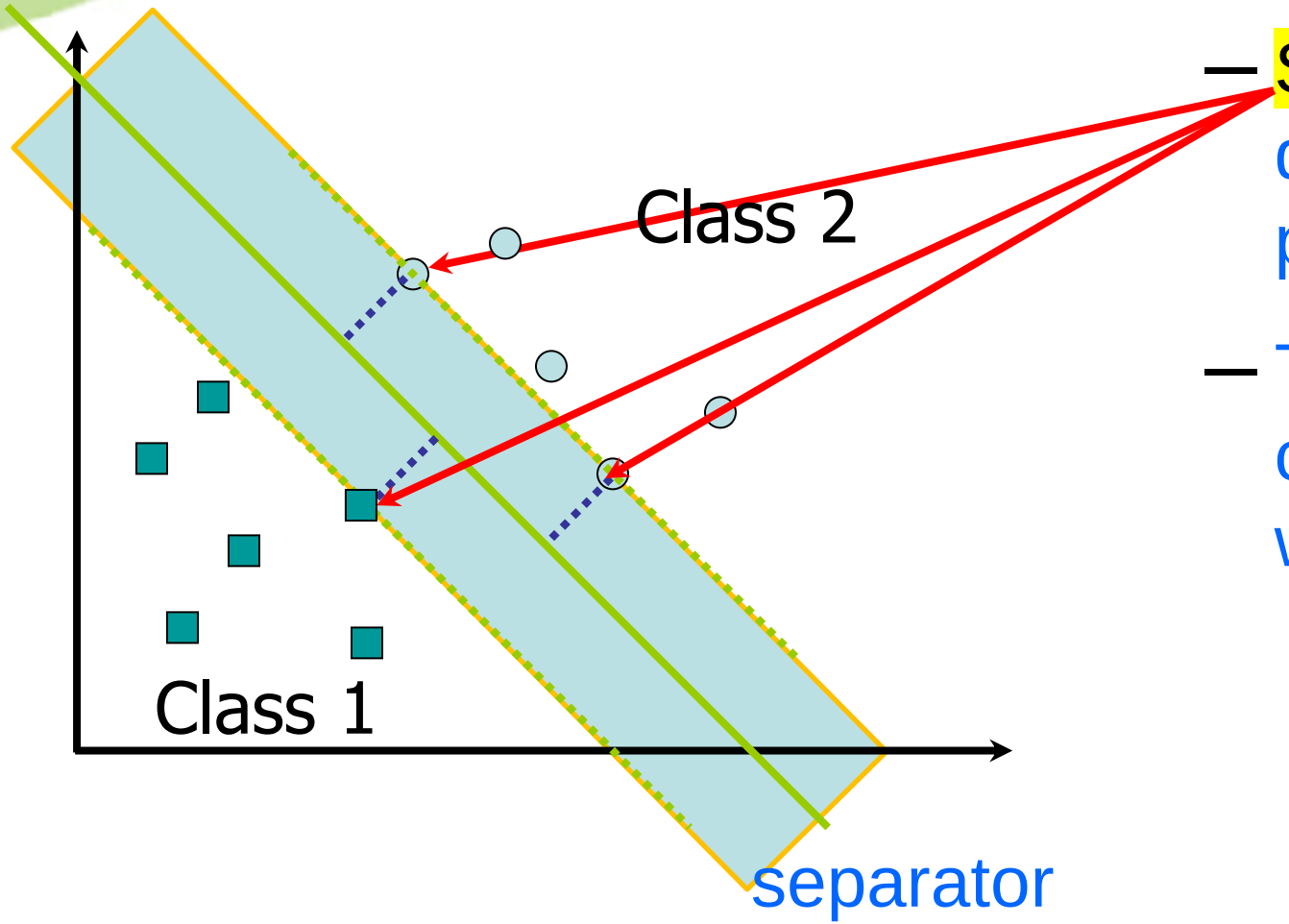


- Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.
- Margin là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với các phân lớp.

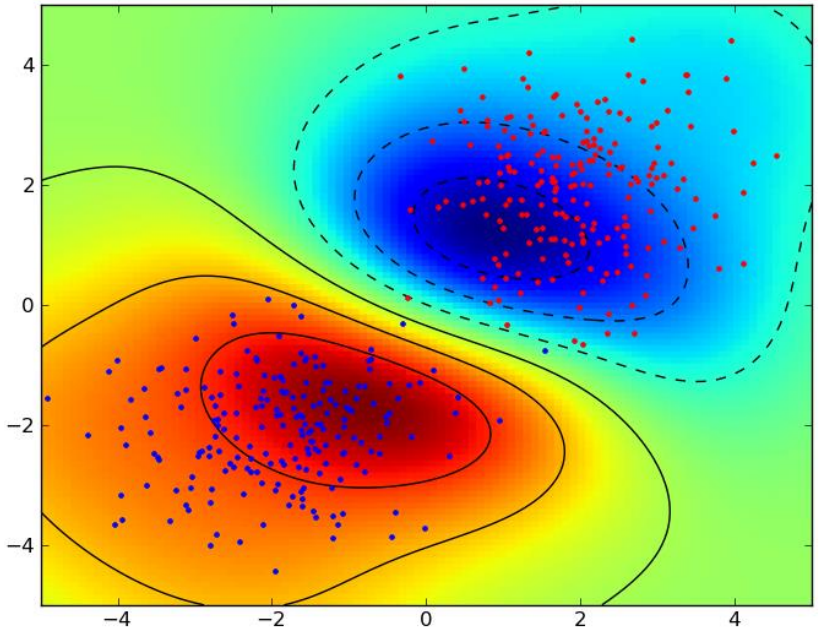
before hitting a datapoint

margin of a linear classifier as the width

General idea



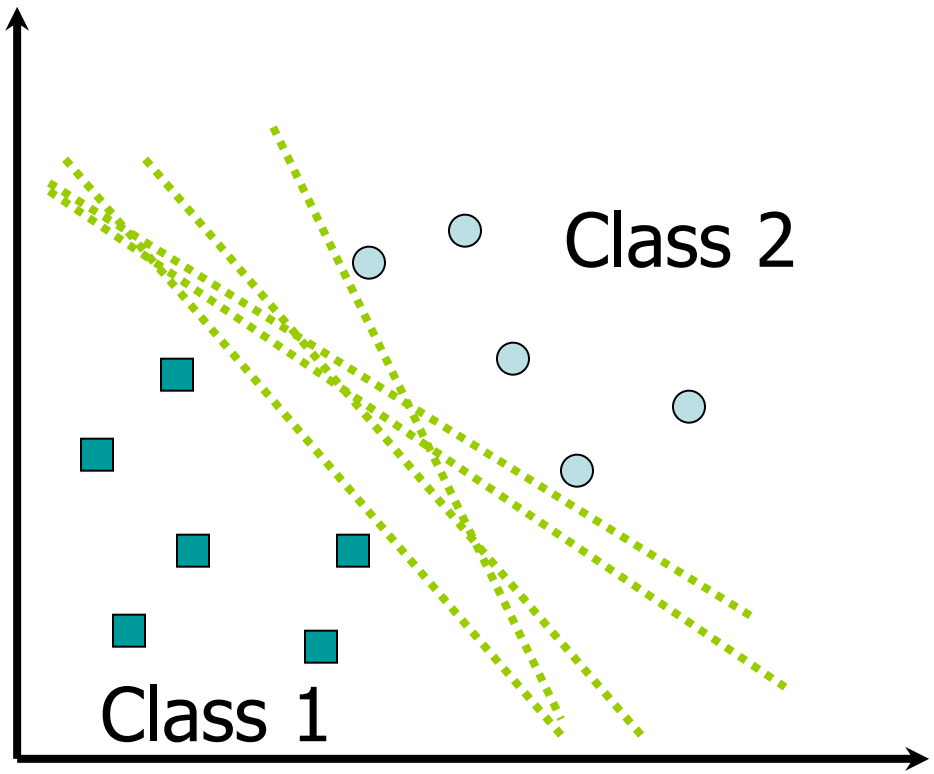
- Support Vectors are those datapoints that the margin pushes up against.
- The maximum margin linear classifier is the linear classifier with the maximum margin.



Two class – linear SVM

TWO CLASS – LINEAR SVM

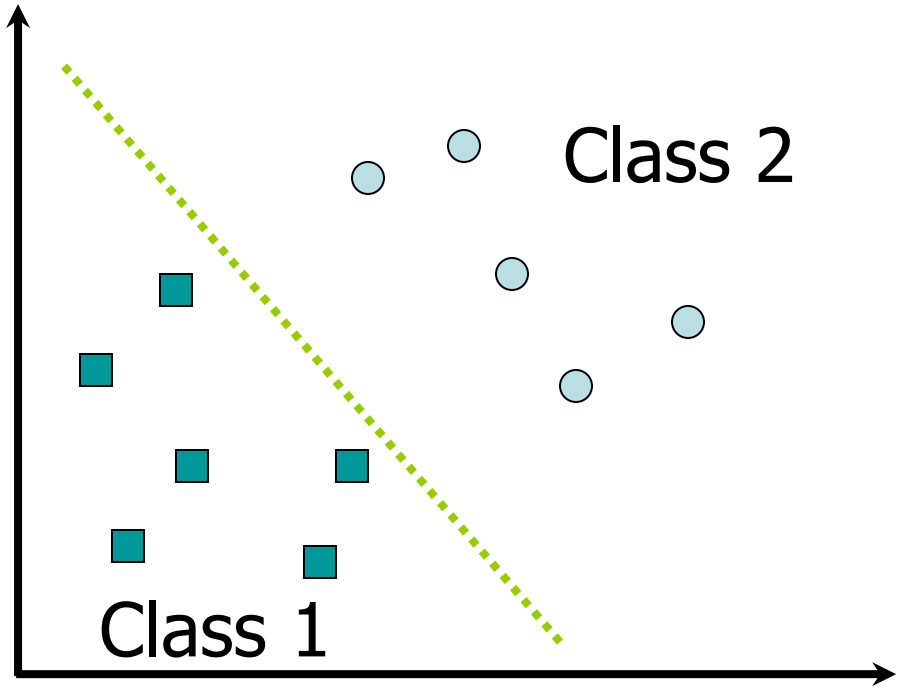
Two Class – Linear SVM

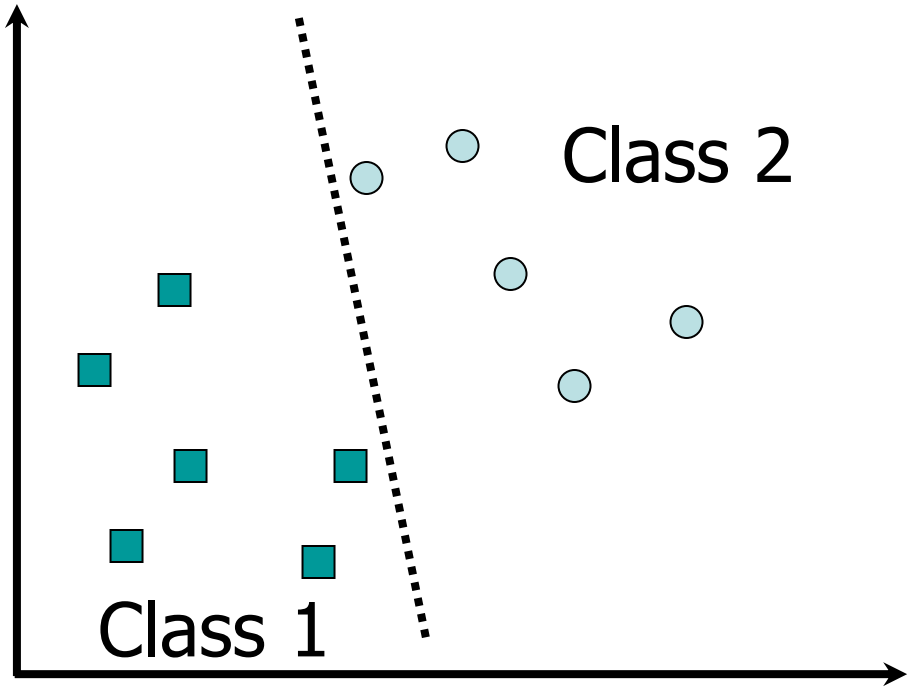


- Many decision boundaries can separate these two classes.
- Có rất nhiều siêu phẳng để phân lớp tập dữ liệu.

