CS 2750 Machine Learning
Lecture 8

Classification learning:
- Logistic regression
- Generative classification model

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

Classification

- **Data:** $D = \{d_1, d_2, .., d_n\}$
  $d_i = < \mathbf{x}_i, y_i >$
  - y_i **represents a discrete class value**
- **Goal: learn** $f : X \rightarrow Y$

- **Binary classification**
  - **A special case when** $Y \in \{0,1\}$

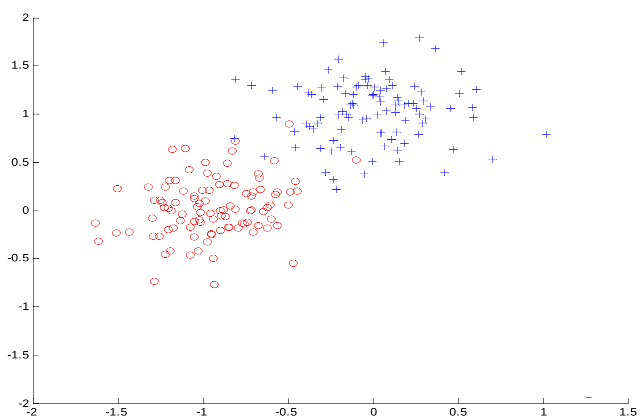
- **First step:**
  - we need to devise a model of the function f

Discriminant functions

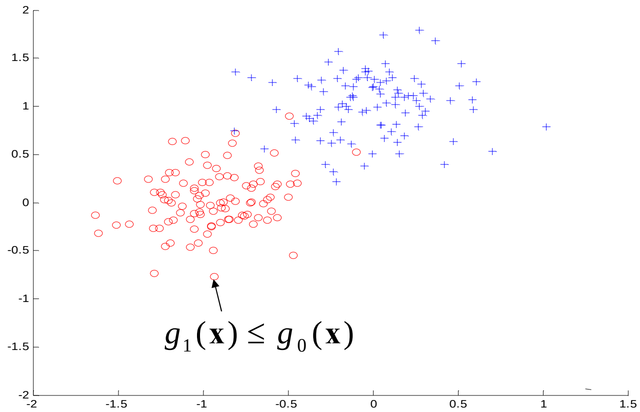
- A common way to represent a **classifier is by using**
  - **Discriminant functions**
- **Works for both the binary and multi-way classification**
- **Idea:**
  - For every class $i = 0,1, \ldots k$ define a function $g_i(\mathbf{x})$ mapping $X \to \Re$
  - When the decision on input **x** should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

