# Linear Regression
# Gradient Descent

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

# Today

- Recap – kNN, MNIST
- Linear Regression
- Gradient Descent
- Next week: SVM

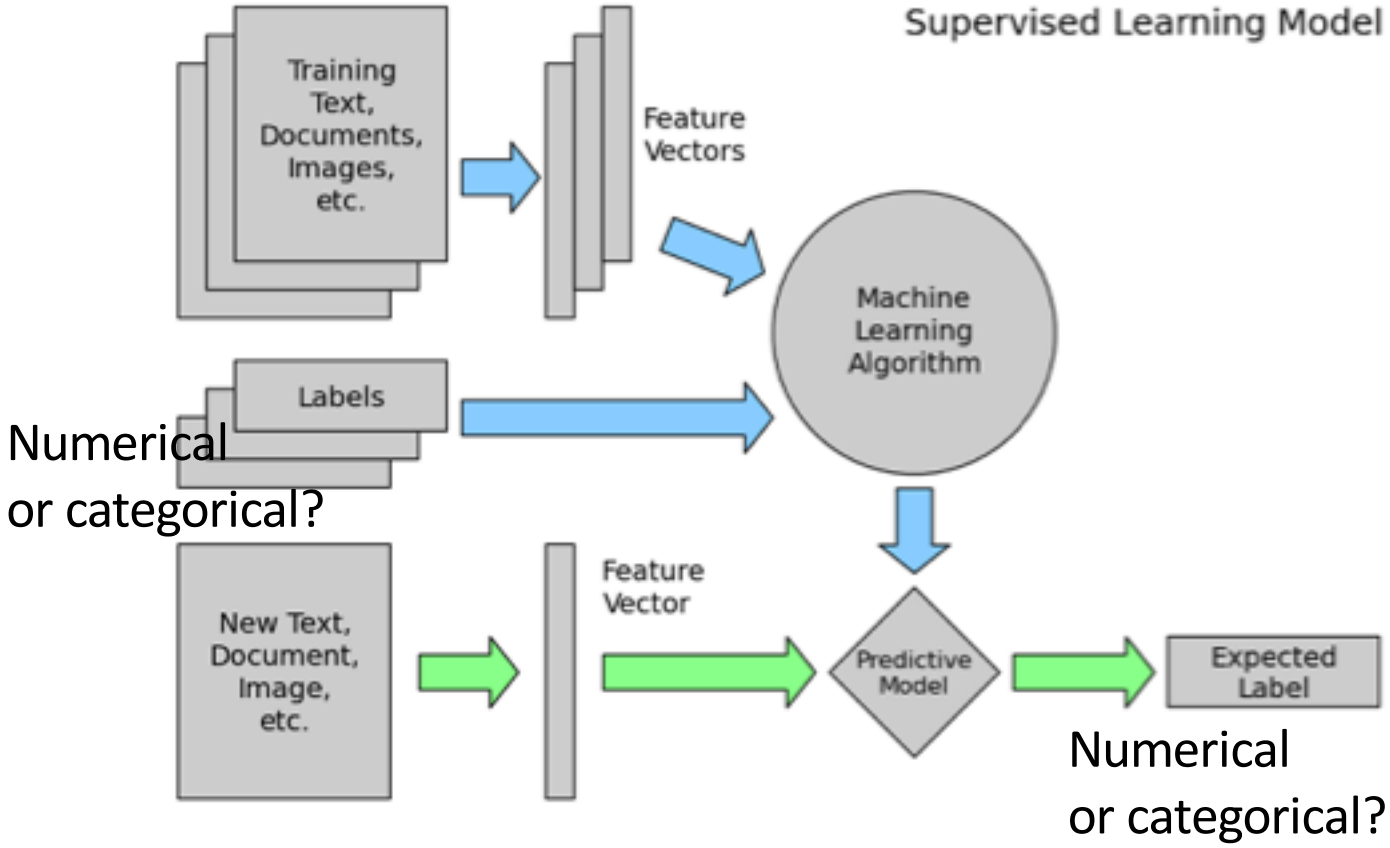
# Recap: Supervised Learning

kNN

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

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

Linear Regression



# Recap: K-Nearest Neighbor Algorithm

```
def train(train_images, labels):
      # ML
      return model
```

จดจำทุก **training data** และ **training label**

```
def predict(test_images):
      # use model to predict labels
      return test_labels
```

หา **train_image** ที่ใกล้เคียงกับ **test_image** มากที่สุด แล้วทำนายว่าเป็น **label** ของ **train_image** นั้น

X คือ feature ของข้อมูล (เช่น น้ำหนัก อายุ)
Y คือ label (เช่น เป็นมะเร็ง/สุขภาพดี)