- Which one should we choose?
- Chúng ta nên chọn siêu phẳng nào?

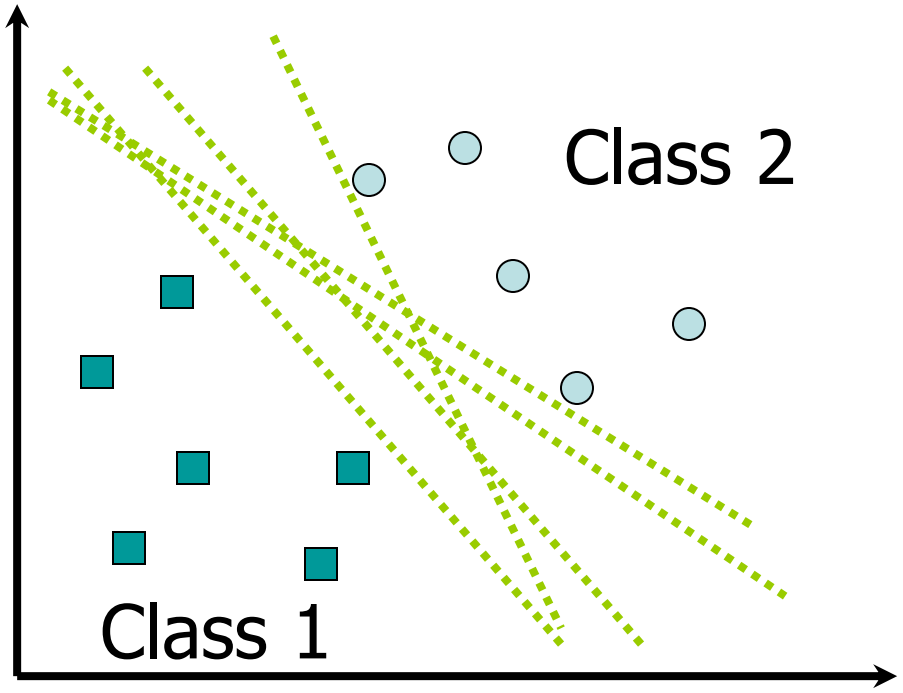
Two Class – Linear SVM





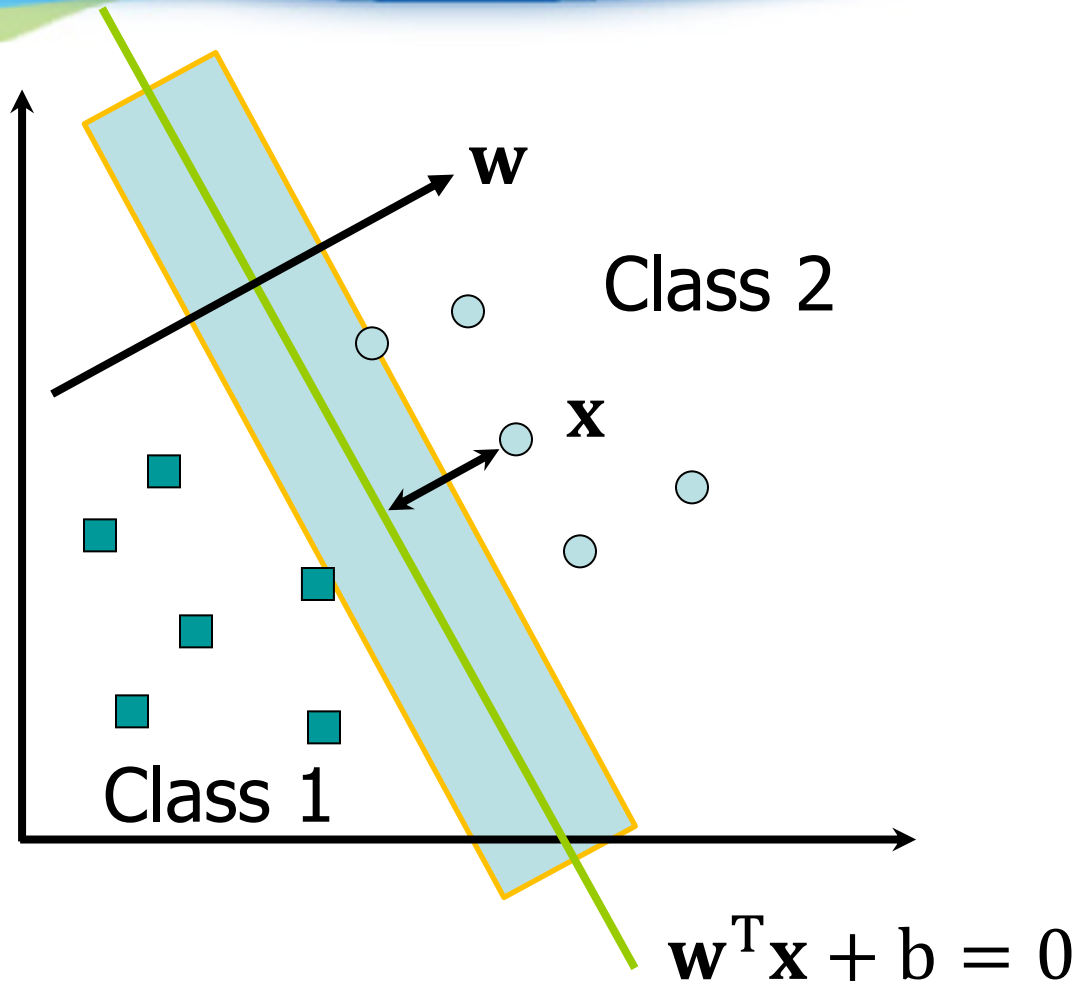
Example of Bad Decision Boundaries

Two Class – Linear SVM



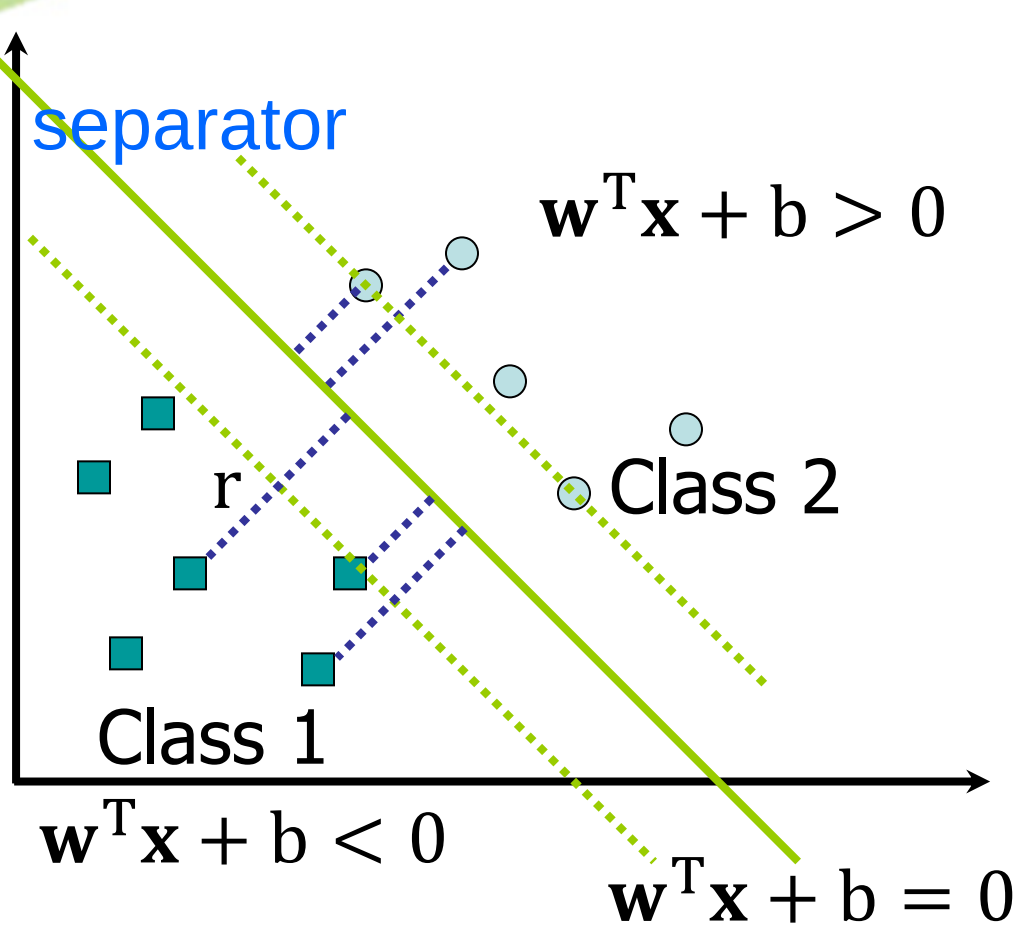
- Which of the linear separators is optimal?
- Siêu phẳng nào là tốt nhất?

Two Class – Linear SVM



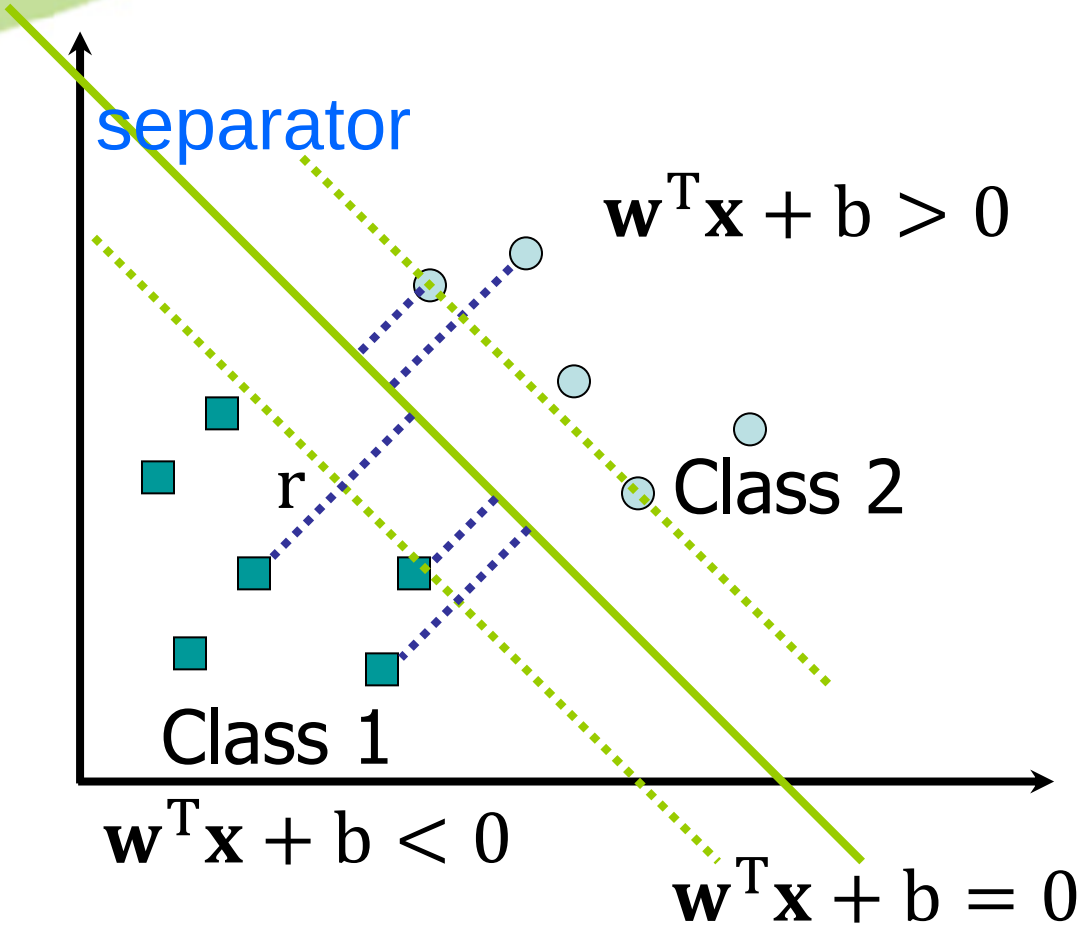
- **x** là véc tơ.
- **w** là véc tơ pháp tuyến của siêu phẳng.
- b là Scalar value.

