

## CS 2750 Machine Learning Lecture 9

### Linear models for classification

Milos Hauskrecht  
[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)  
5329 Sennott Square

---

### Supervised learning

**Data:**  $D = \{d_1, d_2, \dots, d_n\}$  a set of  $n$  examples

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

$\mathbf{x}_i$  is input vector, and  $y$  is desired output (given by a teacher)

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

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

**Two types of problems:**

- **Regression:**  $X$  discrete or continuous  $\rightarrow$   
 $Y$  is **continuous**
- **Classification:**  $X$  discrete or continuous  $\rightarrow$   
 $Y$  is **discrete**

**Last lecture**



## Supervised learning examples

- **Classification:** Y is discrete

```
# #####
  ##
  ##
#####
  ##
#   ##
#  ##
#####
```



Label "3"

```
50419213
47604567
20271864
23591762
86375809
87609757
23949216
56799370
```

Handwritten digit (array of 0,1s)

Data:

```
#####
  ##
  ##
#####
  ##
#   ##
#  ##
#####
```



image



digit

3

7

5

....

## Classification

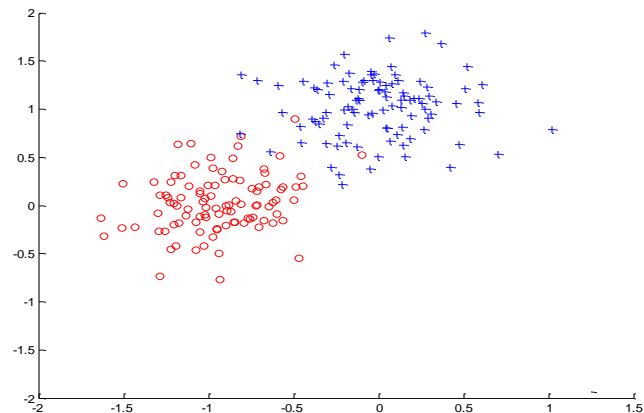
- **Data:**  $D = \{d_1, d_2, \dots, d_n\}$   
 $d_i = \langle \mathbf{x}_i, y_i \rangle$ 
  - $y_i$  represents a discrete class value
- **Goal: learn**  $f : X \rightarrow Y$
- **Binary classification**
  - A special case when  $Y \in \{0,1\}$
- **First step:**
  - we need to devise a model of the function f

## Discriminant functions

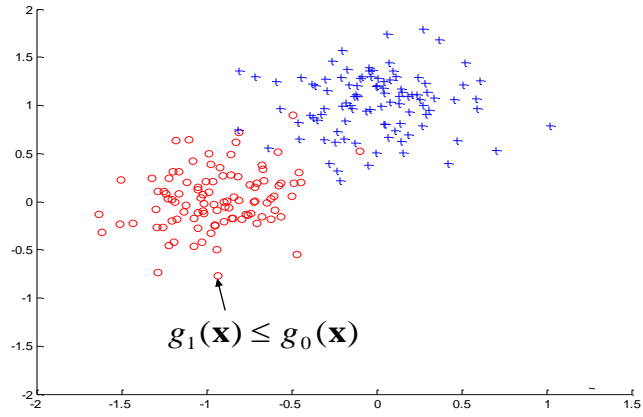
- A common way to represent a classifier is by using
  - Discriminant functions
- Works for both the binary and multi-way classification
- Idea:
  - For every class  $i = 0, 1, \dots, k$  define a function  $g_i(\mathbf{x})$  mapping  $X \rightarrow \mathcal{R}$
  - When the decision on input  $\mathbf{x}$  should be made choose the class with the highest value of  $g_i(\mathbf{x})$