# ขั้นการ test/predict ใน kNN

- ในการ test เราจะหาเพื่อนบ้าน k คน ที่ใกล้ที่สุด แล้วดูว่าเพื่อนบ้านเป็นมะเร็ง (Y1) หรือไม่เป็น (Y0) มากกว่ากัน เป็นต้น
- ในการหาเพื่อนบ้านที่ใกล้ เราวัดระยะทางจากอะไร -> นิยมใช้ Euclidean Distance

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

- formally, เราต้องการหาฟังก์ชัน h:X->Y ซึ่ง h(x) = P(y=มะเร็ง? | X=x)

$$P(y = j|X = x) = \frac{1}{K} \sum_{i \in \mathcal{A}} I(y^{(i)} = j)$$

โดยที่ I() เป็น indicator จะเท่ากับ 1 ถ้า y_i = j หรือเท่ากับ 0 ถ้า y_i != j

# ทำ kNN ด้วย scikit-learn

- โหลดข้อมูล MNIST ซึ่งประกอบด้วย
  - images ภาพตัวเลข
  - target label เฉลย
- แบ่งข้อมูลเป็นชุด train ชุด test
- สร้าง KNeighborsClassifier
- ทำการ train โดยเรียก classifier.fit(ชุด train)
- ทำการ test โดยเรียก classifier.predict(ชุด test)
- เปรียบเทียบผลการ predict กับเฉลยใน target แล้วประเมินประสิทธิภาพ
  - accuracy = จำนวนชุด test ที่ทายถูก / จำนวนชุด test ทั้งหมด
  - ใช้ metrics.confusion_matrix(เฉลย,ทาย) เพื่อหาว่าทำนายผิดอย่างไร

X คือ feature ของข้อมูล (เช่น น้ำหนัก อายุ)
Y คือ label (เช่น เป็นมะเร็ง/สุขภาพดี)

# สรุป kNN

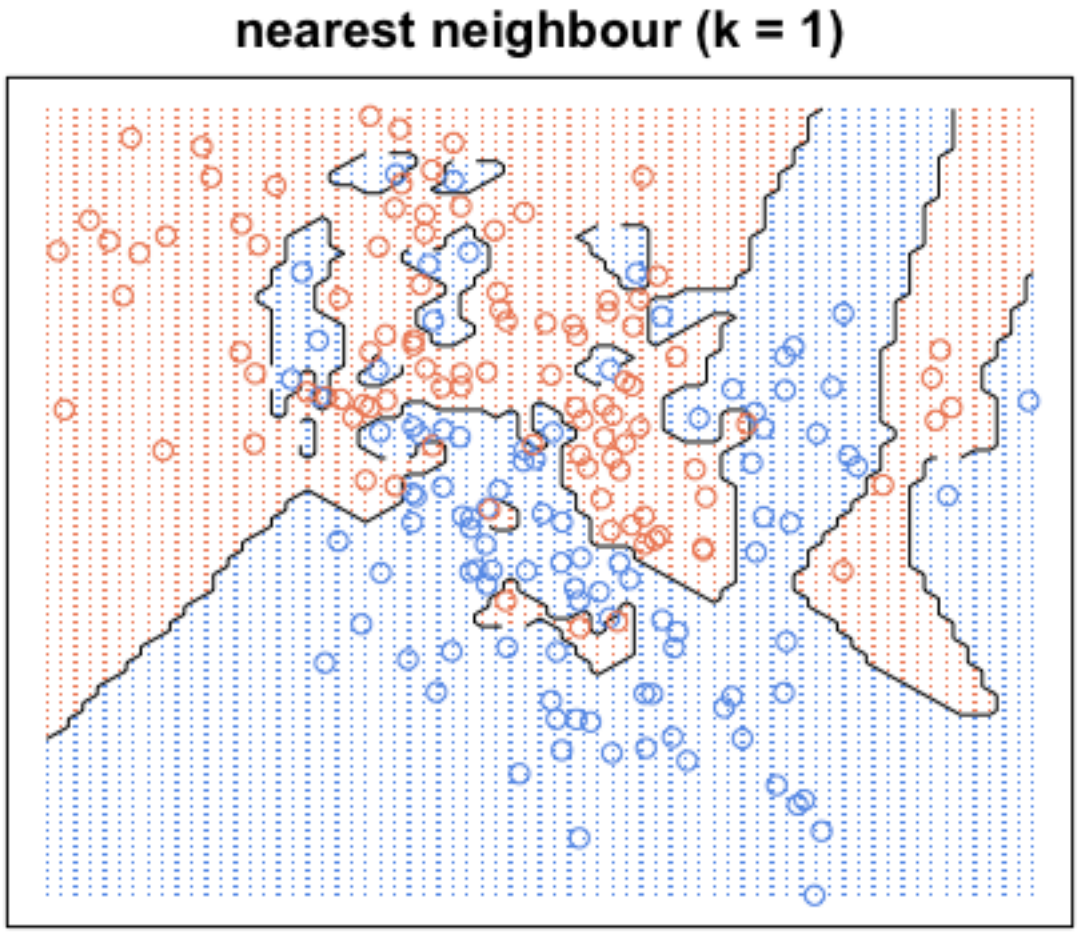
- เป็นอัลกอริธึมแบบ
  - supervised — มีผู้ช่วย (ใช้ข้อมูล train ที่มี label เฉลย ในการหา h:X->Y)
  - non-parametric — ไม่มี assumption เกี่ยวกับหน้าตาของฟังก์ชัน h: X -> Y ไม่มีโมเดลทางคณิตศาสตร์ที่ประกอบด้วยตัวแปร (parameter)
  - ใช้สำหรับทำ classification — จำแนกหมวดหมู่