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

Discriminant functions

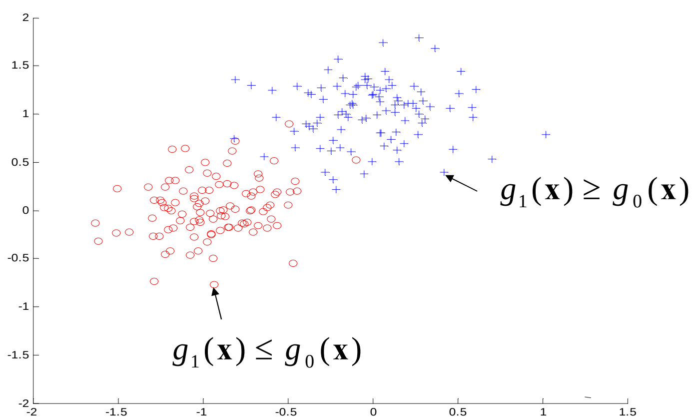


Discriminant functions



CS 2750 Machine Learning

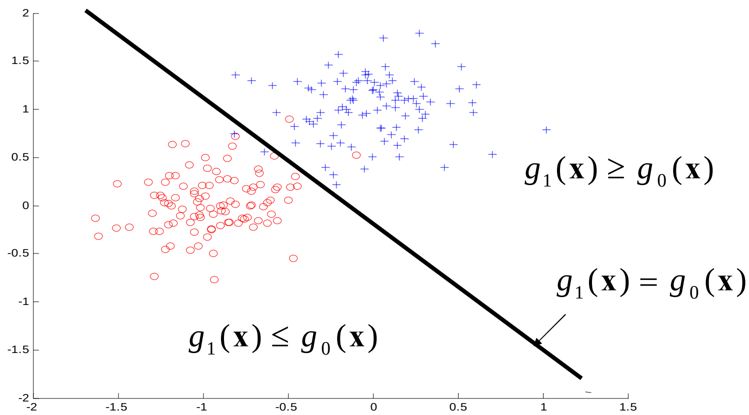
Discriminant functions



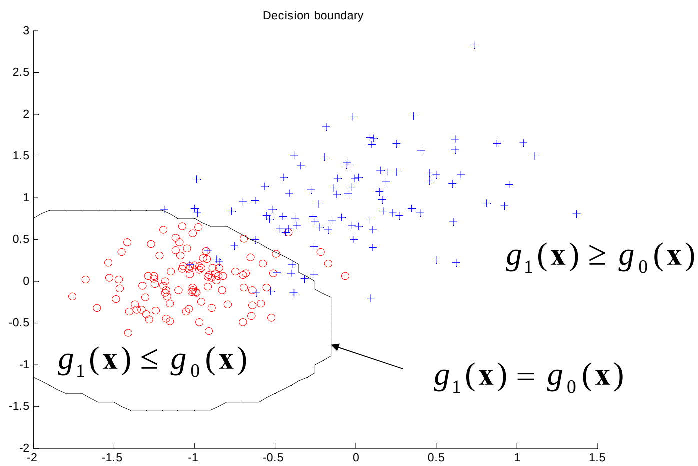
CS 2750 Machine Learning

Discriminant functions

- Define **decision boundary**



Quadratic decision boundary



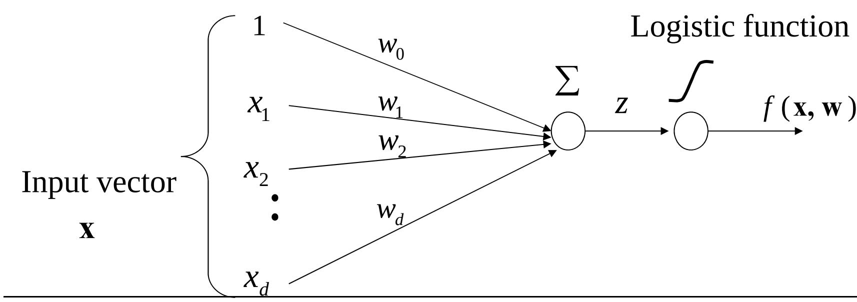
Logistic regression model

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

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

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

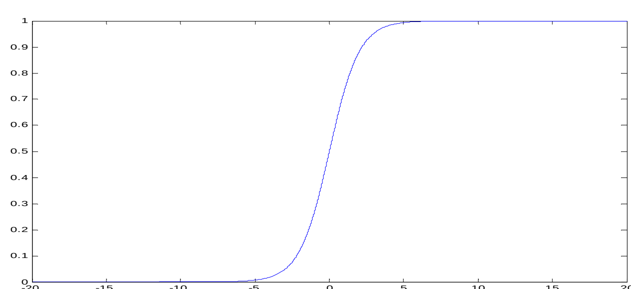
$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$



Logistic function

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

- 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]



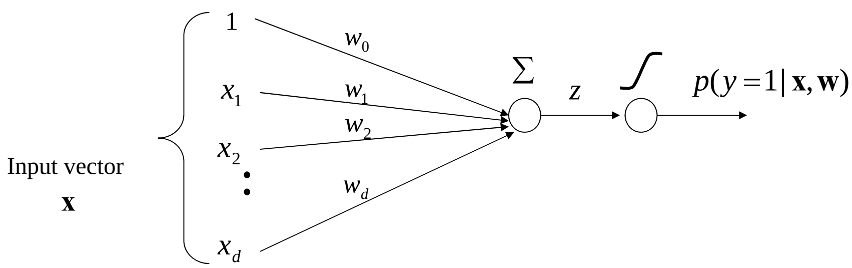
Logistic regression model

- **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**

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



Logistic regression

- We learn **a probabilistic function**

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

  - where *f* describes the probability of class 1 given **x**

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

Note that:

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

- Transformation to binary class values:

> If $p(y = 1 \mid \mathbf{x}) \geq 1/2$ 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:

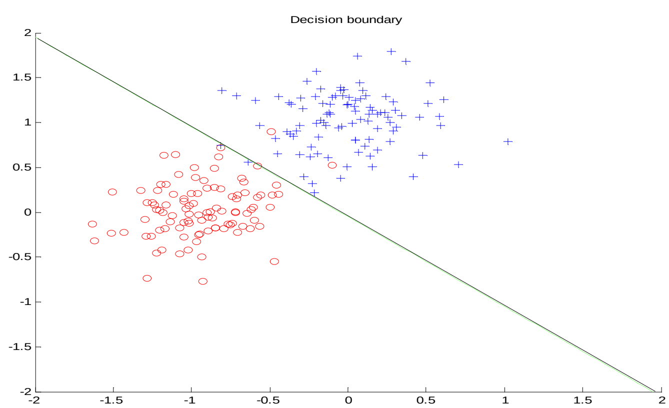
$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^{\mathbf{T}}\mathbf{x})}{g(\mathbf{w}^{\mathbf{T}}\mathbf{x})} = 0$$

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

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**

Nonlinear in weights !!

