

CS 1571 Introduction to AI Lecture 26

Supervised learning. Linear and logistic regression.

Milos Hauskrecht

milos@cs.pitt.edu

5329 Sennott Square

CS 1571 Intro to AI

Announcements

Homeworks:

- Homework 10 due on Tuesday, December 2, 2003

Final exam: December 08, 2003 at 10:00-11:50am

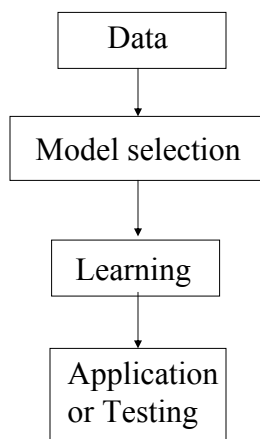
- Location: Sennott Square 5321
- Closed book
- Cumulative
- Format similar to the midterm exam

CS 1571 Intro to AI

Topics

- **Supervised Learning**
- **Linear regression**
 - **Exact Solution**
 - **Gradient methods**
- **Logistic regression**
 - **Motivation**
 - **Decision boundary**
 - **Gradient methods**
- **Extensions to nonlinear models**

Design of a learning system



Design of a learning system.

1. Data: $D = \{D_1, D_2, \dots, D_n\}$

2. Model selection:

- **Select a model** or a set of models (with parameters)

E.g. $y = ax + b$

- **Select the error function** to be optimized

E.g. $\frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$

3. Learning:

- **Find the set of parameters optimizing the error function**
 - The model and parameters with the smallest error

4. Application:

- **Apply the learned model**
 - E.g. predict y s for new inputs $\mathbf{x}$ using learned $f(\mathbf{x})$

CS 1571 Intro to AI

Supervised learning

Data: $D = \{D_1, D_2, \dots, D_n\}$ a set of n examples

$$D_i = \langle \mathbf{x}_i, y_i \rangle$$

$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d})$ is an input vector of size d

y_i is the desired output (given by a teacher)

Objective: learn the mapping $f: X \rightarrow Y$

$$\text{s.t. } y_i \approx f(\mathbf{x}_i) \text{ for all } i = 1, \dots, n$$

- **Regression:** Y is **continuous**

Example: earnings, product orders $\rightarrow$ company stock price

- **Classification:** Y is **discrete**

Example: handwritten digit



digit label

CS 1571 Intro to AI

Linear regression.

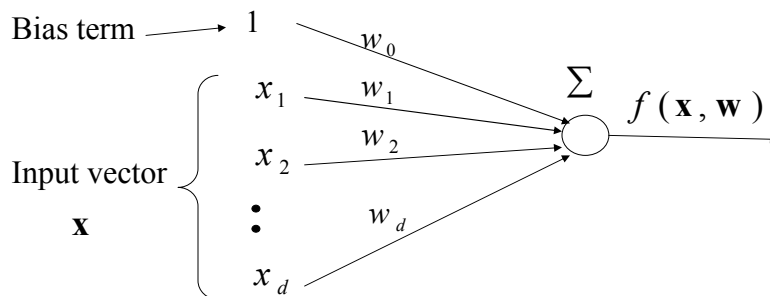
CS 1571 Intro to AI

Linear regression

- **Function** $f: X \rightarrow Y$ is a linear combination of input components

$$f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \dots w_dx_d = w_0 + \sum_{j=1}^d w_jx_j$$

$w_0, w_1, \dots, w_k$ - **parameters (weights)**



CS 1571 Intro to AI

Linear regression

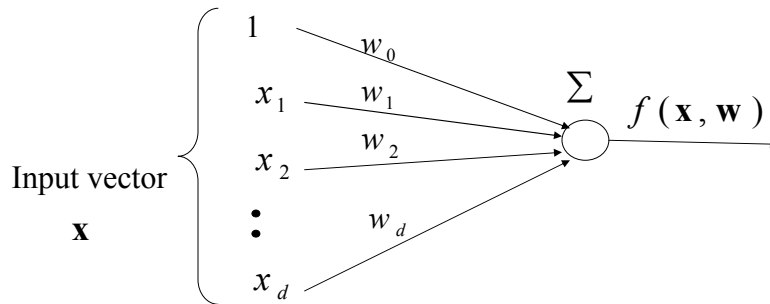
- **Shorter (vector) definition of the model**

- Include bias constant in the input vector

$$\mathbf{x} = (1, x_1, x_2, \dots, x_d)$$

$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d = \mathbf{w}^T \mathbf{x}$$

$w_0, w_1, \dots, w_k$ - **parameters (weights)**



CS 1571 Intro to AI

Linear regression. Error.

