

Machine Learning Basics

Perceptron

HKUST MSBD 6000B

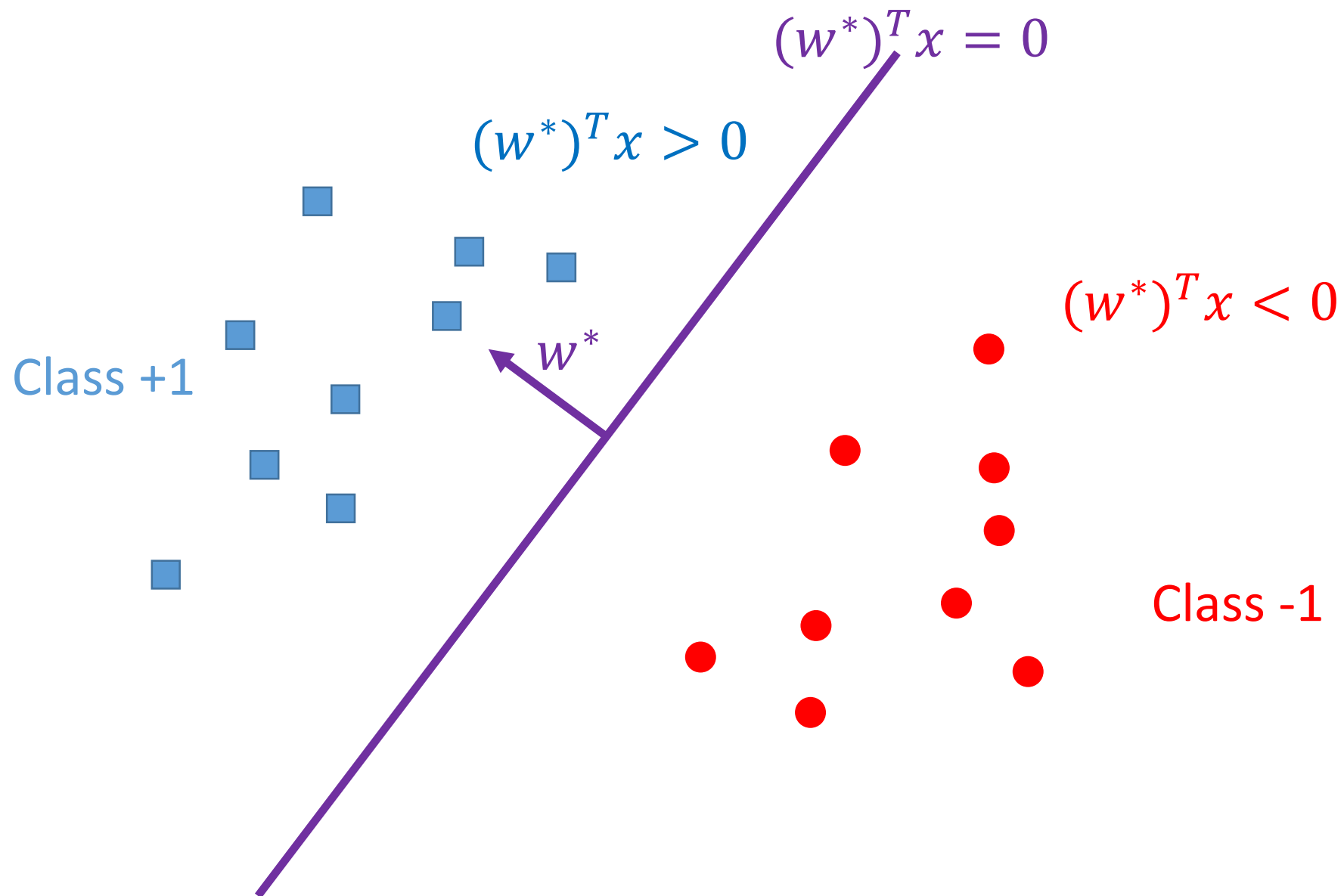
Instructor: Yu Zhang

Perceptron

Overview

- Previous lectures: (Principle for loss function) MLE to derive loss
 - Example: linear regression; some linear classification models
- This lecture: (Principle for optimization) local improvement
 - Example: Perceptron; SGD

Task



Attempt

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Hypothesis $f_w(x) = w^T x$
 - $y = +1$ if $w^T x > 0$
 - $y = -1$ if $w^T x < 0$
- Prediction: $y = \text{sign}(f_w(x)) = \text{sign}(w^T x)$
- Goal: minimize classification error

Perceptron Algorithm

- Assume for simplicity: all x_i has length 1

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example \mathbf{x} , predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
 - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
 - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$. $t \leftarrow t + 1$.

