Support Vector Machines

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

- Recap – Logistic Regression
- SVM
  - Large Margin Classification
  - Hard Margin vs Soft Margin
  - Kernel (non-linear classification)
  - Online SVM
  - (Optional) Math behind SVM
  - Lab

Recap: Supervised Learning

- Classification
  - Predicts class labels/categories
  - ทำนายค่าที่เป็นหมวดหมู่ = จำแนกประเภท
  - อาจมองเป็นการหา **boundary** ที่แบ่งข้อมูลในแต่ละหมวดหมู่ ออกจากกัน

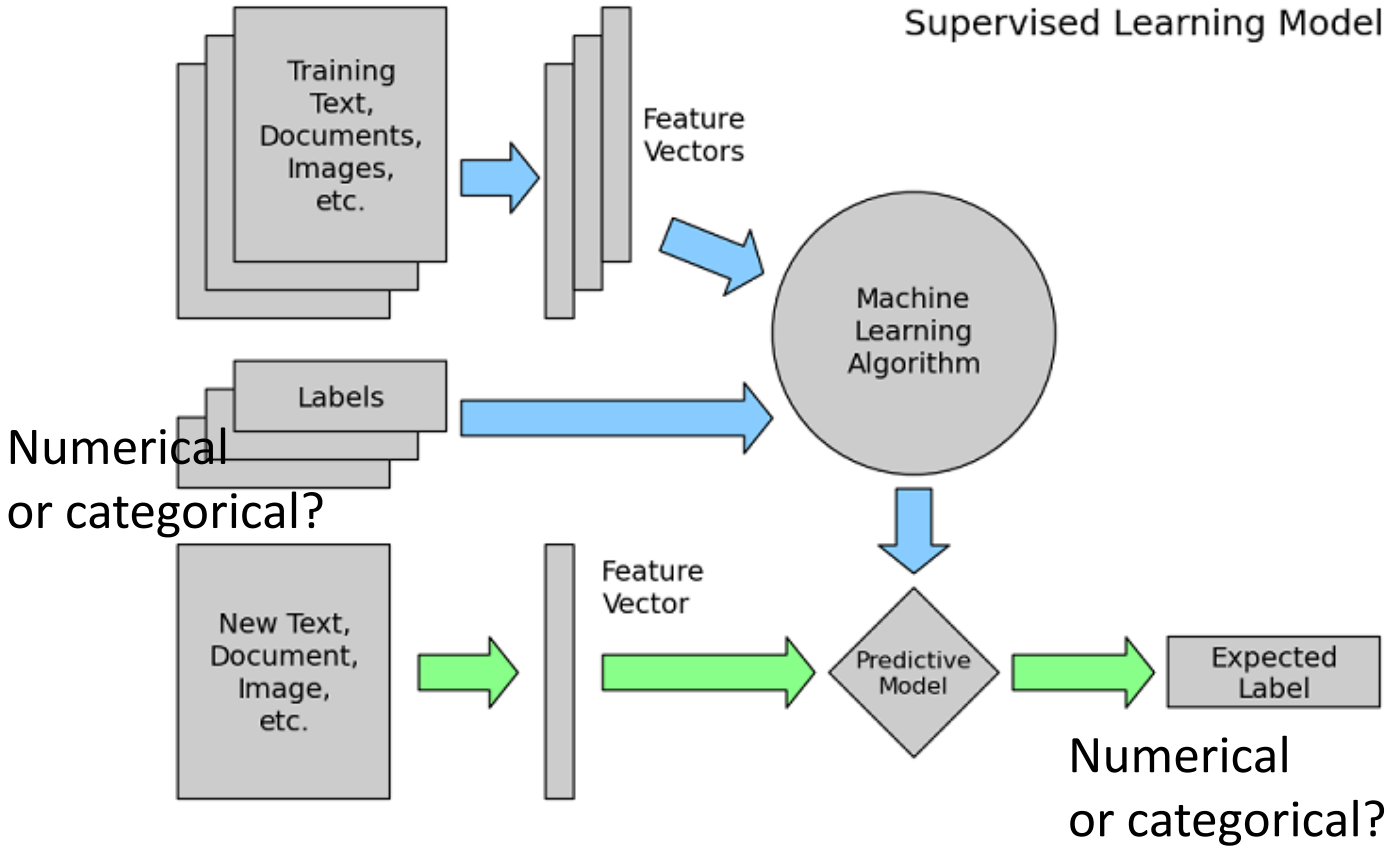
kNN

Logistic Regression

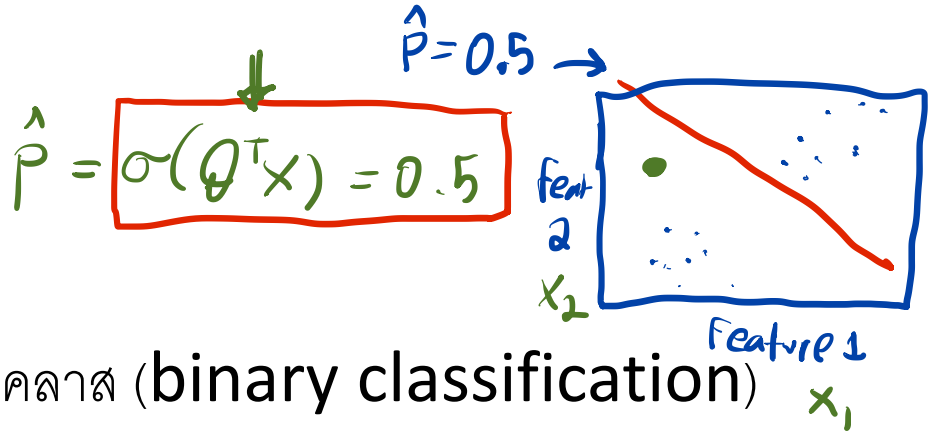
SVM

- Regression
  - Predicts continuous values
  - ทำนายค่าที่เป็นจำนวนจริง
  - อาจมองเป็นการหา **hyperplane** ที่ **fit** กับข้อมูลที่มีมากที่สุด