- **Data:** $D_i = \langle \mathbf{x}_i, y_i \rangle$
- **Function:** $\mathbf{x}_i \rightarrow f(\mathbf{x}_i)$
- We would like to have $y_i \approx f(\mathbf{x}_i)$ for all $i = 1, \dots, n$

- **Error function**

- measures how much our predictions deviate from the desired answers

Mean-squared error
$$J_n = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i))^2$$

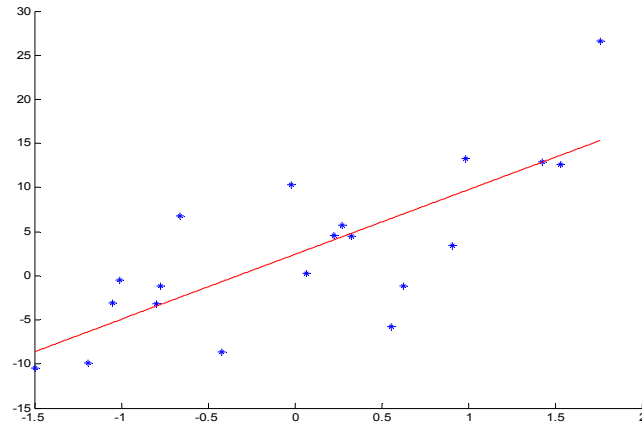
- **Learning:**

We want to find the weights minimizing the error !

CS 1571 Intro to AI

Linear regression. Example

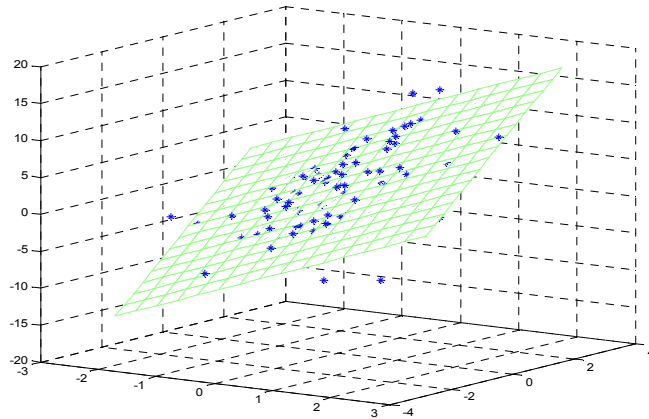
- 1 dimensional input $\mathbf{x} = (x_1)$



CS 1571 Intro to AI

Linear regression. Example.

- 2 dimensional input $\mathbf{x} = (x_1, x_2)$



CS 1571 Intro to AI

Linear regression. Optimization.

- We want the **weights minimizing the error**

$$J_n = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i))^2 = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,j} = 0$$

- Vector of derivatives:**

$$\text{grad}_{\mathbf{w}} (J_n(\mathbf{w})) = \nabla_{\mathbf{w}} (J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \bar{\mathbf{0}}$$

Linear regression. Optimization.

- $\text{grad}_{\mathbf{w}} (J_n(\mathbf{w})) = \bar{\mathbf{0}}$ defines a set of equations in $\mathbf{w}$

$$\frac{\partial}{\partial w_0} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) = 0$$

$$\frac{\partial}{\partial w_1} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,1} = 0$$

...

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,j} = 0$$

...

$$\frac{\partial}{\partial w_d} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,d} = 0$$

Solving linear regression

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,j} = 0$$

By rearranging the terms we get a **system of linear equations** with $d+1$ unknowns

$$\mathbf{Aw} = \mathbf{b}$$

$$\begin{aligned} w_0 \sum_{i=1}^n x_{i,0} 1 + w_1 \sum_{i=1}^n x_{i,1} 1 + \dots + w_j \sum_{i=1}^n x_{i,j} 1 + \dots + w_d \sum_{i=1}^n x_{i,d} 1 &= \sum_{i=1}^n y_i 1 \\ w_0 \sum_{i=1}^n x_{i,0} x_{i,1} + w_1 \sum_{i=1}^n x_{i,1} x_{i,1} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,1} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,1} &= \sum_{i=1}^n y_i x_{i,1} \\ &\vdots \\ w_0 \sum_{i=1}^n x_{i,0} x_{i,j} + w_1 \sum_{i=1}^n x_{i,1} x_{i,j} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,j} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,j} &= \sum_{i=1}^n y_i x_{i,j} \\ &\vdots \end{aligned}$$

