Support Vector Machine

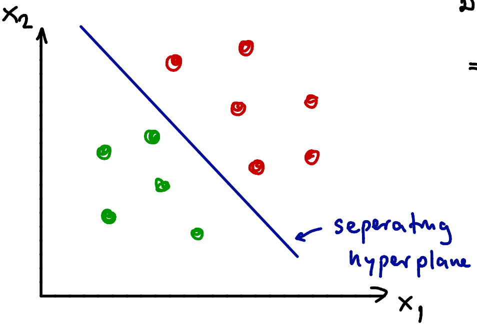
Petchara Pattarakijwanich

Introduction to Data Science, 18 November 2022

Goal of this week

- Support Vector Machine
  - Linearly Separable Data
  - Hyperplanes
  - Maximal Margin Classifier
  - Soft Margin Classifier
  - Kernels
  - Support Vector Machine
- Neural Network?

Linearly Separable Data & Separating Hyperplane



มี Hyperplane ที่แบ่งข้อมูลได้ 100%

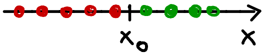
⇒ Linearly Seperable Data

แบ่ง แดง/เขียว ด้วย "Hyperplane"

ระนาบใน n มิติ

Linearly Separable Data & Separating Hyperplane

1 มิติ

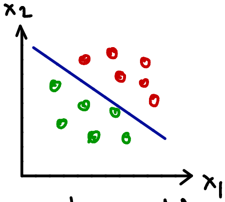


$\Rightarrow$ hyperplane 0 มิติ (จุด)

$x = x_0 =$ คงที่

$\Rightarrow \boxed{\beta_0 + \beta_1 x = 0}$

2 มิติ

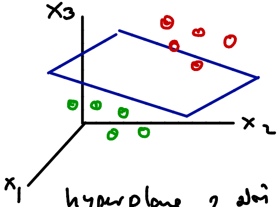


hyperplane 1 มิติ (เส้นตรง)

$x_2 = m x_1 + c$

$\Rightarrow \boxed{\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0}$

3 มิติ



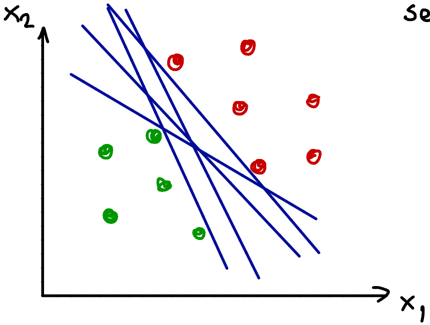
hyperplane 2 มิติ (ระนาบ)

$\Rightarrow \boxed{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 = 0}$

n มิติ $\Rightarrow$ hyperplane n-1 มิติ

รูปแบบ hyperplane $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n = 0 \Rightarrow \boxed{\beta_0 + \vec{\beta} \cdot \vec{x} = 0}$

Linearly Separable Data & Separating Hyperplane



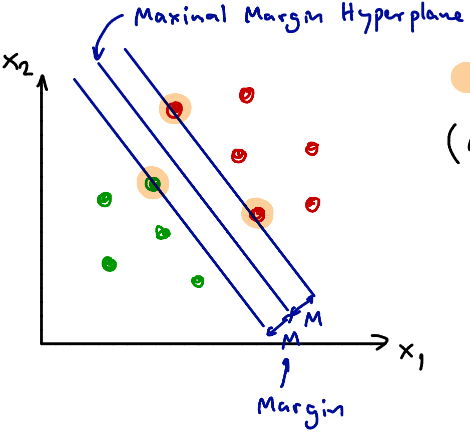
separatng hyperplane ไม่ unique

⇒ อันไหนดีสุด ?