Linear Regression



สรุป Logistic Regression



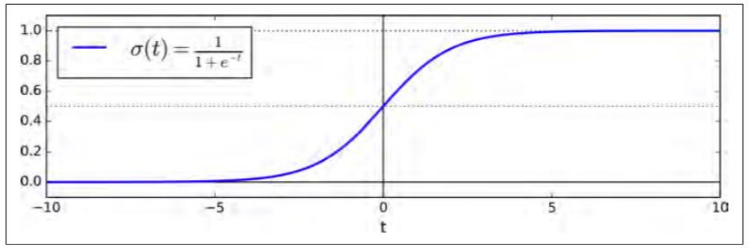
- เป็นอัลกอริทึมแบบมีผู้สอน (supervised) ใช้ในการจำแนก 2 คลาส (binary classification)
- model: $\hat{p} = h_\theta(x) = \sigma(\theta^T x)$ โดยที่ σ คือฟังก์ชัน sigmoid มีค่าในช่วง 0-1
- ถ้า $\hat{p} < 0.5$ ให้เป็นคลาส- otherwise ให้เป็นคลาส+
- cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1-y^{(i)}\right)log\left(1-\hat{p}^{(i)}\right)\right]$$

- สามารถ train ด้วย Gradient Descent ⇒ หา θ ที่ดี

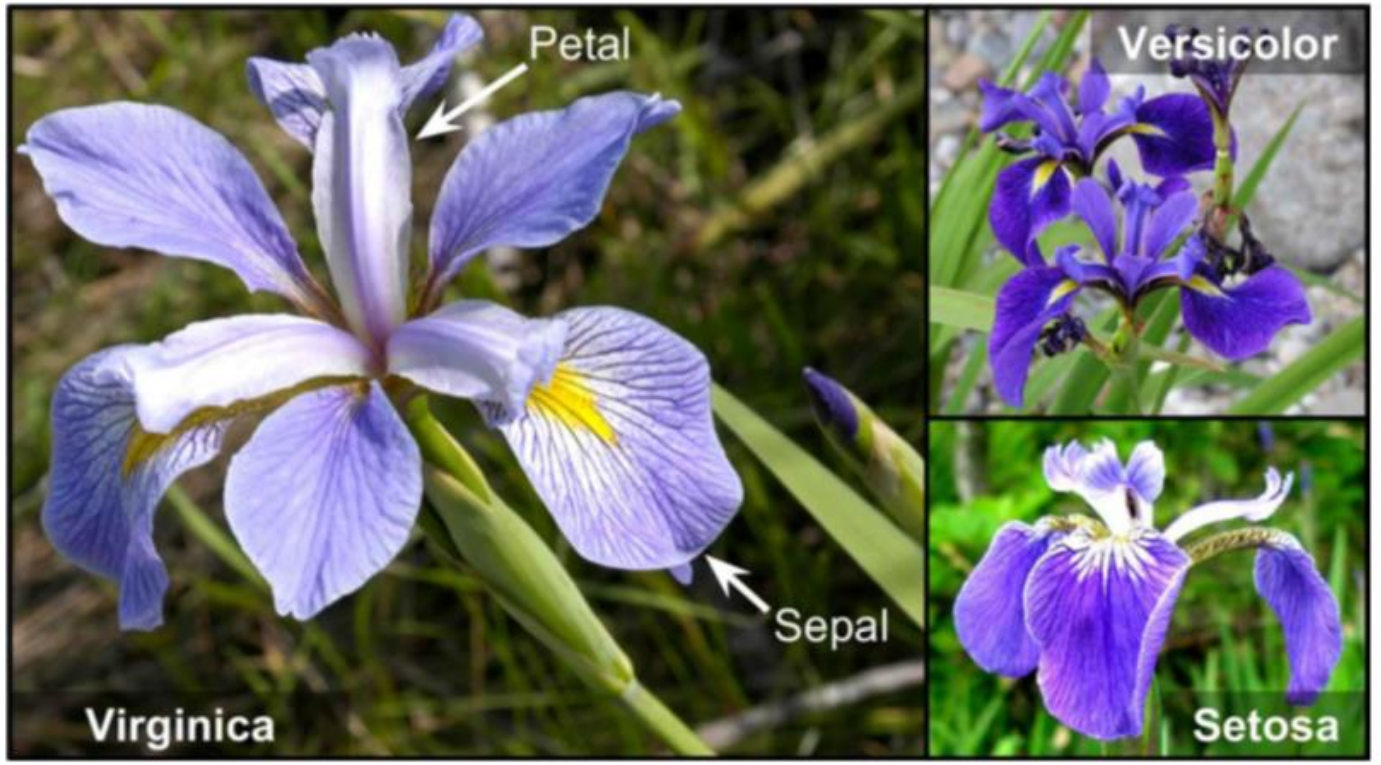
$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\theta^T\cdot\mathbf{x}^{(i)}\right)-y^{(i)}\right)x_j^{(i)}$$

$$\sigma(t) = \frac{1}{1+\exp(-t)}$$



Lab: ใช้ Logistic Regression จำแนกพันธุ์ดอกไม้

```
from sklearn.linear_model import LogisticRegression
```



เป้าหมาย: จำแนก Iris-Virginica ออกจากชนิดอื่น โดยใช้ขนาดของกลีบ Sepal/Petal - width/length

Support Vector Machines (SVM)

