

CS 2750 Machine Learning Lecture 8

Classification : Logistic regression. Generative classification model.

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

CS 2750 Machine Learning

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

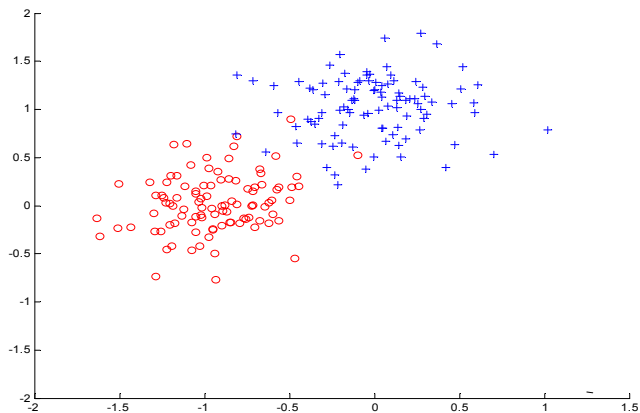
CS 2750 Machine Learning

Discriminant functions

- One way to represent a classifier is by using
 - Discriminant functions
- Works for 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 $g_i(\mathbf{x})$
- So what happens with the input space? Assume a binary case.

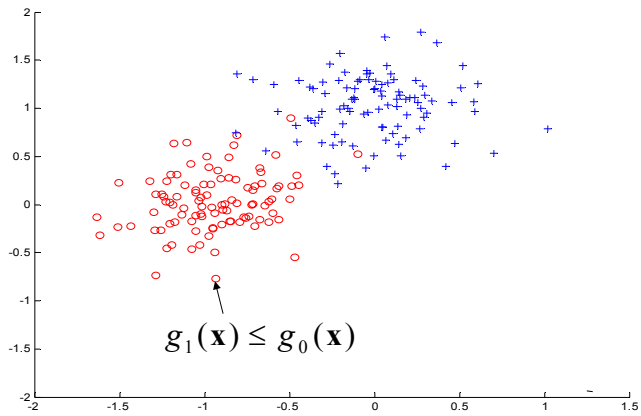
CS 2750 Machine Learning

Discriminant functions



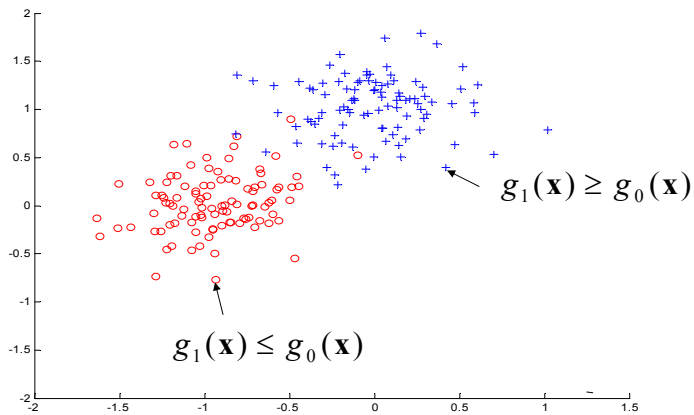
CS 2750 Machine Learning

Discriminant functions



CS 2750 Machine Learning

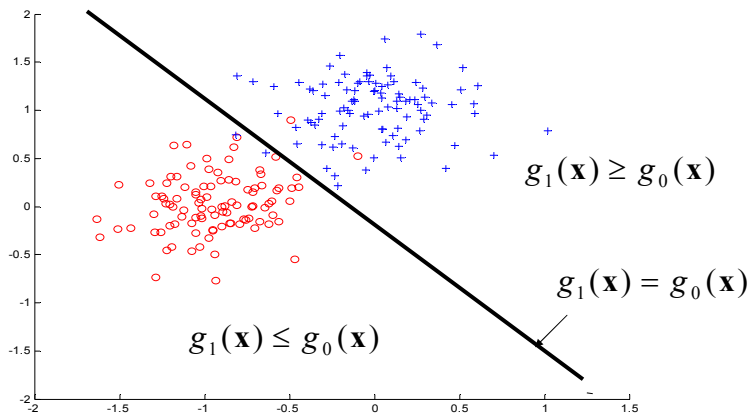
Discriminant functions



CS 2750 Machine Learning

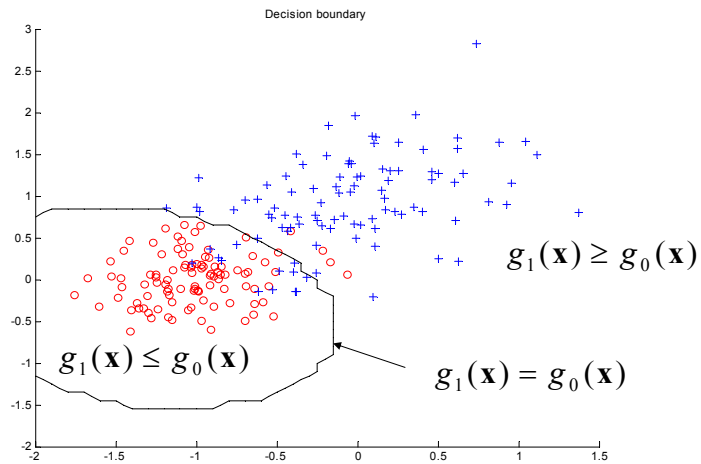
Discriminant functions