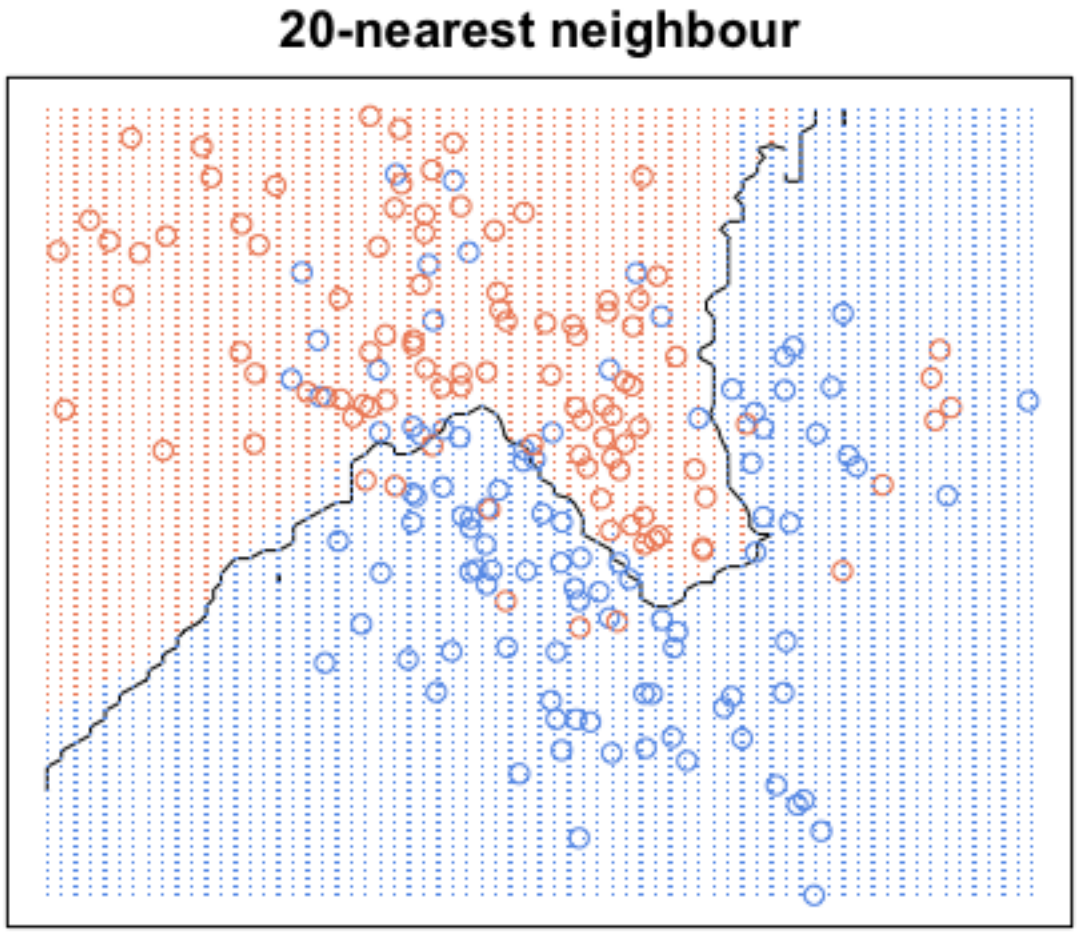
- ในการ train แค่จำข้อมูลทั้งหมดไว้ -> **ข้อเสีย:** เปลืองเนื้อที่
- ในการ test ทำการหาเพื่อนบ้าน k คน ที่ใกล้ที่สุด แล้วดูว่าเพื่อนบ้านเป็นมะเร็ง (Y1) หรือไม่เป็น (Y0) มากกว่ากัน -> **ข้อเสีย:** ใช้เวลาคำนวณนาน
- ค่าของ k เป็น hyperparameter ที่เราเลือกปรับได้

# More on kNN…

- Kevin Zakka's complete guide
  - https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/
- Stanford's CS231n slides
  - http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture02.pdf
- Visualize kNN classifier's boundary
  - http://wittawat.com/posts/knn_boundary.html
  - http://vision.stanford.edu/teaching/cs231n-demos/knn/