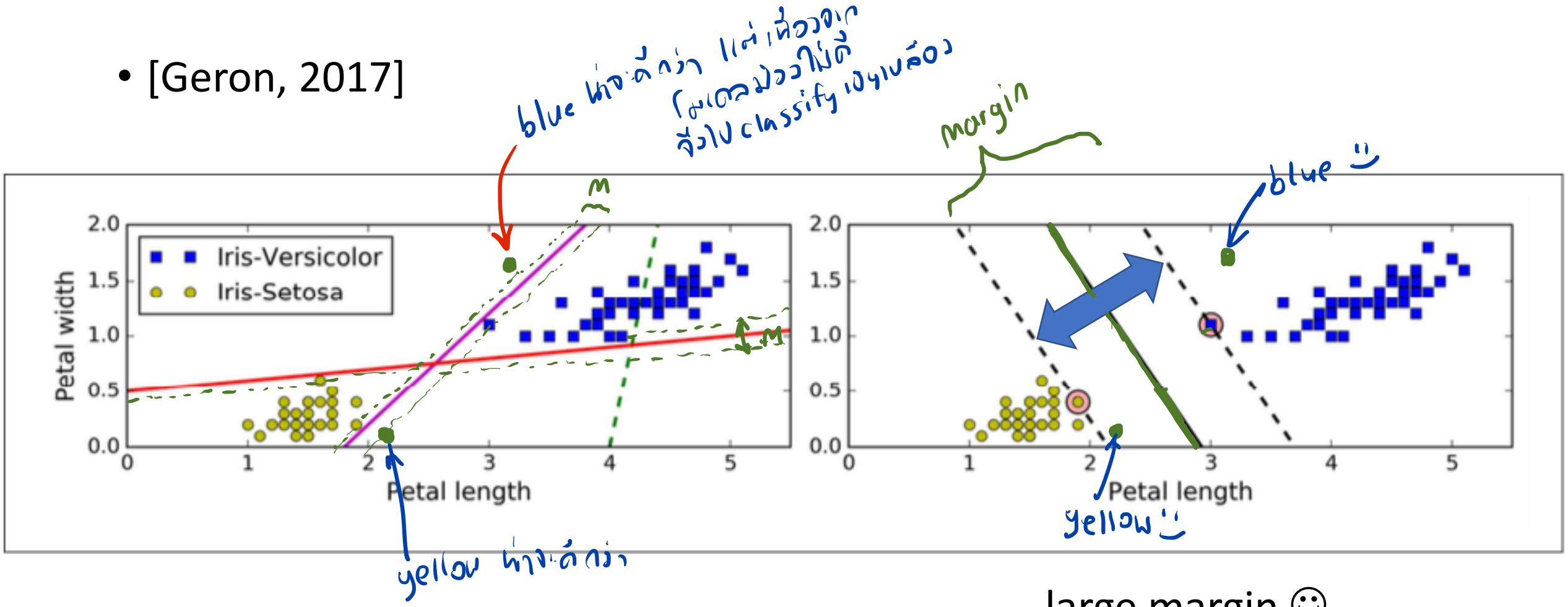
SVM

⇒OCR ⇒ 1234
⇒D "โมะ"

- เป็น supervised learning ใช้ทำ classification เป็นหลัก
- คิดค้นโดย Vapnik (1963)
  เริ่มเป็นที่รู้จักเมื่อนำมาใช้กับงาน OCR ในปี 1992 https://www.svms.org/classification/BoGV92.pdf
- เป็นที่นิยม เพราะ
  - สามารถใช้ทำได้ครอบจักรวาลทั้งงาน linear/non-linear classification หรือ regression หรือแม้กระทั่ง outlier detection
  - ใช้กับหลายงานแล้วพบว่ามีความแม่นยำสูง
  - (!) แต่ปัจจุบันความแม่นยำแพ้อัลกอพวก neural net, gradient boosting (xgboost)
- เหมาะกับ dataset ที่ฟีเจอร์เยอะ/ซับซ้อน (เช่น gene expression) แต่มีจำนวนข้อมูลน้อย - ปานกลาง
- หลักการคร่าวๆ คือหาเส้นแบ่งแดนระหว่างคลาส ให้มีขอบกั้นที่กว้างที่สุด (large margin)

SVM – Large Margin Classification

- [Geron, 2017]



large margin ☺

SVM – Large Margin Model

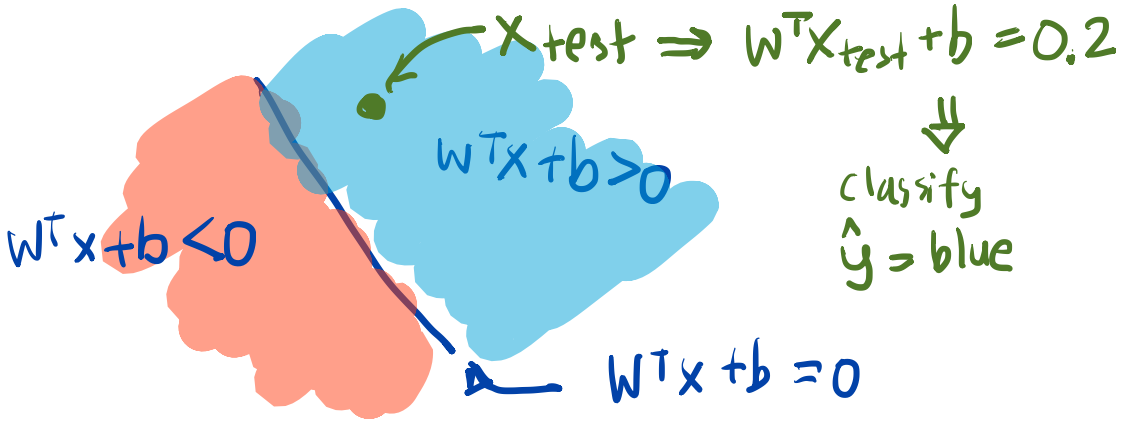
เทียบกับ log reg

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} = \sigma(\theta^T x) < 0.5 \\ 1 & \text{if } \hat{p} = \sigma(\theta^T x) > 0.5 \end{cases}$$

- โมเดล SVM แบบ linear:

- $$\hat{y} = \begin{cases} -1 & if\ w^T.x + b < 0 \\ 1 & if\ w^T.x + b \geq 0 \end{cases}$$

w คือค่า weight
เหมือน θ ใน linear regression

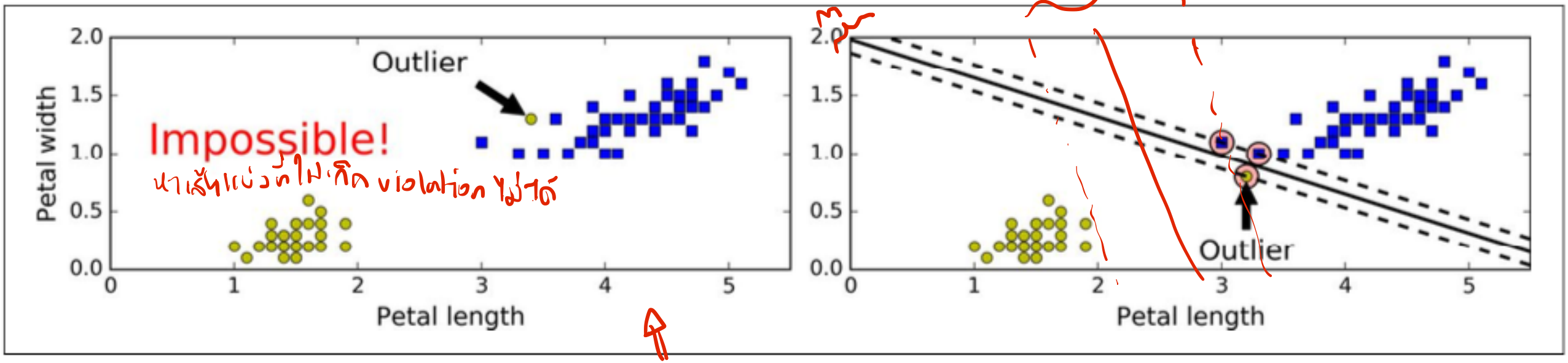


- $w^T.x + b = 0$ คือเส้นแบ่งคลาส
- เป้าหมายคือ (เริ่มต่างจาก logres)
  - อยากให้ margin กว้างที่สุด
  - แต่ห้ามกว้างเกินจนเกิด violation (ห้ามมีจุดข้อมูลอยู่ในบริเวณถนนแบ่งแดน)

(constraint)

SVM – Hard Margin

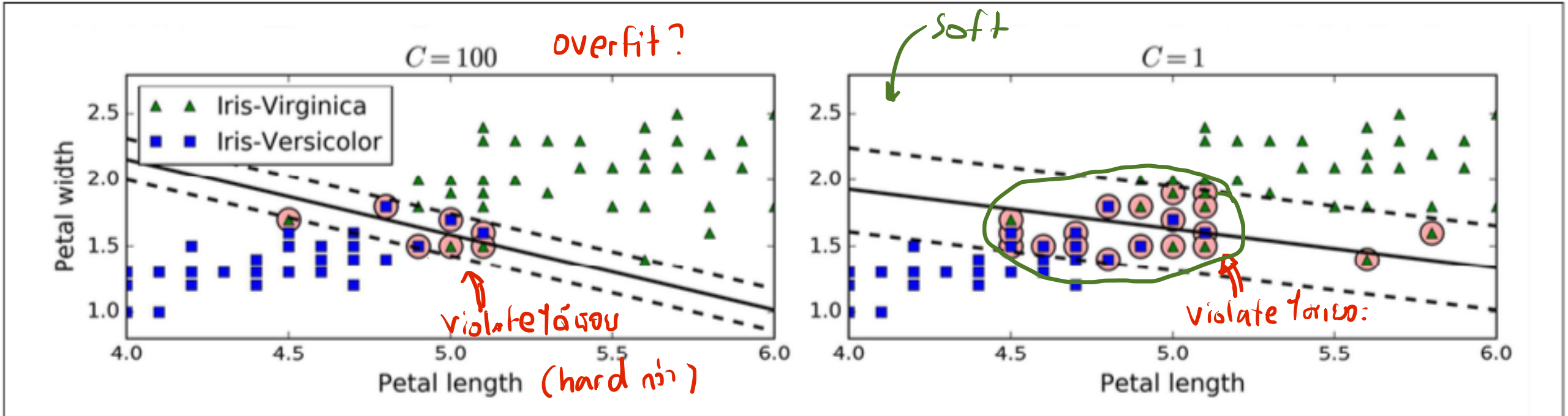
- หากใช้เส้นแบ่งแดนแบบ **hard margin** คือห้ามมีจุดข้อมูลอยู่ในถนนแบ่งแดน จะเกิดปัญหา
  - **outlier** อาจทำให้หาเส้นแบ่งไม่ได้เลย (ภาพซ้าย)
  - **outlier** อาจทำให้เส้นแบ่งเคลื่อนไปจากที่ควรมาก (ภาพขวา)



- ดังนั้นจึงควรใช้ **soft margin**

SVM – Soft Margin Classification

- ใช้ **soft margin** คือยอมให้มีบางจุดเข้ามาอยู่ในถนน (**violate margin**) เพื่อเปิดถนนให้กว้างขึ้น
- ค่า C คือ **hyperparameter** ที่บอกถึงความ **hard** ของ **margin**
  - C เยอะ = **violate** ได้น้อย ถนนแคบ
  - C น้อย = **violate** ได้เยอะ ถนนกว้าง



การ implement SVM ด้วย Scikit-learn

```
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

X = …                          # feature: petal length/width
y = …                          # label: iris-virginica
svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge"))
])
svm_clf.fit(X, y)
```

! ปรับ scale ของข้อมูล

SVM

การ implement SVM ด้วย Scikit-learn

```
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

X = …                          # feature: petal length/width
y = …                          # label: iris-virginica
svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge"))
])
svm_clf.fit(X, y)
```

???