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

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

- **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 $D_k = < \mathbf{x}_k, y_k >$

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

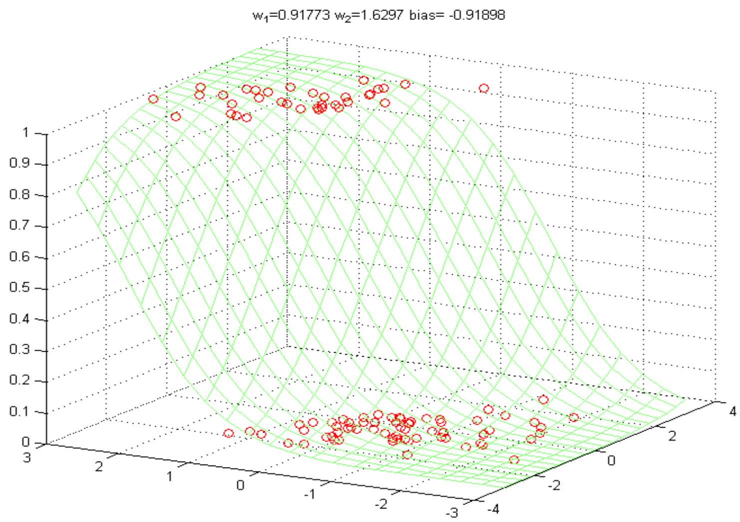
Online logistic regression algorithm

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)

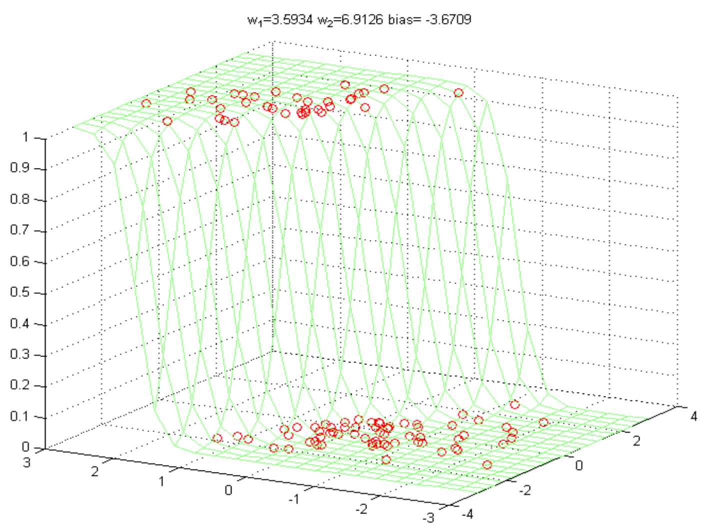
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)[y_i - f(\mathbf{w}, \mathbf{x}_i)]\mathbf{x}_i$$

    end for
    return weights $\mathbf{w}$

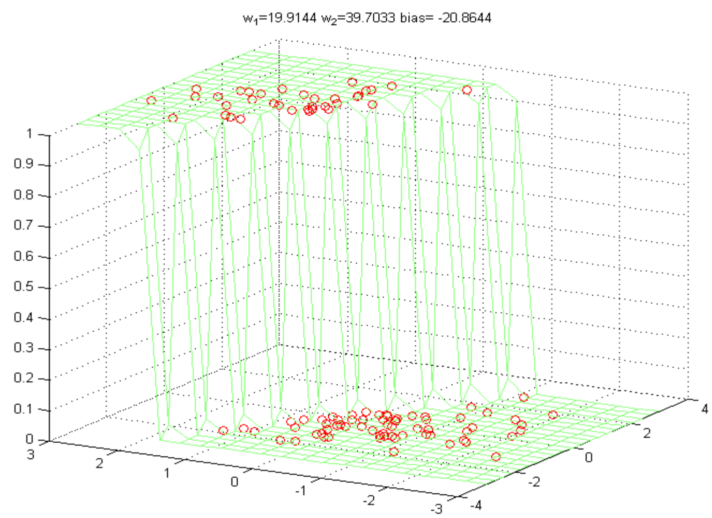
Online algorithm. Example.