Perceptron: figure from the lecture note of Nina Balcan

Intuition: correct the current mistake

- If mistake on a positive example

$$w_{t+1}^T x = (w_t + x)^T x = w_t^T x + x^T x = w_t^T x + 1$$

- If mistake on a negative example

$$w_{t+1}^T x = (w_t - x)^T x = w_t^T x - x^T x = w_t^T x - 1$$

The Perceptron Theorem

- Suppose there exists w^* that correctly classifies $\{(x_i, y_i)\}$
- W.L.O.G., all x_i and w^* have length 1, so the minimum distance of any example to the decision boundary is

$$\gamma = \min_i |(w^*)^T x_i|$$

- Then Perceptron makes **at most $\left(\frac{1}{\gamma}\right)^2$ mistakes**

The Perceptron Theorem

- Suppose there exists w^* that correctly classifies $\{(x_i, y_i)\}$
- W.L.O.G., all x_i and w^* have length 1, so the minimum distance of any example to the decision boundary is

$$\gamma = \min_i |(w^*)^T x_i|$$

Need not be i.i.d. !

- Then Perceptron makes at most $\left(\frac{1}{\gamma}\right)^2$ mistakes

Do not depend on n , the length of the data sequence!

Analysis

- First look at the quantity $w_t^T w^*$
- Claim 1: $w_{t+1}^T w^* \geq w_t^T w^* + \gamma$
- Proof: If mistake on a positive example x

$$w_{t+1}^T w^* = (w_t + x)^T w^* = w_t^T w^* + x^T w^* \geq w_t^T w^* + \gamma$$

- If mistake on a negative example

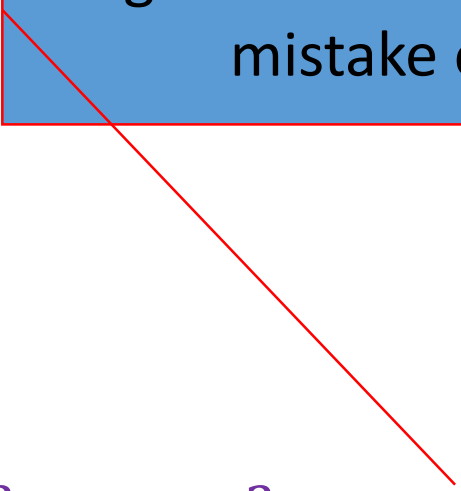
$$w_{t+1}^T w^* = (w_t - x)^T w^* = w_t^T w^* - x^T w^* \geq w_t^T w^* + \gamma$$

Analysis

- Next look at the quantity $\|w_t\|$
- Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$
- Proof: If mistake on a positive example x

$$\|w_{t+1}\|^2 = \|w_t + x\|^2 = \|w_t\|^2 + \|x\|^2 + 2w_t^T x$$

Negative since we made a mistake on x



Analysis: putting things together

- Claim 1: $w_{t+1}^T w^* \geq w_t^T w^* + \gamma$
- Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$

After M mistakes:

- $w_{M+1}^T w^* \geq \gamma M$
- $\|w_{M+1}\| \leq \sqrt{M}$
- $w_{M+1}^T w^* \leq \|w_{M+1}\|$

So $\gamma M \leq \sqrt{M}$, and thus $M \leq \left(\frac{1}{\gamma}\right)^2$

Intuition

- Claim 1: $w_{t+1}^T w^* \geq w_t^T w^* + \gamma$
- Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$

The correlation gets larger. Could be:

1. w_{t+1} gets closer to w^*
2. w_{t+1} gets much longer

Rules out the bad case “2. w_{t+1} gets much longer”