$$y^* = \arg \max_i g_i(\mathbf{x})$$

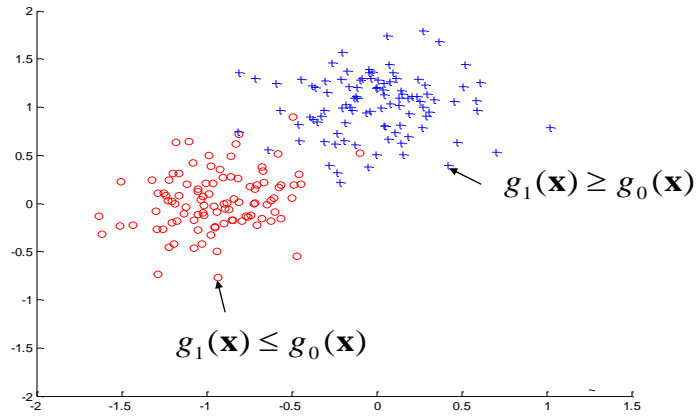
## Discriminant functions



## Discriminant functions

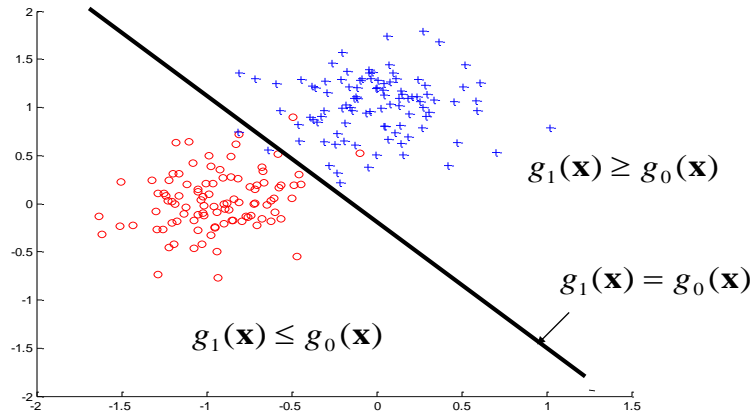


## Discriminant functions

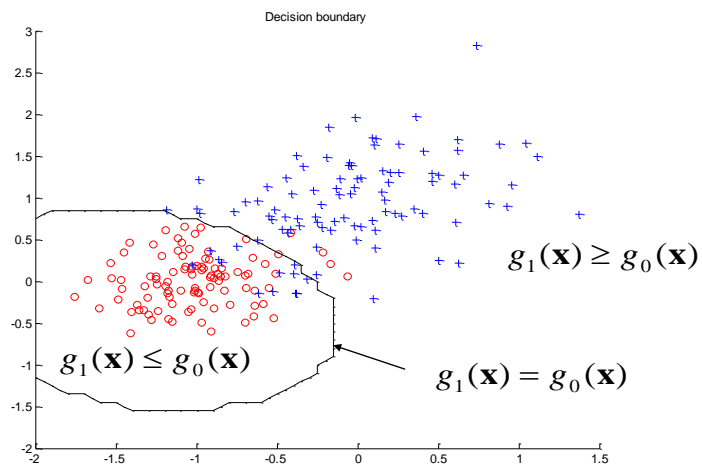


## Discriminant functions

- **Decision boundary:** discriminant functions are equal



## Quadratic decision boundary



## How to design discriminant functions?

- Assume two linear models for classes 0, 1

$$g_1(\mathbf{x}) = \mathbf{w}_1^T \mathbf{x} \qquad g_0(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x}$$

- Class decision:  $y^* = \arg \max_i g_i(\mathbf{x})$

- Training via regression:

– if  $(\mathbf{x}, 1)$

• Train  $g_1(\mathbf{x}) = \mathbf{w}_1^T \mathbf{x}$  with y value 1

• Train  $g_0(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x}$  with y value 0

– if  $(\mathbf{x}, 0)$

• Train  $g_1(\mathbf{x}) = \mathbf{w}_1^T \mathbf{x}$  with y value 0

• Train  $g_0(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x}$  with y value 1

- Use least squares error to find both

$$g_1(\mathbf{x}) = \mathbf{w}_1^T \mathbf{x} \qquad g_0(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x}$$

---

## How to design discriminant functions?

- Previous design used two discriminant functions one for each class  $g_1(\mathbf{x}) = \mathbf{w}_1^T \mathbf{x}$   $g_0(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x}$

- Binary classification is simpler:

– We can use just one set of weights  $\mathbf{w}$

$$g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \qquad g_0(\mathbf{x}) = -\mathbf{w}^T \mathbf{x} = -g_1(\mathbf{x})$$

- Training via regression:

– if  $(\mathbf{x}, 1)$   $y^* = \arg \max_i g_i(\mathbf{x})$

• Train  $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$  with y value 1

– if  $(\mathbf{x}, 0)$

• Train  $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$  with y value -1

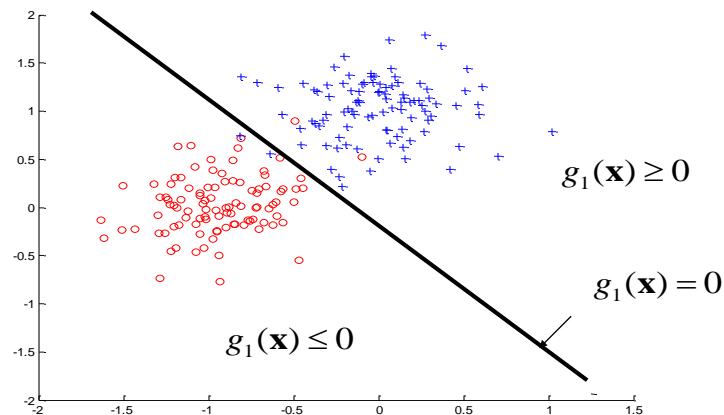
- How to make a class decision?
-

## How to design discriminant functions?

- Previous design used two discriminant functions one for each class
- Binary classification is simpler – only two classes:
  - We can use one set of shared weights  $\mathbf{w}$ 
$$g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \qquad g_0(\mathbf{x}) = -\mathbf{w}^T \mathbf{x} = -g_1(\mathbf{x})$$
- Training via regression:
  - if  $(\mathbf{x}, 1)$ 
    - Train  $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$  with y value 1
  - if  $(\mathbf{x}, 0)$ 
    - Train  $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$  with y value -1
- How to make a class decision?
$$g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x} > 0 \qquad \text{Class 1}$$
$$g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x} < 0 \qquad \text{Class 0}$$

## Discriminant functions and decision boundary

- Linear decision boundary



## How to design discriminant functions?

Property of the above model:

$$g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \quad g_0(\mathbf{x}) = -\mathbf{w}^T \mathbf{x} = -g_1(\mathbf{x})$$

- Defines a linear decision boundary

Limitations of the above model

- Regression is related to a Gaussian
  - But we have only two different values we try to fit
- We would like to have a probabilistic model for classification
  - Is it possible to properly define  $p(y=1|\mathbf{x})$  and  $p(y=0|\mathbf{x})$  ?

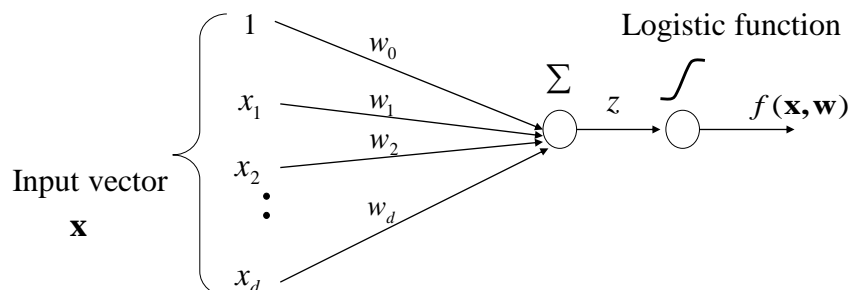
## Logistic regression model

- Defines a linear decision boundary
- Discriminant functions:

$$g_1(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) \quad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T \mathbf{x})$$

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

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$

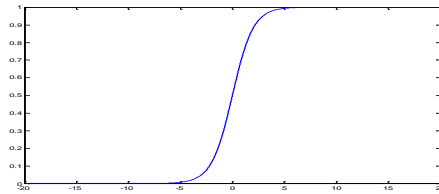




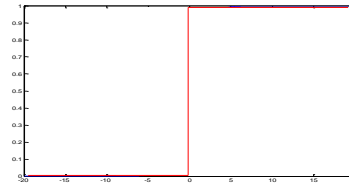
## Logistic function

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

- Is also referred to as a **sigmoid function**
- takes a real number and outputs the number in the interval [0,1]
- Models a smooth switching function; replaces hard threshold function