Two Class – Linear SVM



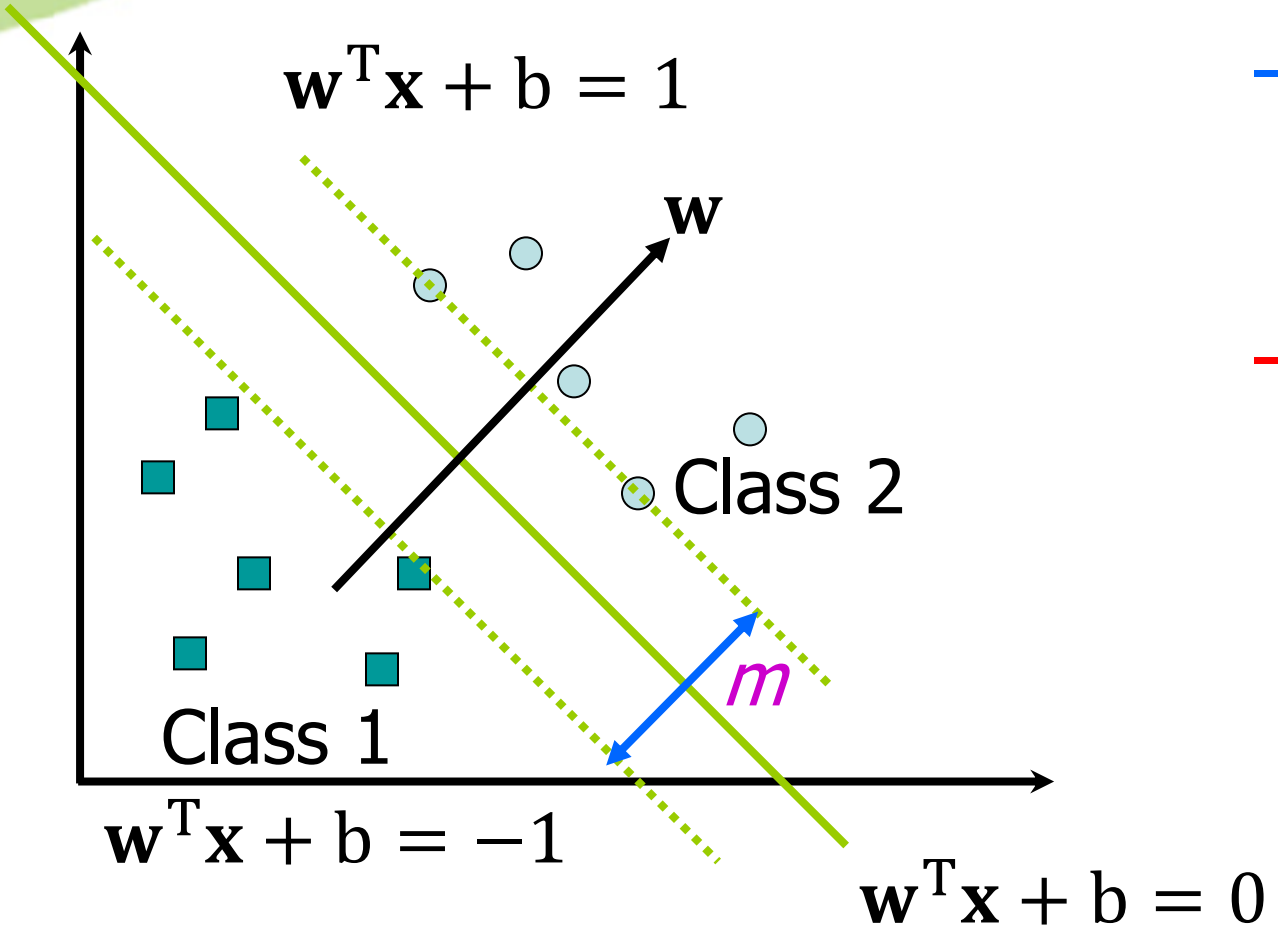
- Distance from véc tơ x_i to the separator is $r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$.
- Khoảng cách từ véc tơ $\mathbf{x}_i$ tới siêu phẳng $w^Tx + b = 0$ là $r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$.

Two Class – Linear SVM



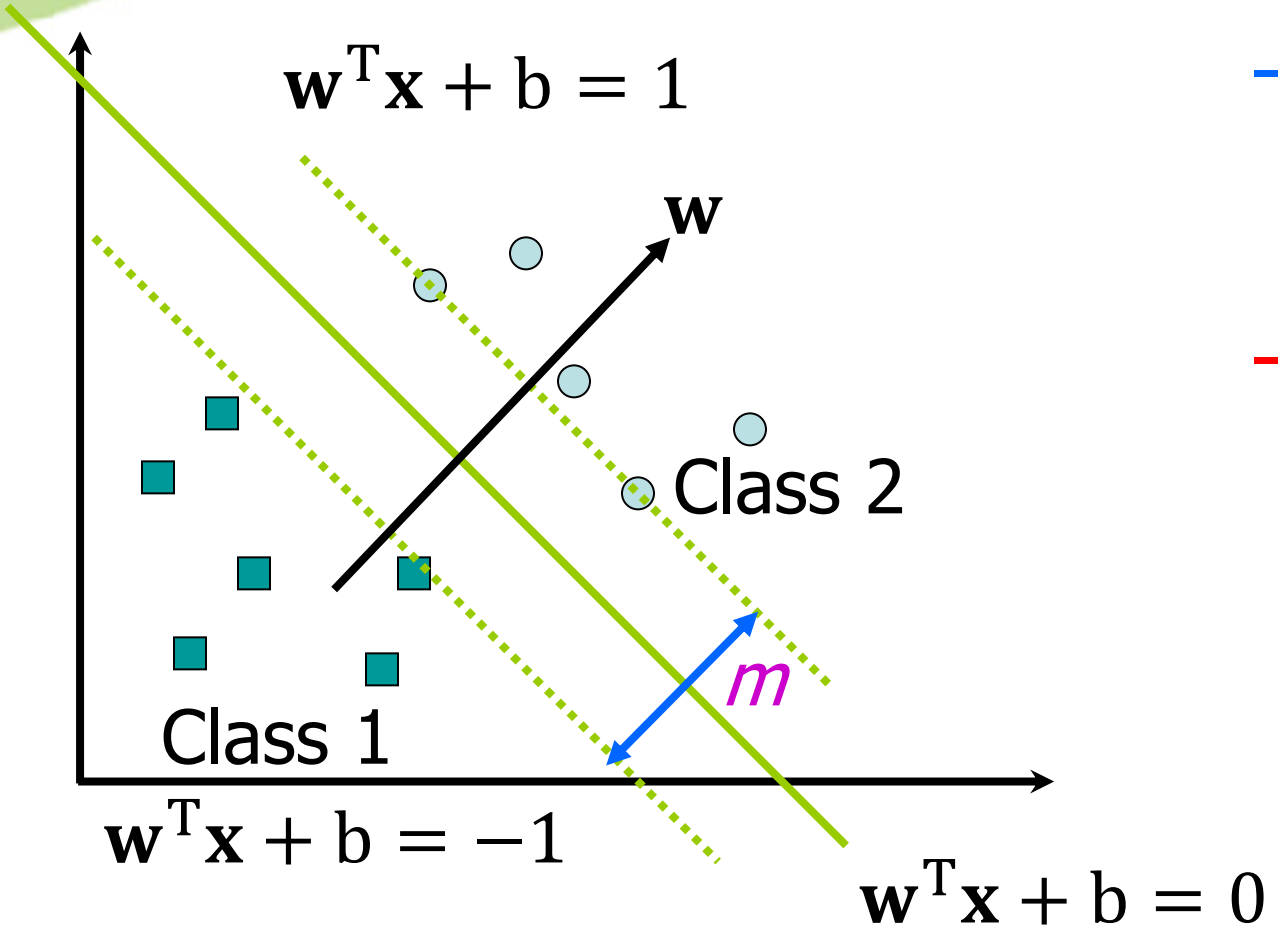
- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.

Two Class – Linear SVM



- The decision boundary should be as far away from the data of both classes as possible.
- Quyết định chọn siêu phẳng nào nên chọn siêu phẳng càng xa càng tốt dữ liệu của cả hai lớp có thể.

Two Class – Linear SVM



- Margin m of the separator is the distance between support vectors.
- Margin m của siêu phẳng là khoảng cách giữa các support vector.

Two Class – Linear SVM

- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$ be separated by a hyperplane with margin m.
- Then for each training example $(\mathbf{x}_i, y_i)$:
  - $\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b} \leq \frac{-m}{2}$ nếu $y_i = -1$.
  - $\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b} \geq \frac{m}{2}$ nếu $y_i = +1$.
- Suy ra: $y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b}) \geq \frac{m}{2}$.

Two Class – Linear SVM

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\frac{m}{2}$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is $\mathrm{r} = \frac{\mathrm{y}_s(\mathbf{w}^{\mathrm{T}}\mathbf{x}_s + \mathrm{b})}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$.
- Then the margin can be expressed through (rescaled) **w** and b as: $m = 2r = \frac{2}{\|\mathbf{w}\|}$.
- We should maximize the margin, $m = \frac{2}{\|\mathrm{w}\|}$.

Two Class – Linear SVM

– Then we can formulate the quadratic optimization problem:

+ Find **w** and *b* such that $m = \frac{2}{\|\mathrm{w}\|}$ is maximized.

+ and for all $\{(\mathbf{x}_i, y_i)\}_{i=1..n}: y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + \mathrm{b}) \geq 1.$

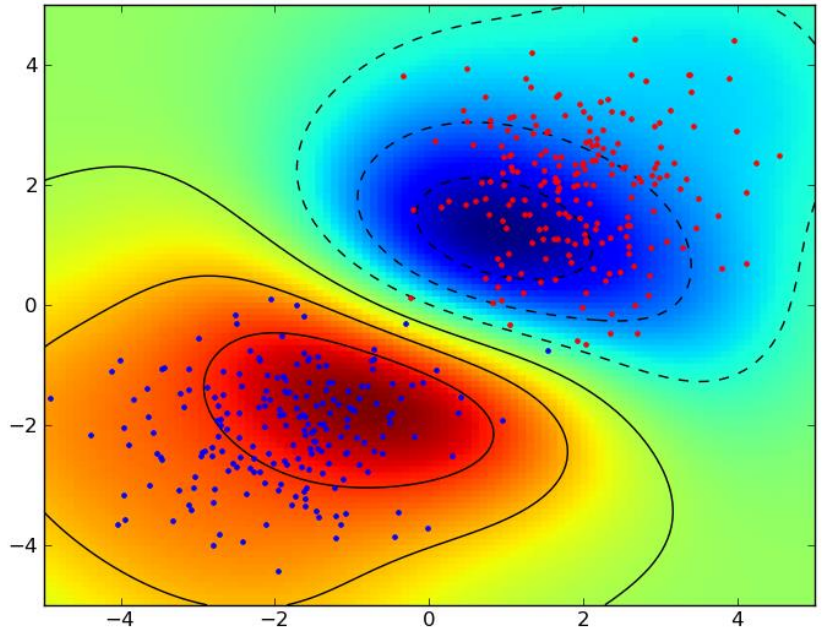
– Which can be reformulated as: Find **w** and *b* such that

+ $\boldsymbol{\Phi}(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized

+ and for all $\{(\mathbf{x}_i, y_i)\}_{i=1..n}: y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + \mathrm{b}) \geq 1.$

Linear SVMs Mathematically

- Which can be reformulated as: Find **w** and *b* such that
  - + $\boldsymbol{\Phi(\mathbf{w})} = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized
  - + and for all $\{(\mathbf{x}_i, y_i)\}_{i=1..n}: y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + \mathrm{b}) \geq 1.$
- Need to optimize a quadratic function subject to linear constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.



