CS 2750 Machine Learning
Lecture 8

Linear regression (cont.)
Linear methods for classification

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Coefficient shrinkage

- The least squares estimates often have low bias but high variance
- The prediction accuracy can be often improved by setting some coefficients to zero
  - Increases the bias, reduces the variance of estimates
- **Solutions:**
  - **Subset selection**
  - **Ridge regression**
  - **Principal component regression**

- Next: **ridge regression**

Ridge regression

- Error function for the standard least squares estimates:

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

- **We seek:** $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1,..n}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$

- **Ridge regression:**

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

- Where $\|\mathbf{w}\|^2 = \sum_{i=0}^{d} w_i^2$ and $\lambda \geq 0$
- What does the new error function do?

Ridge regression

- **Standard regression:**

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

- **Ridge regression:**

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

- $\|\mathbf{w}\|^2 = \sum_{i=0}^{d} w_i^2$ penalizes non-zero weights with the cost proportional to λ (a shrinkage coefficient)
- If an input attribute x_j has a small effect on improving the error function it is "shut down" by the penalty term
- Inclusion of a shrinkage penalty is often referred to as **regularization**

Supervised learning

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

$d_i = < \mathbf{x}_i, y_i >$

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

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

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

Two types of problems:

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

  Example: earnings, product orders $\rightarrow$ company stock price
- **Classification:** Y is **discrete**

  Example: temperature, heart rate $\rightarrow$ disease

Today: **binary classification problems:**

Binary classification

- **Two classes** $Y = \{0,1\}$
- Our goal is to learn to classify correctly two types of examples
  - Class 0 – labeled as 0,
  - Class 1 – labeled as 1
- We would like to learn $f : X \rightarrow \{0,1\}$
- **Zero-one error (loss) function**

$$Error_1(\mathbf{x}_i, y_i) = \begin{cases} 1 & f(\mathbf{x}_i, \mathbf{w}) \neq y_i \\ 0 & f(\mathbf{x}_i, \mathbf{w}) = y_i \end{cases}$$

- Error we would like to minimize: $E_{(x,y)}(Error_1(\mathbf{x}, y))$
- **First step:** we need to devise a model of the function

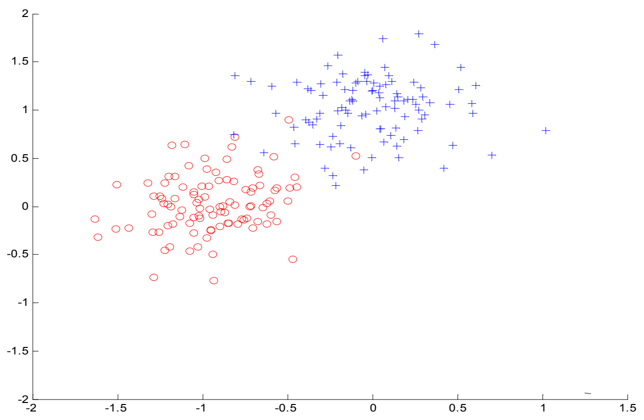
Discriminant functions

- One convenient way to represent classifiers is through
  - **Discriminant functions**
- **Works for binary and multi-way classification**

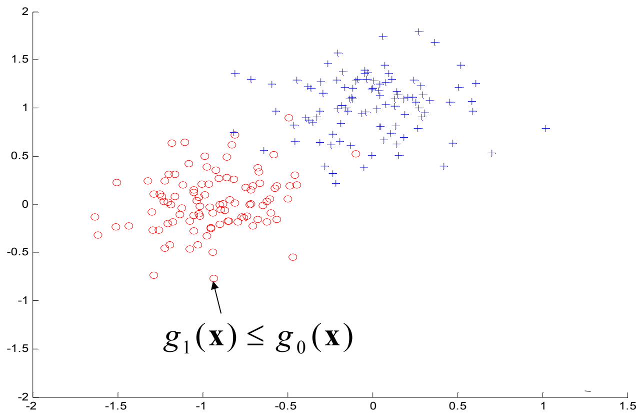
- **Idea:**
  - For every class $i = 0,1, \ldots k$ define a function $g_i(\mathbf{x})$ mapping $X \to \Re$
  - When the decision on input **x** should be made choose the class with the highest value of $g_i(\mathbf{x})$

- So what happens with the input space? Assume a binary case.