Logistic (smooth) switching



Threshold (hard) switching

## Logistic regression model

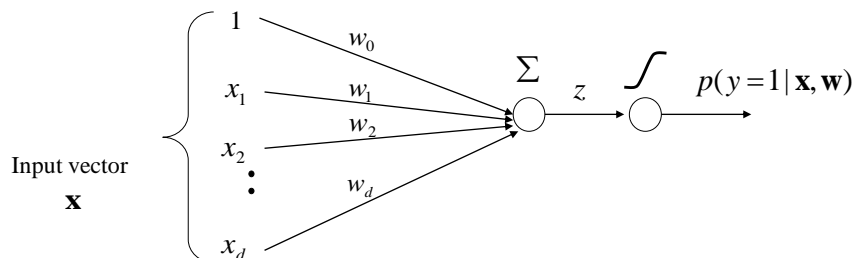
- **Discriminant functions:**

$$g_1(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) \qquad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T \mathbf{x})$$

- Values of discriminant functions vary in interval [0,1]

– **Probabilistic interpretation**

$$f(\mathbf{x}, \mathbf{w}) = p(y = 1 | \mathbf{w}, \mathbf{x}) = g_1(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$



## Logistic regression

- We learn a **probabilistic function**

$$f : X \rightarrow [0,1]$$

- where  $f$  describes the probability of class 1 given  $\mathbf{x}$

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = p(y = 1 | \mathbf{x}, \mathbf{w})$$

**Note that:**

$$p(y = 0 | \mathbf{x}, \mathbf{w}) = 1 - p(y = 1 | \mathbf{x}, \mathbf{w})$$

- Making decisions with the logistic regression model:

?

## Logistic regression

- We learn a **probabilistic function**

$$f : X \rightarrow [0,1]$$

- where  $f$  describes the probability of class 1 given  $\mathbf{x}$

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = p(y = 1 | \mathbf{x}, \mathbf{w})$$

**Note that:**

$$p(y = 0 | \mathbf{x}, \mathbf{w}) = 1 - p(y = 1 | \mathbf{x}, \mathbf{w})$$

- Making decisions with the logistic regression model:

If  $p(y = 1 | \mathbf{x}) \geq 1/2$  then choose **1**  
Else choose **0**

## Linear decision boundary

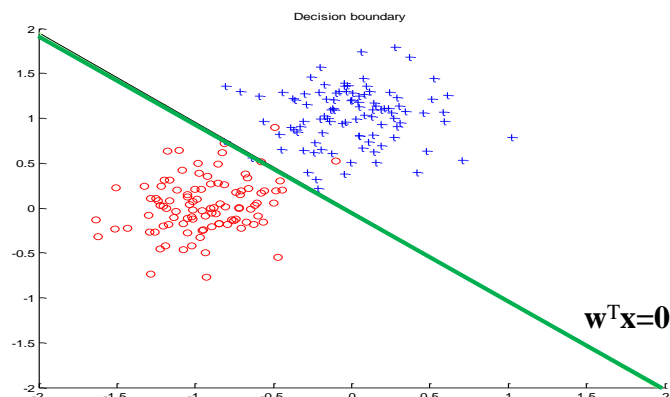
- Logistic regression model defines a **linear decision boundary**
- **Why?**
- **Answer:** Compare two **discriminant functions**.
- **Decision boundary:**  $g_1(\mathbf{x}) = g_0(\mathbf{x})$
- For the boundary it must hold:

$$\log \frac{g_0(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^T \mathbf{x})}{g(\mathbf{w}^T \mathbf{x})} = 0$$

$$\log \frac{g_0(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{\frac{\exp-(\mathbf{w}^T \mathbf{x})}{1 + \exp-(\mathbf{w}^T \mathbf{x})}}{\frac{1}{1 + \exp-(\mathbf{w}^T \mathbf{x})}} = \log \exp-(\mathbf{w}^T \mathbf{x}) = \mathbf{w}^T \mathbf{x} = 0$$

## Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**
- Example:** 2 classes (blue and red points)



## Logistic regression: parameter learning

### Likelihood of outputs

- **Let**  $D_i = \langle \mathbf{x}_i, y_i \rangle$   $\mu_i = p(y_i = 1 | \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x}_i)$