Online algorithm. Example.



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

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

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

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

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

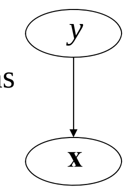
Generative approach to classification

Idea:

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

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

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



- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  binary classification: two class-conditional distributions

  $$p(\mathbf{x} \mid y=0) \qquad p(\mathbf{x} \mid y=1)$$

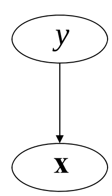
- $p(y)$ = **Priors on classes** - probability of class y

  binary classification: Bernoulli distribution

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

Quadratic discriminant analysis (QDA)

Model:



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

  $$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad \text{for} \quad y=0$$
  $$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{for} \quad y=1$$

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

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

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

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

Learning of parameters of the QDA model

Density estimation in statistics

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

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

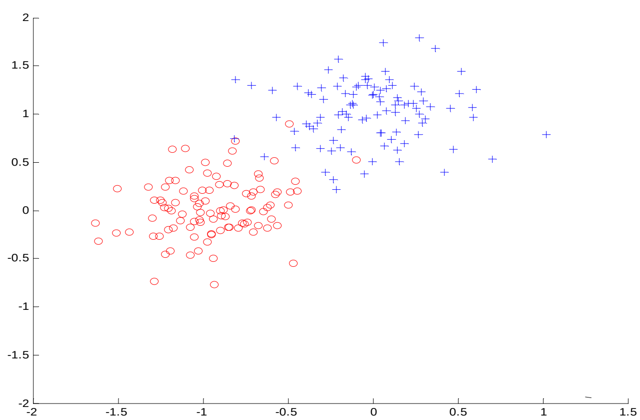
- **ML estimate of parameters** of a multivariate normal $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a set of n examples of $\mathbf{x}$

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

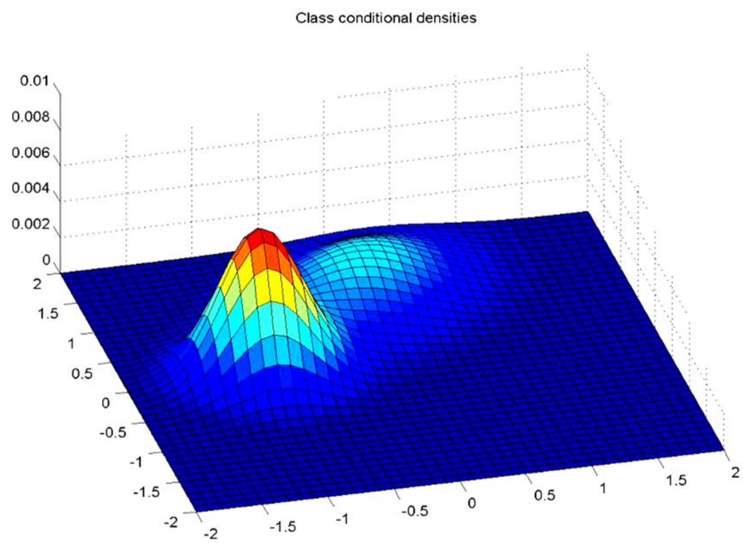
$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$$

- How about **class priors**?

QDA



2 Gaussian class-conditional densities



QDA: 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)

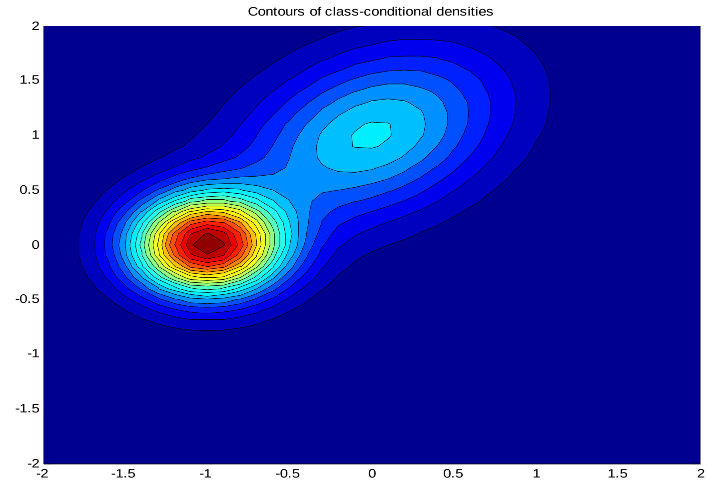
$$\underbrace{p(\mathbf{x} \mid \mu_1, \mathbf{\Sigma}_1)}_{g_1(\mathbf{x})} > \underbrace{p(\mathbf{x} \mid \mu_0, \mathbf{\Sigma}_0)}_{g_0(\mathbf{x})} \Rightarrow \quad \text{then } y=1 \quad \text{else } y=0$$