⇒ อันที่ดีสุดคือ Margin เยอะสุด

ขนาดช่องว่าง ระหว่างสองกลุ่ม

Linearly Separable Data & Separating Hyperplane

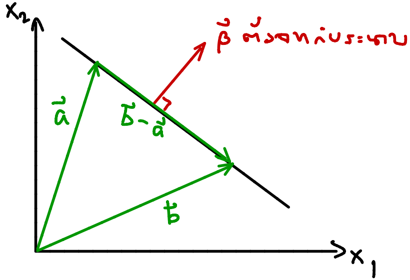


= support vector

(Maximal Margin Hyperplane ขึ้นกับ support vector เท่านั้น ไม่ขึ้นกับข้อมูลจุดอื่นเลย)

Maximal Margin Classifier

Math Background for Hyperplane



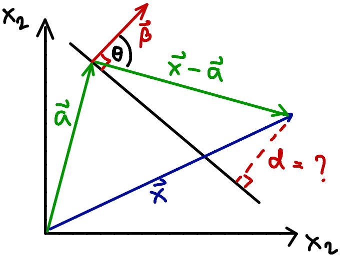
จุด $\vec{a}, \vec{b}$ อยู่บนระนาบ $\Rightarrow$ $\vec{b}-\vec{a}$ อยู่บนระนาบ

$\Rightarrow \vec{\beta} \perp \vec{b}-\vec{a} \Rightarrow \vec{\beta}\cdot(\vec{b}-\vec{a}) = 0 \Rightarrow \vec{\beta}\cdot\vec{b} = \vec{\beta}\cdot\vec{a}$

$\Rightarrow \vec{\beta}\cdot\vec{x}$ = ค่าคงที่ (สำหรับจุด $\vec{x}$ ใดๆ ที่อยู่บนระนาบ)

$\boxed{\beta_0 + \vec{\beta}\cdot\vec{x} = 0 \quad \text{สมการระนาบ}}$

Math Background for Hyperplane



$$\vec{\beta}\cdot(\vec{x}-\vec{a}) = |\vec{\beta}|\overbrace{|\vec{x}-\vec{a}|\cos\theta}^{=d}$$

ให้ $|\vec{\beta}| = 1$, $\vec{\beta}$ = unit vector

$$\Rightarrow \quad d = \vec{\beta}\cdot(\vec{x}-\vec{a})$$

ระยะตั้งฉากของ $\vec{x}$ จากระนาบ

normal vector

จุดหนึ่ง(ใดๆ) บนระนาบ

$\beta_0 + \vec{\beta}\cdot\vec{x} = 0$ สำหรับ $\vec{x}$ บนระนาบ

$\beta_0 + \vec{\beta}\cdot\vec{a} = 0$

$\Rightarrow \quad \vec{\beta}\cdot\vec{a} = -\beta_0$

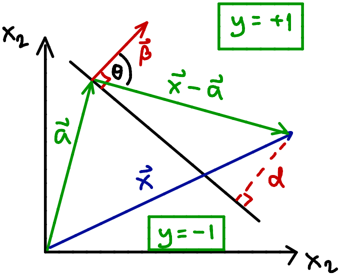
$$\boxed{d = \beta_0 + \vec{\beta}\cdot\vec{x}}$$

$d > 0$ ด้านหนึ่งของระนาบ

$d < 0$ อีกด้านหนึ่งของระนาบ

$d = 0$ บนระนาบ

Math Background for Hyperplane



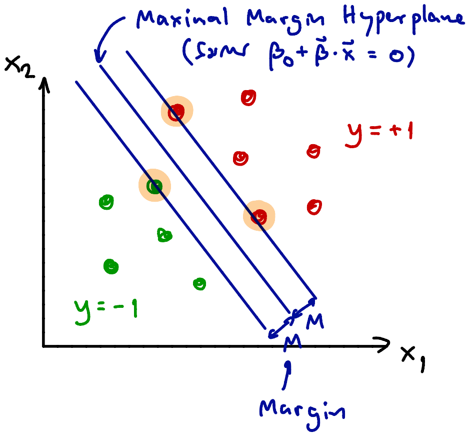
$$d = \beta_0 + \vec{\beta} \cdot \vec{x}$$

[ระยะทาง ที่มีเครื่องหมาย
ขึ้นกับ $\vec{x}$ อยู่ฝั่งไหนของระนาบ
ระยะจริง $= |d|$]

$$\boxed{d = y\left(\beta_0 + \vec{\beta} \cdot \vec{x}\right)}$$

[ระยะจริง เป็นบวกเสมอ]