- **Then**

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

- **Find weights  $\mathbf{w}$  that maximize the likelihood of outputs**
  - Apply the log-likelihood trick. The optimal weights are the same for both the likelihood and the log-likelihood

$$\begin{aligned} l(D, \mathbf{w}) &= \log \prod_{i=1}^n \mu_i^{y_i} (1 - \mu_i)^{1-y_i} = \sum_{i=1}^n \log \mu_i^{y_i} (1 - \mu_i)^{1-y_i} = \\ &= \sum_{i=1}^n y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i) \end{aligned}$$

## Logistic regression: parameter learning

- **Notation:**  $\mu_i = p(y_i = 1 | \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x}_i)$
- **Log likelihood**

$$l(D, \mathbf{w}) = \sum_{i=1}^n y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

- **Derivatives of the loglikelihood**

$$\frac{\partial}{\partial w_j} l(D, \mathbf{w}) = \sum_{i=1}^n x_{i,j} (y_i - g(z_i))$$

**Nonlinear in weights !!**

$$\nabla_{\mathbf{w}} l(D, \mathbf{w}) = \sum_{i=1}^n \mathbf{x}_i (y_i - g(\mathbf{w}^T \mathbf{x}_i)) = \sum_{i=1}^n \mathbf{x}_i (y_i - f(\mathbf{w}, \mathbf{x}_i))$$

- **Gradient descent:**

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [-l(D, \mathbf{w})] |_{\mathbf{w}^{(k-1)}}$$

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} + \alpha(k) \sum_{i=1}^n [y_i - f(\mathbf{w}^{(k-1)}, \mathbf{x}_i)] \mathbf{x}_i$$

## Derivation of the gradient

- Log likelihood**  $l(D, \mathbf{w}) = \sum_{i=1}^n y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$

- Derivatives of the loglikelihood**

$$\frac{\partial}{\partial w_j} l(D, \mathbf{w}) = \sum_{i=1}^n \frac{\partial}{\partial z_i} [y_i \log g(z_i) + (1 - y_i) \log(1 - g(z_i))] \frac{\partial z_i}{\partial w_j}$$

### Derivative of a logistic function

$$\frac{\partial z_i}{\partial w_j} = x_{i,j}$$

$$\frac{\partial g(z_i)}{\partial z_i} = g(z_i)(1 - g(z_i))$$

$$\frac{\partial}{\partial z_i} [y_i \log g(z_i) + (1 - y_i) \log(1 - g(z_i))] = y_i \frac{1}{g(z_i)} \frac{\partial g(z_i)}{\partial z_i} + (1 - y_i) \frac{-1}{1 - g(z_i)} \frac{\partial g(z_i)}{\partial z_i}$$

$$= y_i(1 - g(z_i)) + (1 - y_i)(-g(z_i)) = y_i - g(z_i)$$

$$\nabla_{\mathbf{w}} l(D, \mathbf{w}) = \sum_{i=1}^n -\mathbf{x}_i (y_i - g(\mathbf{w}^T \mathbf{x}_i)) = \sum_{i=1}^n -\mathbf{x}_i (y_i - f(\mathbf{w}, \mathbf{x}_i))$$

## Logistic regression. Online gradient descent

- On-line component of the loglikelihood**

$$J_{\text{online}}(D_i, \mathbf{w}) = -[y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)]$$

- On-line learning update for weight  $\mathbf{w}$**   $J_{\text{online}}(D_k, \mathbf{w})$

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [J_{\text{online}}(D_k, \mathbf{w})] |_{\mathbf{w}^{(k-1)}}$$