TÓM TẮT

Tóm tắt

- SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification) hay hồi qui (Regression).
- SVM là mô hình xây dựng một siêu phẳng hoặc một tập hợp các siêu phẳng trong một không gian nhiều chiều hoặc vô hạn chiều, có thể được sử dụng cho phân loại, hồi quy, hoặc các nhiệm vụ khác.

Tóm tắt

- Để phân loại tốt nhất thì phải xác định siêu phẳng (Optimal hyperplane) nằm ở càng xa các điểm dữ liệu của tất cả các lớp (Hàm lề) càng tốt, vì nói chung lề càng lớn thì sai số tổng quát hóa của thuật toán phân loại càng bé.
- Muốn các điểm dữ liệu có thể được chia tách một cách tuyến tính, thì bạn phải cần chọn hai siêu phẳng của lề sao cho không có điểm nào ở giữa chúng và khoảng cách giữa chúng là tối đa.

Tóm tắt

- Trong nhiều trường hợp, không thể phân chia các lớp dữ liệu một cách tuyến tính trong một không gian ban đầu được dùng để mô tả một vấn đề.
- Vì vậy, nhiều khi cần phải ánh xạ các điểm dữ liệu trong không gian ban đầu vào một không gian mới nhiều chiều hơn, để việc phân tách chúng trở nên dễ dàng hơn trong không gian mới.

DATASET

Dataset

- Tên tập dữ liệu: Social Network Ads.
- Nguồn: https://www.superdatascience.com/pages/machine-learning.
- Tập dữ liệu cho biết các thông tin của khách hàng và họ có mua hàng hay không.

Dataset

- Tập dữ liệu chứa 400 điểm dữ liệu, mỗi điểm dữ liệu có 5 thuộc tính gồm:
  + UserID: Mã số định danh của người dùng.
  + Gender: Giới tính của người dùng.
  + Age: Độ tuổi người dùng.
  + Estimated Salary: Mức lương ước đoán của người dùng.
  + Purchased: Là một trong hai số 0 và 1. Số 0 cho biết khách hàng không mua hàng và số 1 cho biết khách hàng có mua hàng.

Dataset

– Dưới đây là 5 điểm dữ liệu ngẫu nhiên trong tập dữ liệu.

UserID	Gender	Age	Estimated Salary	Purchased
15624510	Male	19	19,000	0
15810944	Male	35	20,000	1
15668575	Female	26	43,000	0
15603246	Female	27	57,000	0
15804002	Male	19	76,000	1

Dataset

– Yêu cầu với 2 thuộc tính:

+ Độ tuổi (Age)

+ Mức lương ước đoán (Estimated Salary)

Dự đoán khách hàng sẽ mua hàng hay không?

TIỀN XỬ LÝ DỮ LIỆU

Tiền xử lý dữ liệu

– Ở bài này, ta chỉ quan tâm đến hai thuộc tính tuổi và mức lương ước đoán.

```
1. import pandas as pd
2. import numpy as np
3. dataset = pd.read_csv("Social_Network_Ads.csv")
4. X = dataset.iloc[:, [2, 3]].values
5. Y = dataset.iloc[:, 4].values
```

Tiền xử lý dữ liệu

- Với mục đích:
  + Thuận tiện cho trực quan hóa kết quả sau khi huấn luyện
  + Tăng hiệu quả khi huấn luyện trên mô hình SVM

Ta chuẩn hóa dữ liệu về dạng có kỳ vọng bằng 0 và phương sai bằng 1.

- Lớp `StandardScaler` trong module `sklearn.preprocessing` đã được xây dựng sẵn để chuẩn hóa dữ liệu.

```
7. from sklearn.preprocessing import StandardScaler
8. SC = StandardScaler()
9. X = SC.fit_transform(X)
```

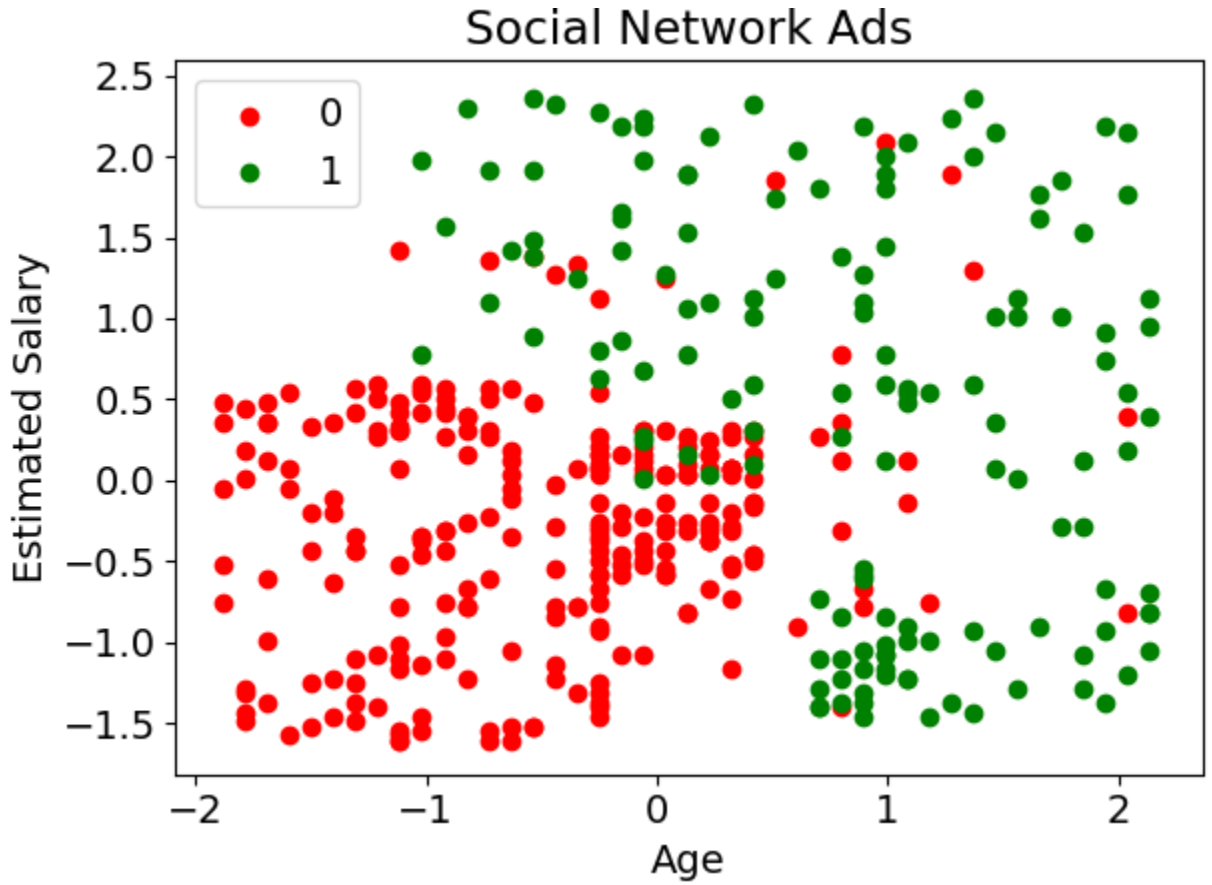
Tiền xử lý dữ liệu

- Chia dữ liệu thành hai tập training set và test set.
- Ta dùng hàm `train_test_split` được cung cấp trong module `sklearn.model_selection`.

```
10.from sklearn.model_selection import train_test_split
11.X_train, X_test, Y_train, Y_test = train_test_split(
   X, Y, train_size = 0.8, random_state = 0)
```

TRỰC QUAN HÓA DỮ LIỆU

Trực quan hóa dữ liệu



Trực quan hóa dữ liệu

– Xây dựng hàm trực quan hóa các điểm dữ liệu.

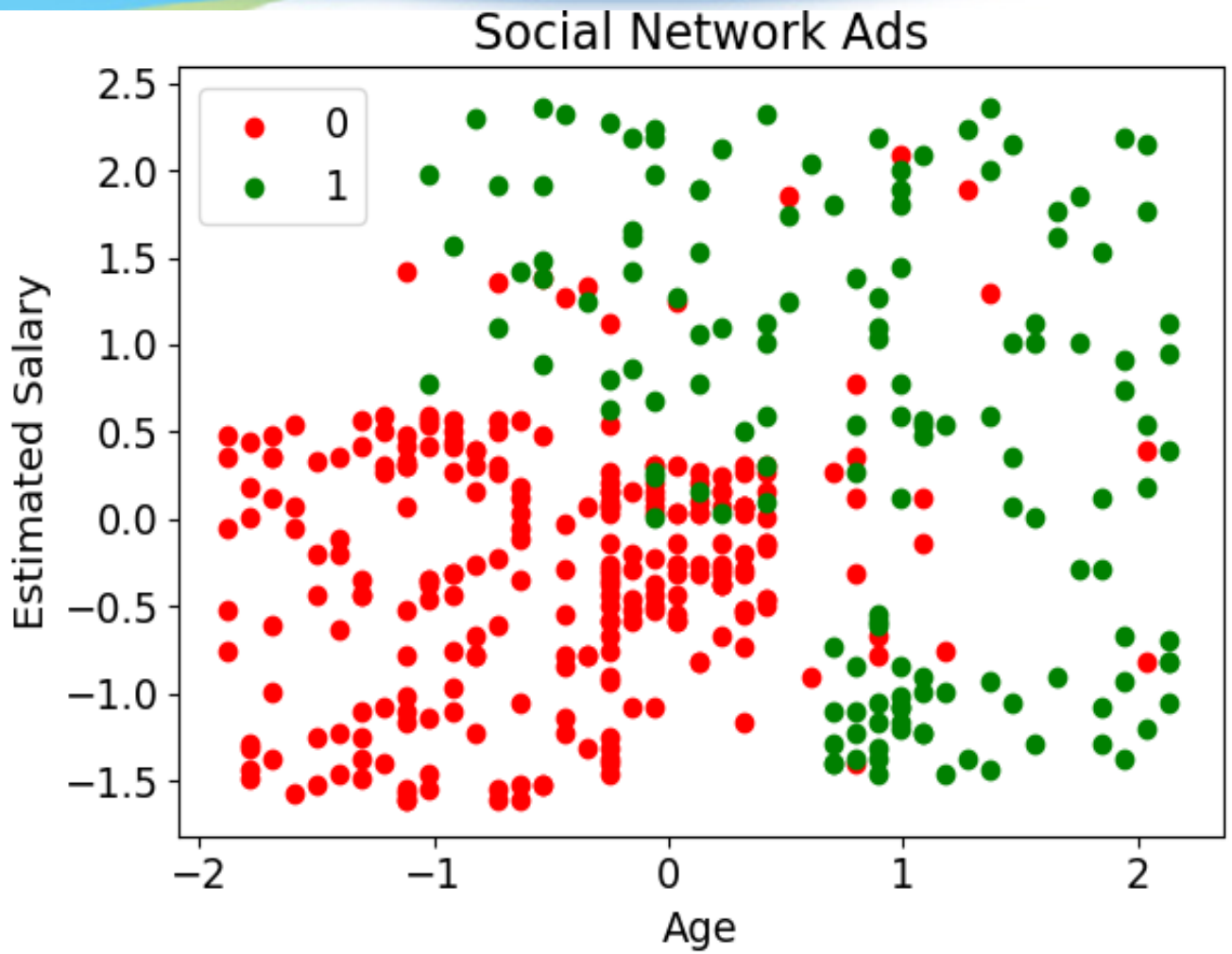
```
11.from matplotlib.colors import ListedColormap
12.import matplotlib.pyplot as plt
13.def VisualizingDataset(X_, Y_):
14.    X1 = X_[:, 0]
15.    X2 = X_[:, 1]
16.    for i, label in enumerate(np.unique(Y_)):
17.        plt.scatter(X1[Y_ == label], X2[Y_ == label]
    ,
```

Trực quan hóa dữ liệu

– Gọi hàm trực quan hóa dữ liệu.

