Introduction to Machine Learning

Lecture 7
Supervised Learning

- Linear Classifiers
- Perceptron
- Sigmoid unit

- Some materials are courtesy of Vibhave Gogate and Tom Mitchell.
- All pictures belong to their creators.

Linear Classifiers

- Input: a real vector $\boldsymbol{x} = (x_1, \dots, x_n)$, some features
- A weight vector $\boldsymbol{\omega} = (\omega_1, \dots, \omega_n)$
- Bias: ω_0
- A constant feature: $x_0 = 1$ for all instance

- Linear classifier:
- $f(\boldsymbol{x}) = positive\ if\ \sum \omega_i \cdot x_i > 0$
- $f(\boldsymbol{x}) = negative\ if\ \sum \omega_i \cdot x_i \le 0$

Linear Classifiers

- Exercise:
- Is logistic regression a linear classifier?

$$\Pr[y_i = 0|x_i] = \frac{1}{1 + \exp(\omega_o + \sum_j \omega_j \cdot x_{i,j})}$$

$$\Pr[y_i = 1|x_i] = \frac{\exp(\omega_o + \sum_j \omega_j \cdot x_{i,j})}{1 + \exp(\omega_o + \sum_j \omega_j \cdot x_{i,j})}$$

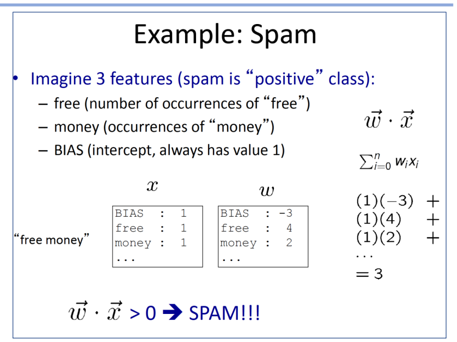
Classification Rule:
Classify a new instance x as 1 iff $\Pr[y_i = 1|x_i] \ge \Pr[y_i = 0|x_i]$

$$\exp(\omega_o + \textstyle\sum_j \omega_j \cdot x_{i,j}) \ge 1 \quad \Rightarrow \quad \omega_o + \sum_j \omega_j \cdot x_{i,j} \ge 0$$

A linear classifier!

Linear Classifiers

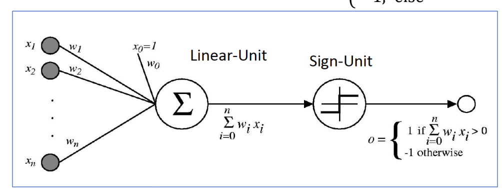
- Example
- (from Gogate)



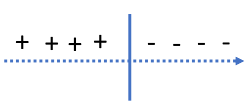
Perceptron

$$o(x_1, \dots, x_n) = \begin{cases} 1, & \text{if } \omega_o + \omega_1 x_1 + \dots + \omega_n x_n > 0 \\ -1, & \text{else} \end{cases}$$

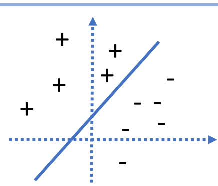
$$o(x_1, \dots, x_n) = \operatorname{sgn}(\boldsymbol{\omega} \cdot \boldsymbol{x}) \text{ where } \operatorname{sgn}(y) = \begin{cases} 1, & \text{if } y > 0 \\ -1, & \text{else} \end{cases}$$



Decision Surface of Perceptron

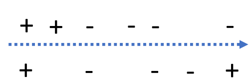


$\omega_o + \omega_1 x_1 > 0$



$\omega_o + \omega_1 x_1 + \omega_2 x_2 > 0$

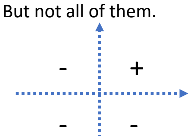
- Representation Power of Perceptron:
- All **linearly separable** data.



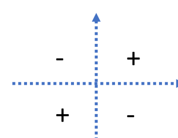
Decision Surface of Perceptron

- Representation Power of Perceptron:
- All **linearly separable** data.

- Special cases: Boolean function $x_i = 0$ or 1.
- Perceptron can represent some Boolean functions such as AND and OR.
- But not all of them.