- Define **decision boundary**



CS 2750 Machine Learning

Quadratic decision boundary



CS 2750 Machine Learning

Logistic regression model

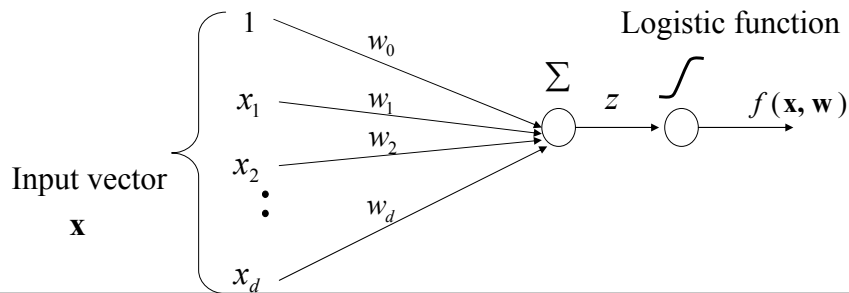
- Defines a linear decision boundary

- 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

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

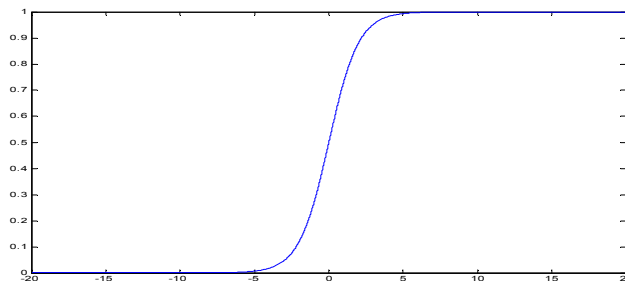


CS 2750 Machine Learning

Logistic function

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

- Is 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 $[0,1]$



CS 2750 Machine Learning

Logistic regression model

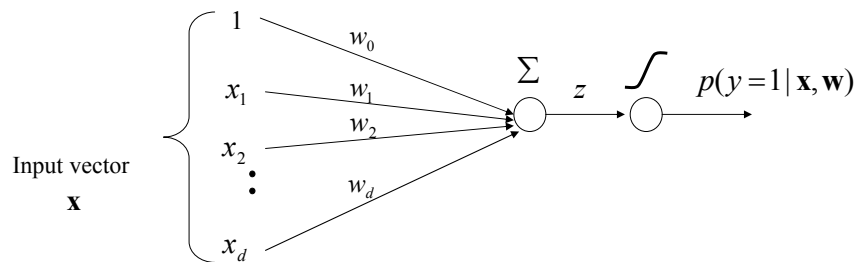
- **Discriminant functions:**

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

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

– **Probabilistic interpretation**

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



CS 2750 Machine Learning

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:

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

- Transformation to binary class values:

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

CS 2750 Machine Learning

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:

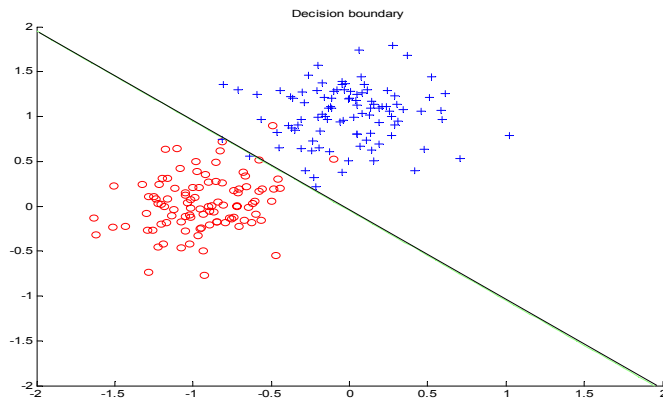
$$\log \frac{g_0(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^T \mathbf{x})}{g(\mathbf{w}^T \mathbf{x})} = 0$$

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

CS 2750 Machine Learning

Logistic regression model. Decision boundary

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



CS 2750 Machine Learning

Logistic regression: parameter learning

Likelihood of outputs

- Let

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

- 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

- 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

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

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

- Gradient descent:

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

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

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 $D_k = \langle \mathbf{x}_k, y_k \rangle$

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

Online logistic regression algorithm

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

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

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

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

set $\alpha = 1/i$

update weights (in parallel)

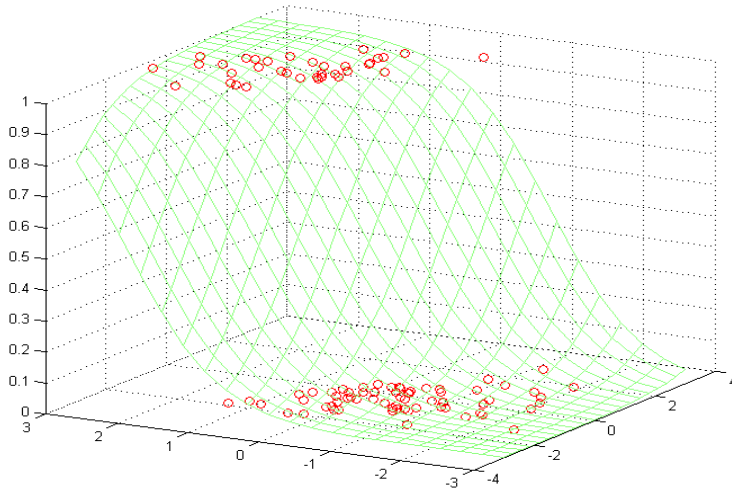
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(i) [y_i - f(\mathbf{w}, \mathbf{x}_i)] \mathbf{x}_i$$

end for

return weights \mathbf{w}

Online algorithm. Example.

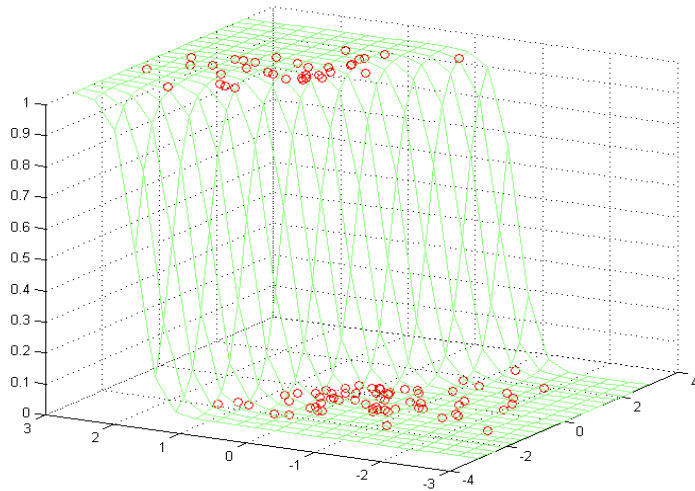
$w_1=0.91773$ $w_2=1.6297$ bias= -0.91898