ก่อนทำ SVM ควร normalize feature ด้วย scaler

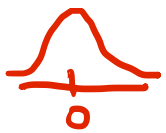
```
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

X = …                          # feature: petal length/width
y = …                          # label: iris-virginica
svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge"))
])
svm_clf.fit(X, y)
```

1000 – 10000

⇓

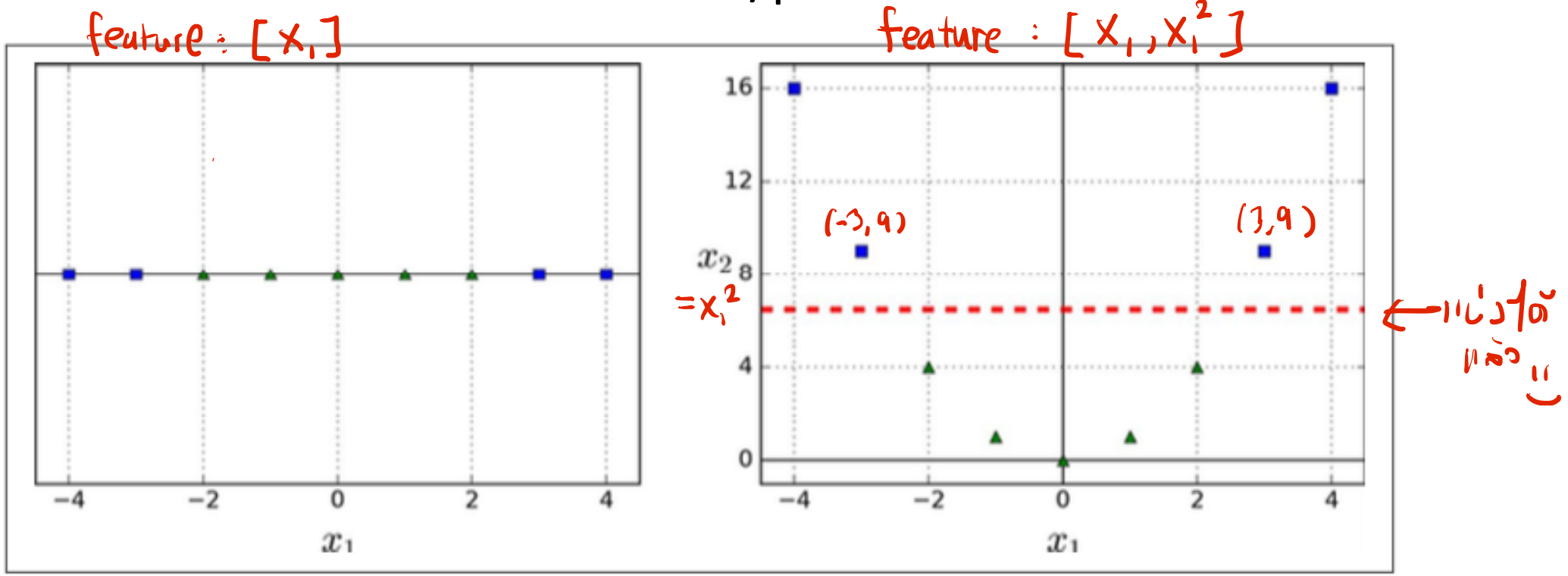
-3 – 3



scaler จะทำการ normalize ทุก feature ให้มี mean=0, var=1 เพื่อไม่ให้ feature ที่มีค่าใหญ่ส่งอิทธิพลต่อเส้นแดนมากกว่า feature ที่มีค่าเล็ก

SVM – Non-linear classification

- หากต้องการ **classify** แบบ **non-linear** (เส้นแบ่งแดนโค้งได้) สามารถเพิ่ม **feature** ที่เป็น **polynomial** กำลังสูง ๆ ได้ เช่นเดียวกับการทำ **polynomial regression** ใน **lecture** ที่แล้ว
- (!) แต่ไม่ควร เพราะ **feature** ที่เยอะจะทำให้ **train/predict** ช้าลงมาก



SVM – Kernels

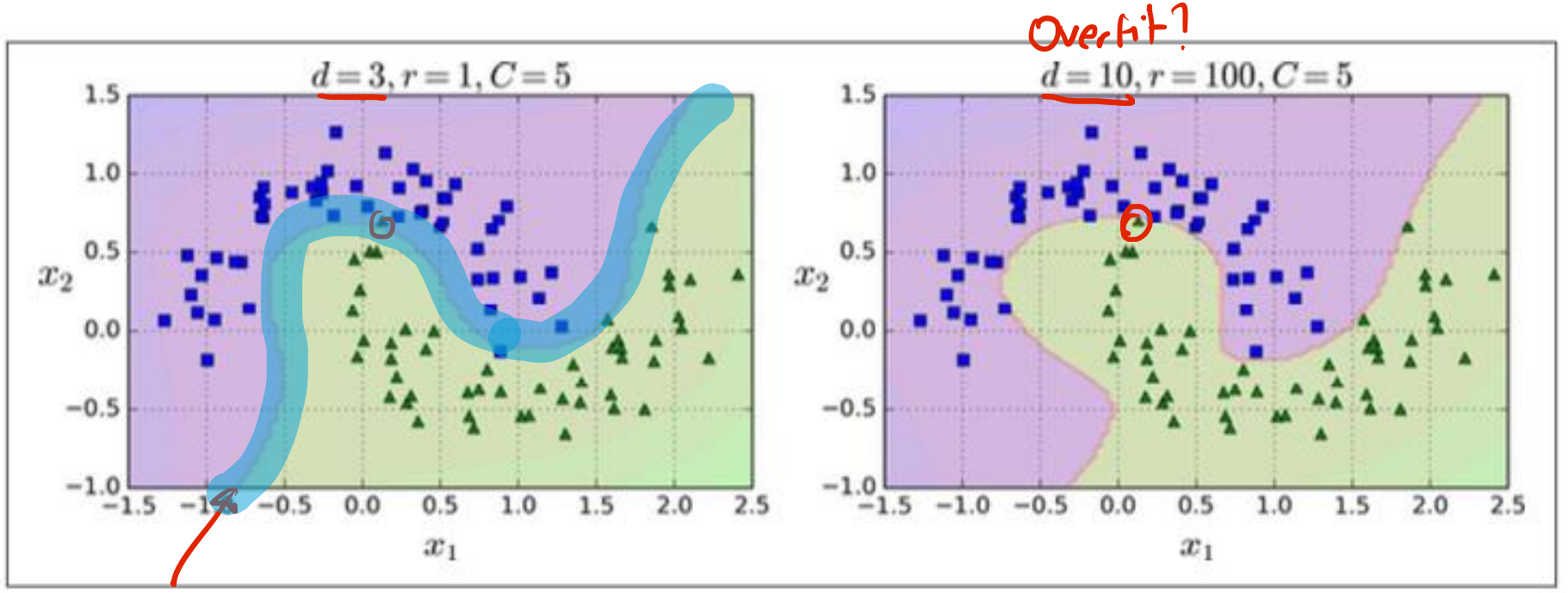
- วิธีที่ดีกว่าคือใช้ "Kernel Trick" เป็นการพลิกแพลงทางคณิตศาสตร์ทำให้สามารถทำ non-linear classification โดยไม่ต้องเพิ่มจำนวน feature เลยได้