AND



XOR

Training a Perceptron

- Training a Perceptron.
- Method 1: perceptron training rule
- Method 2: LSM and gradient descent.

$$o(x_1, \dots, x_n) = \begin{cases} 1, & \text{if } \omega_o + \omega_1 x_1 + \dots + \omega_n x_n > 0 \\ -1, & \text{else} \end{cases}$$

1	2	3	3	+
2	6	3	3	-
9	2	4	4	+
2	7	5	3	-

Initialize ω_i as some small values

Update $\omega_i \leftarrow \omega_i + \Delta\omega_i$

Until some criteria met

Training a Perceptron

- **Method 1: perceptron training rule**

$$o(x_1, \dots, x_n) = \begin{cases} 1, & \text{if } \omega_o + \omega_1 x_1 + \dots + \omega_n x_n > 0 \\ -1, & \text{else} \end{cases}$$

Update $\omega_i \leftarrow \omega_i + \Delta\omega_i$

$\Delta\omega_i = \eta(t - o)x_i$

Do until converge
For each x in D
For each ω_i
$\omega_i \leftarrow \omega_i + \Delta\omega_i$

- t: target value
- o: current prediction
- η: learning rate. small value.

Training a Perceptron

- **Method 1: perceptron training rule**

$$o(x_1, \dots, x_n) = \begin{cases} 1, & \text{if } \omega_o + \omega_1 x_1 + \dots + \omega_n x_n > 0 \\ -1, & \text{else} \end{cases}$$

Update $\omega_i \leftarrow \omega_i + \Delta\omega_i$

$\Delta\omega_i = \eta(t - o)x_i$

- t: target value
- o: current prediction
- η: learning rate. small value.

Training a Perceptron

- **Method 1: perceptron training rule**

Do until converge
For each x in D
For each ω_i
$\omega_i \leftarrow \omega_i + \Delta\omega_i$

- Converges if data is linearly separable.
- If learning rate is small enough
- May not converge if data is not linearly separable
- Your training data may not be linearly separable, even if the underlying truth is linear. Noise.
- Any approach that always converges? **LSM+Gradient Descent.**

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**
- Linear unit: $o_l(\boldsymbol{x}) = \omega_o + \omega_1 x_1 + \dots + \omega_n x_n$
- Define Error $E_D(\boldsymbol{\omega}) = \frac{1}{2}\sum_{d \in D}\left(t_d - o_l(d)\right)^2$
- Least squares method.

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**
- Linear unit: $o_l(\boldsymbol{x}) = \omega_o + \omega_1 x_1 + \dots + \omega_n x_n$
- Define Error $E_D(\boldsymbol{\omega}) = \frac{1}{2}\sum_{d \in D}\left(t_d - o_l(d)\right)^2$
- Least squares method.
- **Gradient descent.**

Do until converge
For each d in D
For each ω_i
$\omega_i \leftarrow \omega_i + \Delta\omega_i$

Gradient:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta\vec{w} = -\eta\nabla E[\vec{w}]$$

I.e.:

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i}$$

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**
- Linear unit: $o_l(\boldsymbol{x}) = \omega_o + \omega_1 x_1 + \dots + \omega_n x_n$
- $E_D(\boldsymbol{\omega}) = \frac{1}{2}\sum_{d \in D}\left(t_d - o_l(d)\right)^2$

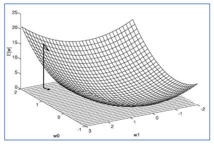
Update $\omega_i \leftarrow \omega_i + \Delta\omega_i$

- $\frac{\partial E_D}{\partial \omega_i} = \sum_d (t_d - o_l(d))(-x_{d,i})$
- $\Delta\omega_i = -\eta\frac{\partial E_D}{\partial \omega_i}$

Do until converge
For each ω_i
$\omega_i \leftarrow \omega_i + \eta\sum_d (t_d - o_l(d))(x_{d,i})$

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**



- Linear unit: $o_l(\boldsymbol{x}) = \omega_o + \omega_1 x_1 + \dots + \omega_n x_n$
- $E_D(\boldsymbol{\omega}) = \frac{1}{2}\sum_{d \in D}\left(t_d - o_l(d)\right)^2$
- $\frac{\partial E_D}{\partial \omega_i} = \sum_d (t_d - o_l(d))(-x_{d,i})$
- $\Delta\omega_i = -\eta\frac{\partial E_D}{\partial \omega_i}$

Converges to the hypothesis that can minimize the error.
If the learning rate is sufficiently small
Even if data is not linearly separable.
Even if data has noise.

