Discussion on the *k*-NN Algorithm

→ มีตัวสำคัญอยู่ 2 อย่าง คือ
1. k
2. หาระยะห่าง

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors ⇒ ถามคนที่อยู่ใกล้เคียง เป็นจำนวนคี่เพื่อเอกฉันท์
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q

    เชื่อคนที่อยู่ใกล้เรามากที่สุด

    ถามคนที่ใกล้เคียง คนที่ 1, 2, 3, ...

    - Give greater weight to closer neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2} \Rightarrow \frac{1}{\text{ระยะทาง}}$$

→ ยิ่งใกล้ยิ่งมาก

→ ยิ่งไกลยิ่งน้อย

→ จำนวนเพื่อนบ้านที่จะถาม

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

Chapter 9. Classification: Advanced Methods

หาวิธีเลียนแบบคอมพิวเตอร์ให้สามารถตัดสินใจแทนเราได้ เช่น การถามเพื่อนบ้าน, การสร้างต้นไม้

- Bayesian Belief Networks
- Support Vector Machines

เลียนแบบเซลล์ประสาทของคน

- Neural Networks and Deep Learning
- Pattern-Based Classification
- Lazy Learners and K-Nearest Neighbors
- Other Classification Methods
- Summary

ของเทียม

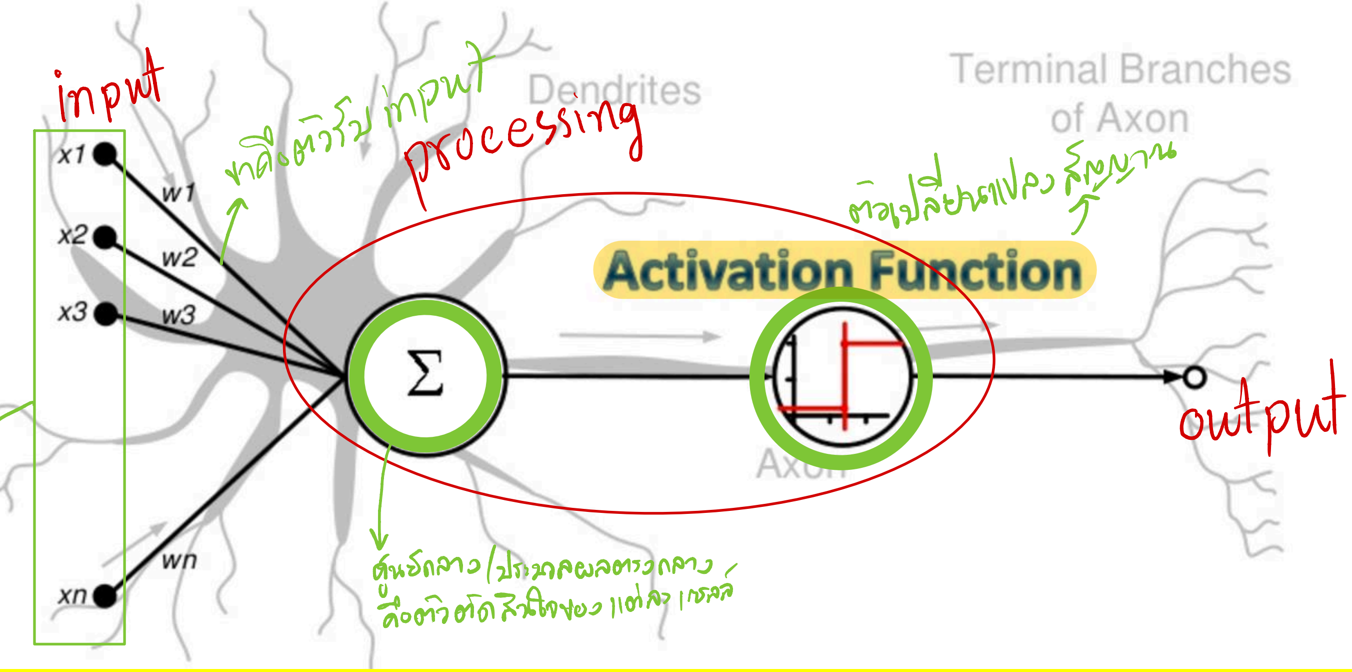
Artification +

Neural Network for Classification

เลียนแบบเซลล์ประสาทของคน

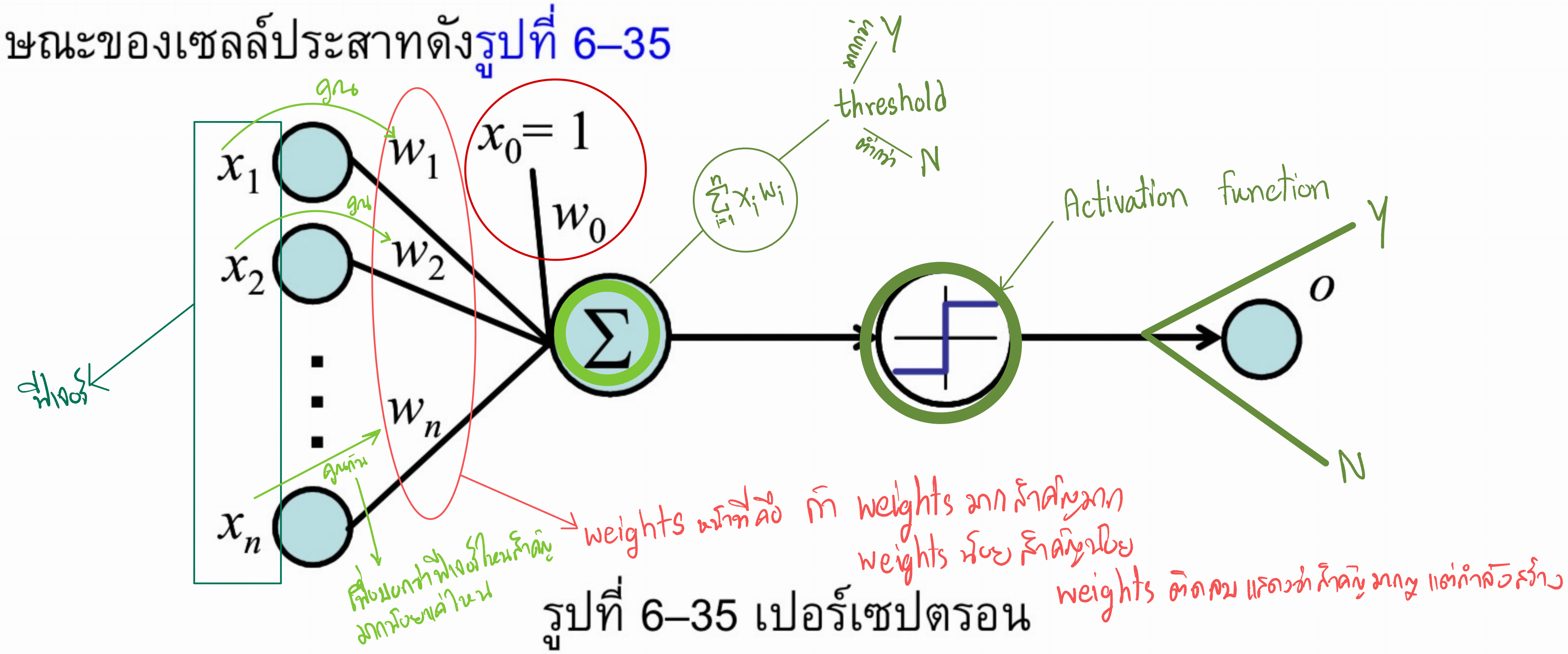
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
  - During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples

ฟีเจอร์



Artificial Neural Networks as an analogy of Biological Neural Networks

เพอร์เซปตรอน (perceptron) เป็นข่ายงานประสาทเทียมแบบง่ายมีหน่วยเดียวที่จำลองลักษณะของเซลล์ประสาทดังรูปที่ 6–35



