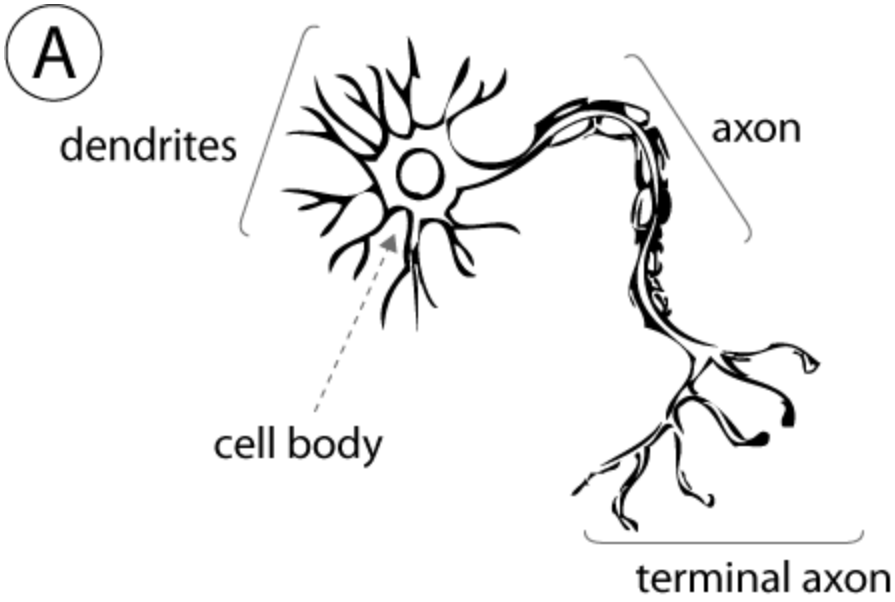
Lecture 4
Perceptron

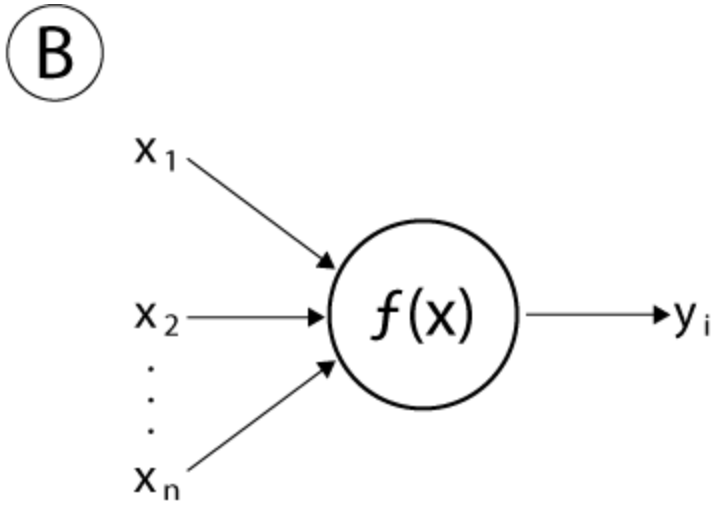
Rui Xia
School of Computer Science & Engineering
Nanjing University of Science & Technology
http://www.nustm.cn/~rxia