CS 2750 Machine Learning

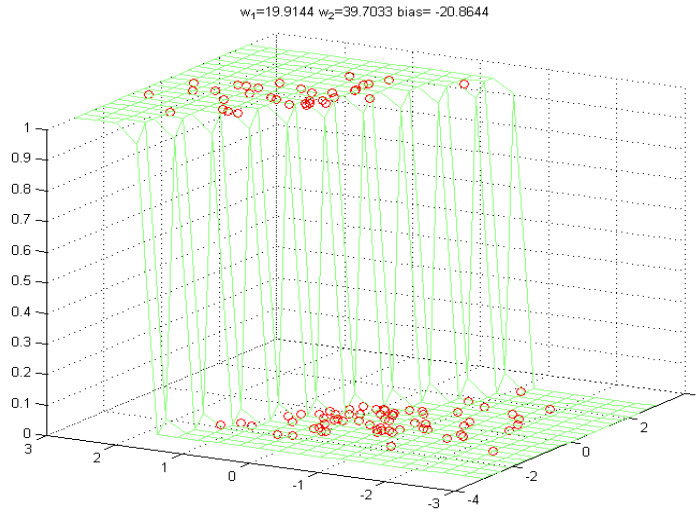
Online algorithm. Example.

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



CS 2750 Machine Learning

Online algorithm. Example.



CS 2750 Machine Learning

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**

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

Derivative of a logistic function

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

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

$$\begin{aligned} \frac{\partial}{\partial z_i} [y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)] &= y_i \frac{1}{g(z_i)} \frac{\partial g(z_i)}{\partial z_i} + (1 - y_i) \frac{-1}{1 - g(z_i)} \frac{\partial g(z_i)}{\partial z_i} \\ &= y_i(1 - g(z_i)) + (1 - y_i)(-g(z_i)) = y_i - g(z_i) \end{aligned}$$

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

CS 2750 Machine Learning

Generative approach to classification

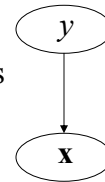
Idea:

1. **Represent and learn the distribution** $p(\mathbf{x}, y)$
2. **Use it to define probabilistic discriminant functions**

E.g. $g_0(\mathbf{x}) = p(y = 0 | \mathbf{x}) \quad g_1(\mathbf{x}) = p(y = 1 | \mathbf{x})$

Typical model $p(\mathbf{x}, y) = p(\mathbf{x} | y)p(y)$

- $p(\mathbf{x} | y) =$ **Class-conditional distributions (densities)**
 binary classification: two class-conditional distributions
 $p(\mathbf{x} | y = 0) \quad p(\mathbf{x} | y = 1)$
- $p(y) =$ **Priors on classes** - probability of class y
 binary classification: Bernoulli distribution



$$p(y = 0) + p(y = 1) = 1$$

Generative approach to classification

Example:

- **Class-conditional distributions**
 - **multivariate normal distributions**

$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad \text{for } y = 0$$

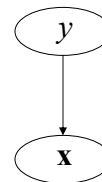
$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{for } y = 1$$

Multivariate normal $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

- **Priors on classes (class 0,1)** $y \sim \text{Bernoulli}$
 - **Bernoulli distribution**

$$p(y, \theta) = \theta^y (1 - \theta)^{1-y} \quad y \in \{0,1\}$$



Learning of parameters of the model

Density estimation in statistics

- We see examples – we do not know the parameters of Gaussians (class-conditional densities)

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

- **ML estimate of parameters** of a multivariate normal $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a set of n examples of \mathbf{x}