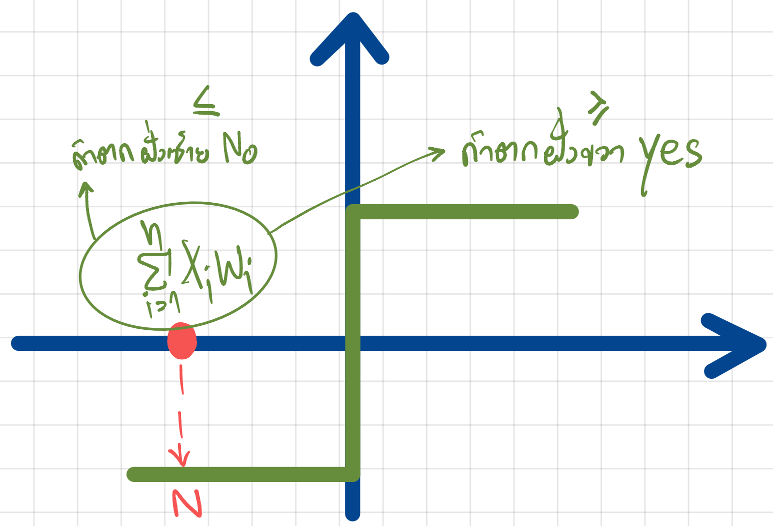
รูปที่ 6–35 เปอร์เซปตรอน

ตารางที่ 6–18 ฟังก์ชัน AND(x1,x2)

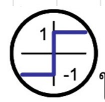
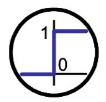
x_1	x_2	เอาต์พุตเป้าหมาย
0	0	0
0	1	0
1	0	0
1	1	1

X Y

$T \wedge T \equiv T$
$T \wedge F \equiv F$
$F \wedge T \equiv F$
$F \wedge F \equiv F$



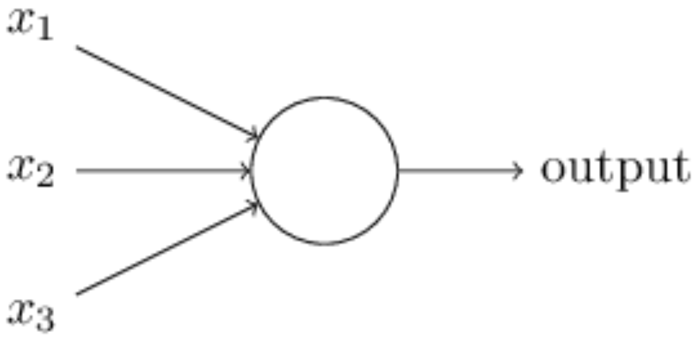
ฟังก์ชันกระตุ้น

 ในรูปแสดงฟังก์ชันกระตุ้น (activation function) ชนิดที่เรียกว่าฟังก์ชันสองขั้ว (bipolar function) ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ -1 ฟังก์ชันกระตุ้นอื่นๆ ที่นิยมใช้ก็อย่างเช่น ฟังก์ชันไบนารี (binary function) ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ 0 และเขียนแทนด้วยรูป 

เราสามารถแสดงเอาต์พุต (o) ในรูปของฟังก์ชันของอินพุต ($x_1, x_2, \ldots, x_n$) ได้ดังนี้

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + \cdots + w_nx_n > \theta \\ -1 & \text{if } w_1x_1 + w_2x_2 + \cdots + w_nx_n < \theta \end{cases} \quad (6.7)$$

Perceptron: Predecessor of a Neural Network



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \le \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

- The perceptron algorithm: invented in 1957 by Frank Rosenblatt
- Input: An *n*-dimensional input vector **x** (with n variables)
- Output: 1 or 0 depending on if the weighted sum passes a threshold
- Perceptron: A device that makes decisions by weighing up evidence
- Often written in the vector form, using bias (*b*) instead of threshold, as