CS 1571 Intro to AI

Solving linear regression

- The optimal set of weights satisfies:

$$\nabla_{\mathbf{w}} (J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \mathbf{0}$$

Leads to a **system of linear equations (SLE)** with $d+1$ unknowns of the form

$$\mathbf{Aw} = \mathbf{b}$$

$$w_0 \sum_{i=1}^n x_{i,0} x_{i,j} + w_1 \sum_{i=1}^n x_{i,1} x_{i,j} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,j} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,j} = \sum_{i=1}^n y_i x_{i,j}$$

Solution to SLE: ?

CS 1571 Intro to AI

Solving linear regression

- The optimal set of weights satisfies:

$$\nabla_{\mathbf{w}} (J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \bar{\mathbf{0}}$$

Leads to a **system of linear equations (SLE)** with $d+1$ unknowns of the form

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

$$w_0 \sum_{i=1}^n x_{i,0} x_{i,j} + w_1 \sum_{i=1}^n x_{i,1} x_{i,j} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,j} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,j} = \sum_{i=1}^n y_i x_{i,j}$$

Solution to SLE:

$$\mathbf{w} = \mathbf{A}^{-1} \mathbf{b}$$

- matrix inversion

Gradient descent solution

Goal: the weight optimization in the linear regression model

$$J_n = \text{Error}(\mathbf{w}) = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

An alternative to SLE solution:

- Gradient descent**

Idea:

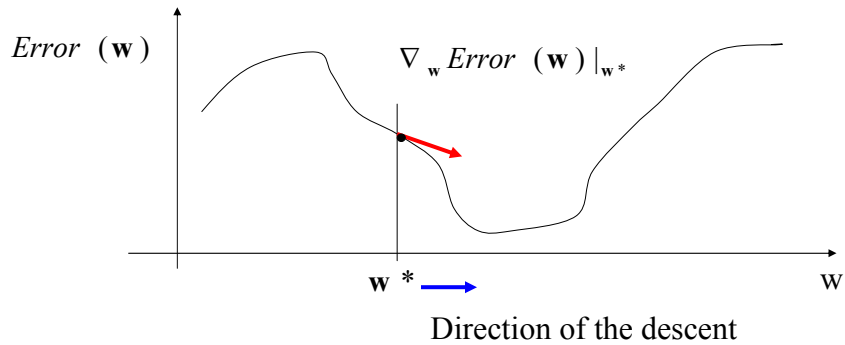
- Adjust weights in the direction that improves the Error
- The gradient tells us what is the right direction

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \text{Error}_i(\mathbf{w})$$

$\alpha > 0$ - a **learning rate** (scales the gradient changes)

Gradient descent method

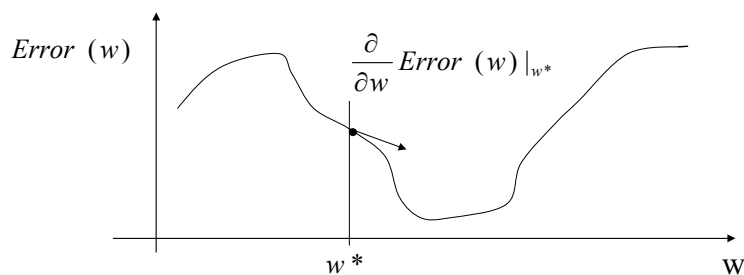
- Descend using the gradient information



- Change the value of w according to the gradient

$$w \leftarrow w - \alpha \nabla_w Error_i(w)$$

Gradient descent method



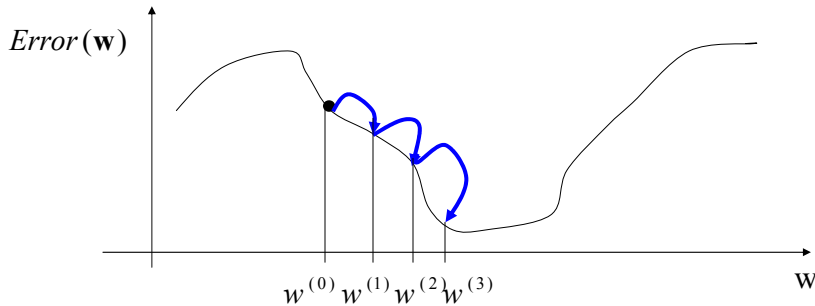
- New value of the parameter

$$w_j \leftarrow w_j^* - \alpha \frac{\partial}{\partial w_j} Error(w)|_{w^*} \quad \text{For all } j$$