Some side notes on Perceptron

Note: connectionism vs symbolism

- Symbolism: AI can be achieved by representing concepts as symbols
 - Example: rule-based expert system, formal grammar
- Connectionism: explain intellectual abilities using connections between neurons (i.e., artificial neural networks)
 - Example: perceptron, larger scale neural networks

Symbolism example: Credit Risk Analysis

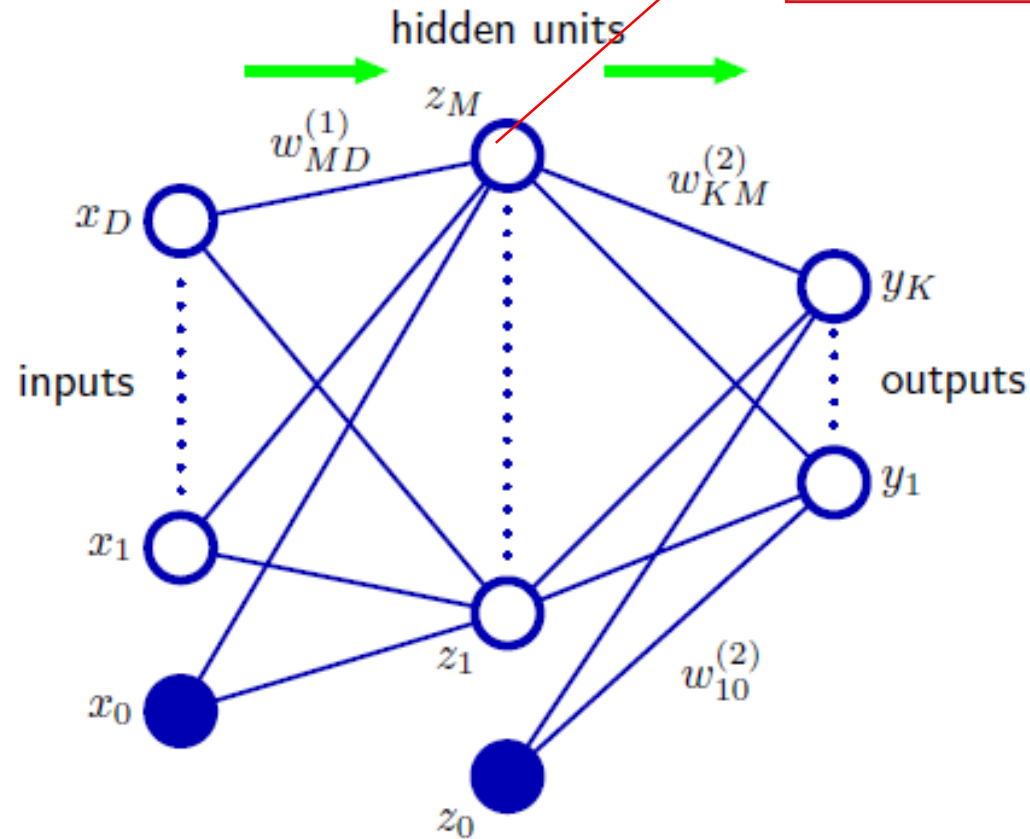
```
If    Other-Delinquent-Accounts > 2, and  
      Number-Delinquent-Billing-Cycles > 1  
Then Profitable-Customer? = No  
    [Deny Credit Card application]
```

```
If    Other-Delinquent-Accounts = 0, and  
      (Income > $30k) OR (Years-of-Credit > 3)  
Then Profitable-Customer? = Yes  
    [Accept Credit Card application]
```

Example from Machine learning lecture notes by Tom Mitchell

Connectionism example

Network diagram for the two-layer neural network corresponding to (5.7). The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables x_0 and z_0 . Arrows denote the direction of information flow through the network during forward propagation.