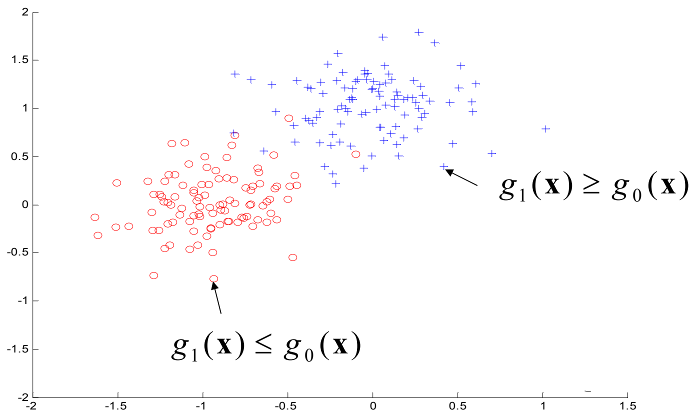
Discriminant functions



Discriminant functions

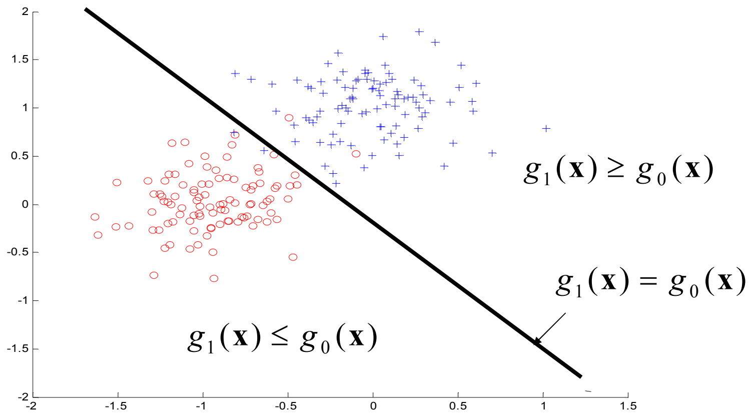


Discriminant functions

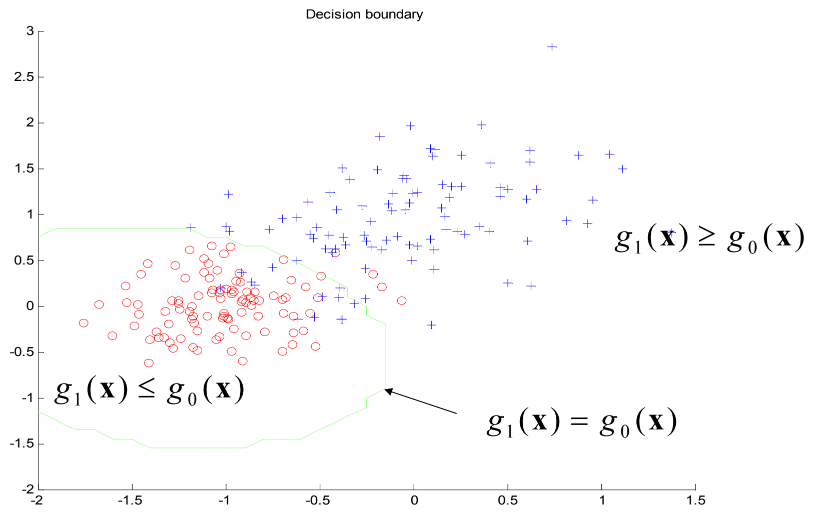


Discriminant functions

- Define **decision boundary.**



Quadratic decision boundary



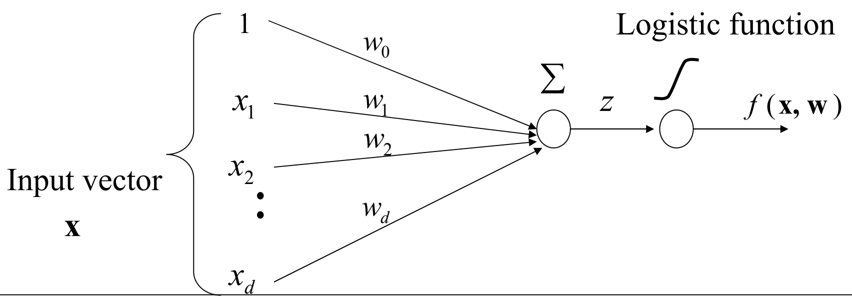
Logistic regression model

- **Defines a linear decision boundary**
- **Discriminant functions:**

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

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

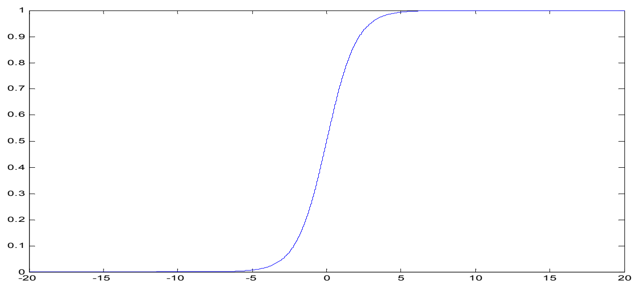
$$f(\mathbf{x},\mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



Logistic function

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

- also referred to as a **sigmoid function**
- Replaces the threshold function with smooth switching
- takes a real number and outputs the number in the interval [0,1]

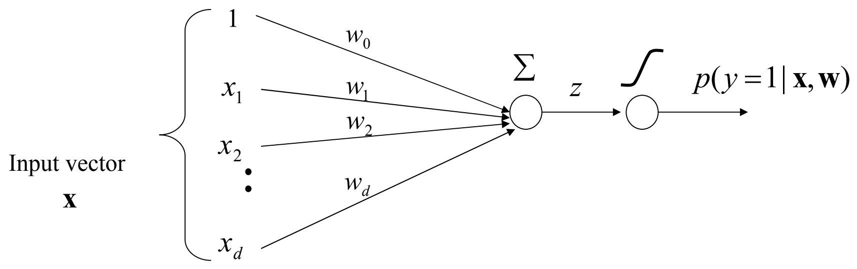


Logistic regression model

- **Discriminant functions:**

  $g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$ $g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x})$
- **Where** $g(z) = 1/(1+e^{-z})$ - is a logistic function
- **Values of discriminant functions vary in [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

- Instead of learning the mapping to discrete values 0,1

  $$f : X \rightarrow \{0,1\}$$
- we learn **a probabilistic function**

  $$f : X \rightarrow [0,1]$$