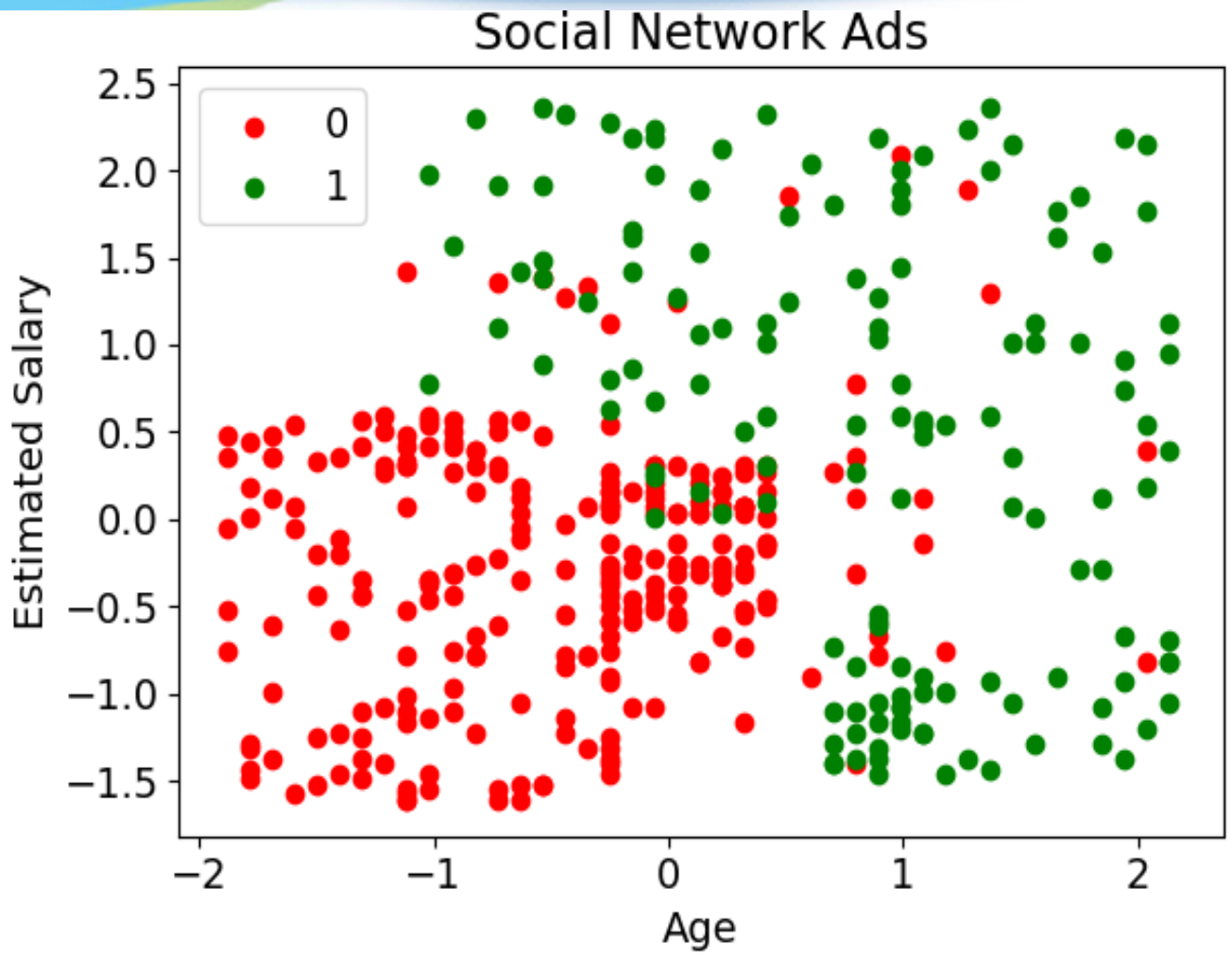
```
18.VisualizingDataset(X, Y)
19.plt.show()
```

Trực quan hóa dữ liệu



- Theo hình vẽ, ta thấy các điểm có sự phân bố thành 2 mảng.
  - + Mảng dưới trái phần lớn có màu đỏ, tức khách hàng không mua hàng.
  - + Mảng bên phải và mảng bên trên phần lớn có màu xanh, tức khách hàng có mua hàng.

Trực quan hóa dữ liệu



- Điều này là phù hợp vì các khách hàng trẻ và có mức lương thấp sẽ thường không mua hàng.
- Ngược lại, khách hàng cao tuổi hoặc có lương cao sẽ thường mua hàng nhiều hơn.

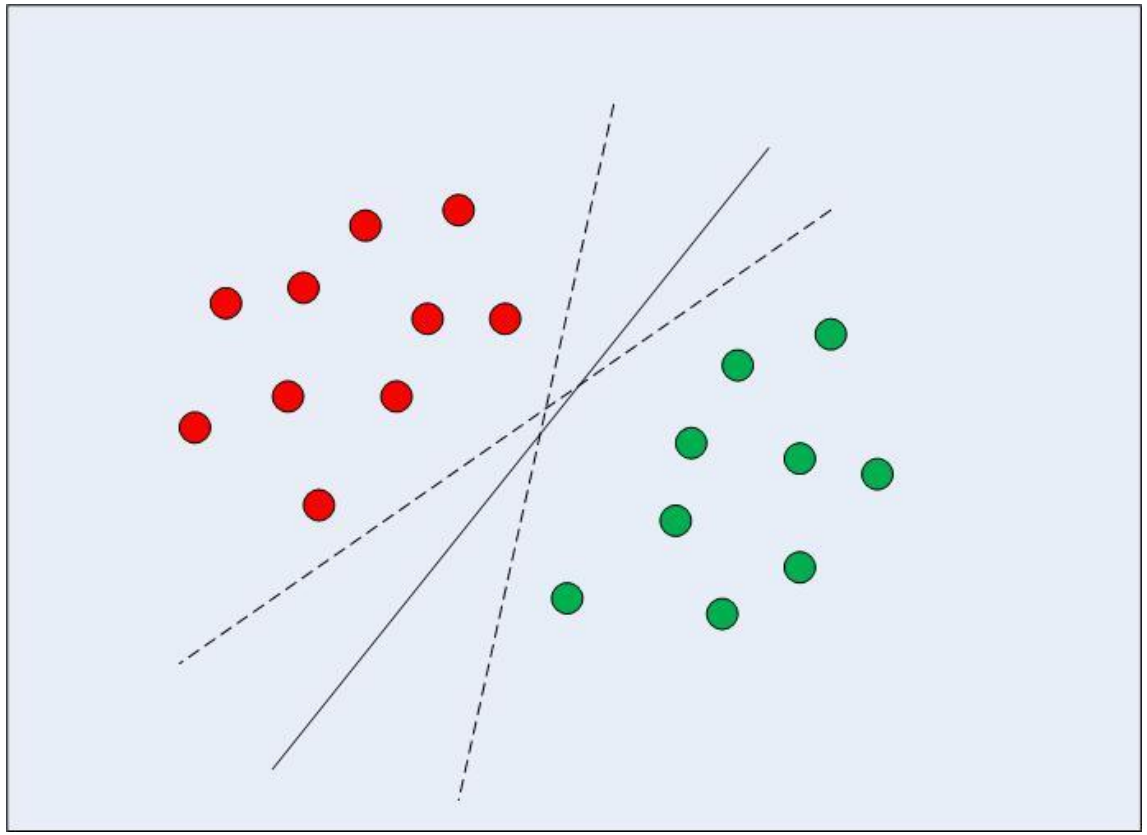
SUPPORT VECTOR MACHINE

Support Vector Machine

- *Support Vector Machine* là một trong các thuật toán phân loại được sử dụng phổ biến nhất trong Machine Learning.
- Đây là một thuật toán phân loại tuyến tính. Tức là sau khi huấn luyện, ta thu được các siêu phẳng phân chia các lớp dữ liệu với nhau, giống như thuật toán Logistic Regression.
- Mục tiêu của thuật toán này không những phân chia được các lớp dữ liệu với nhau, mà còn tìm cách để tối đa khoảng cách giữa đường phân chia với các điểm dữ liệu giữa các lớp (maximum margin).

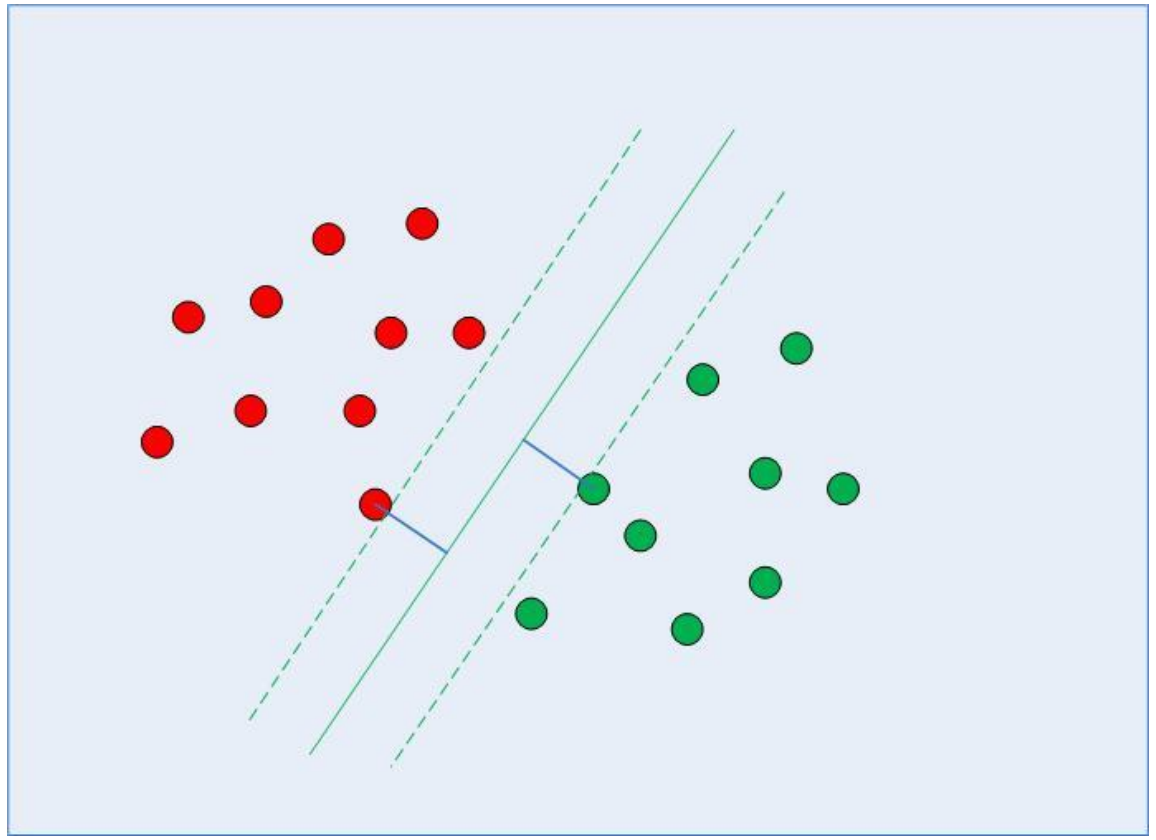
Support Vector Machine

– Các mô hình phân loại tuyến tính khác.