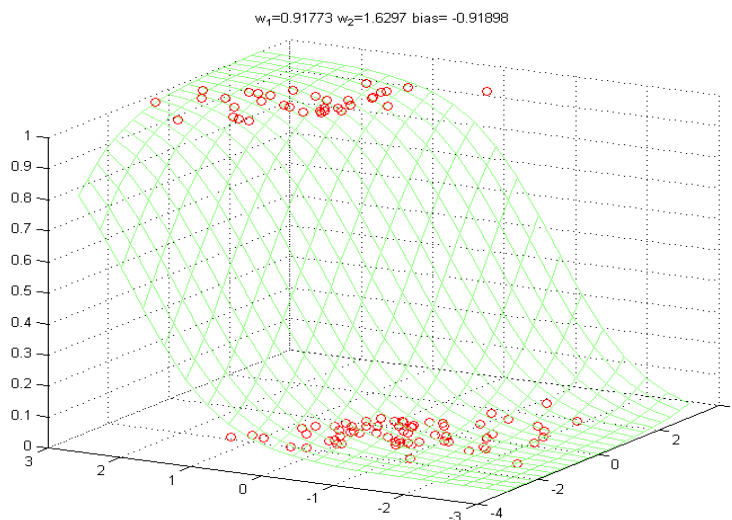
- $i$ th update for the logistic regression** and  $D_k = \langle \mathbf{x}_k, y_k \rangle$

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} + \alpha(k) [y_i - f(\mathbf{w}^{(k-1)}, \mathbf{x}_k)] \mathbf{x}_k$$

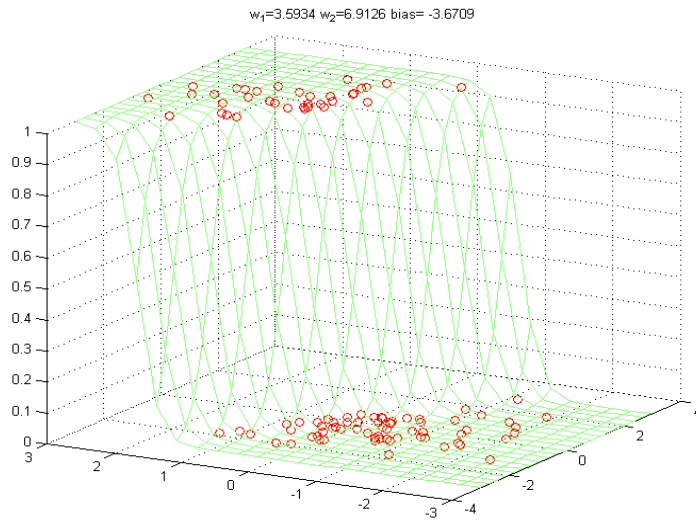
## Online logistic regression algorithm

```
Online-logistic-regression (stopping_criterion)  
initialize weights  $\mathbf{w} = (w_0, w_1, w_2 \dots w_d)$   
while stopping_criterion = FALSE  
  do    select next data point  $D_i = \langle \mathbf{x}_i, y_i \rangle$   
        set  $\alpha(i)$   
        update weights (in parallel)  
             $\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)[y_i - f(\mathbf{w}, \mathbf{x}_i)]\mathbf{x}_i$   
  end  
return weights  $\mathbf{w}$ 
```

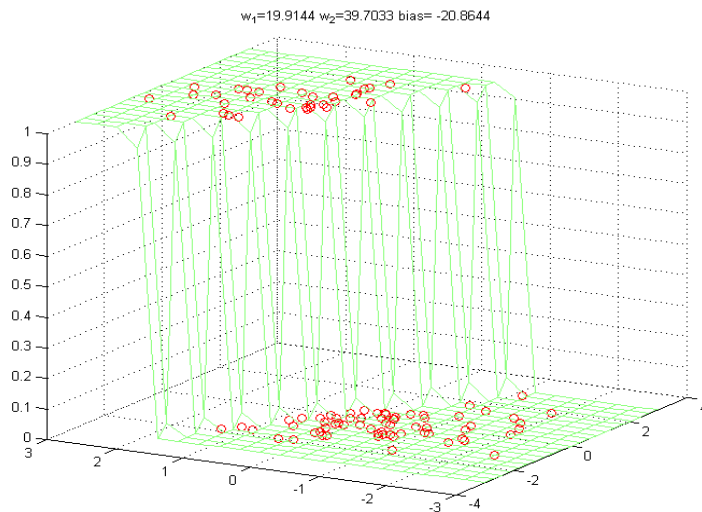
## Online algorithm. Example.