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \le 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Bias: a measure of how easy it is to get the perceptron to output a 1

ตารางที่ 6–17 อัลกอริทึมกฎการเรียนรู้เพอร์เซปตรอน

ตอบถูกหมด

ครบกำหนด epoch

```
Algorithm: Perceptron-Learning-Rule

1. Initialize weights w_i of the perceptron.
2. UNTIL the termination condition is met DO
   2.1 FOR EACH training example DO
       - Input the example and compute the output.
       - Change the weights if the output from the
         perceptron is not equal to the target
         output using the following rule.
```

สำหรับ Data แต่ละตัว

นำไปคำนวณ คือ เอาไป sum

$f(\Sigma wx)$

weight ใหม่ weight เก่า Penalty คือ ลงโทษ

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i \leftarrow \alpha \boxed{(t-o)} x_i$$

t = Target-label ที่แท้จริง

o = output - model ทำนาย

input ของฟีเจอร์ i , input ของแต่ละตัว $x_0, x_1, x_2, \dots$

learning rate

where t, o and α are the target output, the output from the perceptron and the learning rate, respectively.

weight ใหม่ weight เก่า Penalty คือ ลงโทษ

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i \leftarrow \alpha \boxed{(t-o)} x_i$$

t = Target-label ที่แท้จริง

o = output - model ทำนาย

input ของฟีเจอร์ i , input ของแต่ละตัว $x_0, x_1, x_2, \dots$

learning rate

ตารางที่ 6–19 ผลการเรียนรู้ฟังก์ชัน AND โดยกฎการเรียนรู้เพอร์เซปตรอน

ถ้าปรับสูงจะเรียนรู้เร็ว เร็วมากก็ไม่ดี, ช้าไปก็ไม่ดี

learning rate = ค่าที่จะปรับให้เรียนรู้เร็วหรือช้า

① ปรับได้ เวลาเขียนโค้ด

Target – Actual

เปลี่ยนไปเรื่อยๆ ตาม Data

② ค่าเริ่มต้นที่กำหนดขึ้นมา

$\sum_{i=0}^{n}(x_i w_i)$ $\quad f(\sum_{i=0}^{n} x_i w_i)$ $\quad$ Y ที่แท้จริง

มากกว่า=1 น้อยกว่า=0

Perceptron Learning Example - Function AND										
		Bias Input x0=+1					Alpha =	0.5		
Input	Input				Net Sum	Target	Actual	Alpha*	Weight Values	
x1	x2	1.0*w0	x1*w1	x2*w2	Input	Output	Output	Error	w0	w1	w2
									0.1	0.1	0.1
0	0	0.10	+ 0.00	+ 0.00	= 0.10	0	1	-0.50	-0.40	0.10	0.10
0	1	-0.40	0.00	0.10	-0.30	0	0	0.00	-0.40	0.10	0.10
1	0	-0.40	0.10	0.00	-0.30	0	0	0.00	-0.40	0.10	0.10
1	1	-0.40	0.10	0.10	-0.20	1	0	0.50	0.10	0.60	0.60
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	0.60	0.60
0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	0.60	0.10
1	0	-0.90	0.60	0.00	-0.30	0	0	0.00	-0.90	0.60	0.10
1	1	-0.90	0.60	0.10	-0.20	1	0	0.50	-0.40	1.10	0.60
0	0	-0.40	0.00	0.00	-0.40	0	0	0.00	-0.40	1.10	0.60
0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	1.10	0.10
1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60	0.10
1	1	-1.40	0.60	0.10	-0.70	1	0	0.50	-0.90	1.10	0.60
0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10	0.60
0	1	-0.90	0.00	0.60	-0.30	0	0	0.00	-0.90	1.10	0.60
1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60	0.60
1	1	-1.40	0.60	0.60	-0.20	1	0	0.50	-0.90	1.10	1.10
0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10	1.10
0	1	-0.90	0.00	1.10	0.20	0	1	-0.50	-1.40	1.10	0.60
1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10	0.60
1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10	0.60
0	0	-1.40	0.00	0.00	-1.40	0	0	0.00	-1.40	1.10	0.60
0	1	-1.40	0.00	0.60	-0.80	0	0	0.00	-1.40	1.10	0.60
1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10	0.60
1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10	0.60