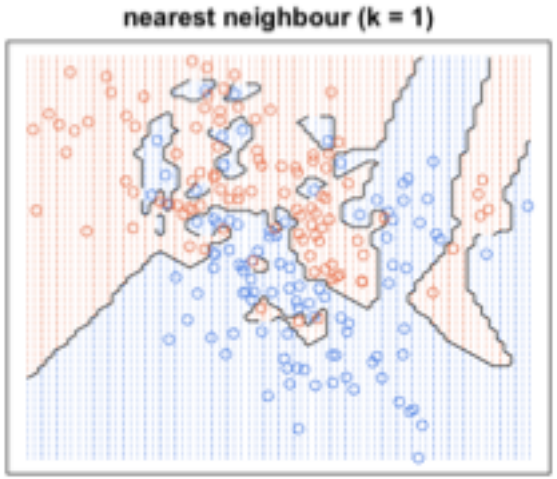
# How to choose the best k?



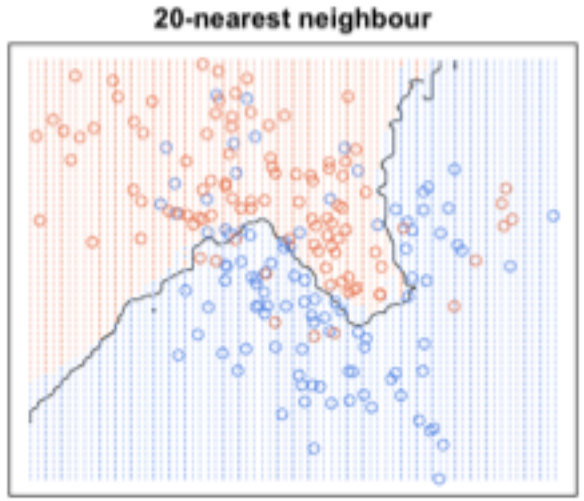



# Choosing k:  Bias-Variance Tradeoff

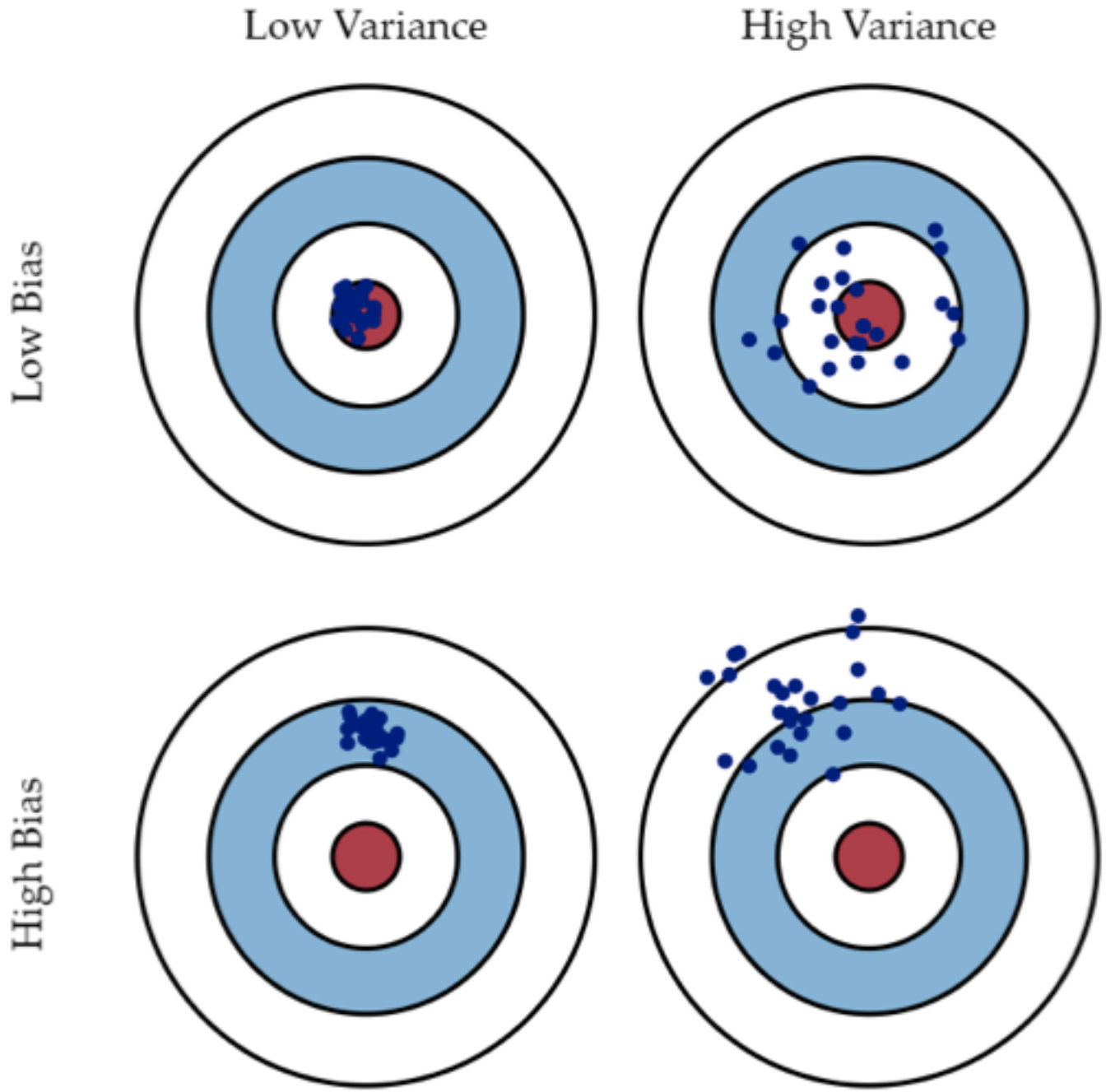
- ค่าของ k เป็น hyperparameter ที่เราเลือกปรับได้ **แต่**เราจะเลือกค่าที่ดีสุดได้อย่างไร?
- error ของการเรียนรู้ มีสองประเภท
  - bias  = ทำนายผิดเพราะเราใช้ model ซึ่งเป็นการประมาณจากโลกจริง อาจเกิดจากการที่เราลืมพิจารณาบางปัจจัย บาง feature ไป
  - variance = ทำนายผิดเพราะความผันผวนของข้อมูล เจออะไรที่ไม่คาดฝัน อาจเกิดจาก noise ในข้อมูล




