Support Vector Machines

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

- Recap – Gradient Descent, Regularization, Logistic Regression
- SVM
- Midterm

Recap: Supervised Learning

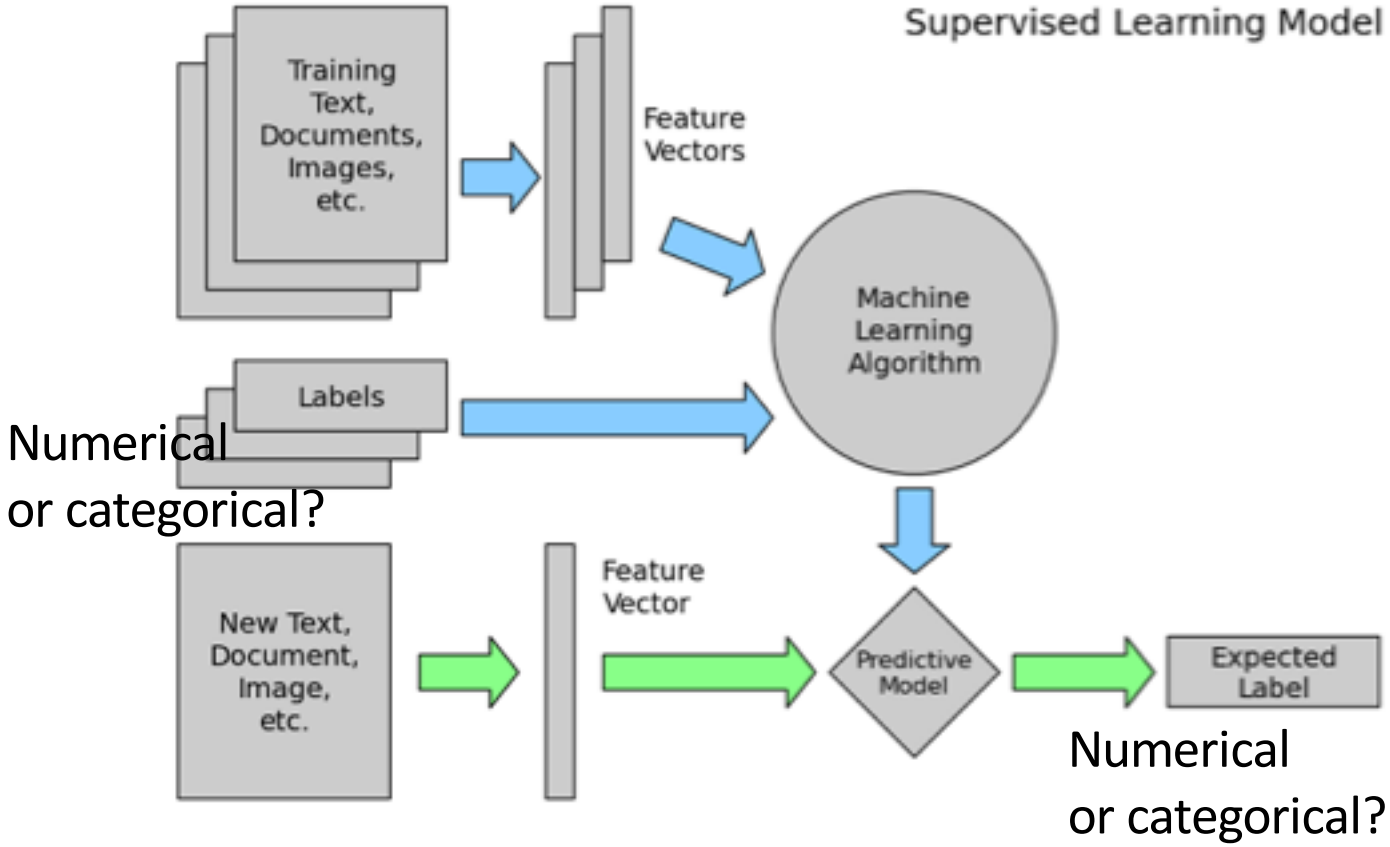
- Classification
  - Predicts class labels/categories
  - ทำนายค่าที่เป็นหมวดหมู่ = จำแนกประเภท
  - อาจมองเป็นการหา **boundary** ที่แบ่งข้อมูลในแต่ละหมวดหมู่ ออกจากกัน
- Regression
  - Predicts continuous values
  - ทำนายค่าที่เป็นจำนวนจริง
  - อาจมองเป็นการหา **hyperplane** ที่ **fit** กับข้อมูลที่มีมากที่สุด