!!

```
from sklearn.svm import SVC
svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svc", SVC(kernel="poly", degree=3, coef0=0, C=5))
])
svm_clf.fit(X, y)
```

SVM poly kernel – ค่า degree และ coef0

- hyperparameter ของ poly kernel มี
  - degree คือค่ากำลังของ polynomial (ในรูปกำลัง 3 vs กำลัง 10)
  - coef0 คือค่าที่ทำให้ poly กำลังสูงหรือต่ำส่งผลกับ model มากกว่ากัน **ปกติให้เป็น 0**



สามารถใช้ grid search ในการหา hyperparam ที่ดีที่สุด

Commonly used Kernel functions

Kernal ที่ใช้นิยมใช้:

SVC(kernel = "____")	ฟังก์ชัน **Kernel** ที่บอกถึงความคล้ายคลึง (**Similarity**) ระหว่าง **a** กับ **b**
linear	$K(a,b) = a^T.b$
poly	$K(a,b) = (\gamma a^T.b + r)^d$
rbf	$K(a,b) = \exp(-\gamma\|a-b\|^2)$ (Gaussian Radial Basis Function)
sigmoid	$K(a,b) = \tanh(\gamma a^T.b + r)$

← most popular

อ่านเพิ่มเติมเกี่ยวกับ SVM

- http://pyml.sourceforge.net/doc/howto.pdf (อธิบายดีมาก แนะนำ)
- https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (เชิง practical)
  - แนะนำให้เริ่มจาก scale ข้อมูล
  - ลองใช้ RBF kernel* ก่อน (ถ้าขนาด feature ไม่ใหญ่เว่อร์)
  - ทำ cross-validate เพื่อหาค่า c, gamma ที่ดีที่สุด
  - ถ้าข้อมูลเป็นแบบ categorical (เช่น sunny, cloudy, rainy) ให้เข้ารหัสแบบ one-hot (เช่น (0,0,1), (0,1,0),(1,0,0))
- http://scikit-learn.org/stable/modules/svm.html (ประกอบการใช้งาน library)
- http://cs229.stanford.edu/notes/cs229-notes3.pdf (อ่านยากหน่อย แต่ math ละเอียด)

*เนื่องจาก RBF สามารถจำลองความซับซ้อนได้อย่างไม่มีขีดจำกัด -- อ่านเพิ่มเติม: Learning Theory, VC dimension

Math behind SVM (ไม่ออกสอบ)

- โมเดล SVM แบบ linear:

$$\hat{y} = \begin{cases} -1 & if\ w^T.x + b < 0 \\ 1 & if\ w^T.x + b \geq 0 \end{cases}$$

w คือค่า weight
เหมือนกับ θ ใน linear regression

- เป้าหมายคือ
  - อยากให้ margin กว้างที่สุด
  - แต่ห้ามกว้างเกินจนเกิด violation ← constraint ทำให้ซับซ้อน ใช้ gradient ตรงๆ ไม่ได้

- เราสามารถเขียนเป้าหมายนี้ในรูปปัญหา optimization ได้

Math behind SVM (ไม่ออกสอบ)

- Hard Margin SVM ในรูปปัญหา optimization
- objective:

$$\begin{aligned} \underset{w,\,b}{\text{minimize}} \quad & \frac{1}{2} w^T . w \\ \text{subject to} \quad & y^{(i)}\left(w^T . x^{(i)} + b\right) \geq 1 \quad \text{for } i = 1, 2, \ldots, m \end{aligned}$$

Math behind SVM (ไม่ออกสอบ)

- Hard Margin SVM ในรูปปัญหา optimization
- objective:

$$\underset{w,\,b}{\text{minimize}} \quad \frac{1}{2} w^T . w$$

$$\text{subject to} \quad y^{(i)}\left(w^T . x^{(i)} + b\right) \geq 1 \quad \text{for } i = 1, 2, \ldots, m$$

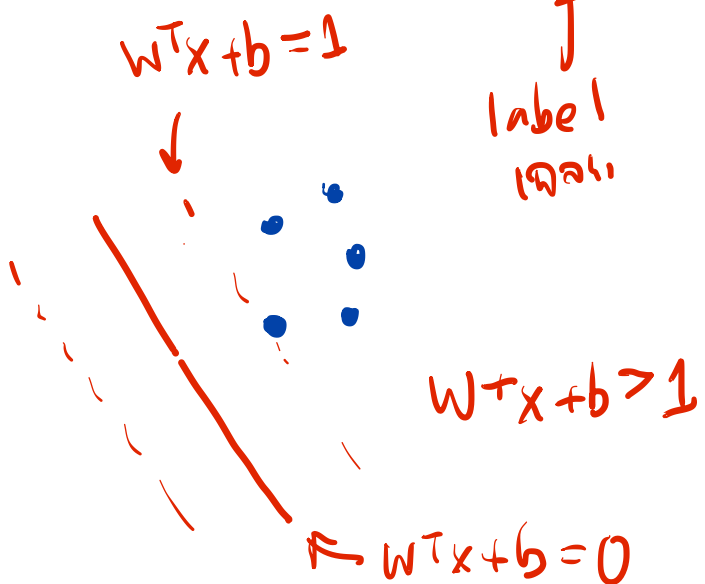
$\frac{1}{2}\|w\|^2$ ยิ่งน้อย margin ยิ่งกว้าง