Support Vector Machine

– Mô hình SVM.



Support Vector Machine

- Thuật toán SVM sẽ tìm một số vector đặc biệt (gọi là support vectors).
- Mô hình (Model) dự đoán (predict) kết quả đầu ra của những điểm dữ liệu mới dựa trên các vector đặc biệt này.

Support Vector Machine

- Điểm đặc biệt của Support Vector Machine:
  - + Hầu hết các thuật toán Machine Learning khác đều phân chia dữ liệu dựa trên các điểm dữ liệu đặc trưng nhất của lớp dữ liệu đó.
  - + Trong khi đó, Support Vector Machine phân chia dữ liệu dựa trên các điểm dữ liệu dễ gây nhầm lẫn nhất giữa các lớp dữ liệu.

HUẤN LUYỆN MÔ HÌNH

Huấn luyện mô hình

- Ta sử dụng lớp `SVC` trong module `sklearn.svm` để huấn luyện mô hình.
- Mô hình Support Vector Machine Classification chuẩn được thiết lập với tham số `kernel = "linear"`.

```
20.from sklearn.svm import SVC
21.classifier = SVC(kernel = "linear")
22.classifier.fit(X_train, Y_train)
```

TRỰC QUAN HÓA KẾT QUẢ MÔ HÌNH

Trực quan hóa kết quả mô hình

- Ta tạo một *confusion matrix*. Đây là một ma trận có kích thước là $p \times p$ với p là số phân lớp trong bài toán đang xét, ở đây là 2.
- Phần tử ở dòng thứ i, cột thứ j của confusion matrix biểu thị số lượng phần tử có loại là i và được phân vào loại j.
- Hàm `confusion_matrix` trong module `sklearn.metrics` sẽ hỗ trợ ta xây dựng confusion matrix.

```
23.from sklearn.metrics import confusion_matrix
24.cm = confusion_matrix(Y_train, classifier.predict(X_train))
25.print(cm)
```

Trực quan hóa kết quả mô hình

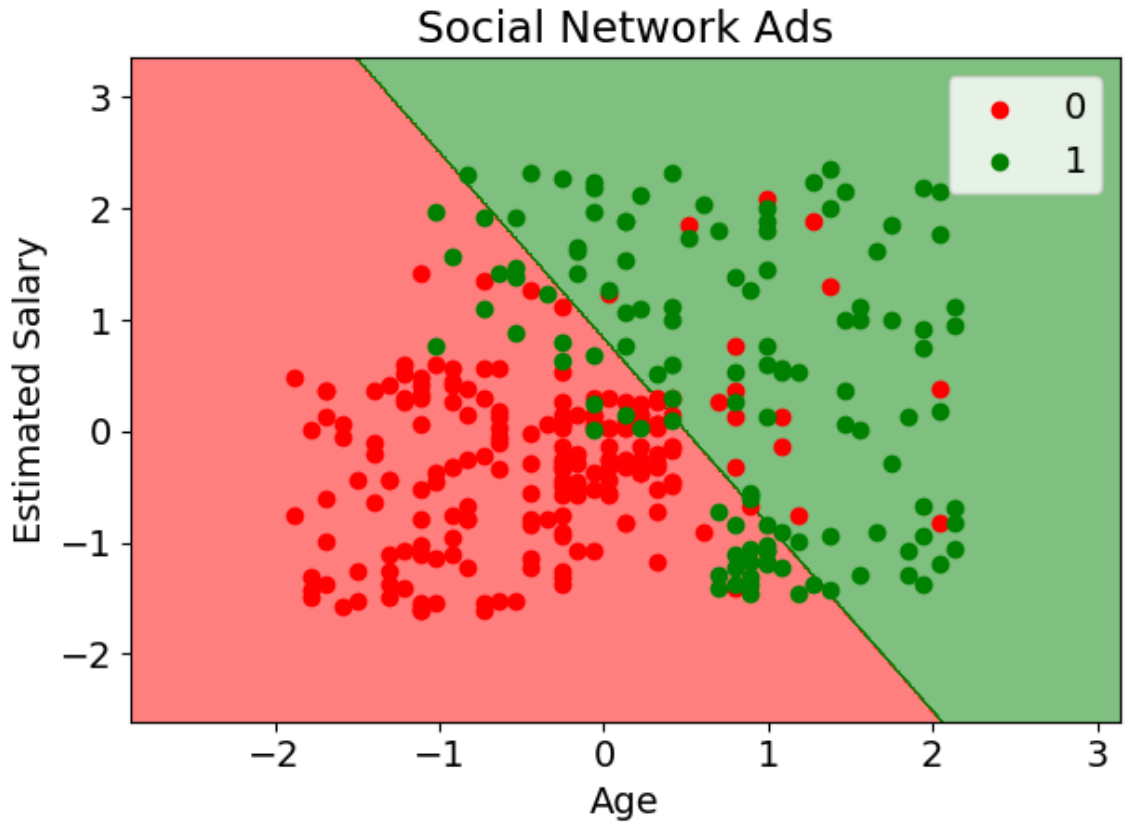
- Confusion Matrix được in ra là:

	0	1
0	181	18
1	39	82

- Theo ma trận trên, số lượng dữ liệu được phân loại đúng là $181 + 82 = 263$ điểm dữ liệu.
- Số lượng dữ liệu phân loại sai là $18 + 39 = 57$ điểm dữ liệu.
- Tỉ lệ điểm dữ liệu phân loại sai là $57/320 \approx 0.18$.

Trực quan hóa kết quả mô hình

- Ta trực quan hóa kết quả mô hình trên mặt phẳng tọa độ bằng cách vẽ 2 vùng phân chia mà mô hình thu được sau quá trình huấn luyện.



Trực quan hóa kết quả mô hình

– Xây dựng hàm trực quan hóa kết quả bằng cách tạo 2 vùng phân chia mà mô hình đạt được.

```
26. def VisualizingResult(model, X_):
27.     X1 = X_[:, 0]
28.     X2 = X_[:, 1]
29.     X1_range = np.arange(start= X1.min()-1, stop= X1.max()+1, step =
    0.01)
30.     X2_range = np.arange(start= X2.min()-1, stop= X2.max()+1, step =
    0.01)
31.     X1_matrix, X2_matrix = np.meshgrid(X1_range, X2_range)
```

Trực quan hóa kết quả mô hình

– Xây dựng hàm trực quan hóa kết quả bằng cách tạo 2 vùng phân chia mà mô hình đạt được.

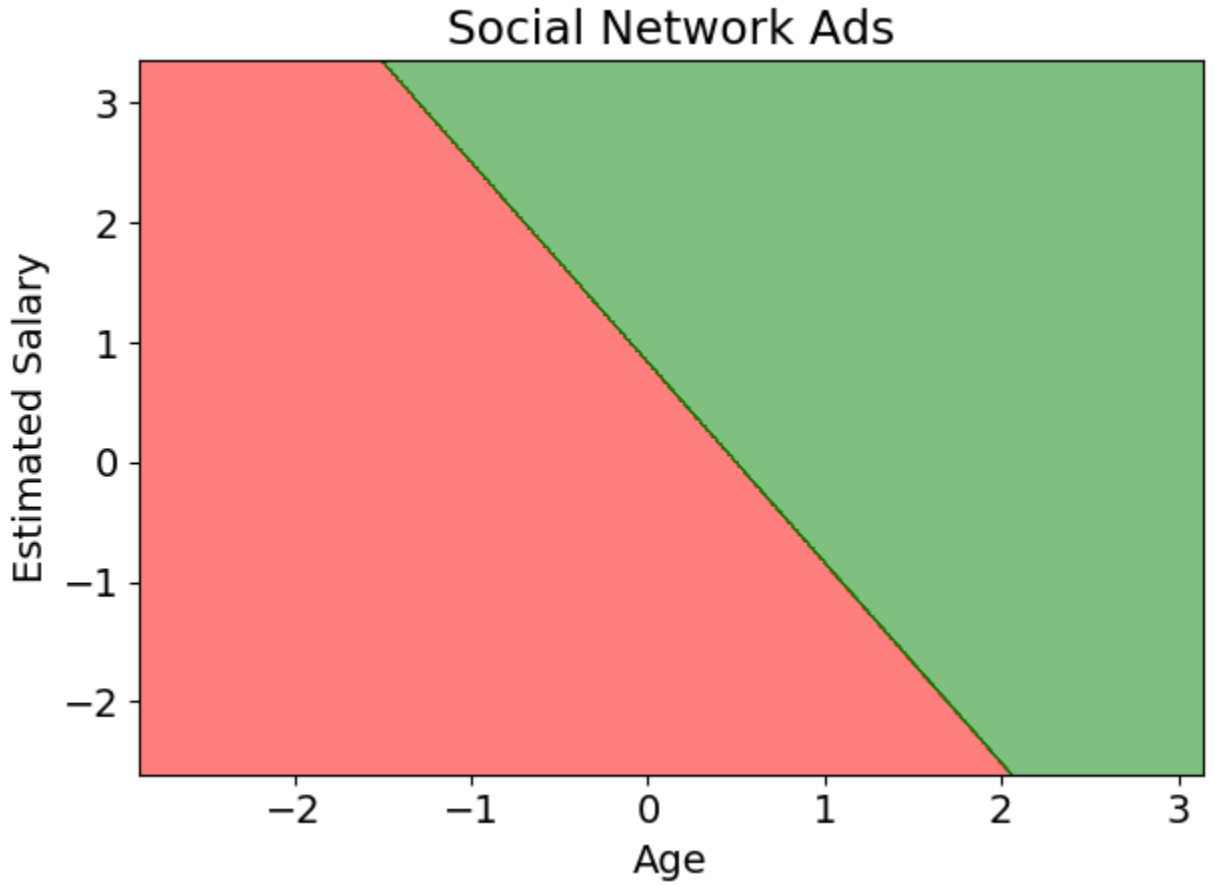
```
26.def VisualizingResult(model, X_):
31.     ...
32.     X_grid= np.array([X1_matrix.ravel(), X2_matrix.ravel()]).T
33.     Y_grid= model.predict(X_grid).reshape(X1_matrix.shape)
34.     plt.contourf(X1_matrix, X2_matrix, Y_grid, alpha = 0.5,
    cmap = ListedColormap(("red", "green")))
```

Trực quan hóa kết quả mô hình

– Trực quan hóa kết quả mô hình.

```
35.VisualizingResult(classifier, X_train)
36.plt.show()
```