kNN

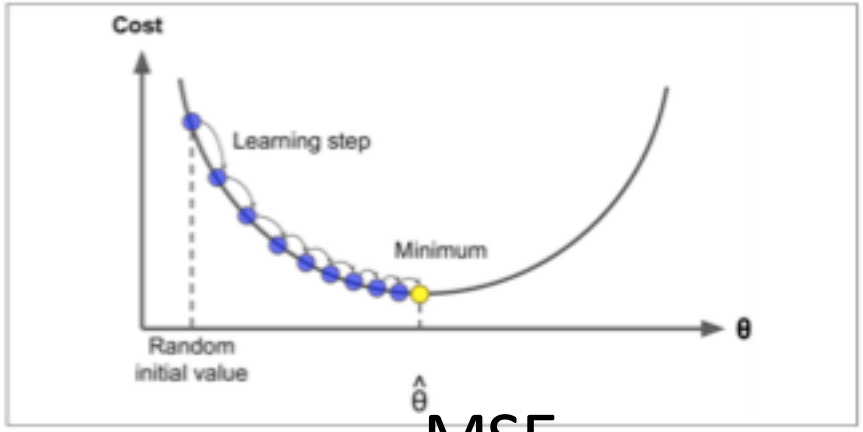
Logistic Regression

SVM

Linear Regression



Gradient Descent



ภูเขา MSE

- เป้าหมาย: minimize $J(\theta) = MSE(\theta)$
- อัลกอริทึม: วนทำสมการนี้ซ้ำไปเรื่อยๆ จนกว่า MSE จะเล็กมากพอ

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \text{MSE}(\theta)$$

learning rate

gradient ของ MSE(θ) w.r.t. θ

$$\nabla_\theta \text{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\theta) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Regularization

- แก้ปัญหา **overfitting** โดยเพิ่มพจน์ **regularization** ใน **cost function**

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Ridge (L2) regularization

- ช่วยลดอิทธิพลของ **polynomial degree** สูงๆ อย่าง x^4, x^5 = ลด **overfitting**