## Online algorithm. Example.



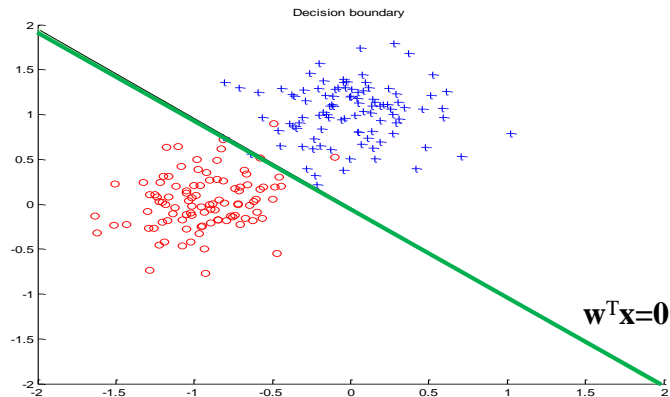
## Online algorithm. Example.



## Logistic regression model. Decision boundary

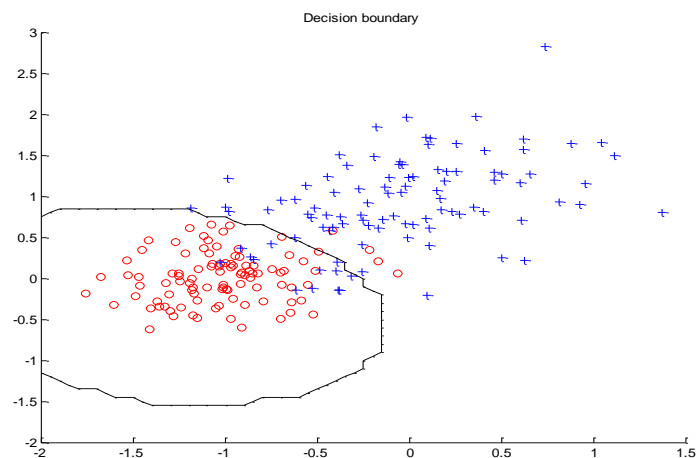
- LR defines a linear decision boundary

**Example:** 2 classes (blue and red points)



## When does the logistic regression fail?

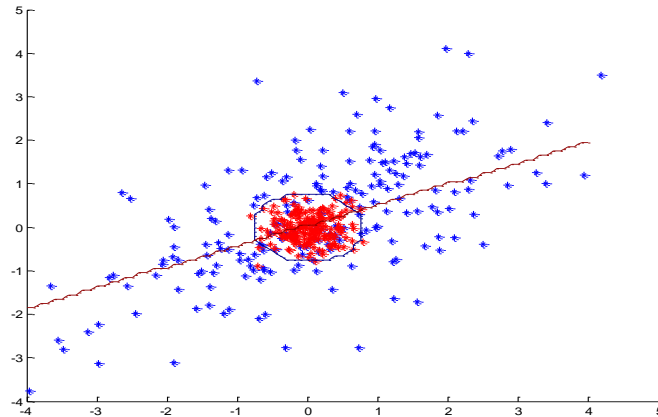
- Nonlinear decision boundary





## When does the logistic regression fail?

- Another example of a non-linear decision boundary



## Non-linear extension of logistic regression

- use **feature (basis) functions** to model **nonlinearities**
  - the same trick as used for the linear regression

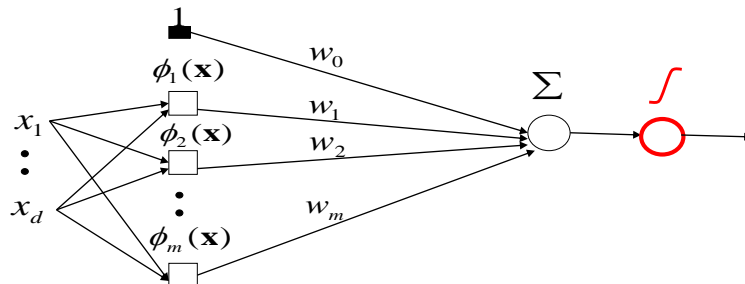
**Linear regression**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^m w_j \phi_j(\mathbf{x})$$