Trực quan hóa kết quả mô hình

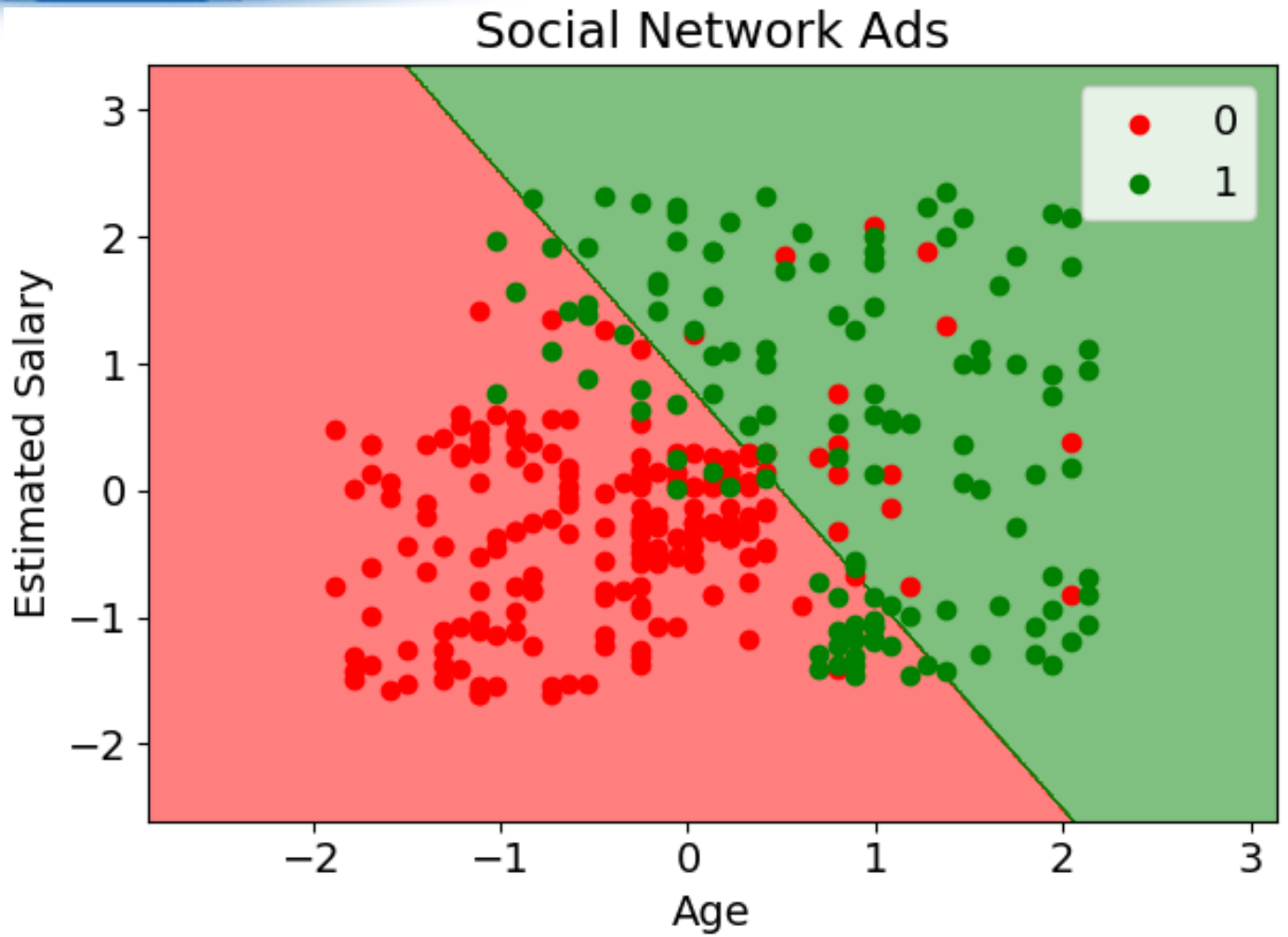


Trực quan hóa kết quả mô hình

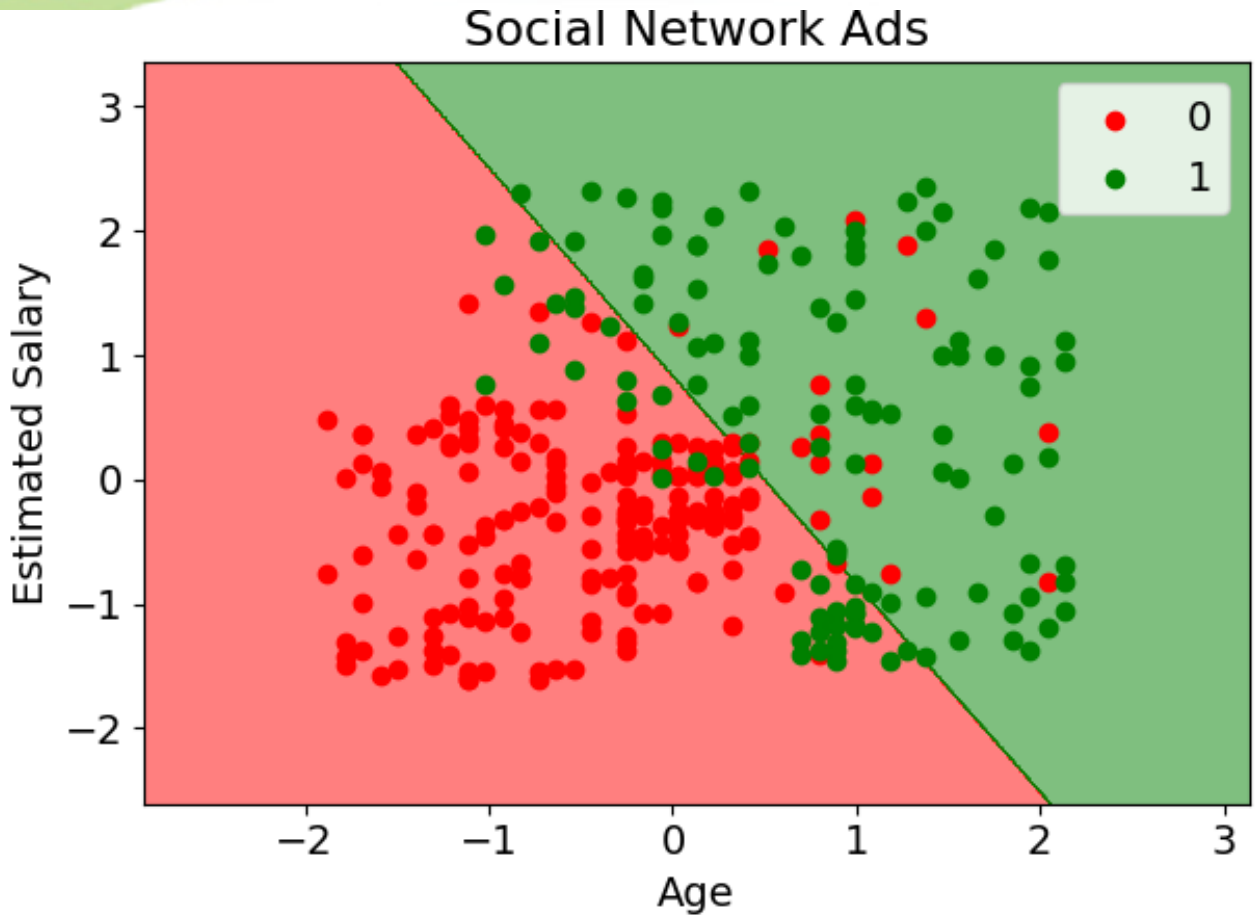
- Hoàn thiện quá trình trực quan bằng cách vẽ thêm các điểm dữ liệu huấn luyện lên mặt phẳng tọa độ.

```
37.VisualizingResult(classifier, X_train)
38.VisualizingDataset(X_train, Y_train)
39.plt.show()
```

Trực quan hóa kết quả mô hình



Trực quan hóa kết quả mô hình



- Nhận xét:
  - + Mô hình có độ chính xác chấp nhận được, vẫn có nhiều điểm phân chia nhầm.
  - + Mô hình phân chia theo một đường thẳng, vì SVM là một mô hình phân loại tuyến tính.

KIỂM TRA KẾT QUẢ TRÊN TẬP TEST

Kiểm tra kết quả trên tập test

– Tạo *confusion matrix* trên tập test.

```
40.cm = confusion_matrix(Y_test, classifier.predict(X_t
   est))
41.print(cm)
```

Kiểm tra kết quả trên tập test

- Confusion Matrix được in ra là:

	0	1
0	57	1
1	6	16

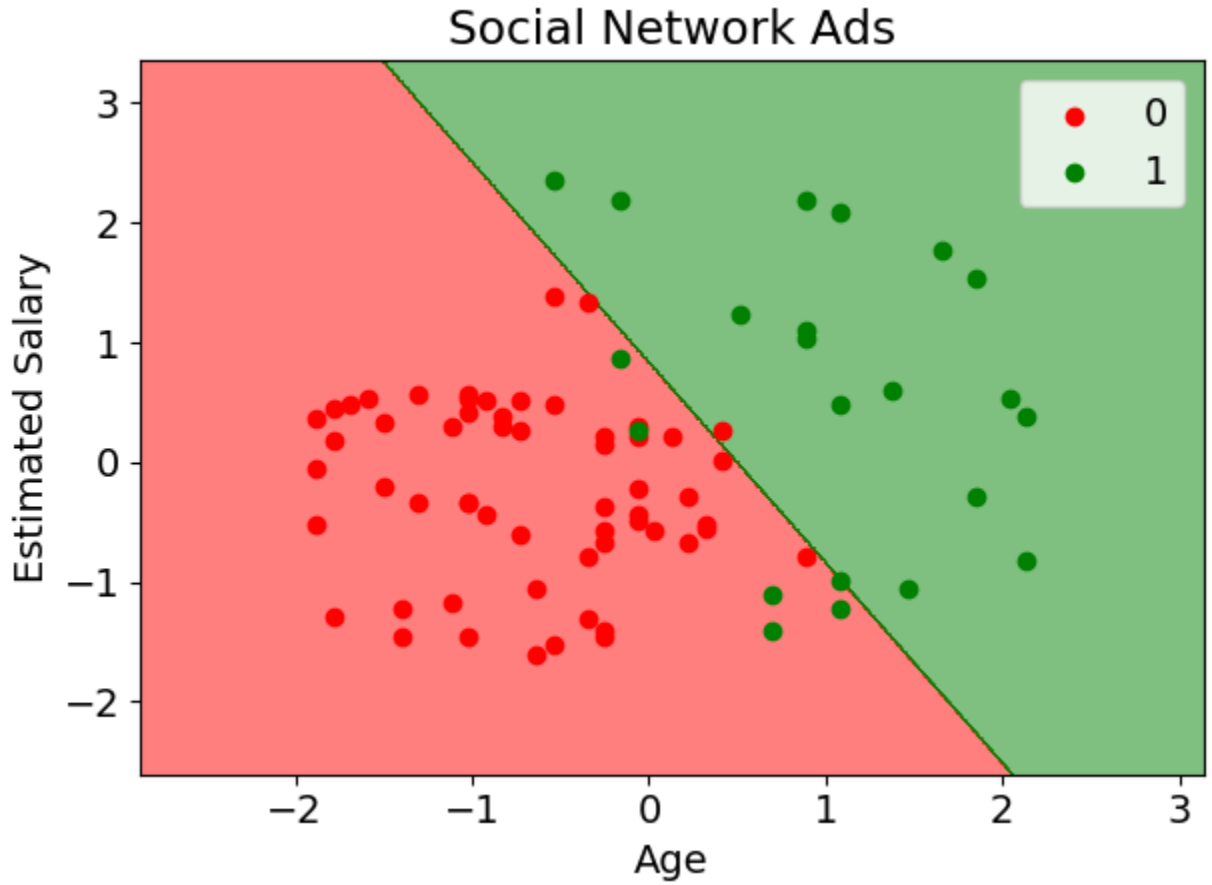
- Theo ma trận trên, số lượng dữ liệu được phân loại đúng là $57 + 16 = 73$ điểm dữ liệu.
- Số lượng dữ liệu phân loại sai là $5 + 2 = 7$ điểm dữ liệu.
- Tỉ lệ điểm dữ liệu phân loại sai là $7/80 \approx 0.0875$.