ทำให้ y กับ $\hat{y}$ มีเครื่องหมายตรงกัน และห่างออกนอกขอบ margin

label เฉลย

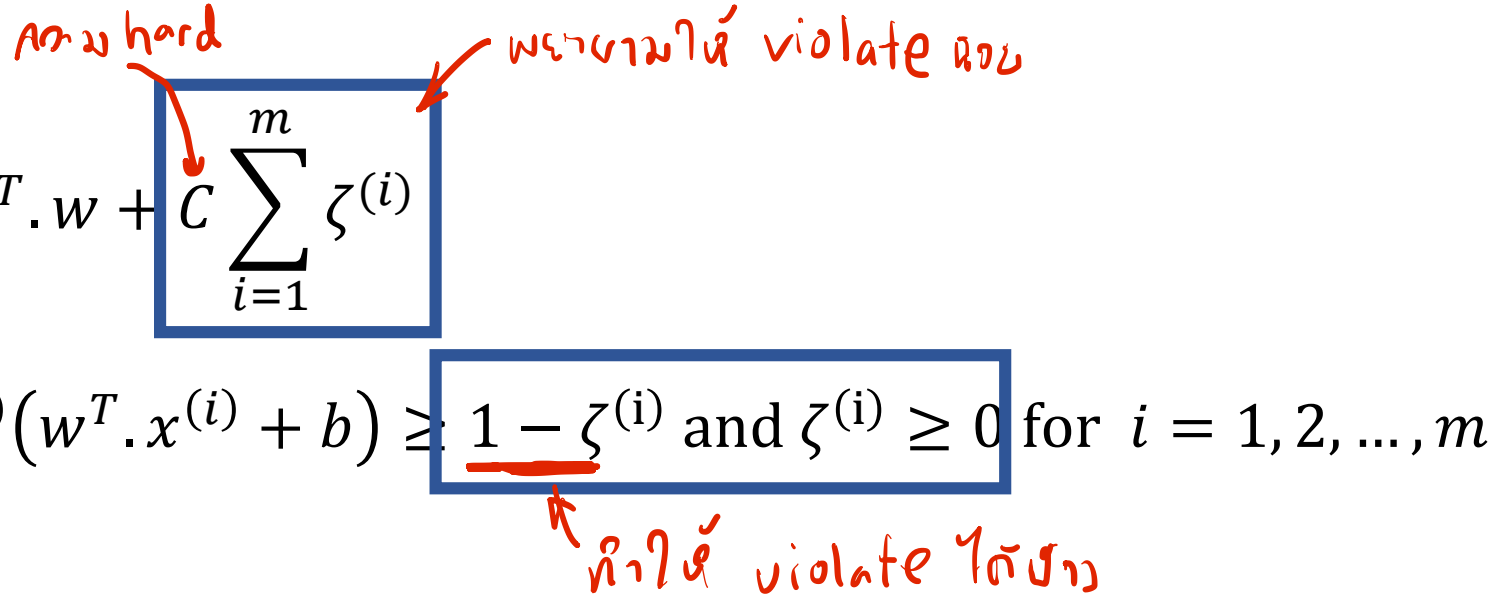
ค่าที่ model ทำนาย ($\hat{y}$)

เงื่อนไข: ห้ามเกิด violation



Math behind SVM (ไม่ออกสอบ)

- หากต้องการ **Soft Margin** จะต้องเพิ่ม **slack term**
- **new objective:**



$$\underset{w,\,b}{\text{minimize}} \quad \frac{1}{2} w^T . w + C \sum_{i=1}^{m} \zeta^{(i)}$$

$$\text{subject to} \quad y^{(i)}\left(w^T . x^{(i)} + b\right) \geq 1 - \zeta^{(i)} \text{ and } \zeta^{(i)} \geq 0 \text{ for } i = 1, 2, \ldots, m$$

- สังเกตว่าเกิด **hyperparameter C**

Math behind SVM (ไม่ออกสอบ)

- Soft Margin SVM objective (primal form):

$$\underset{w,\,b}{\text{minimize}} \quad \frac{1}{2} w^T . w + C \sum_{i=1}^{m} \zeta^{(i)}$$

quadratic

$$\text{subject to} \quad y^{(i)}\left(w^T . x^{(i)} + b\right) \geq 1 - \zeta^{(i)} \text{ and } \zeta^{(i)} \geq 0 \text{ for } i = 1, 2, \ldots, m$$

linear constraints

ปัญหานี้เป็นที่รู้จักในนาม **Quadratic Programming**
สามารถแปลงให้อยู่ในรูป **dual form** แล้วใช้
QP Solver สำเร็จรูปหาคำตอบได้อย่างรวดเร็ว ☺

Deriving SVM math:
Geometric Margin, Primal/Dual form, KKT

(whiteboard)

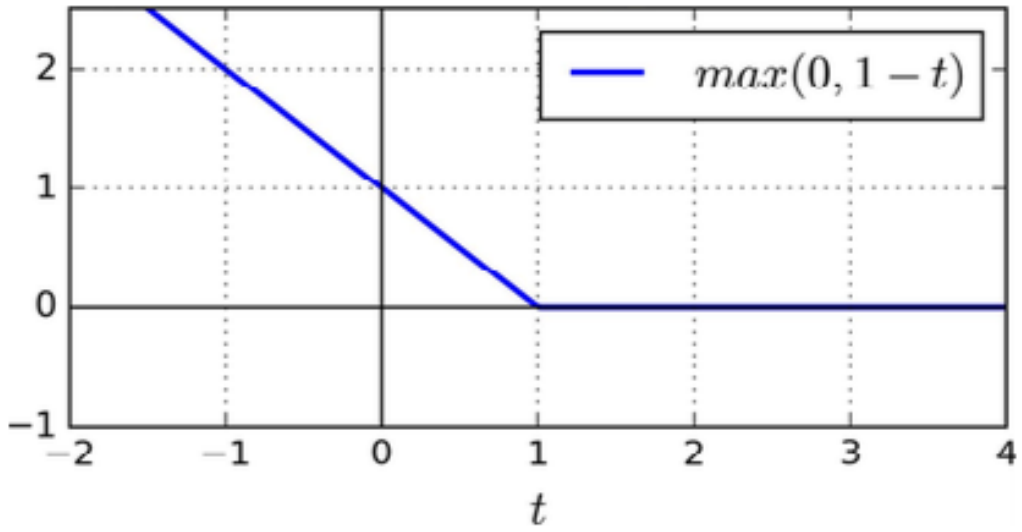
ควรรู้มาบ้าง: Dot product, Vector Projection, Lagrange Multiplier, Partial Derivative