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**
- Batch Mode:
- Define the error for the whole training data, and consider it as a whole.
- $E_D(\boldsymbol{\omega}) = \frac{1}{2}\sum_{d \in D}\left(t_d - o_l(d)\right)^2$
- Incremental Mode (online mode):
- Define the error for each training instance, and consider it one by one.
- $E_d(\boldsymbol{\omega}) = \frac{1}{2}\left(t_d - o_l(d)\right)^2$

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**
- **Batch Mode:**
- Define the error for the whole training data, and consider it as a whole.
- $E_D(\boldsymbol{\omega}) = \frac{1}{2}\sum_{d \in D}\left(t_d - o_l(d)\right)^2$

Do until converge
For each ω_i, update ω_i
$\omega_i \leftarrow \omega_i - \eta\frac{\partial E_D}{\partial \omega_i}$

- **Incremental (Online)Mode:**
- Define the error for each training instance, and consider it one by one.
- $E_d(\boldsymbol{\omega}) = \frac{1}{2}\left(t_d - o_l(d)\right)^2$

Do until converge
For each d in D
For each ω_i, update ω_i
$\omega_i \leftarrow \omega_i - \eta\frac{\partial E_d}{\partial \omega_i}$

Training a Perceptron

- **Method 2: LSM and Gradient Descent.**
- **Batch Mode vs Incremental (Online) Mode:**
- Incremental mode can approximate batch model with an arbitrary small error.
- Incremental mode can better avoid local minima. Update ω more frequently. It is faster. It deals with dynamic datasets. Learning rate is usually smaller in incremental mode.
- Batch mode is more stable. Easy to check convergence.

Do until converge
For each ω_i, update ω_i
$$\omega_i \leftarrow \omega_i - \eta \frac{\partial E_D}{\partial \omega_i}$$

Do until converge
For each d in D
For each ω_i, update ω_i
$$\omega_i \leftarrow \omega_i - \eta \frac{\partial E_d}{\partial \omega_i}$$

Training a Perceptron

Two Methods

- **Perceptron Rule:**
- $\Delta\omega_i = \eta(t-o)x_i$
- Consider the threshold error output by the perceptron.
- Converges if data is separable by a perceptron.

- **LSM and Gradient Descent :**
- $\frac{1}{2}\sum_{d\in D}\left(t_d - o_l(d)\right)^2$
- Consider the un-threshold error.
- Always converges to the hypothesis that can minimize the error.

Minimizing the error does not necessarily minimize the misclassified samples.

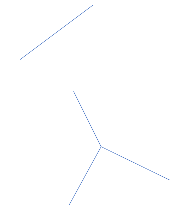
Training a Perceptron

- **Multi Class Perceptron**

$$o(x_1,\dots,x_n) = \begin{cases} 1, & \text{if } \omega_o + \omega_1 x_1 + \cdots + \omega_n x_n > 0 \\ -1, & \text{else} \end{cases}$$

$$f(x) = \omega_o + \omega_1 x_1 + \cdots + \omega_n x_n$$

- Class: {red, orange, blue}
- Task 1: red or not red. $f_r(x)$
- Task 2: orange or not orange. $f_o(x)$
- Task 3: blue or not blue. $f_b(x)$
- Class(x)= $\text{argmax}_{r,o,b}\{f_r(x), f_o(x), f_b(x)\}$

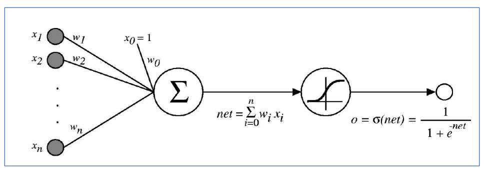


Sigmoid Unit

- Input : $(x_1,\dots,x_n)$ real vector
- Output: real value

$$o(x_1,\dots,x_n) = \frac{1}{1+e^{-\boldsymbol{\omega}\cdot\boldsymbol{x}}}$$

$$o(x_1,\dots,x_n) = \sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}) \text{ where } \sigma(y) = \frac{1}{1+e^{-y}}$$



Sigmoid Unit

- Input : $(x_1,\dots,x_n)$ real vector
- Output: real value
- Parameters: $(\omega_0,\dots,\omega_n)$
- Train a sigmoid unit.
- Define the error
- **Calculate partial derivatives**

$$o(x_1,\dots,x_n) = \frac{1}{1+e^{-\boldsymbol{\omega}\cdot\boldsymbol{x}}} = \sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}) \qquad \sigma(y) = \frac{1}{1+e^{-y}}$$