**Logistic regression**

$$p(y=1|x) = g\left(w_0 + \sum_{j=1}^m w_j \phi_j(\mathbf{x})\right)$$

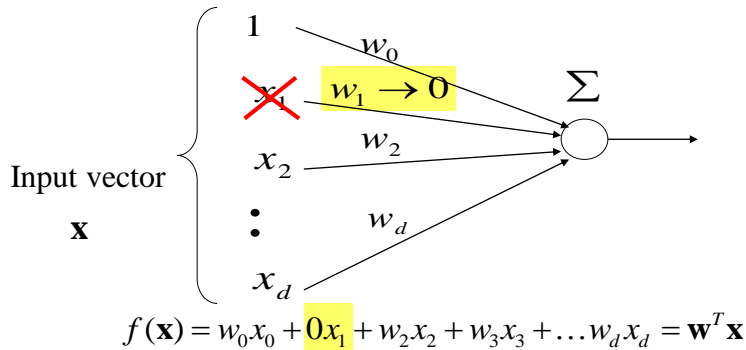
$\phi_j(\mathbf{x})$  - an arbitrary function of  $\mathbf{x}$



## Regularized logistic regression

- If the model is too complex and can cause overfitting, its prediction accuracy can be improved by **removing some inputs from the model = setting their coefficients to zero**
- **Recall the linear model:**

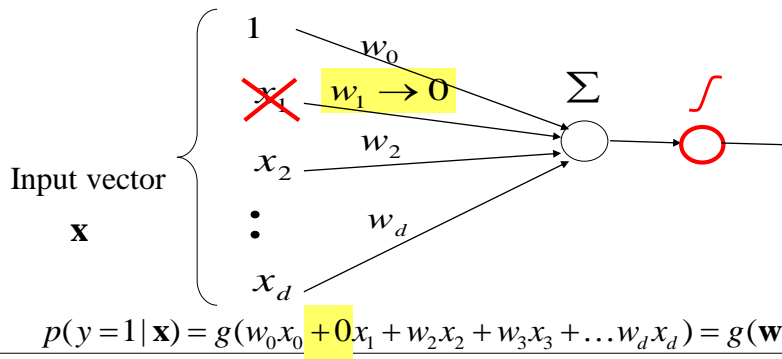
$$f(\mathbf{x}) = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots w_dx_d = \mathbf{w}^T \mathbf{x}$$



## Regularized logistic regression

- If the model is too complex and can cause overfitting, its prediction accuracy can be improved by **removing some inputs from the model = setting their coefficients to zero**
- **We can apply the same idea to the logistic regression:**

$$p(y=1 | \mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) \quad w_0, w_1, \dots, w_k \text{ - parameters (weights)}$$



## Ridge (L2) penalty

### Linear regression – Ridge penalty:

$$J_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_{L2}^2$$

**Fit to data**                      **Model complexity penalty**

$$\|\mathbf{w}\|_{L2}^2 = \sum_{i=0}^d w_i^2 = \mathbf{w}^T \mathbf{w} \quad \text{and} \quad \lambda \geq 0$$

### Logistic regression:

$$J_n(\mathbf{w}) = -\log P(D | \mathbf{w}) + \lambda \|\mathbf{w}\|_{L2}^2$$

**Fit to data**                      **Model complexity penalty**

$$J_n(\mathbf{w}) = -\left[ \sum_{i=1}^n y_i \log g(\mathbf{w}^T x_i) + (1 - y_i) \log(1 - g(\mathbf{w}^T x_i)) \right] + \lambda \|\mathbf{w}\|_{L2}^2$$

**Fit to data measured using the negative log likelihood**

## Lasso (L1) penalty

### Linear regression – Lasso penalty:

$$J_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_{L1}$$

**Fit to data**                      **Model complexity penalty**

$$\|\mathbf{w}\|_{L1} = \sum_{i=0}^d |w_i| \quad \text{and} \quad \lambda \geq 0$$

### Logistic regression:

$$J_n(\mathbf{w}) = -\log P(D | \mathbf{w}) + \lambda \|\mathbf{w}\|_{L1}$$

**Fit to data**                      **Model complexity penalty**

$$J_n(\mathbf{w}) = -\left[ \sum_{i=1}^n y_i \log g(\mathbf{w}^T x_i) + (1 - y_i) \log(1 - g(\mathbf{w}^T x_i)) \right] + \lambda \|\mathbf{w}\|_{L1}$$

**Fit to data measured using the negative log likelihood**