  - where f describes the probability of class 1 given $\mathbf{x}$

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

  Note that: $p(y=0\,|\,\mathbf{x},\mathbf{w}) = 1 - p(y=1\,|\,\mathbf{x},\mathbf{w})$
- Transformation to discrete class values:

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:

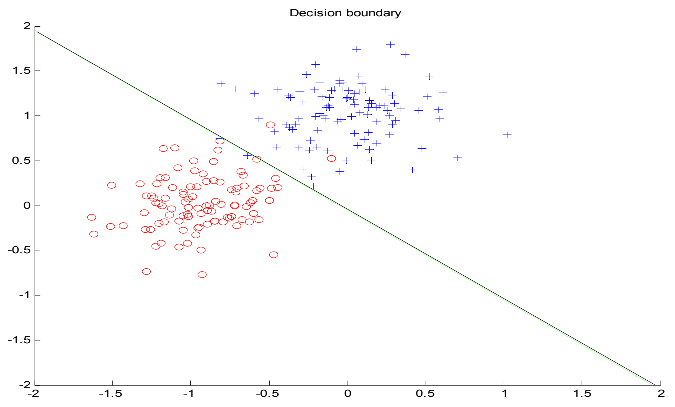
$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^{\mathbf{T}}\mathbf{x})}{g(\mathbf{w}^{\mathbf{T}}\mathbf{x})} = 0$$

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

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**

$$D_i =< \mathbf{x}_i, y_i > \qquad \mu_i = p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$$

- **Then**

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

- **Find weights 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

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

$$= \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

Logistic regression: parameter learning

- **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**

<u>**Nonlinear in weights !!**</u>

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

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

- **Gradient descent:**

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

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

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 w** $J_{online}(D_k, \mathbf{w})$