Maximal Margin Classifier



หา β_0, $\vec{\beta} = \beta_1\hat{e}_1 + \beta_2\hat{e}_2 + \dots + \beta_n\hat{e}_n$ ที่ทำให้ M มากสุด โดยที่ยังทำให้ constraints พวกนี้เป็นจริง

① $\beta_1^2 + \beta_2^2 + \dots + \beta_n^2 = 1$

[$\vec{\beta}$ เป็น unit vector]

② $y_i(\beta_0 + \vec{\beta}\cdot\vec{x}_i) \geq M$

สำหรับทุกจุดข้อมูล i

[ทุกจุดอยู่ห่างจาก hyperplane เป็นระยะอย่างน้อย M]

Convex Quadratic Optimization

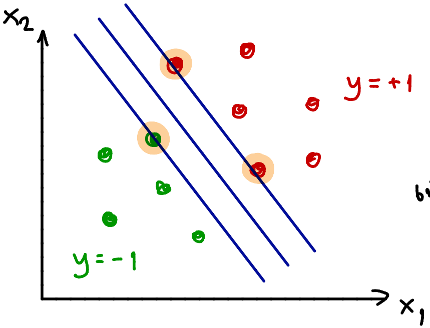
(แก้ได้ถ้า linearly separable, มี algorithm ที่แก้ได้เร็ว)

Maximal Margin Classifier

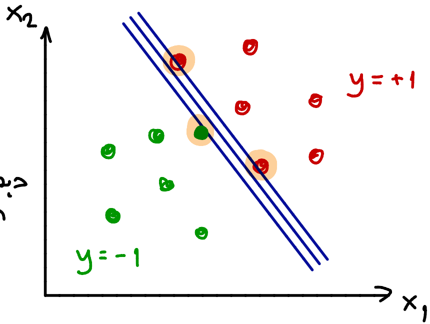
ข้อดี : ง่าย ตรงไปตรงมา

ข้อเสีย : - มักจะใช้ไม่ได้ (ไม่ linearly separable)

- Variance เยอะ



⇒ เพิ่มข้อมูล 1 จุด

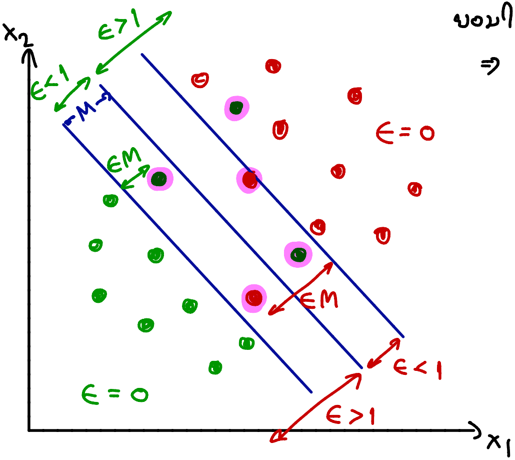


⇒ $\boxed{\text{Hard Margin}}$

เพิ่มข้อมูลนิดเดียว ⇒ model เปลี่ยนเยอะ

⇒ Variance มาก

Soft Margin Classifier



ยอมให้บางจุดล้ำ Margin

$\Rightarrow$ แต่มีราคาของการล้ำเส้น ϵ

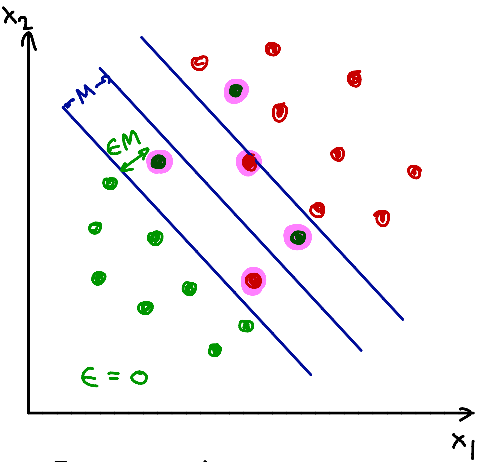
ϵ = ราคาของการล้ำ margin

$\epsilon \geq 0$ (ค่าปรับ)

