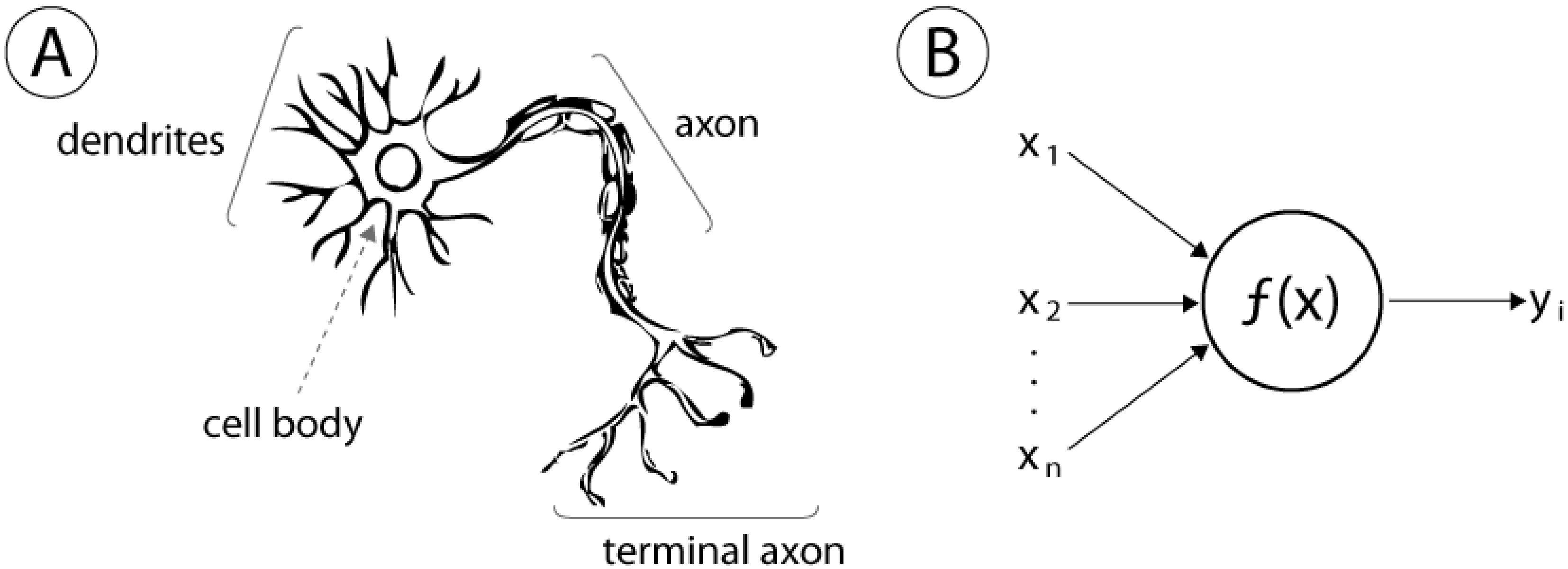
# Lecture 4
# Perceptron

Rui Xia
**T**ext **M**ining Group
**N**anjing **U**niversity of **S**cience & **T**echnology
rxia@njust.edu.cn

# Perceptron


- The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt.
- Perceptron is an algorithm for supervised classification.
- It is a type of linear classifier.
- It lays the foundation of artificial neural networks (ANN).