Sigmoid Unit

- Input : $(x_1,\dots,x_n)$ real vector
- Output: real value
- Parameters: $(\omega_0,\dots,\omega_n)$
- Train a sigmoid unit.
- Define the error
- **Calculate partial derivatives**

$$\begin{aligned}\frac{\partial\,\sigma(y)}{\partial\, y} &= \frac{\partial(1+e^{-y})^{-1}}{\partial\, y} \\ &= -(1+e^{-y})^{-2}\cdot\frac{\partial(1+e^{-y})}{\partial\, y} \\ &= -(1+e^{-y})^{-2}\cdot e^{-y}\frac{\partial -y}{\partial\, y} \\ &= -(1+e^{-y})^{-2}\cdot e^{-y}(-1) \\ &= \frac{e^{-y}}{(1+e^{-y})^2} = \frac{e^{-y}}{(1+e^{-y})(1+e^{-y})} \\ &= \sigma(y)\,(1-\sigma(y))\end{aligned}$$

$$o(x_1,\dots,x_n) = \frac{1}{1+e^{-\boldsymbol{\omega}\cdot\boldsymbol{x}}} = \sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}) \qquad \sigma(y) = \frac{1}{1+e^{-y}}$$

Sigmoid Unit

- Input : $(x_1,\dots,x_n)$ real vector
- Output: real value
- Parameters: $(\omega_0,\dots,\omega_n)$
- Train a sigmoid unit.
- Define the error
- **Calculate partial derivatives**

$$\frac{\partial\,\sigma(y)}{\partial\, y} = \frac{\partial(1+e^{-y})^{-1}}{\partial\, y} = \sigma(y)\,(1-\sigma(y))$$

$$\frac{\partial\,\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x})}{\partial\,\omega_i} = \frac{\partial\,\boldsymbol{\omega}\cdot\boldsymbol{x}}{\partial\,\omega_i}\cdot\frac{\partial\,\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x})}{\partial\,\boldsymbol{\omega}\cdot\boldsymbol{x}}$$
$$= x_i\cdot\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x})\,(1-\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}))$$

$$o(x_1,\dots,x_n) = \frac{1}{1+e^{-\boldsymbol{\omega}\cdot\boldsymbol{x}}} = \sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}) \qquad \sigma(y) = \frac{1}{1+e^{-y}}$$

Sigmoid Unit

- Input : $(x_1,\dots,x_n)$ real vector
- Output: real value
- Parameters: $(\omega_0,\dots,\omega_n)$
- Train a sigmoid unit.
- **Define the error**
- Calculate partial derivatives

$$\frac{\partial\,\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x})}{\partial\,\omega_i} = x_i\cdot\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x})\,(1-\sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}))$$

$$E_D(\omega) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

$$\frac{\partial E_D}{\partial\,\omega_i} = \sum_{d\in D} -(t_d-o_d)\frac{\partial o_d}{\partial\,\omega_i} = \sum_{d\in D} -(t_d-o_d)\cdot x_{d,i}\cdot o_d(1-o_d)$$

$$o_d = o(x_1,\dots,x_n) = \frac{1}{1+e^{-\boldsymbol{\omega}\cdot\boldsymbol{x}}} = \sigma(\boldsymbol{\omega}\cdot\boldsymbol{x}) \qquad \sigma(y) = \frac{1}{1+e^{-y}}$$

Perceptron and Sigmoid Unit

- **Perceptron:**
- $o(x_1,\dots,x_n) = \begin{cases} 1, & \text{if } \boldsymbol{\omega}\cdot\boldsymbol{x} > 0 \\ -1, & \text{else} \end{cases}$
- Output a class
- Not differentiable.

- **Sigmoid Unit:**
- $o(x_1,\dots,x_n) = \frac{1}{1+e^{-\boldsymbol{\omega}\cdot\boldsymbol{x}}}$
- Output a value
- Differentiable.
- (0,1)

- **Linear:**
- $o(x_1,\dots,x_n) = \boldsymbol{\omega}\cdot\boldsymbol{x}$
- Output a value
- Differentiable.
- unbounded

Summary

- Perceptron (definition)
- Two methods to train a perceptron
- Batch mode vs Online mode

- Sigmoid (definition)
- Formulas to train a sigmoid.