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

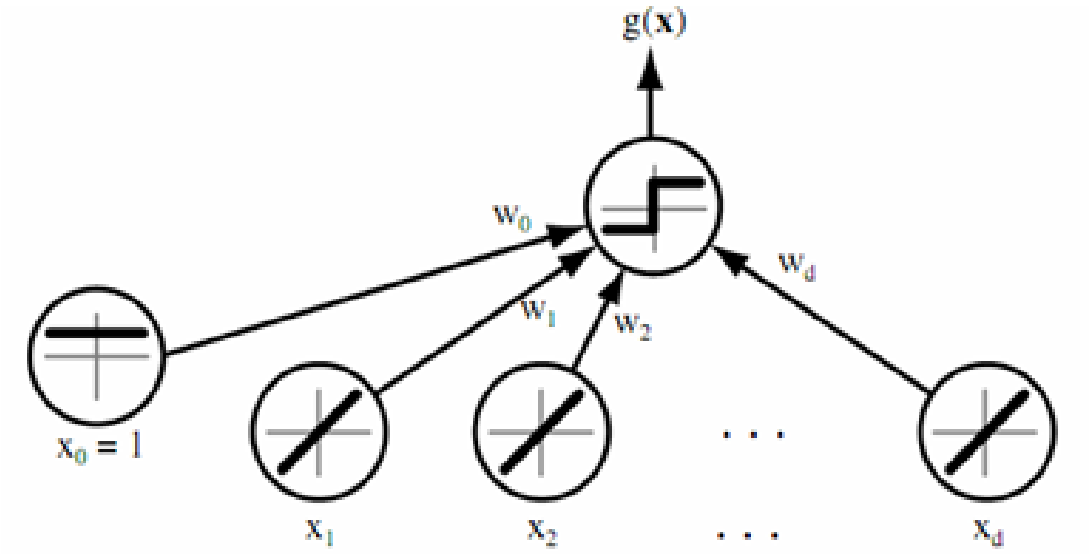




Hypothesis

- Model hypothesis

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



Learning

- Perceptron cost function

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

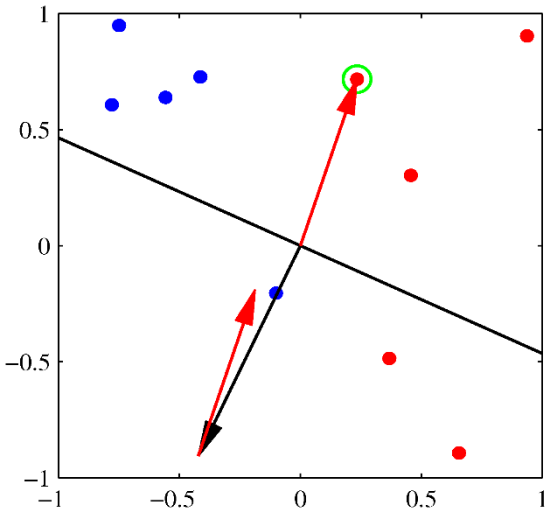
- Perceptron updating rule (by applying SGD)

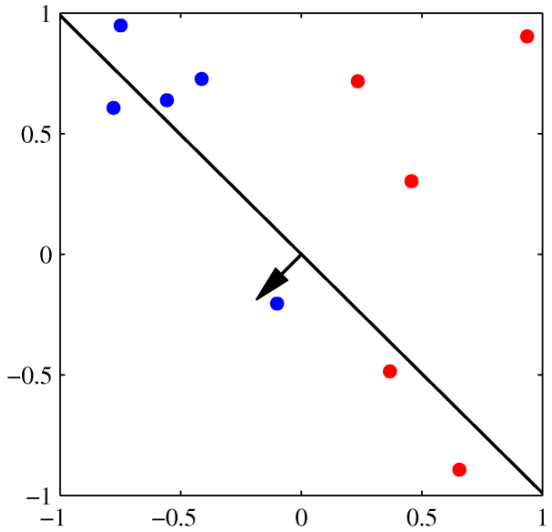
$$
\boldsymbol{w} = \boldsymbol{w} + \alpha \left(y - h_{\boldsymbol{w}}(\boldsymbol{x}) \right) \boldsymbol{x}
$$