low bias ☺
high variance ☹



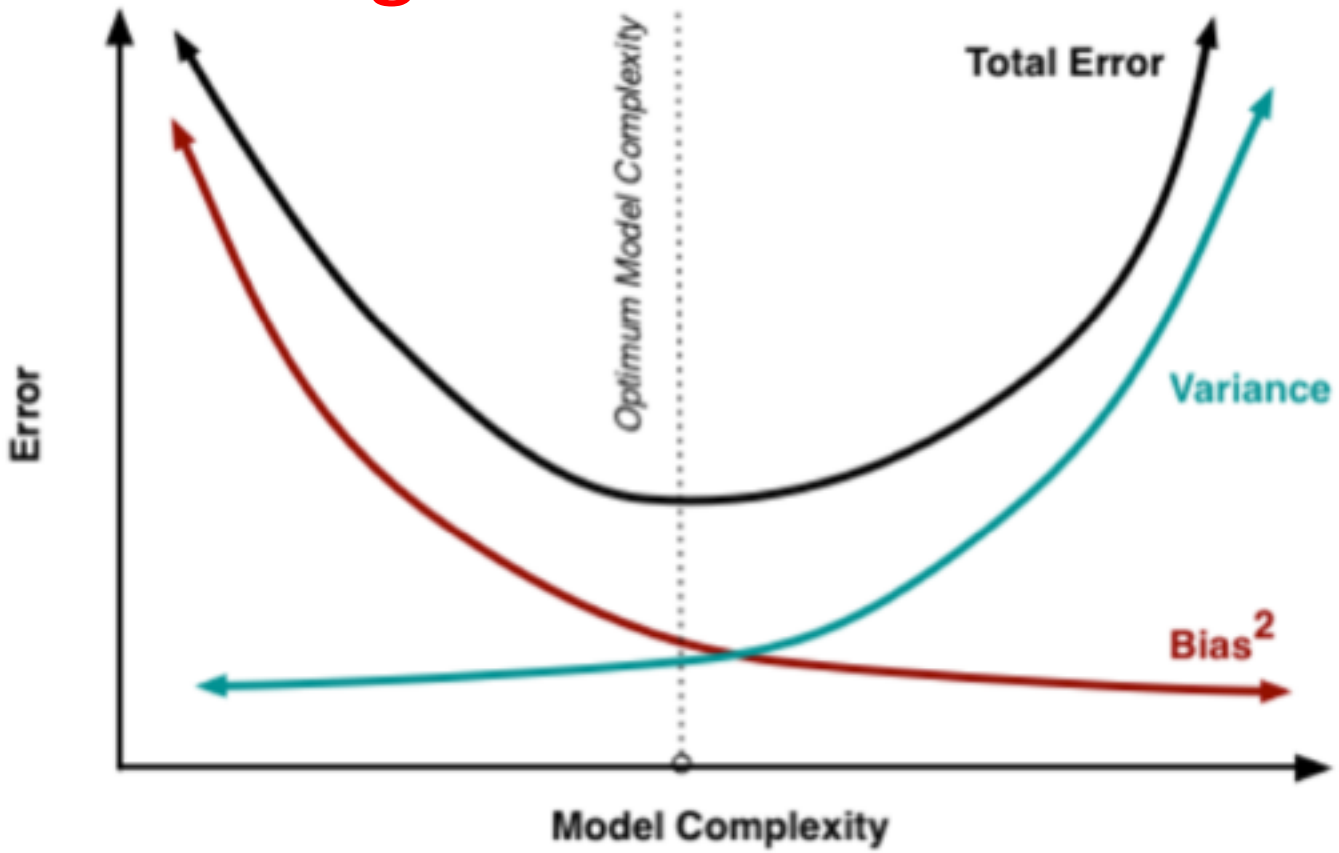

high bias ☹
low variance ☺


Low Variance
High Variance
Low Bias
High Bias

# Bias-Variance Tradeoff

underfitting

overfitting




k มาก