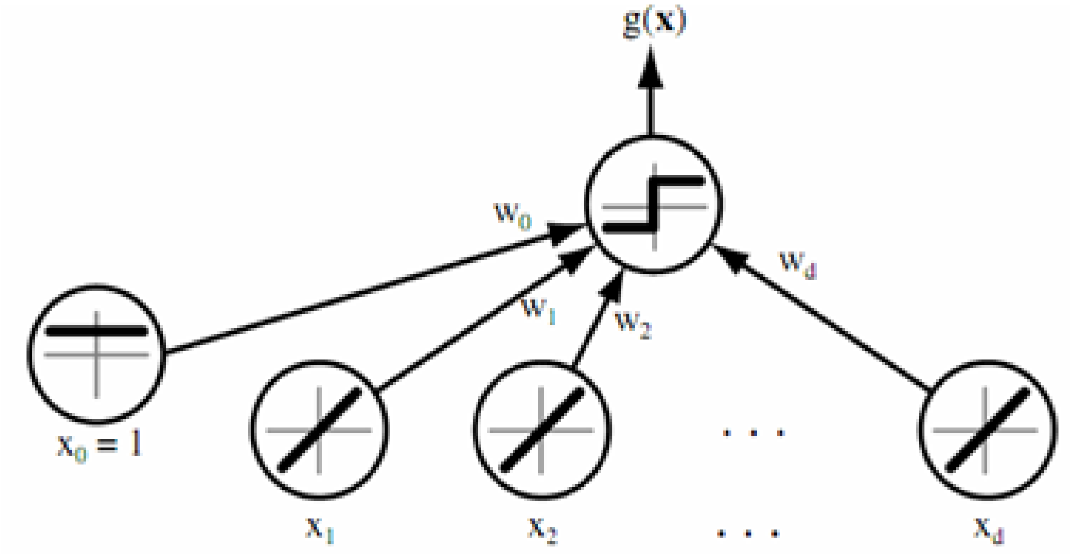
# Inspired from Neural Networks



# Model Description

- Model hypothesis

$$h_w(x) = \begin{cases} 1 & \text{if } \; w^T x \geq 0 \\ 0 & \text{if } \; w^T x < 0 \end{cases}$$



# Perceptron Algorithm

- Perceptron cost function

$$
\begin{aligned}
J_P(x) &= \sum_{x^{(i)} \epsilon M_0} \omega^T x^{(i)} - \sum_{x^{(j)} \epsilon M_1} \omega^T x^{(j)} \\