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

- **ith update for the logistic regression** and $D_k = <\mathbf{x}_k, y_k>$

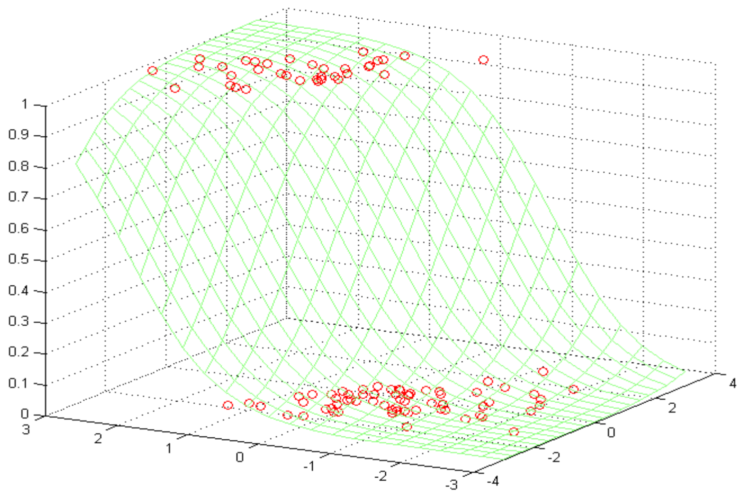
$$\mathbf{w}^{(i)} \leftarrow \mathbf{w}^{(k-1)} + \alpha(k)[y_i - f(\mathbf{w}^{(k-1)}, \mathbf{x}_k)]\mathbf{x}_k$$

Online logistic regression algorithm

```
Online-logistic-regression (D, number of iterations)
    initialize weights  $\mathbf{w} = (w_0, w_1, w_2 \ldots w_d)$
    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 weights (in parallel)
                    $\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)[y_i - f(\mathbf{w}, \mathbf{x}_i)]\mathbf{x}_i$
    end for
    return weights  $\mathbf{w}$
```

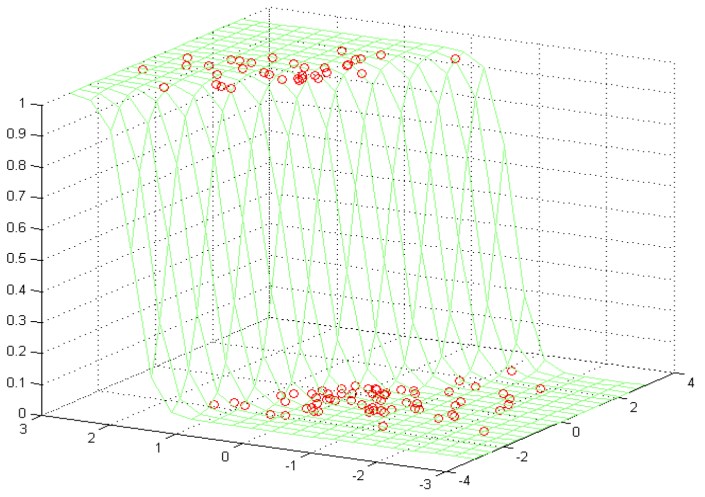
Online algorithm. Example.

w_1=0.91773 w_2=1.6297 bias= -0.91898

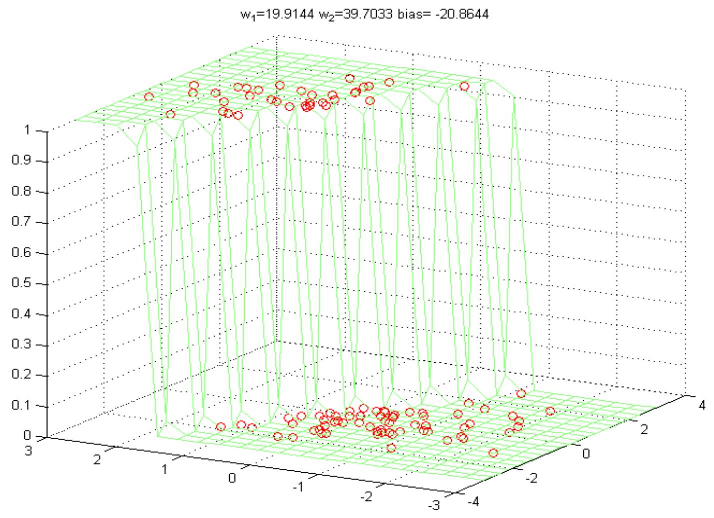


Online algorithm. Example.

w_1=3.5934 w_2=6.9126 bias= -3.6709



Online algorithm. Example.



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**

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

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

Derivative of a logistic function

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

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

$$= y_i(1-g(z_i)) + (1-y_i)(-g(z_i)) \boxed{= y_i - g(z_i)}$$

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