$\alpha > 0$ - a learning rate (scales the gradient changes)

Gradient descent method

- Iteratively approaches the optimum of the Error function



CS 1571 Intro to AI

Online gradient algorithm

- The error function is defined for the whole dataset D

$$J_n = Error(w) = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(x_i, w))^2$$

- error for a sample** $D_i = \langle x_i, y_i \rangle$

$$J_{\text{online}} = Error_i(w) = \frac{1}{2} (y_i - f(x_i, w))^2$$

- Online gradient method: changes weights after every sample**

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} Error_i(w)$$

- vector form:**

$$w \leftarrow w - \alpha \nabla_w Error_i(w)$$

$\alpha > 0$ - Learning rate that depends on the number of updates

CS 1571 Intro to AI

Online gradient method

Linear model

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

On-line error $J_{\text{online}} = \text{Error}_i(\mathbf{w}) = \frac{1}{2}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2$

On-line algorithm: generates a sequence of online updates

(i)-th update step with : $D_i = \langle \mathbf{x}_i, y_i \rangle$

j-th weight:

$$w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial \text{Error}_i(\mathbf{w})}{\partial w_j} \Big|_{\mathbf{w}^{(i-1)}}$$

$$w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)}))x_{i,j}$$

Annealed learning rate: $\alpha(i) \approx \frac{1}{i}$

- Gradually rescales changes in weights

CS 1571 Intro to AI

Online regression algorithm

Online-linear-regression (D , number of iterations)

Initialize weights $\mathbf{w} = (w_0, w_1, w_2 \dots w_d)$

for $i=1:1$: number of iterations

do select a data point $D_i = (\mathbf{x}_i, y_i)$ from D

set $\alpha = 1/i$

update weight vector

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y_i - f(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$$

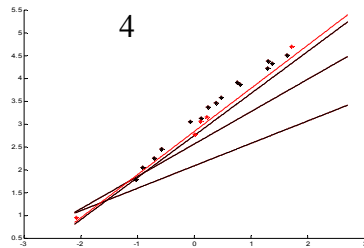
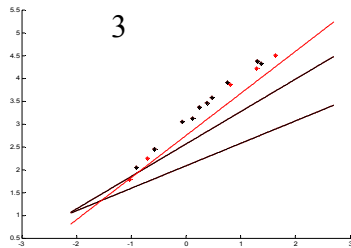
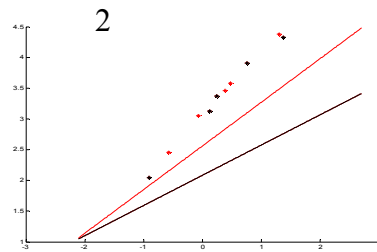
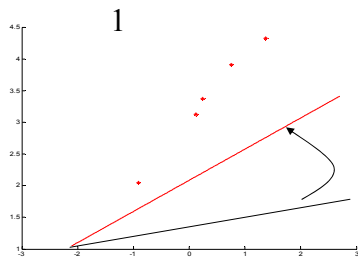
end for