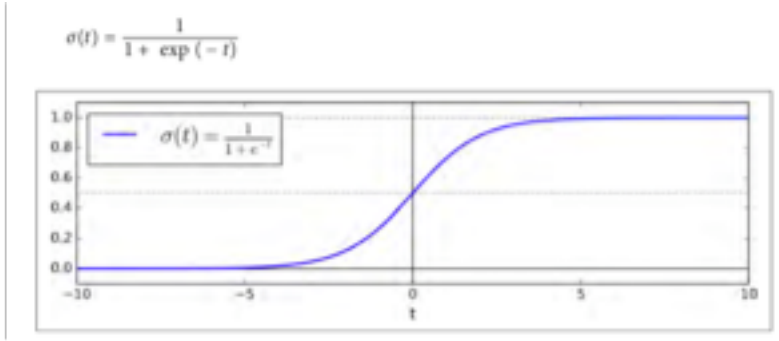
สรุป Logistic Regression

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

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

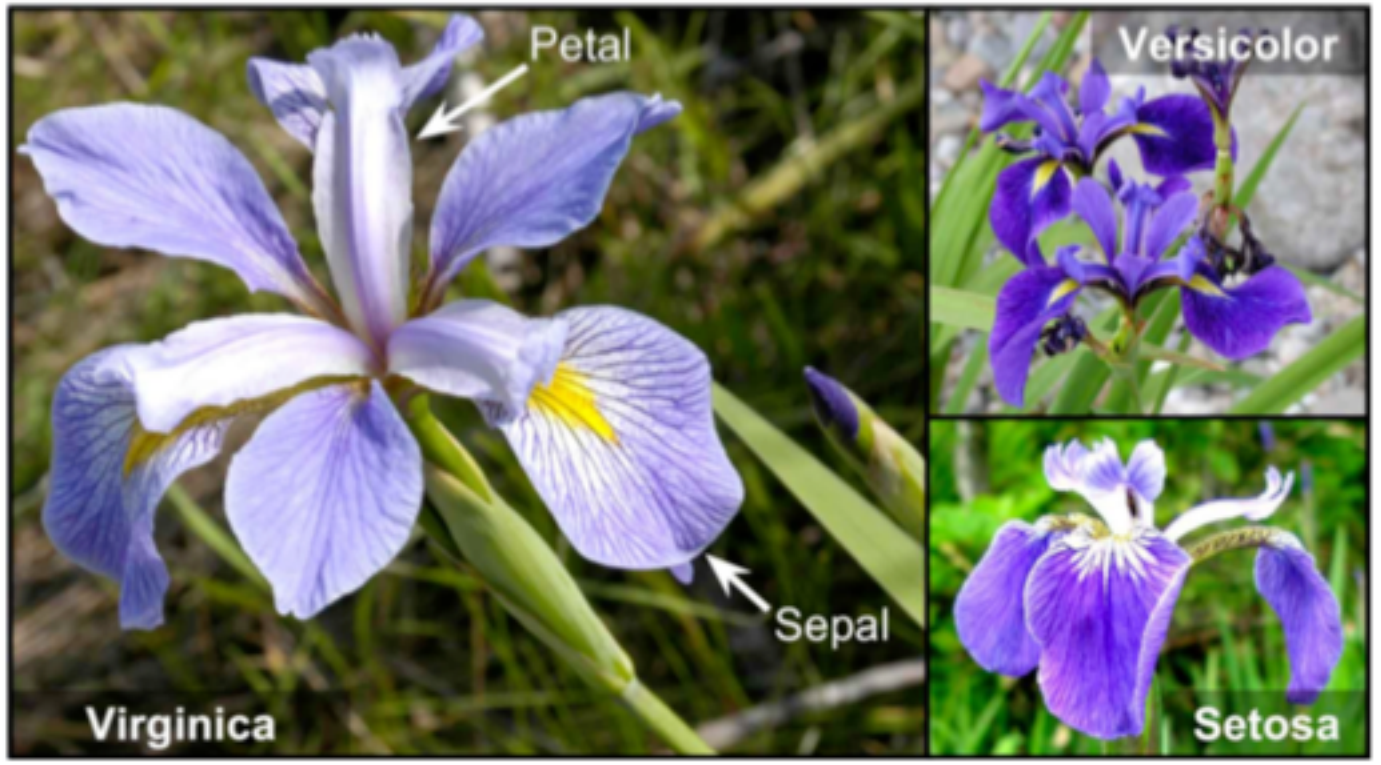
- สามารถ train ด้วย Gradient Descent

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



Lab: ใช้ Logistic Regression จำแนกพันธุ์ดอกไม้

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



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

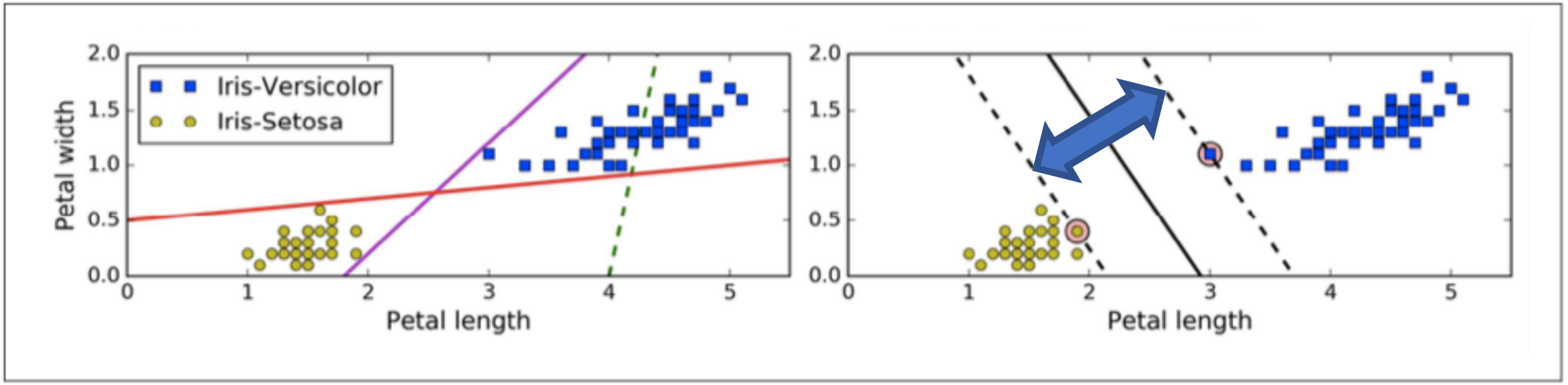
Support Vector Machines (SVM)