&= \sum_{i=1}^{N} \left( \left(1 - y^{(i)}\right) h_w\left(x^{(i)}\right) - y^{(i)} \left(1 - h_w\left(x^{(i)}\right)\right) \right) \omega^T x^{(i)} \\
&= \sum_{i=1}^{N} \left(h_w\left(x^{(i)}\right) - y^{(i)}\right) \omega^T x^{(i)}
\end{aligned}
$$

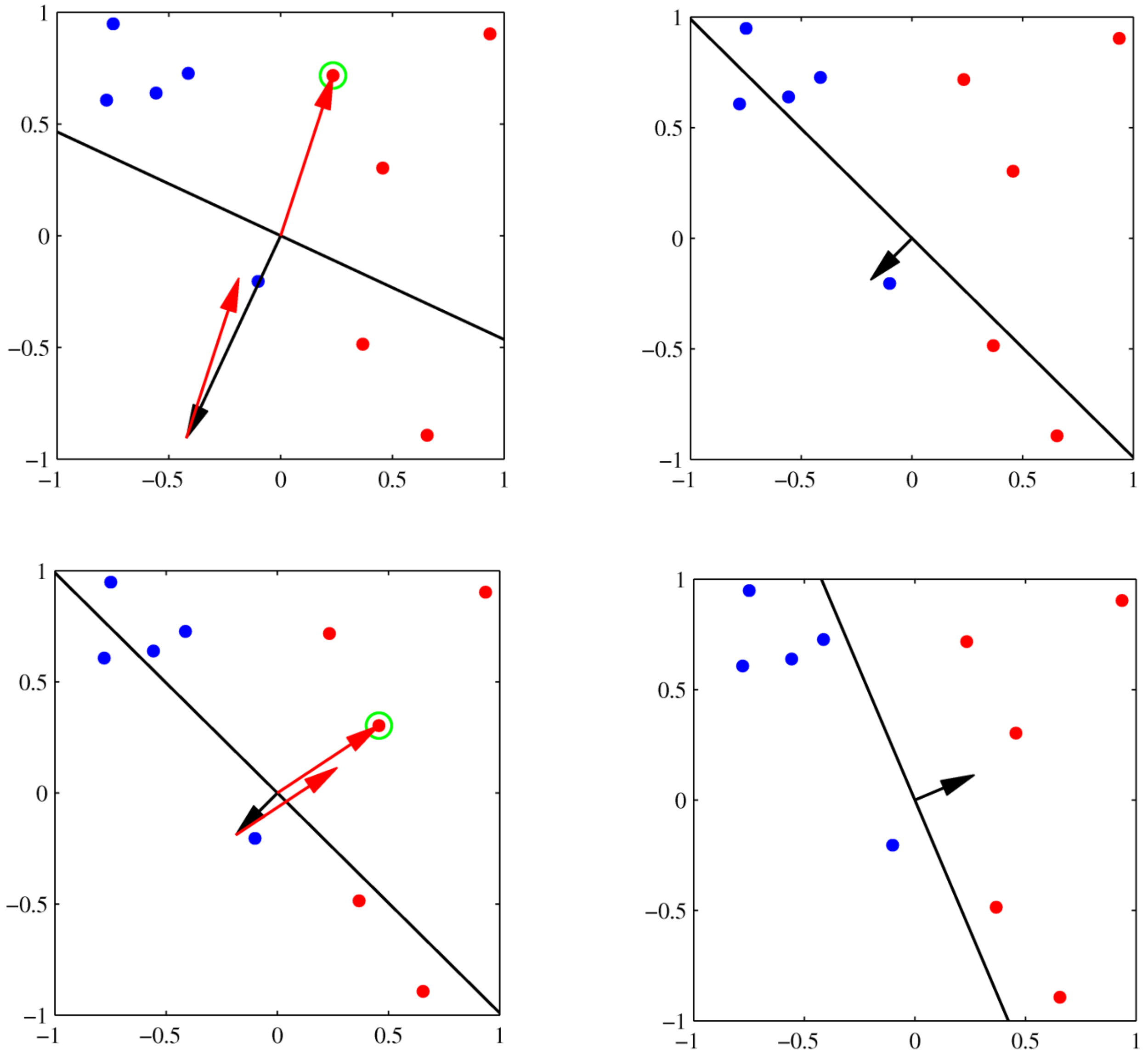
- Perceptron updating rule (by applying SGD)