C = งบรวม จ่ายค่าปรับ รวมกันไม่เกินค่านี้

$$\sum_i \epsilon_i \leq C$$

Soft Margin Classifier



หา $\beta_0, \beta_1, \beta_2, \dots, \beta_n$ ที่ทำให้ M มากที่สุด โดยที่

1. $\beta_1^2 + \beta_2^2 + \dots + \beta_n^2 = 1$

   [$\vec{\beta}$ = unit vector]

2. $y_i(\beta_0 + \vec{\beta}\cdot\vec{x}_i) \geq M(1-\epsilon_i)$

   สำหรับทุกจุดข้อมูล i

   [ไม่ล้ำ $\epsilon_i = 0$, ล้ำ $\epsilon_i > 0$]

3. $\epsilon_i \geq 0$ สำหรับทุกจุดข้อมูล i

4. $\sum \epsilon_i \leq C$

   [ค่าปรับรวมมีจำกัด $\leq C$]

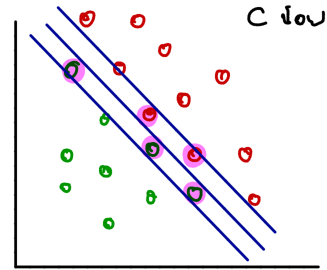
Convex Quadratic Optimization

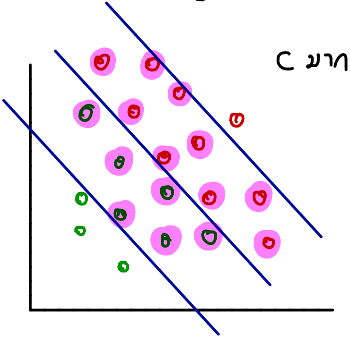
ค่า C ปรับได้
- $C = 0 \Rightarrow$ MMC
- C มาก Margin กว้าง
- C น้อย Margin แคบ

$\Rightarrow$ ข้อมูลที่ไม่ Linearly separable ยังใช้ได้ ถ้า $C \geq C_{crit}$

Soft Margin Classifier

C ตั้งค่าได้ [test data , cross validation]





C น้อย ⇒ M น้อย

⇒ จำนวน SV น้อย

⇒ Variance สูง

⇒ Bias ต่ำ

Bias-Variance Tradeoff

C มาก ⇒ M มาก

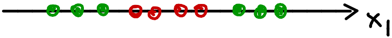
⇒ จำนวน SV มาก

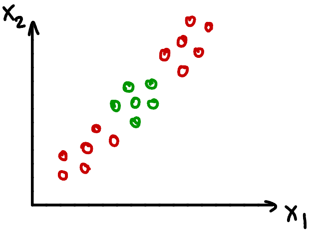
⇒ Variance ต่ำ

⇒ Bias สูง

Kernels

ทำไงกับข้อมูลแบบนี้?

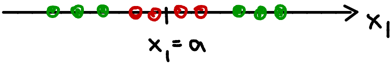




$\Rightarrow$ ใช้ Kernel

Kernels

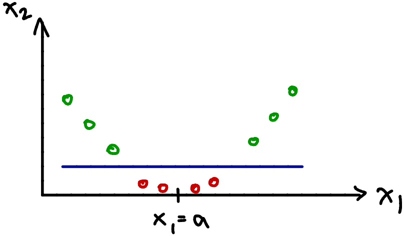
ข้อมูลเดิม



ไม่ separable ⇒ separable

เพิ่มมิติใหม่

$$x_2 = (x_1 - a)^2$$



Soft Margin (Linear) Classifier

หา $\beta_0, \vec{\beta}$ ที่ทำให้ M มากสุด
เงื่อนไข ...