SVM

- เป็น supervised learning ใช้ทำ classification เป็นหลัก
- เป็นที่นิยมมาก เพราะ
  - สามารถใช้ทำได้ครอบจักรวาลทั้งงาน linear/non-linear classification หรือ regression หรือแม้กระทั่ง outlier detection
  - ใช้กับหลายงานแล้วพบว่ามีความแม่นยำสูง
  - (!) แต่ปัจจุบันแพ้อัลกอพวก neural net, gradient boosting (xgboost)
- เหมาะกับ dataset ที่ฟีเจอร์เยอะ แต่มีขนาดเล็ก - ปานกลาง
- หลักการคร่าวๆ คือหาเส้นแบ่งแดนระหว่างคลาส ให้มีขอบกั้นที่กว้างที่สุด (large margin)

SVM – Large Margin Classification

- [Geron, 2017]



large margin ☺

SVM – Large Margin Model

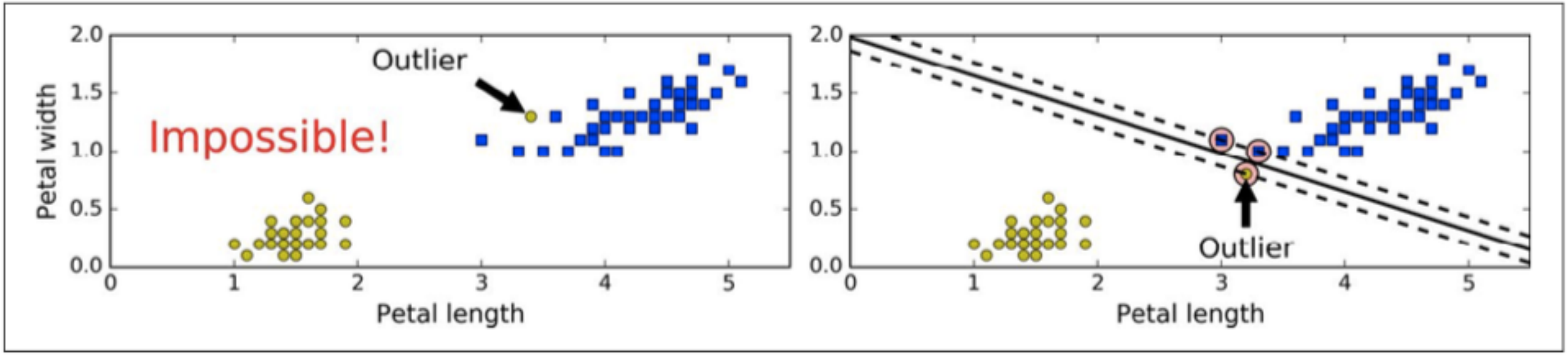
- โมเดล SVM แบบ linear:
- $\hat{y} = \begin{cases} 0 & if\ w^T.x + b < 0 \\ 1 & if\ w^T.x + b \geq 0 \end{cases}$

w คือค่า weight
เหมือนกับ θ ใน linear regression

- $w^T.x + b$ คือเส้นแบ่งคลาส
- เป้าหมายคือ
  - อยากให้ margin กว้างที่สุด
  - แต่ห้ามกว้างเกินจนเกิด violation