Kiểm tra kết quả trên tập test

- Thực hiện tương tự trực quan hóa kết quả mô hình trên tập traning.

```
42.VisualizingResult(classifier, X_test)
43.VisualizingDataset(X_test, Y_test)
44.plt.show()
```

Kiểm tra kết quả trên tập test



	0	1
0	57	1
1	6	16

Kiểm tra kết quả trên tập test

– Xây dựng hàm so sánh kết quả trên một điểm dữ liệu trong tập test.

```
45.def compare(i_example):
46.     x = X_test[i_example : i_example + 1]
47.     y = Y_test[i_example]
48.     y_pred = classifier.predict(x)
49.     x_inv = SC.inverse_transform(x)
50.     print(x_inv, y, y_pred)
```

Kiểm tra kết quả trên tập test

– Gọi thực hiện hàm so sánh trên 5 điểm dữ liệu, có chỉ mục từ thứ 7 đến 11 trong tập kiểm thử.

```
51.for i in range(7, 12):
52.     compare(i)
```

Kiểm tra kết quả trên tập test

Age	Estimated Salary	Purchased	Predicted Purchased
36	144,000	1	1
18	68,000	0	0
47	43,000	0	0
30	49,000	0	0
28	53,000	0	0

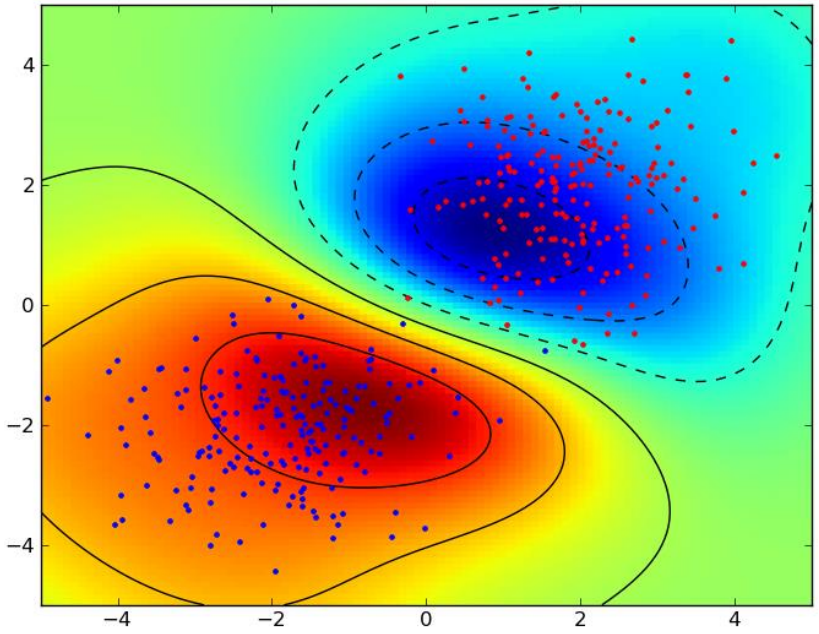
Chúc các bạn học tốt

Thân ái chào tạm biệt các bạn

ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN TP.HCM

TOÀN DIỆN – SÁNG TẠO – PHỤNG SỰ



Kernel function

HÀM KERNEL

Hàm kernel là gì

- A kernel function is a function that is eqiuvalent to an inner product in some feature space.
- For some functions $K(\mathbf{x}_i, \mathbf{x}_j)$ checking that $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem: Every semi-positive definite symmetric function is a kernel.

Hàm kernel là gì

– Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & K(\mathbf{x}_1, \mathbf{x}_3) & \dots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & K(\mathbf{x}_2, \mathbf{x}_3) & \dots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \dots & \dots & \dots & \dots & \dots \\ K(\mathbf{x}_{n-1}, \mathbf{x}_1) & K(\mathbf{x}_{n-1}, \mathbf{x}_2) & K(\mathbf{x}_{n-1}, \mathbf{x}_3) & \dots & K(\mathbf{x}_{n-1}, \mathbf{x}_n) \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & K(\mathbf{x}_n, \mathbf{x}_3) & \dots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Một số hàm kernel

– Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.

+ Mapping $\Phi: \mathbf{x} \to \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

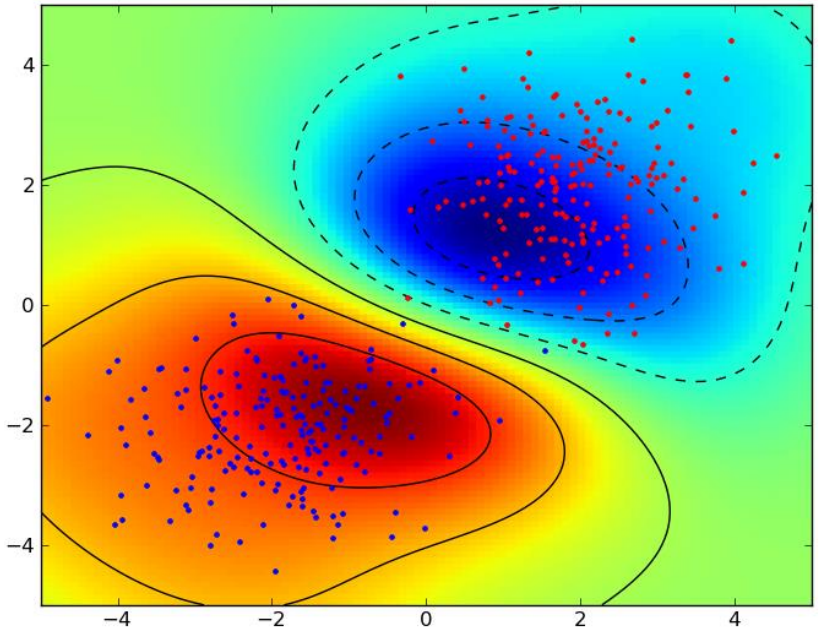
– Polynomial of power p: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$.

+ Mapping $\Phi: \mathbf{x} \to \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions.

Một số hàm kernel

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

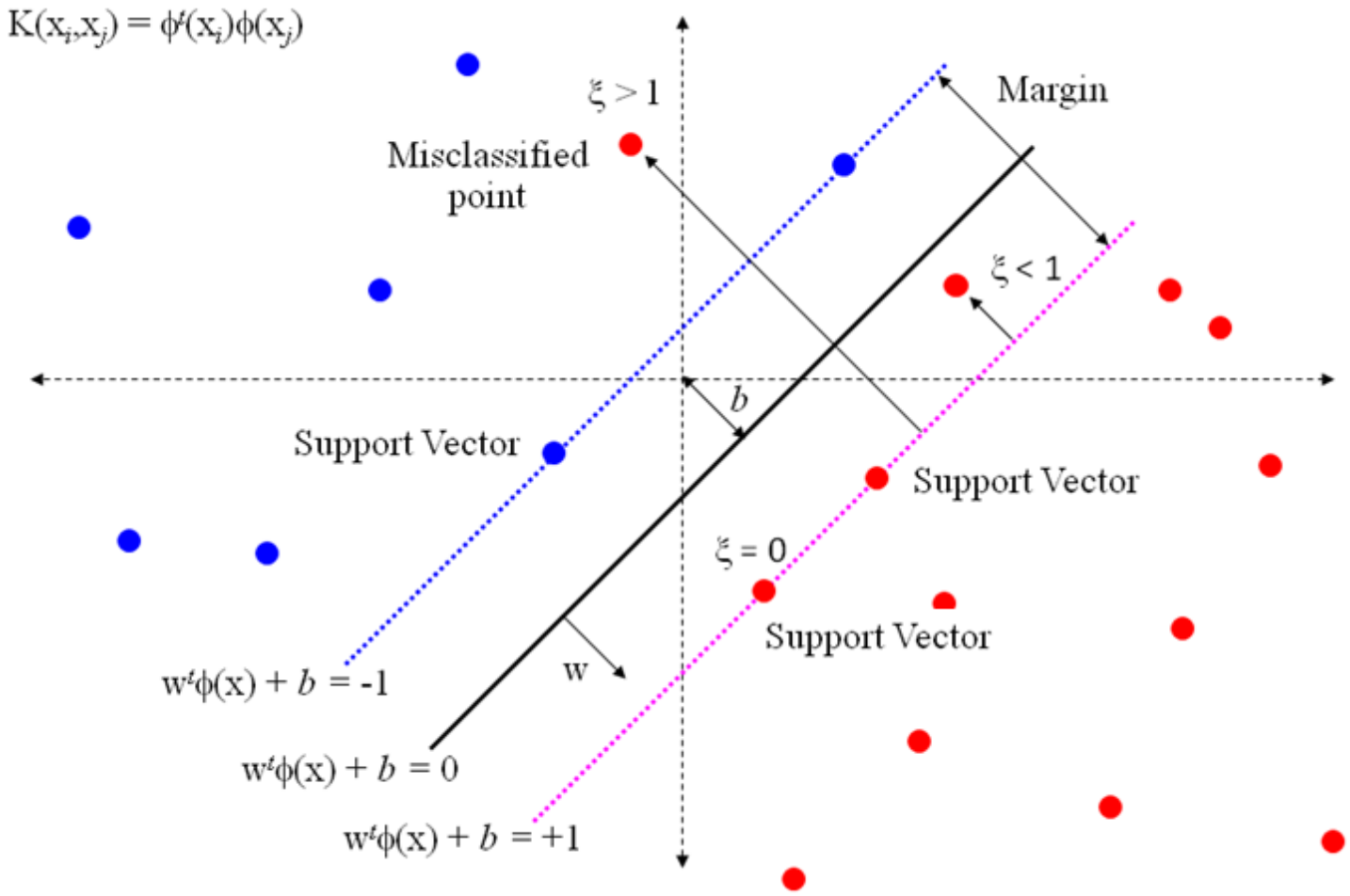
- Higher-dimensional space still has intrinsic dimensionality d (the mapping is not onto), but linear separators in it correspond to non-linear separators in original space.

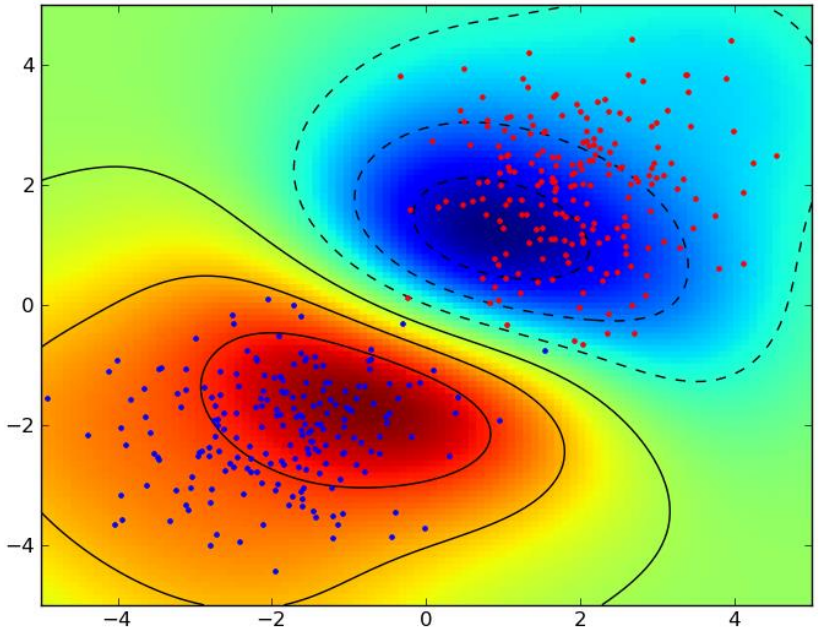


Example

VÍ DỤ MINH HỌA

Ví dụ minh họa





Programming Exercise

BÀI TẬP THỰC HÀNH

Bài tập thực hành trên Python

- Problem Statement: Use Machine Learning to predict cases of breast cancer using patient treatment history and health data.
- Dataset: Breast Cancer Wisconsin (Diagnostic).
- Tham khảo: https://intellipaat.com/blog/tutorial/machine-learning-tutorial/svm-algorithm-in-python/

Chúc các bạn học tốt

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN TP.HCM

Nhóm UIT-Together

Nguyễn Tấn Trần Minh Khang