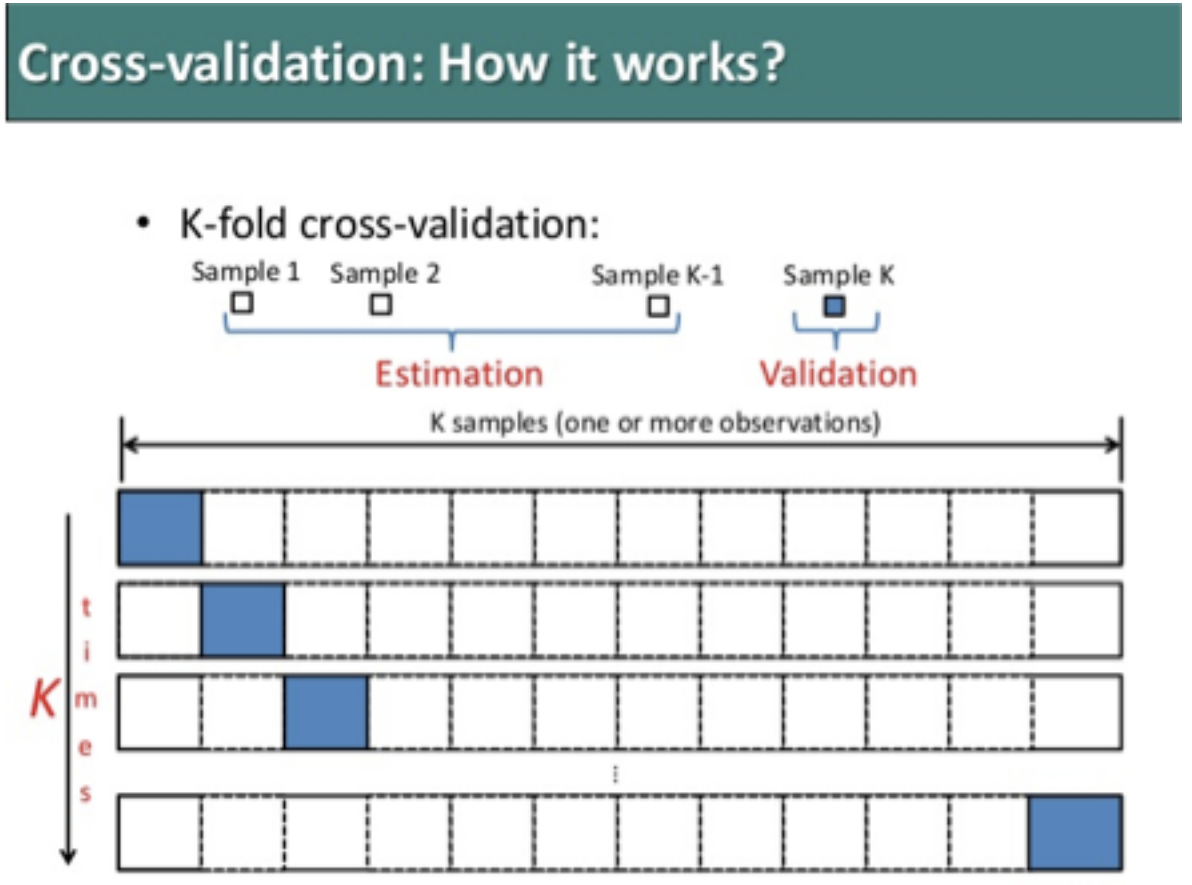
# Choosing k

- เป้าหมาย: เราต้องการค่า k ที่ทำให้ test error ต่ำสุด
- solution1: ทำการ train, test ซ้ำๆ กับข้อมูลชุดเดิม โดยเปลี่ยนค่า k ไปเรื่อยๆ แล้วหา k ที่ทำให้ test error ต่ำสุด
  - problem?