$W_i + \alpha(t-O)X_0$

$0.1+0.5(0-1)1 = 0.1+(-0.5) = 0.40$

$0.1+0.5(0-1) = 0.1$

$-0.4+0.5(1-0)1 = 0.1$

$0.1+0.5(1-0)1 = 0.6$

1 รอบ

1 epoch กำหนดรอบ ปรับได้

1 epoch

จบ

ข้อสอบ!!!

ลองทำดู!!!

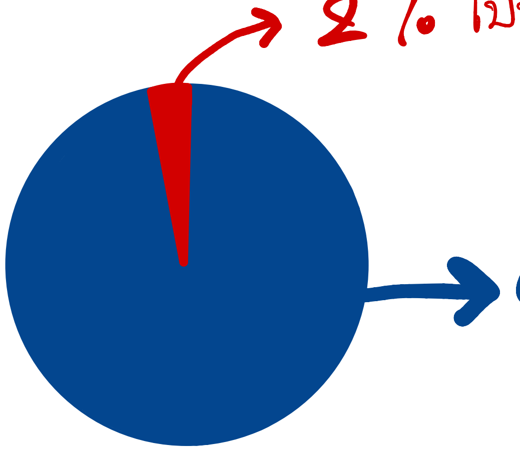
เติมตาราง อย่าลืม!!!

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Additional Concepts on Classification
- Summary

Model Evaluation and Selection

- Evaluation metrics
  - How can we measure accuracy?
  - Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier’s accuracy
  - Holdout method
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - ROC Curves



Classifier Evaluation Metrics: Confusion Matrix

- **Confusion Matrix:**

Class ที่แท้จริง

Precision

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

Recall

- In a confusion matrix w. *m* classes, $CM_{i,j}$ indicates # of tuples in class *i* that were labeled by the classifier as class *j*
  - May have extra rows/columns to provide totals
- **Example of Confusion Matrix:**

test

Positive Positive Negative

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

Negative

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P’	**N’**	**All**

- **Classifier accuracy,** or recognition rate
  - Percentage of test set tuples that are correctly classified

    ความแม่นยำ

    Accuracy = (TP + TN)/All
- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class imbalance problem**
  - One class may be *rare*
    - E.g., fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - Measures handle the class imbalance problem
    - **Sensitivity** (recall): True positive recognition rate
      - **Sensitivity = TP/P**
    - **Specificity**: True negative recognition rate
      - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: Exactness: what % of tuples that the classifier labeled as positive are actually positive?

$$P = \text{Precision} = \frac{TP}{TP + FP}$$

ตัวที่ Model ทายว่าเป็น Positive
ถูกต้องมากแค่ไหน

- **Recall:** Completeness: what % of positive tuples did the classifier label as positive?

$$R = \text{Recall} = \frac{TP}{TP + FN}$$

โมเดลเราหา ตัวที่เป็น Positive จริงๆ
เจอมากน้อยแค่ไหน

  - Range: [0, 1]
- The "inverse" relationship between precision & recall
- ***F* measure (or *F*-score):** harmonic mean of precision and recall
  - In general, it is the weighted measure of precision & recall

$$F_\beta = \frac{1}{\alpha \cdot \frac{1}{P} + (1 - \alpha) \cdot \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Assigning β times as much weight to recall as to precision)

  - ***F1-measure (balanced F-measure)***
    - That is, when β = 1,

ยิ่งสูงยิ่งดี
ต่ำไม่ดี

Precision
Recall

$$\boxed{F_1 = \frac{2PR}{P + R}}$$

P สูง R ต่ำ = F ต่ำ
คือ ต้องให้สมดุลกัน