Soft Margin Classifier + Kernel

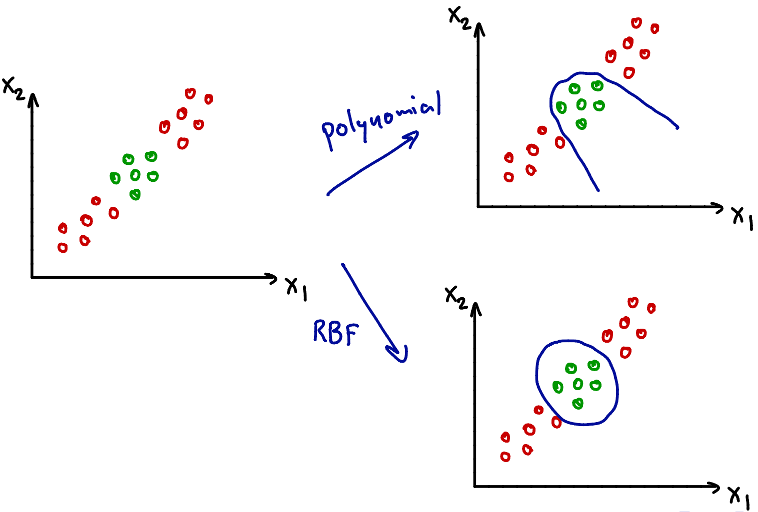
หา $\beta_0, \vec{\beta}, a$ ที่ทำให้ M มากสุด
เงื่อนไข ...

ตัวแปรของ Kernel

Kernels

Kernal ยอดนิยม

- Polynomial (ปรับ order)
- Radial Basis function (ปรับ γ)



Kernels

- Convex Quadratic Optimization
  - พิสูจน์ได้ว่ามีคำตอบ
  - มี algorithm ที่เร็ว
  - มี library พร้อมใช้
- โจทย์ optimize
  - ขึ้นกับ dot product $\vec{x}_i \cdot \vec{x}_j$ ไม่ขึ้นกับข้อมูล $\vec{x}_i$ $\Rightarrow \begin{pmatrix} \vec{x}_1\cdot\vec{x}_1 & \vec{x}_1\cdot\vec{x}_2 & \cdots \\ \vec{x}_2\cdot\vec{x}_1 & \vec{x}_2\cdot\vec{x}_2 & \vdots \\ \vdots & \cdots & \vdots \end{pmatrix}$
  - Kernel $\equiv$ นิยาม dot product ใหม่

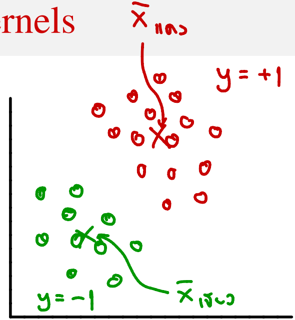
    dot product คือ "ความคล้าย" ของ $\vec{x}_i, \vec{x}_j$

    > 0, ทิศเดียวกัน

    $= 0$, ตั้งฉาก

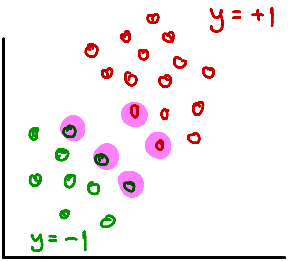
    < 0, ตรงข้ามกัน
    - Linear $K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$
    - Polynomial $K(\vec{x}_i, \vec{x}_j) = P_n(\vec{x}_i \cdot \vec{x}_j)$
    - RBF $K(\vec{x}_i, \vec{x}_j) \propto e^{-\gamma P_n(\vec{x}_i \cdot \vec{x}_j)}$

Kernels



วิธีอื่น

- โฟกัสที่คุณสมบัติ "เฉลี่ย" ของสองกลุ่ม x̄
- ใช้คุณสมบัติเฉลี่ย มาแยก 2 กลุ่ม



SVM

- โฟกัสที่ขอบ
  (จุดแดงที่ใกล้เขียว, จุดเขียวที่ใกล้แดง)
- จุดขอบ = support vector
- ใช้ SV สร้าง classifier

Kernel = ทริคแมพที่ทำให้ non-linear