- Overfitting คือการที่โมเดล fit กับข้อมูลชุดหนึ่งมากเกินไป ทำให้ generalize กับข้อมูลชุดอื่นๆ ไม่ได้

# Cross Validation

- แบ่งข้อมูลออกเป็น **train, validate, test**
- ใช้ชุด **train, validate** ในการหาค่า **k** ที่ดีที่สุด
- ไม่ยุ่งกับข้อมูลชุด **test** จนกว่าจะหา **k** ที่ดีที่สุดได้
- ใช้ชุด **test** ในการประเมินสุดท้าย



# Linear Regression

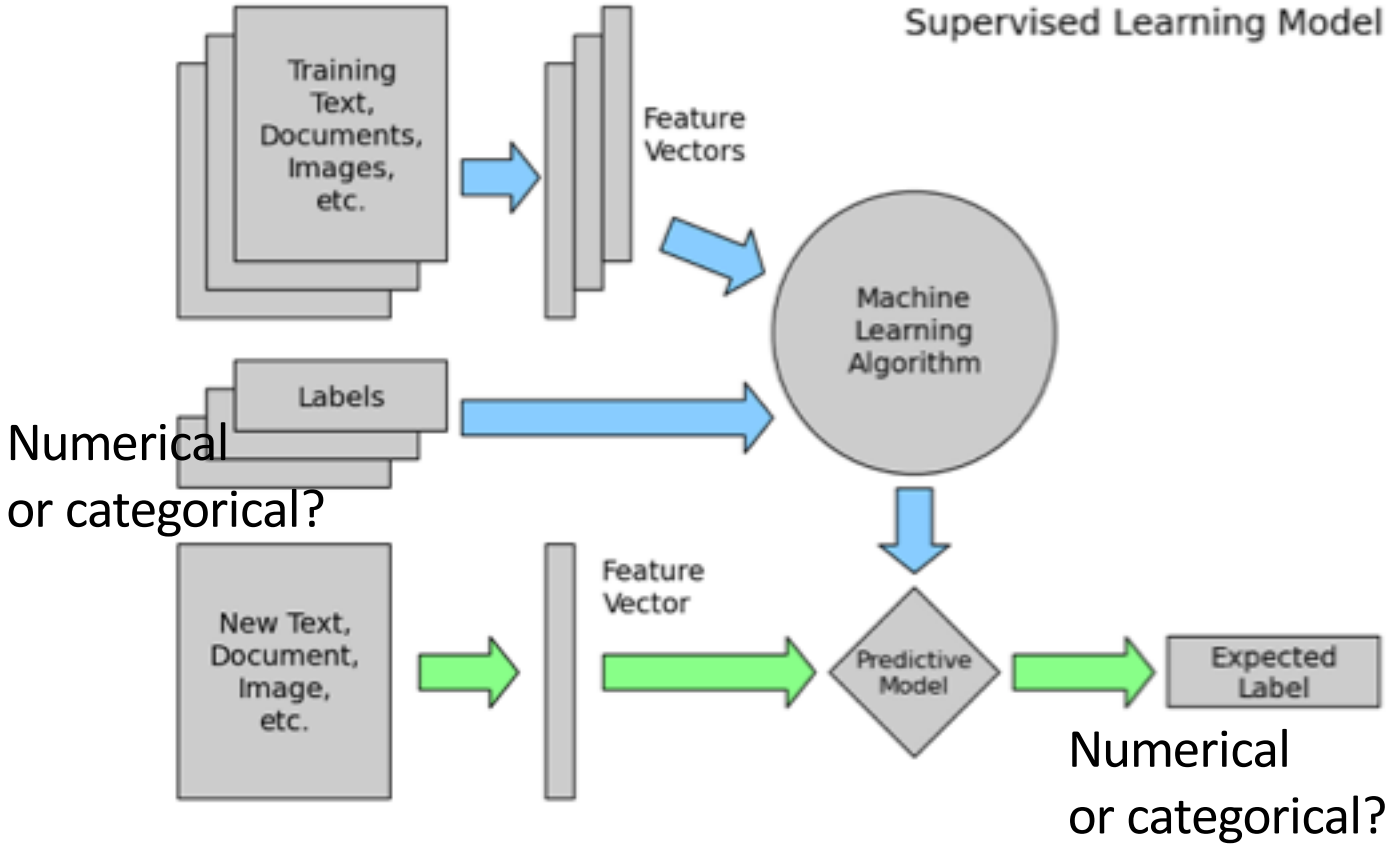
# Recap: Supervised Learning

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

kNN

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

Linear Regression

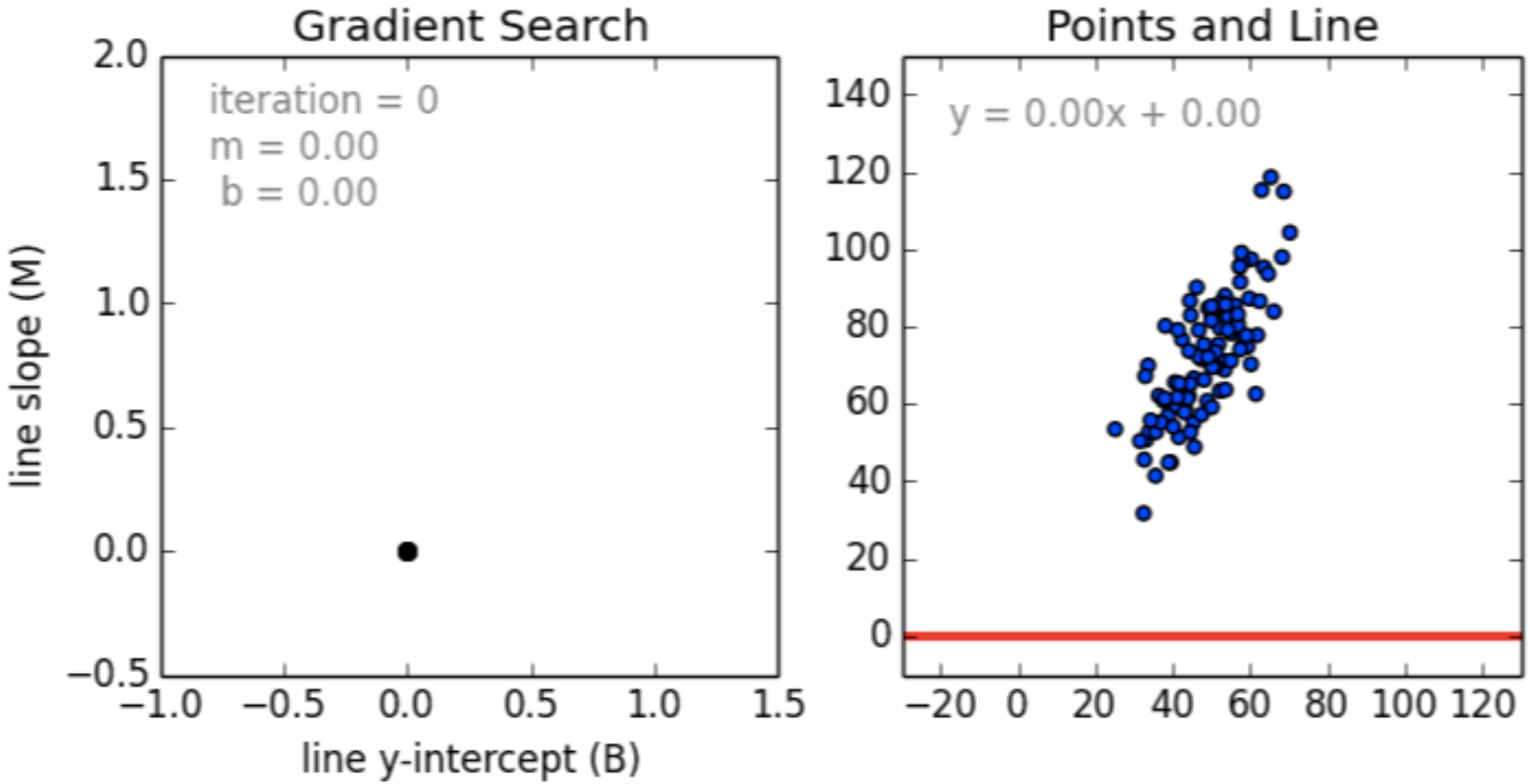


# Linear Regression

- เป็นอัลกอริธึมแบบ
  - supervised — มีผู้ช่วย (ใช้ข้อมูล train ที่มี label เฉลย ในการหา h:X->Y)
  - parametric — มี assumption ว่าฟังก์ชัน h: X -> Y เป็น Linear มีตัวแปร (parameter) ในโมเดล
  - ใช้สำหรับทำ regression — ทำนายค่าจำนวนจริง

- Example
  - โมเดล: life_satisfaction = $h(\theta) = \theta_0 + \theta_1 \times GDP$

# Linear Regression



# Linear Regression

- เราสามารถหาโมเดลที่ **fit** ข้อมูลได้ดีที่สุดด้วย คณิตศาสตร์ ได้อย่างไร**?**
- โมเดลการทำนาย: $\hat{y} = h_\theta(x) = \theta^T x$
- กำหนดเป้าหมาย $\mathrm{J}(\theta)$ เรียกว่า **cost function**
- ต้องการหาพารามิเตอร์ $\theta$ ที่ทำให้ $\mathrm{J}(\theta)$ น้อยที่สุด
- กำหนดให้ $\mathrm{J}(\theta) = MSE(\theta) = \frac{1}{M}\sum_{i=1}^{M}(\theta^T x^{(i)} - y^{(i)})^2$
- จะได้ว่า $\theta_{MSE} = \underset{\theta}{argmin}\ J(\theta)$

# Gradient Descent

- refer to text