SVM – Hard Margin

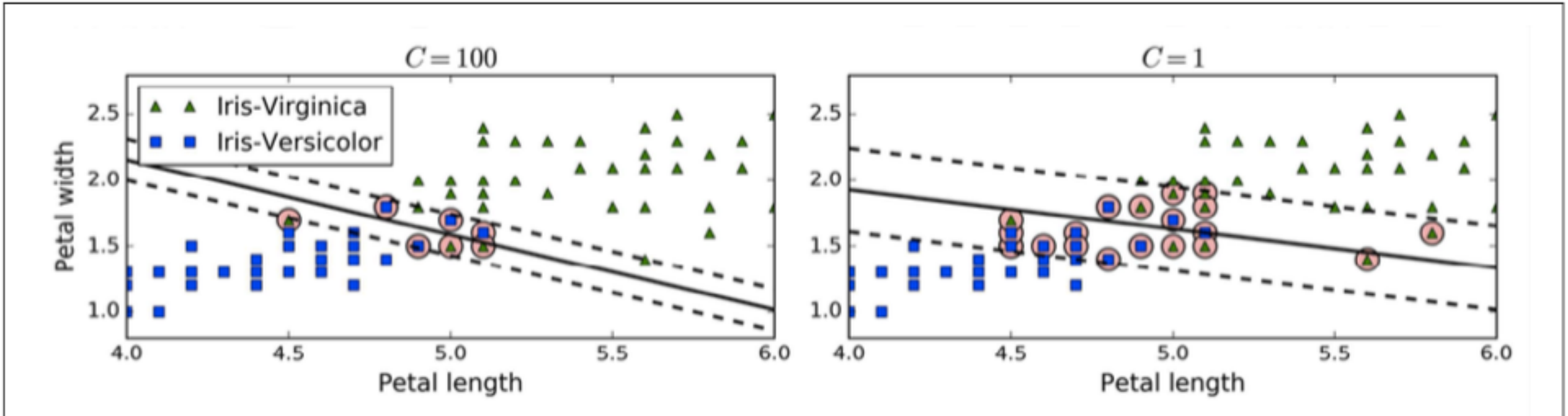
- หากใช้เส้นแบ่งแดนแบบ **hard margin** คือห้ามมีจุดข้อมูลอยู่ในถนนแบ่งแดน จะเกิดปัญหา
  - **outlier** อาจทำให้หาเส้นแบ่งไม่ได้เลย
  - **outlier** อาจทำให้เส้นแบ่งเคลื่อนไปจากที่ควรมาก



- 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.pipline 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)
```

การ implement SVM ด้วย Scikit-learn

```
from sklearn.svm import LinearSVC
from sklearn.pipline 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

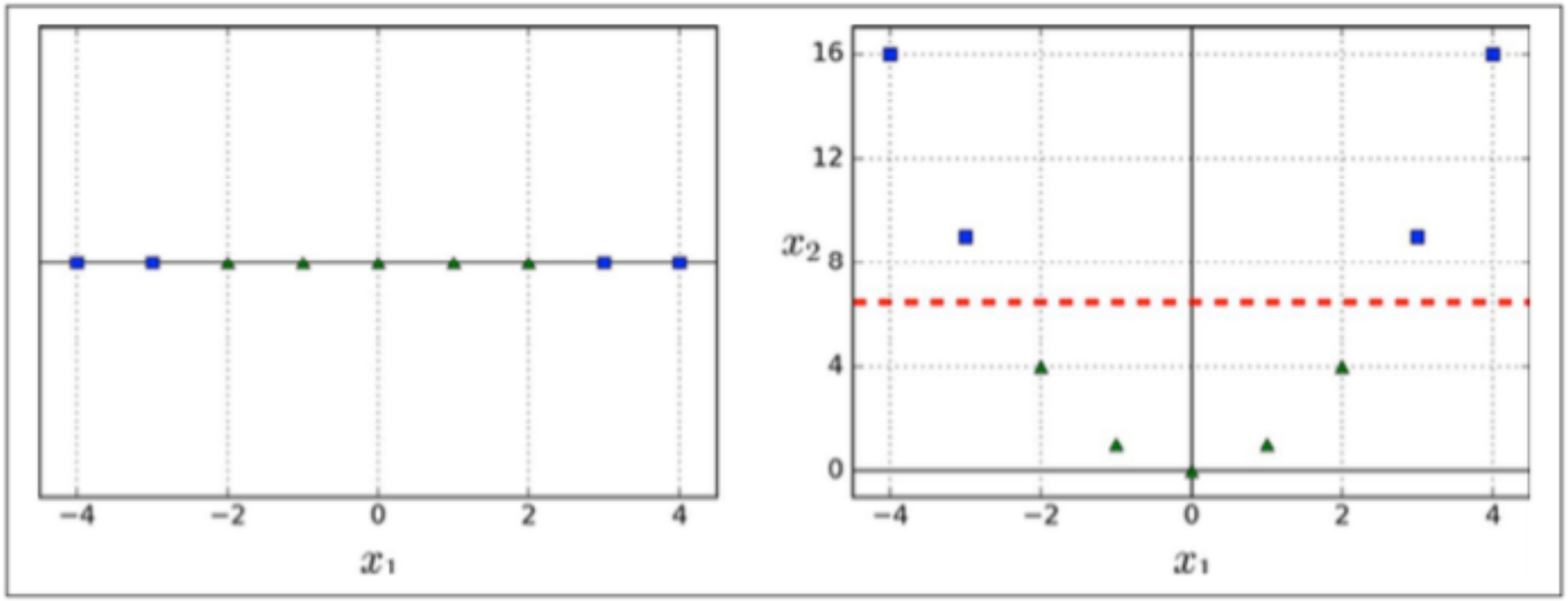
```
from sklearn.svm import LinearSVC
from sklearn.pipline import Pipeline
from sklearn.preprocessing import StandardScaler

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)
```