$$
\begin{aligned}
\omega &:= \omega + \alpha \boxed{\left(y - h_w(x)\right) x} \\
&= \begin{cases} \omega + \alpha x, & \text{if } y = 1 \text{ and } h_w(x) = 0 \\ \omega - \alpha x, & \text{if } y = 0 \text{ and } h_w(x) = 1 \\ \omega, & \text{others} \end{cases}
\end{aligned}
$$

**Error × Feature**

NUSTM

# An Illustration

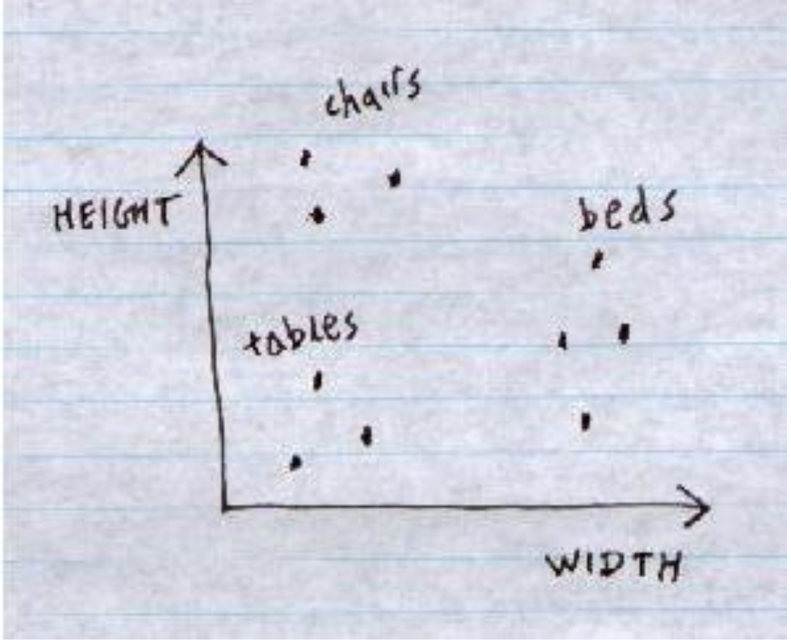


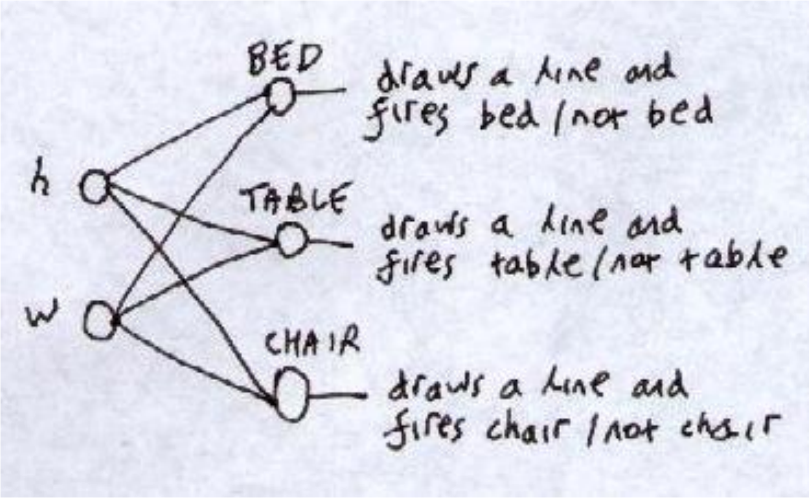
# A Simple Python Code

```
threshold = 0.5
learning_rate = 0.1
weights = [0, 0, 0]
training_set = [((1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1), ((1, 1, 1), 0)]

def dot_product(values, weights):
    return sum(value * weight for value, weight in zip(values, weights))

while True:
    print('-' * 60)
    error_count = 0
    for input_vector, desired_output in training_set:
        print(weights)
        result = dot_product(input_vector, weights) > threshold
        error = desired_output - result
        if error != 0:
            error_count += 1
            for index, value in enumerate(input_vector):
                weights[index] += learning_rate * error * value
    if error_count == 0:
        break
```

# Multi-class Perceptron

- Multi-class perceptron is an extension of standard perceptron to solve multi-class classification problems;
- Multi-class perceptron is widely used in NLP.



# Model Description

- Hypothesis

$$C^* = \arg\max_{j=1,\dots,C} \omega_j^T x$$

- Cost function

$$J_p(w) = \sum_{k=1}^{N} \left( \max_{j=1,\dots,C} \omega_j^T x^{(k)} - \omega_{y^{(k)}}^T x^{(k)} \right)$$

- Parameter update rule

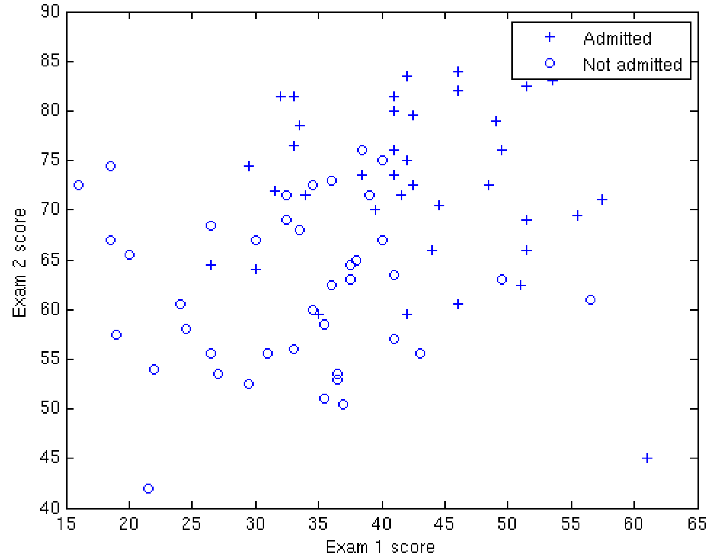
$$\omega_j := \omega_j - \alpha\left(1\{j = c^{(k)}\} - 1\{j = y^{(k)}\}\right)x^{(k)}$$
$$= \begin{cases} \omega_j - \alpha x^{(k)}, & \text{if } j = c^{(k)} \neq y^{(k)} \\ \omega_j + \alpha x^{(k)}, & \text{if } j = y^{(k)} \neq c^{(k)} \\ \omega_j, & \text{others} \end{cases}$$

$$\text{where } c^{(k)} = \arg\max_{j=1,\dots,C} \omega_j^T x^{(k)}$$

NUSTM

# Practice: Perceptron

- Given the following training data:




http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=DeepLearning&doc=exercises/ex4/ex4.html

- Implement perceptron algorithm and compare it with logistic regression (SGD);
- Implement multi-class perceptron algorithm and compare it with softmax regression (SGD).


# Questions?