Online SVM

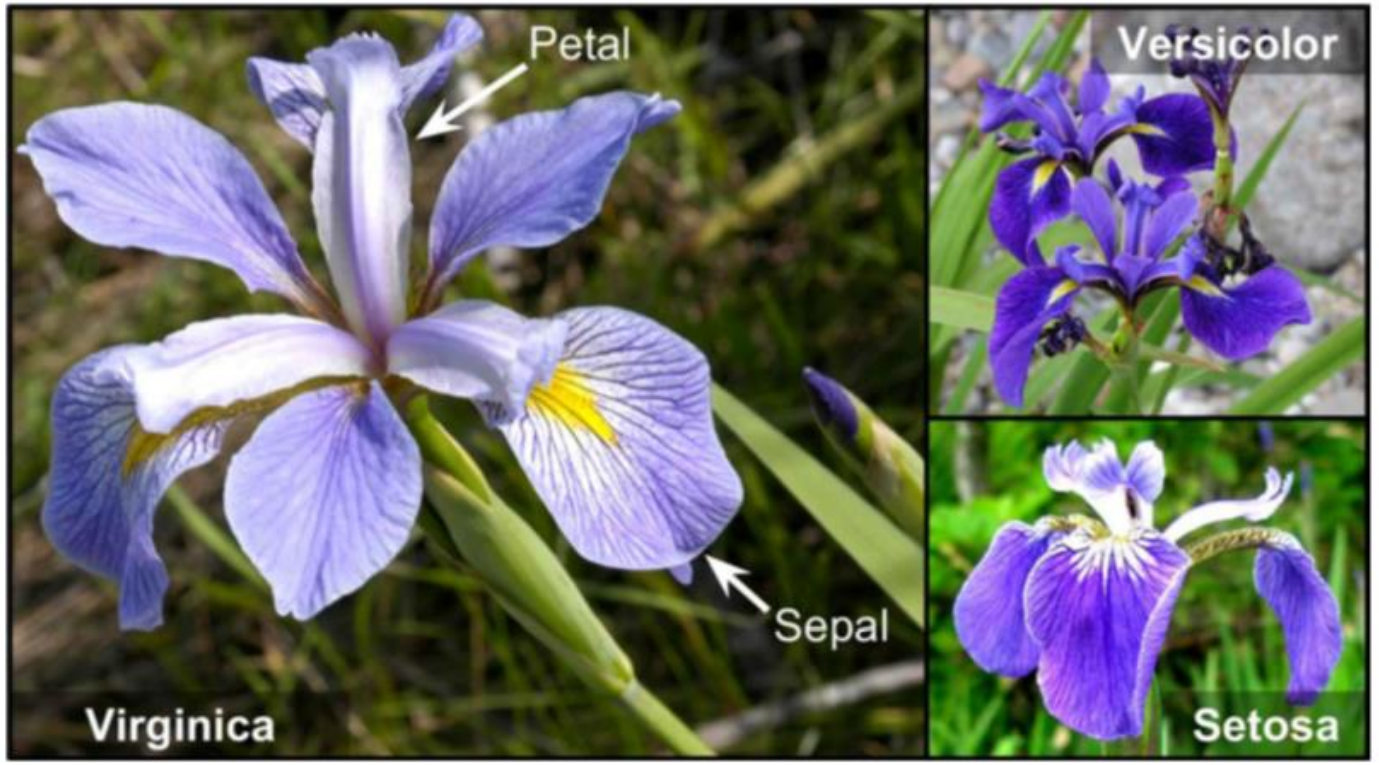
- เราสามารถใช้ **Gradient Descent** ในการแก้ **SVM** ได้ด้วย
- โดยคำนวณ **gradient** ของ **cost function** นี้

$$J(w,b) = \frac{1}{2} w^T . w + C \sum_{i=1}^{m} \max(0, 1 - t^{(i)}(w^T . x^{(i)} + b))$$

Hinge Loss



Lab: ใช้ SVM จำแนกพันธุ์ดอกไม้



เป้าหมาย: จำแนก **Iris-Virginica** ออกจากชนิดอื่น โดยใช้ขนาดของกลีบ **Sepal/Petal - width/length**

Midterm

- AI
  - Strong vs Weak
  - Turing Test
- Machine Learning
  - เป้าหมายของ Machine learning
  - กระบวนการ train, test/predict/inference
  - supervised vs unsupervised, classification vs regression
  - คำศัพท์: feature, class, label, model
- Model Selection
  - evaluation metrics: error, accuracy, FP/FN/TP/TN, precision, recall, F1, confusion matrix
  - bias-variance tradeoff, overfitting problem
  - cross-validation
- Supervised Learning Algorithms
  - kNN (ข้อเสียคืออะไร, เปลี่ยนค่า k ส่งผลกับ boundary อย่างไร) ← ไม่มี model ⇒ เทรนไว แต่ inference ช้า !!
  - Linear Regression (วิธี gradient descent ดีกว่า normal eq อย่างไร, regularize ทำเพื่ออะไร)
    - สมการ model, MSE cost function, normal equation, gradient descent, learning rate, regularization
  - Logistic Regression
    - sigmoid function
  - SVM
    - แนวคิด large margin, soft vs hard margin, kernel function

มี model