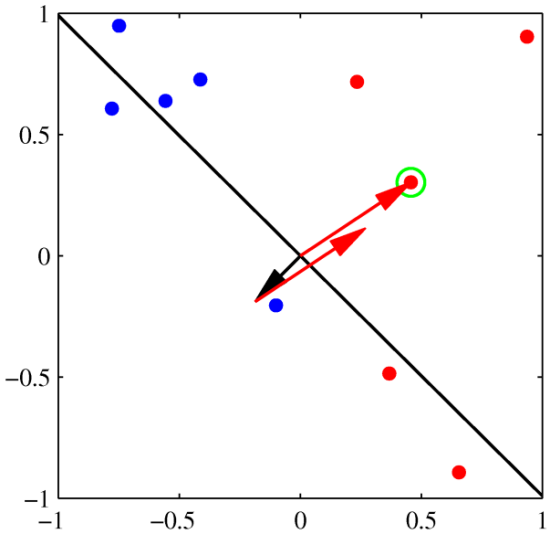
Error × Feature

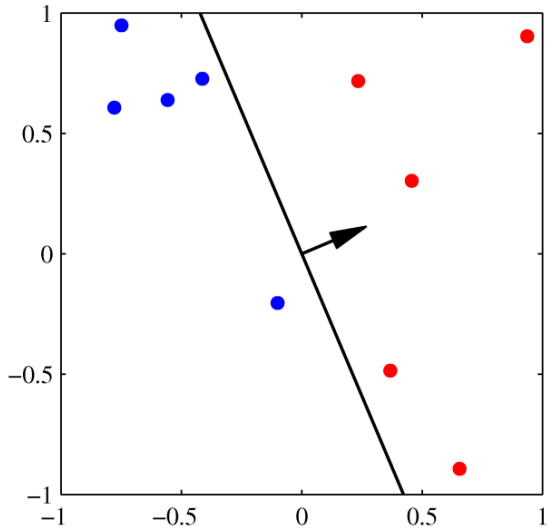
$$
= \begin{cases} \boldsymbol{w} + \alpha \boldsymbol{x}, & if\ y = 1\ and\ h_{\boldsymbol{w}}(\boldsymbol{x}) = 0 \\ \boldsymbol{w} - \alpha \boldsymbol{x}, & if\ y = 0\ and\ h_{\boldsymbol{w}}(\boldsymbol{x}) = 1 \\ \boldsymbol{w}, & otherwise \end{cases}
$$