return weights $\mathbf{w}$

- **Advantages:** very easy to implement, continuous data streams

CS 1571 Intro to AI

On-line learning. Example



CS 1571 Intro to AI

Logistic regression

CS 1571 Intro to AI

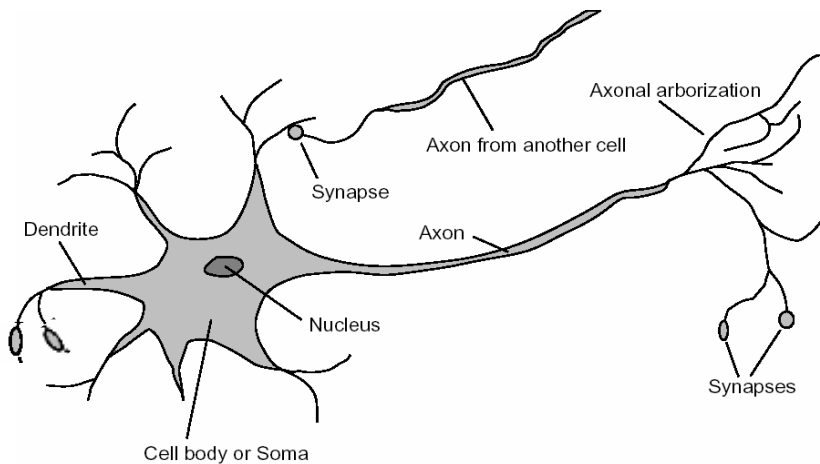
Binary classification

- **Two classes** $Y = \{0,1\}$
- Our goal is to learn how to correctly classify two types of examples
 - Class 0 – labeled as 0,
 - Class 1 – labeled as 1
- We would like to learn $f : X \rightarrow \{0,1\}$
- **First step:** we need to devise a model of the function f
- **Inspiration:** neuron (nerve cells)

CS 1571 Intro to AI

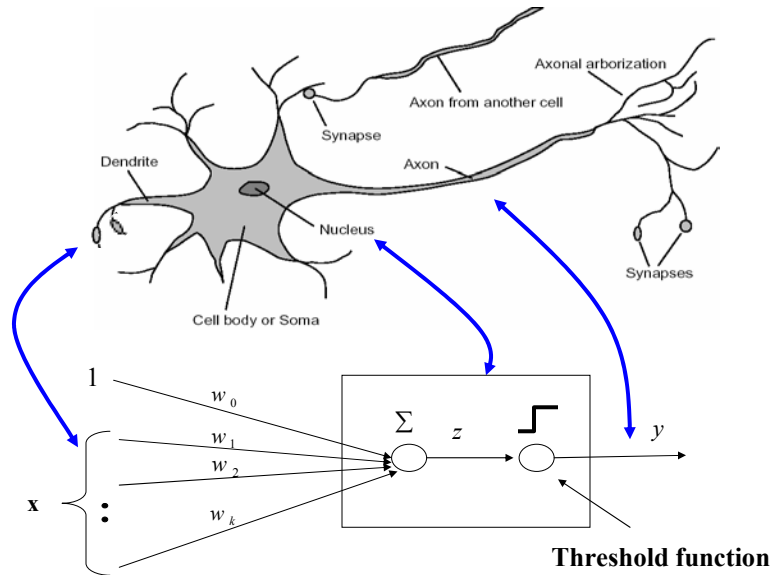
Neuron

- **neuron (nerve cell) and its activities**



CS 1571 Intro to AI

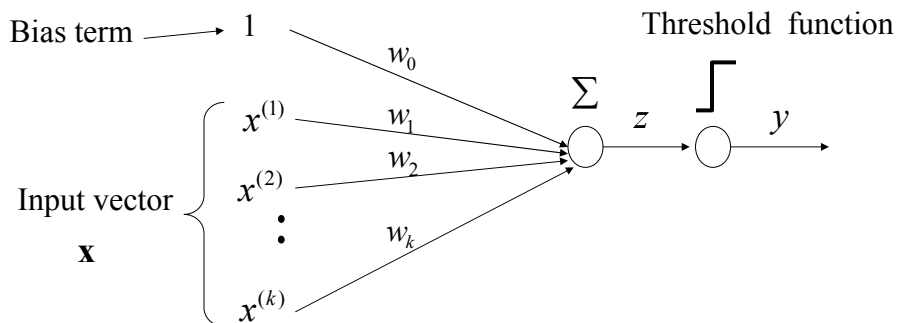
Neuron-based binary classification model