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

SVM – Non-linear classification

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



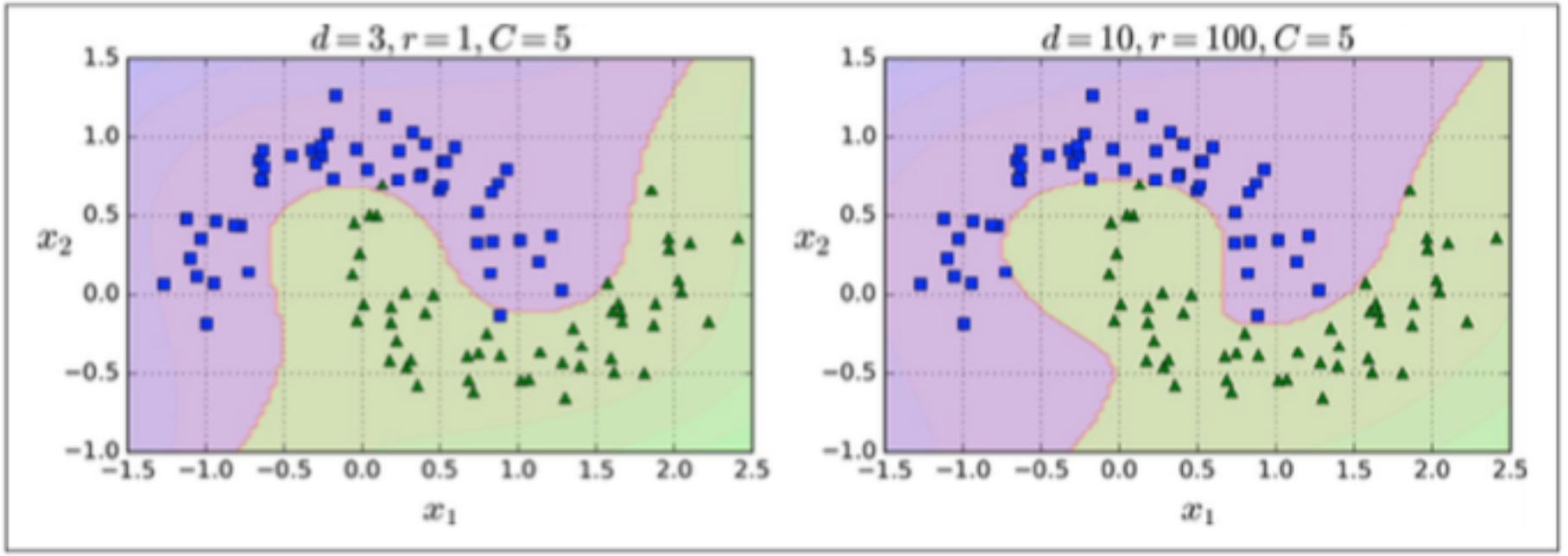
SVM – Kernels

- วิธีที่ดีกว่าคือใช้ "Kernel Trick" เป็นการพลิกแพลงทางคณิตศาสตร์ทำให้สามารถ classify non-linear โดยไม่ต้องเพิ่ม 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

อ่านเพิ่มเติมเกี่ยวกับ SVM

- http://pyml.sourceforge.net/doc/howto.pdf
- http://scikit-learn.org/stable/modules/svm.html

Kernal ที่ใช้นิยมใช้

SVC(kernel = "____")	สมการ Kernel ที่บอกถึงความคล้ายคลึง (Similarity)
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	$K(a,b) = \tanh(\gamma a^T.b + r)$