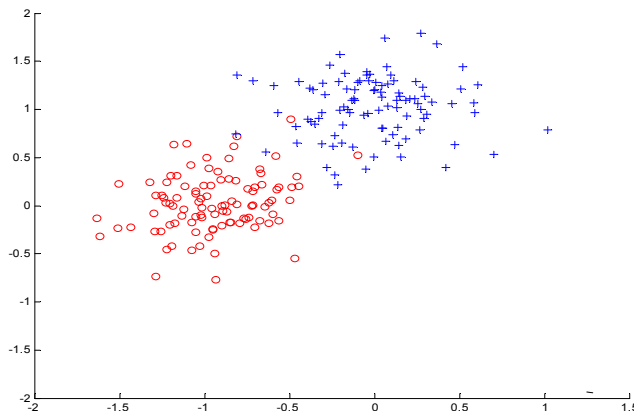
Optimize log-likelihood: $l(D, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^n p(\mathbf{x}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$$

- How about **class priors**?

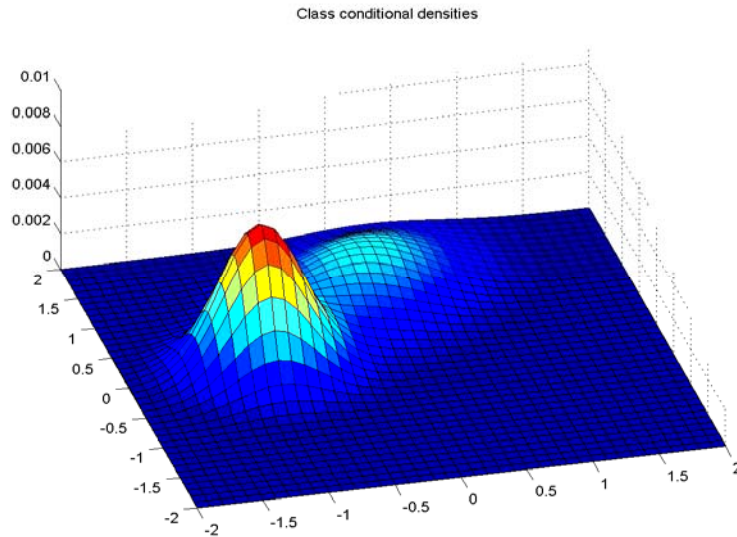
CS 2750 Machine Learning

Generative model



CS 2750 Machine Learning

2 Gaussian class-conditional densities



Making class decision

Basically we need to design discriminant functions

Two possible choices:

- **Likelihood of data** – choose the class (Gaussian) that explains the input data (\mathbf{x}) better (likelihood of the data)

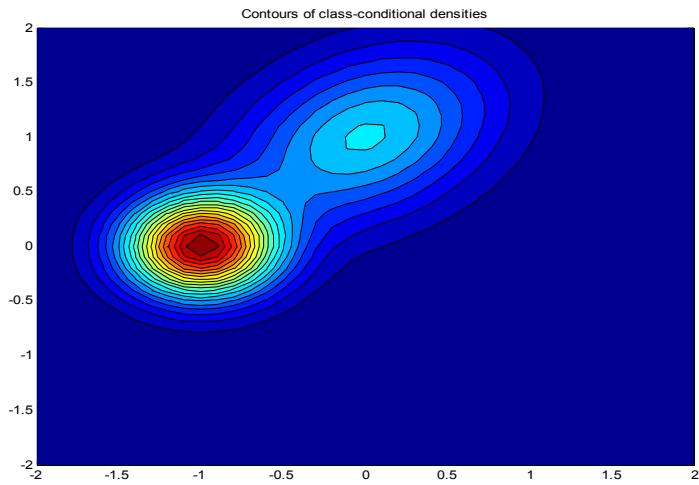
$$\underbrace{p(\mathbf{x} | \mu_1, \Sigma_1)}_{g_1(\mathbf{x})} > \underbrace{p(\mathbf{x} | \mu_0, \Sigma_0)}_{g_0(\mathbf{x})} \quad \longrightarrow \quad \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

- **Posterior of a class** – choose the class with better posterior probability

$$p(y = 1 | \mathbf{x}) > p(y = 0 | \mathbf{x}) \quad \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