CS 1571 Intro to AI

Neuron-based binary classification

- **Function we want to learn** $f : X \rightarrow \{0,1\}$



CS 1571 Intro to AI

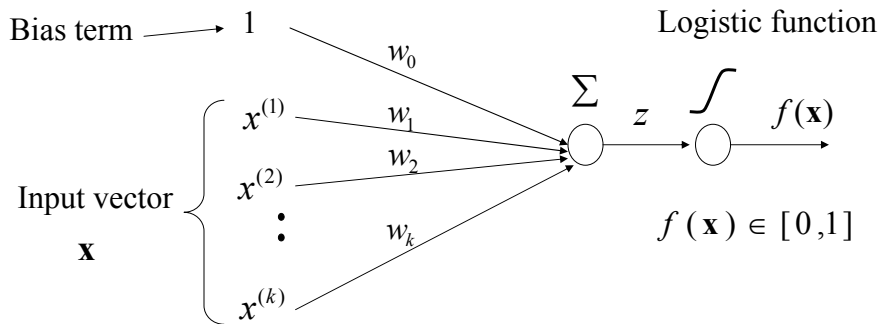
Logistic regression model

- A function model with smooth switching:

$$f(\mathbf{x}) = g(w_0 + w_1 x^{(1)} + \dots w_k x^{(k)})$$

where w are parameters of the models

and $g(z)$ is a **logistic function** $g(z) = 1/(1 + e^{-z})$

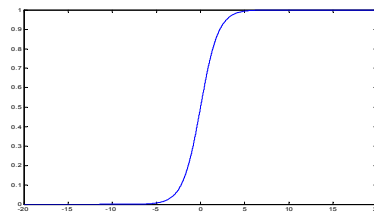


CS 1571 Intro to AI

Logistic function

function

$$g(z) = \frac{1}{(1 + e^{-z})}$$



- also referred to as **sigmoid function**
- **replaces threshold function** with smooth switching
- takes a real number and outputs the number in the interval $[0, 1]$
- Output of the logistic regression has a probabilistic interpretation

$$f(\mathbf{x}) = p(y = 1 | \mathbf{x}) \quad - \text{Probability of class 1}$$

CS 1571 Intro to AI

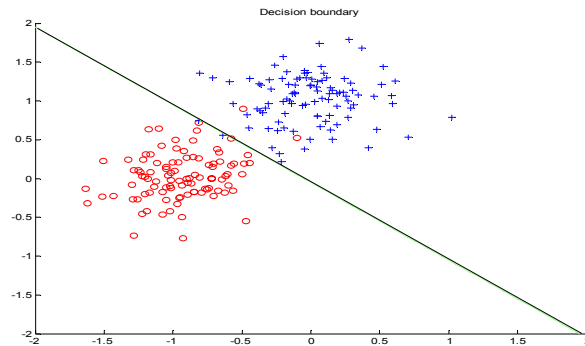
Logistic regression. Decision boundary

Classification with the logistic regression model:

If $f(\mathbf{x}) \geq 1/2$ then choose class 1
Else choose class 0

Logistic regression model defines a linear decision boundary

Example:



CS 1571 Intro to AI

Optimization of weights

- **Two classes:** $Y = \{0,1\}$
- **Data:** $D = \{d_1, d_2, \dots, d_n\}$
 $d_i = \langle \mathbf{x}_i, y_i \rangle$
- We want to find the set of weight $\mathbf{w}$ that explain the data the best
- **Zero-one error function**

$$\text{Error}(x_i, y_i) = \begin{cases} 1 & f(\mathbf{x}_i, \mathbf{w}) \neq y_i \\ 0 & f(\mathbf{x}_i, \mathbf{w}) = y_i \end{cases}$$

- Error we would like to minimize: $E_{(x,y)}(\text{Error}(x, y))$
- The error is minimized if we choose:

$$y = 1 \quad \text{if} \quad p(y = 1 | \mathbf{x}, \mathbf{w}) > p(y = 0 | \mathbf{x}, \mathbf{w})$$
$$y = 0 \quad \text{otherwise}$$

CS 1571 Intro to AI

Logistic regression. Parameter optimization.

- The error is minimized if we choose:

$$\begin{aligned} y &= 1 && \text{if } p(y = 1 \mid \mathbf{x}, \mathbf{w}) > p(y = 0 \mid \mathbf{x}, \mathbf{w}) \\ y &= 0 && \text{otherwise} \end{aligned}$$

- We construct a probabilistic version of the error function based on the **likelihood of the data**

$$L(D, \mathbf{w}) = P(D \mid \mathbf{w}) = \prod_{i=1}^n P(y = y_i \mid \mathbf{x}_i, \mathbf{w})$$

- Likelihood of the data**
 - Measures the goodness of fit

$$Error(D, \mathbf{w}) = -L(D, \mathbf{w})$$

Logistic regression: parameter estimation

- Error function decomposes to online error components**

$$Error(D, \mathbf{w}) = \sum_{i=1}^n Error_i(D_i, \mathbf{w})$$

- Derivatives of the online error component for the LR model (in terms of weights)**

$$\begin{aligned} \frac{\partial}{\partial w_0} Error_i(D_i, \mathbf{w}) &= -(y_i - f(\mathbf{x}_i, \mathbf{w})) \\ &\vdots \\ \frac{\partial}{\partial w_j} Error_i(D_i, \mathbf{w}) &= -(y_i - f(\mathbf{x}_i, \mathbf{w})) x_{i,j} \end{aligned}$$

Logistic regression. Online gradient.