An Illustration







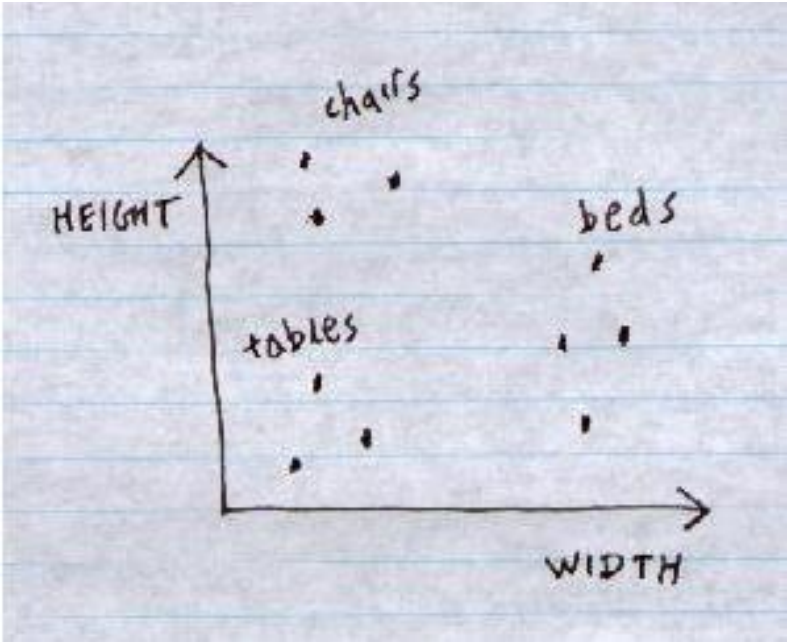


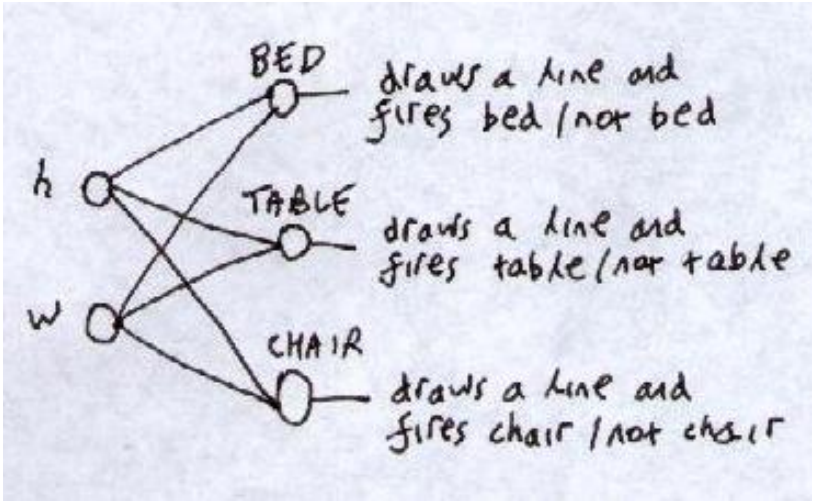
A Simple Python Code

```
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.





Hypothesis and Learning

- Hypothesis

$$C^* = \arg \max_{j=1,\ldots,C} \boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}$$

- Cost function

$$J_p(\boldsymbol{w}) = \sum_{k=1}^{N} \left(\max_{j=1,\ldots,C} \boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}^{(k)} - \boldsymbol{w}_{y^{(k)}}^{\mathrm{T}} \boldsymbol{x}^{(k)} \right)$$

- Parameter update rule

$$\boldsymbol{w}_j := \boldsymbol{w}_j - \alpha\left(1\{j = c^{(k)}\} - 1\{j = y^{(k)}\}\right)\boldsymbol{x}^{(k)}$$

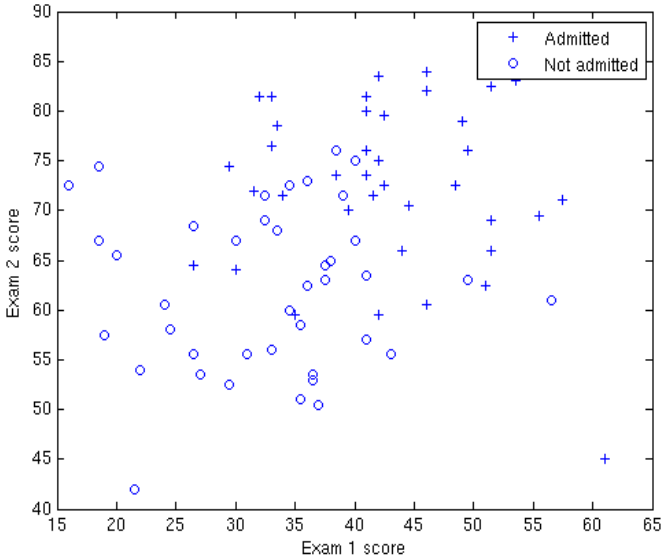
$$= \begin{cases} \boldsymbol{w}_j - \alpha \boldsymbol{x}^{(k)}, & \text{if } j = c^{(k)} \neq y^{(k)} \\ \boldsymbol{w}_j + \alpha \boldsymbol{x}^{(k)}, & \text{if } j = y^{(k)} \neq c^{(k)} \\ \boldsymbol{w}_j, & \text{otherwise} \end{cases}$$

$$\text{where } c^{(k)} = \arg \max_{j=1,\ldots,C} \boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}^{(k)}$$

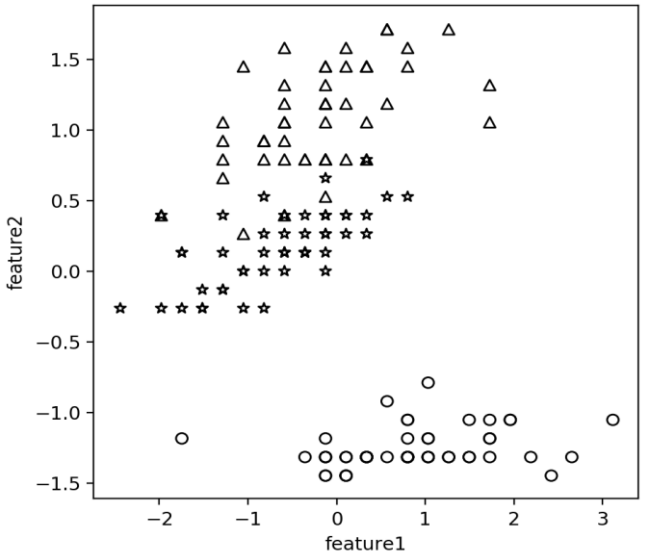
Practice 4: Perceptron

- Given the following training data sets:

(1)



(2)



(1) http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=DeepLearning&doc=exercises/ex4/ex4.html
(2) https://pan.baidu.com/s/1gU81bKsIj8cRokOYEk1Jzw password: w2a8

- For data set (1), implement perceptron algorithm and compare it with logistic regression (SGD);
- For data set (1), implement multi-class perceptron algorithm and compare it with standard perceptron (SGD);
- For data set (2), implement multi-class perceptron algorithm and compare it with softmax regression (SGD).

Any Questions?