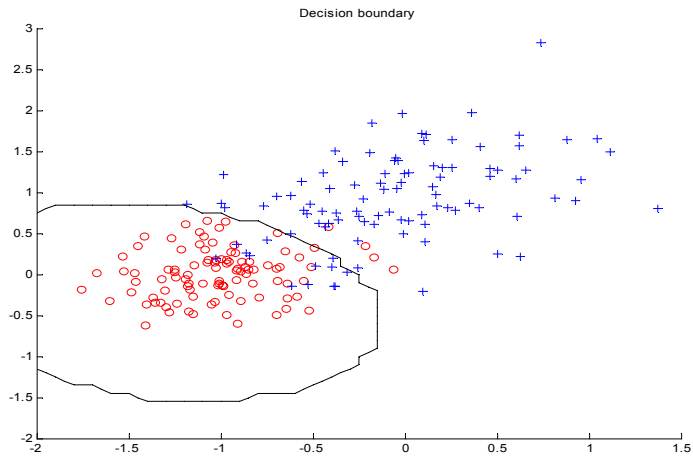
$$p(y = 1 | \mathbf{x}) = \frac{p(\mathbf{x} | \mu_1, \Sigma_1) p(y = 1)}{p(\mathbf{x} | \mu_0, \Sigma_0) p(y = 0) + p(\mathbf{x} | \mu_1, \Sigma_1) p(y = 1)}$$

2 Gaussians: Quadratic decision boundary



CS 2750 Machine Learning

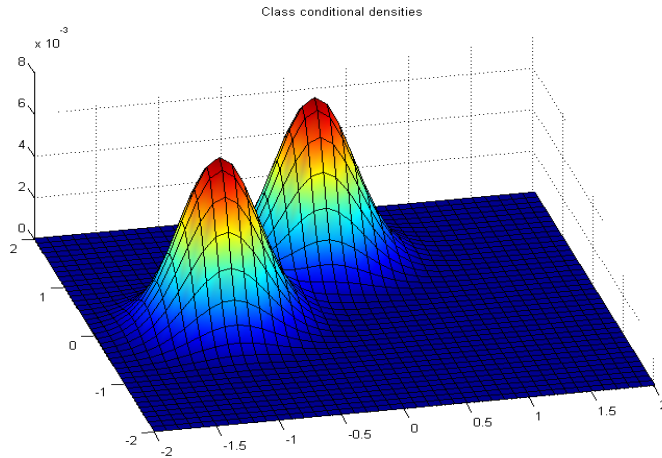
2 Gaussians: Quadratic decision boundary



CS 2750 Machine Learning

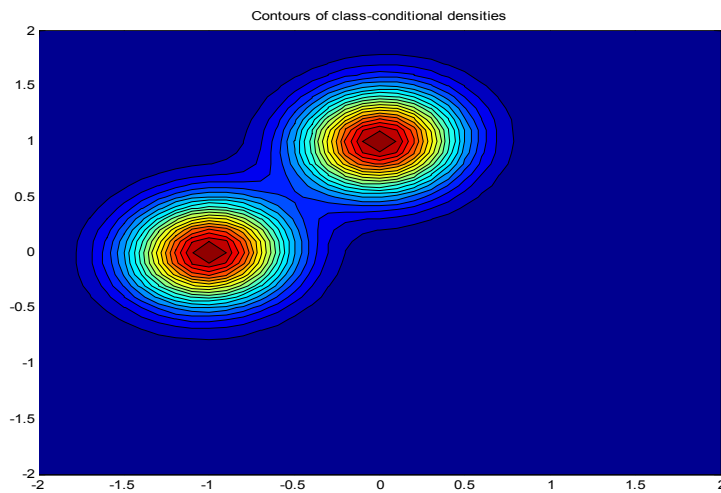
2 Gaussians: Linear decision boundary

- When covariances are the same $\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}), y = 0$
 $\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}), y = 1$



CS 2750 Machine Learning

2 Gaussians: Linear decision boundary



CS 2750 Machine Learning

2 Gaussians: linear decision boundary