Neuron/perceptron

Figure from
*Pattern Recognition
and machine learning*,
Bishop

Note: online vs batch

- Batch: Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$, typically i.i.d.
- Online: data points arrive one by one
 - 1. The algorithm receives an unlabeled example x_i
 - 2. The algorithm predicts a classification of this example.
 - 3. The algorithm is then told the correct answer y_i , and update its model

Stochastic gradient descent (SGD)

Gradient descent

- Minimize loss $\hat{L}(\theta)$, where the hypothesis is parametrized by θ
- Gradient descent
 - Initialize θ_0
 - $\theta_{t+1} = \theta_t - \eta_t \nabla \hat{L}(\theta_t)$

Stochastic gradient descent (SGD)

- Suppose data points arrive one by one
- $\hat{L}(\theta) = \frac{1}{n} \sum_{t=1}^n l(\theta, x_t, y_t)$, but we only know $l(\theta, x_t, y_t)$ at time t
- Idea: simply do what you can based on local information
 - Initialize θ_0
 - $\theta_{t+1} = \theta_t - \eta_t \nabla l(\theta_t, x_t, y_t)$

Example 1: linear regression

- Find $f_w(x) = w^T x$ that minimizes $\hat{L}(f_w) = \frac{1}{n} \sum_{t=1}^n (w^T x_t - y_t)^2$
- $l(w, x_t, y_t) = \frac{1}{n} (w^T x_t - y_t)^2$
- $w_{t+1} = w_t - \eta_t \nabla l(w_t, x_t, y_t) = w_t - \frac{2\eta_t}{n} (w_t^T x_t - y_t) x_t$

Example 2: logistic regression

- Find w that minimizes

$$\hat{L}(w) = -\frac{1}{n} \sum_{y_t=1} \log \sigma(w^T x_t) - \frac{1}{n} \sum_{y_t=-1} \log[1 - \sigma(w^T x_t)]$$

$$\hat{L}(w) = -\frac{1}{n} \sum_t \log \sigma(y_t w^T x_t)$$

$$l(w, x_t, y_t) = \frac{-1}{n} \log \sigma(y_t w^T x_t)$$

Example 2: logistic regression

- Find w that minimizes

$$l(w, x_t, y_t) = \frac{-1}{n} \log \sigma(y_t w^T x_t)$$

$$w_{t+1} = w_t - \eta_t \nabla l(w_t, x_t, y_t) = w_t + \frac{\eta_t}{n} \frac{\sigma(a)(1-\sigma(a))}{\sigma(a)} y_t x_t$$

Where $a = y_t w_t^T x_t$

Example 3: Perceptron

- Hypothesis: $y = \text{sign}(w^T x)$
- Define hinge loss

$$l(w, x_t, y_t) = -y_t w^T x_t \mathbb{I}[\text{mistake on } x_t]$$

$$\hat{L}(w) = - \sum_t y_t w^T x_t \mathbb{I}[\text{mistake on } x_t]$$

$$w_{t+1} = w_t - \eta_t \nabla l(w_t, x_t, y_t) = w_t + \eta_t y_t x_t \mathbb{I}[\text{mistake on } x_t]$$

Example 3: Perceptron

- Hypothesis: $y = \text{sign}(w^T x)$

$$w_{t+1} = w_t - \eta_t \nabla l(w_t, x_t, y_t) = w_t + \eta_t y_t x_t \mathbb{I}[\text{mistake on } x_t]$$

- Set $\eta_t = 1$. If mistake on a positive example

$$w_{t+1} = w_t + y_t x_t = w_t + x$$

- If mistake on a negative example

$$w_{t+1} = w_t + y_t x_t = w_t - x$$

Pros & Cons

Pros:

- Widely applicable
- Easy to implement in most cases
- Guarantees for many losses
- Good performance: error/running time/memory etc.

Cons:

- No guarantees for non-convex opt (e.g., those in deep learning)
- Hyper-parameters: initialization, learning rate

Mini-batch

- Instead of one data point, work with a small batch of b points

$$(x_{tb+1}, y_{tb+1}), \dots, (x_{tb+b}, y_{tb+b})$$

- Update rule

$$\theta_{t+1} = \theta_t - \eta_t \nabla \left(\frac{1}{b} \sum_{1 \leq i \leq b} l(\theta_t, x_{tb+i}, y_{tb+i}) \right)$$

- Other variants: variance reduction etc.