Math behind SVM (ไม่ต้องจำ)

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

w คือค่า weight
เหมือนกับ θ ใน linear regression

- เป้าหมายคือ
  - อยากให้ margin กว้างที่สุด
  - แต่ห้ามกว้างเกินจนเกิด violation

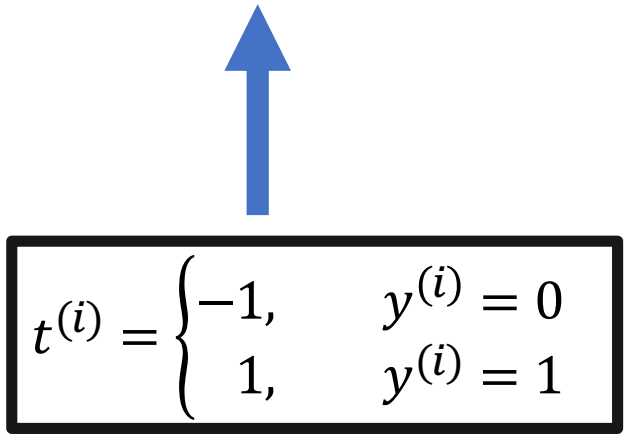
- เราสามารถเขียนเป้าหมายนี้ในรูปปัญหา optimization ได้

Math behind SVM (ไม่ต้องจำ)

- Hard Margin SVM ในรูปปัญหา optimization
- objective:

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

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



$$t^{(i)} = \begin{cases} -1, & y^{(i)} = 0 \\ 1, & y^{(i)} = 1 \end{cases}$$

Math behind SVM (ไม่ต้องจำ)

- Hard Margin SVM ในรูปปัญหา optimization
- objective:

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

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

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

เงื่อนไข: ห้ามเกิด violation

Math behind SVM (ไม่ต้องจำ)

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

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

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

- สังเกตว่าเกิด **hyperparameter C**

Math behind SVM (ไม่ต้องจำ)

- Soft Margin SVM objective:

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

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

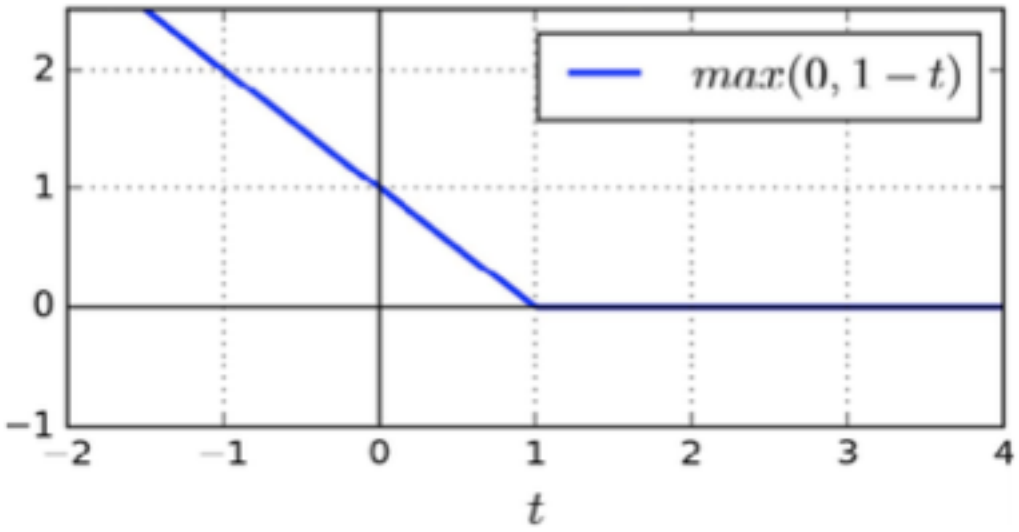
ปัญหานี้เป็นที่รู้จักในนาม **Quadratic Programming**
สามารถใช้ **QP Solver** สำเร็จรูปหาคำตอบได้อย่างรวดเร็ว ☺