- We want to optimize the online Error
- **On-line gradient update for the jth weight and ith step**

$$w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha \frac{\partial}{\partial w_j} [Error_i(D_i, \mathbf{w}) |_{\mathbf{w}^{(i-1)}}]$$

- **(i)th update for the logistic regression** $D_i = \langle \mathbf{x}_i, y_i \rangle$

J-th weight

$$w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i) (y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)})) x_{i,j}$$

α - annealed learning rate (depends on the number of updates)

The same update rule as used in the linear regression !!!

CS 1571 Intro to AI

Online logistic regression algorithm

Online-logistic-regression (D , number of iterations)

initialize weights $w_0, w_1, w_2 \dots w_k$

for $i=1:1$: number of iterations

do select a data point $d=\langle \mathbf{x}, y \rangle$ from D

set $\alpha=1/i$

update weights (in parallel)

$$w_0 = w_0 + \alpha[y - f(\mathbf{x}, \mathbf{w})]$$

$\vdots$

$$w_j = w_j + \alpha[y - f(\mathbf{x}, \mathbf{w})]x_j$$

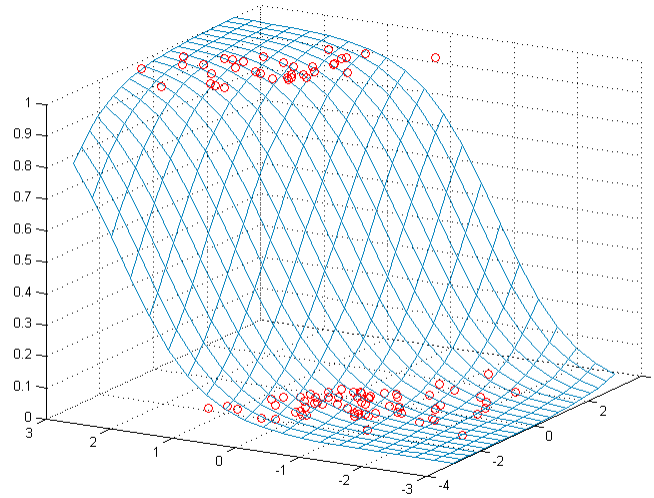
end for

return weights

CS 1571 Intro to AI

Online algorithm. Example.

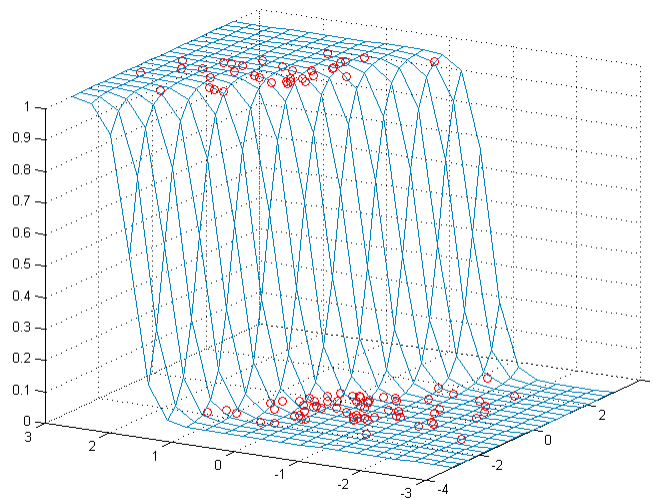
$w_1=0.91773$ $w_2=1.6297$ bias= -0.91898



CS 1571 Intro to AI

Online algorithm. Example.

$w_1=3.5934$ $w_2=6.9126$ bias= -3.6709



CS 1571 Intro to AI

Online algorithm. Example.

$w_1=19.9144$ $w_2=39.7033$ bias= -20.8644