Online SVM

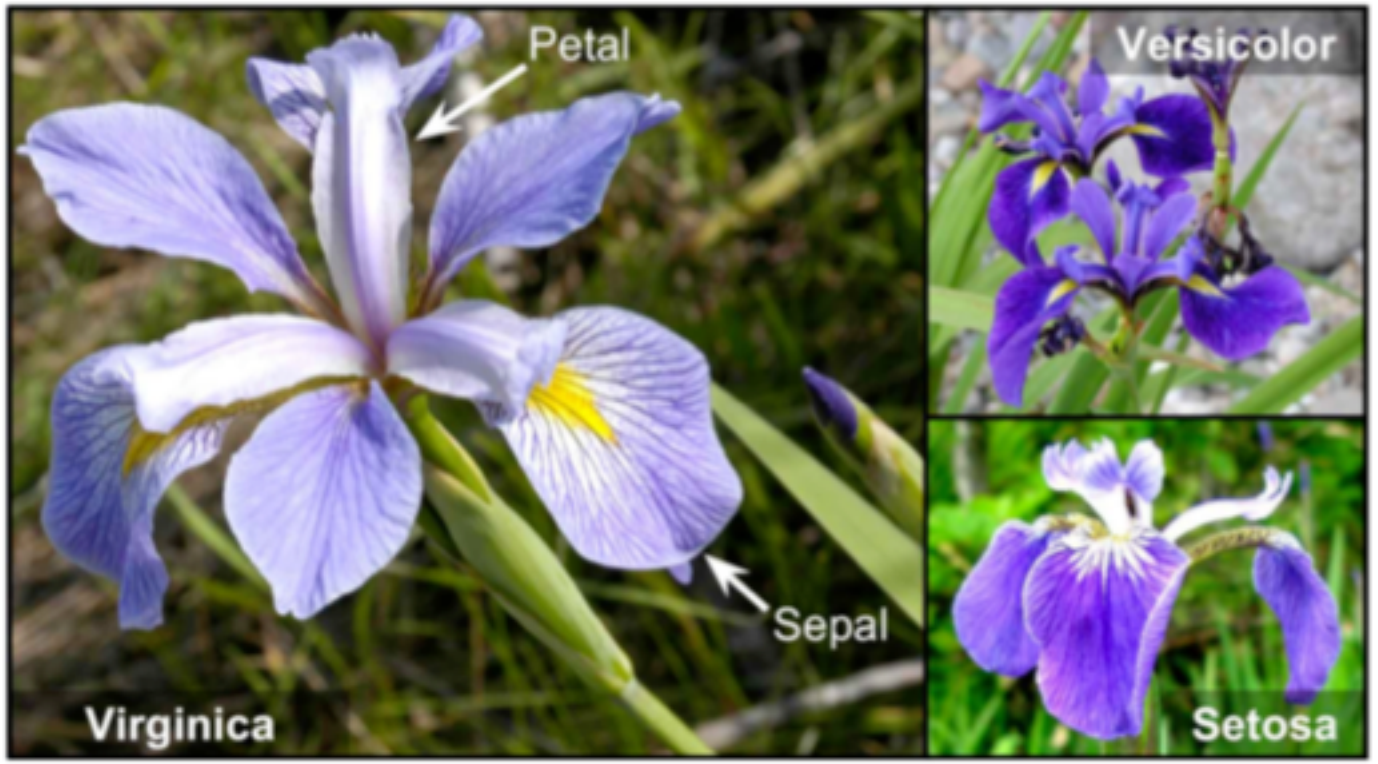
- เราสามารถใช้ **Gradient Descent** ในการแก้ **SVM** ได้ด้วย
- โดยคำนวณ **gradient** ของ **cost function** นี้

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

Hinge Loss



Lab: ใช้ SVM จำแนกพันธุ์ดอกไม้



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

Midterm & TPQI

- AI
  - Strong vs Weak
  - Turing Test
- Machine Learning
  - supervised vs unsupervised
  - classification vs regression
  - model, train, test, feature, class, cross-validation, overfitting
- Supervised Learning
  - kNN (ข้อเสียคืออะไร, ค่า k ส่งผลกับ boundary อย่างไร)
  - Linear Regression (วิธี gradient descent ดีกว่า normal eq อย่างไร, regularize ทำเพื่ออะไร)
    - prediction model, MSE cost function, normal equation, gradient descent, regularization
  - Logistic Regression (ใช้ทำอะไรได้, นึกถึง iris)
    - prediction model, sigmoid cost function
  - SVM (ใช้ทำอะไรได้, ดีอย่างไร, kernel มีประโยชน์อย่างไร)
    - prediction model, hard vs soft margin, kernel