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

$$p(y=1 \mid \mathbf{x}) > p(y=0 \mid \mathbf{x}) \quad \text{then } y=1 \quad \text{else } y=0$$

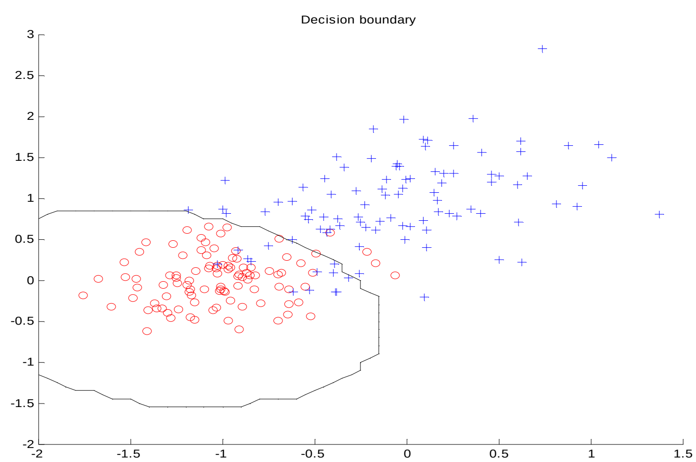
$$p(y=1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mu_1, \mathbf{\Sigma}_1)\, p(y=1)}{p(\mathbf{x} \mid \mu_0, \mathbf{\Sigma}_0)\, p(y=0) + p(\mathbf{x} \mid \mu_1, \mathbf{\Sigma}_1)\, p(y=1)}$$

QDA: Quadratic decision boundary



CS 2750 Machine Learning

QDA: Quadratic decision boundary

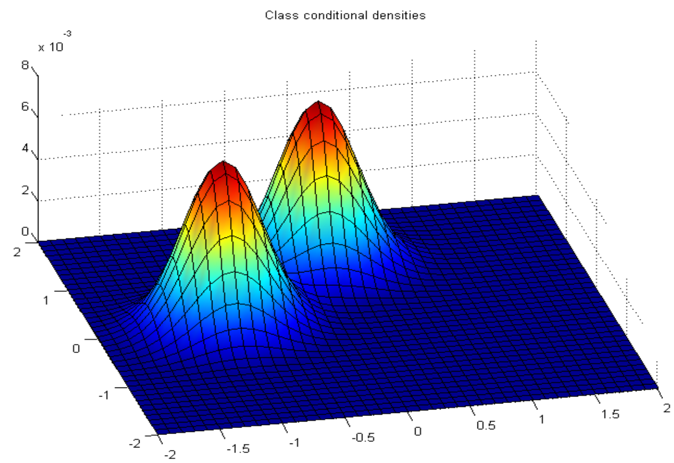


CS 2750 Machine Learning

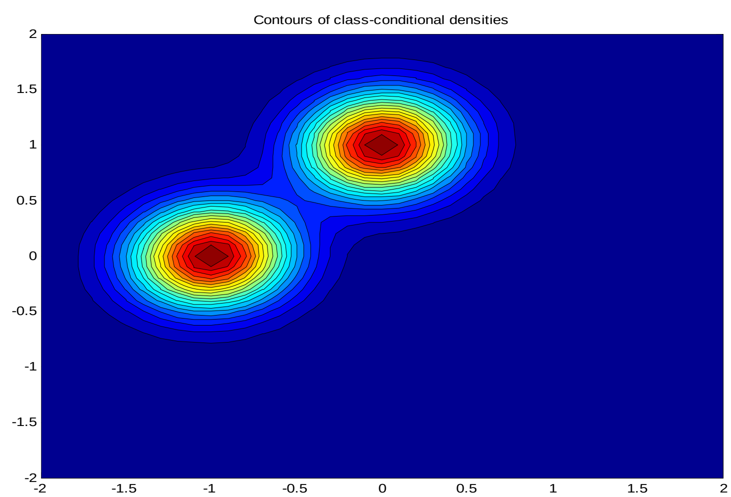
Linear discriminant analysis (LDA)

- When covariances are the same

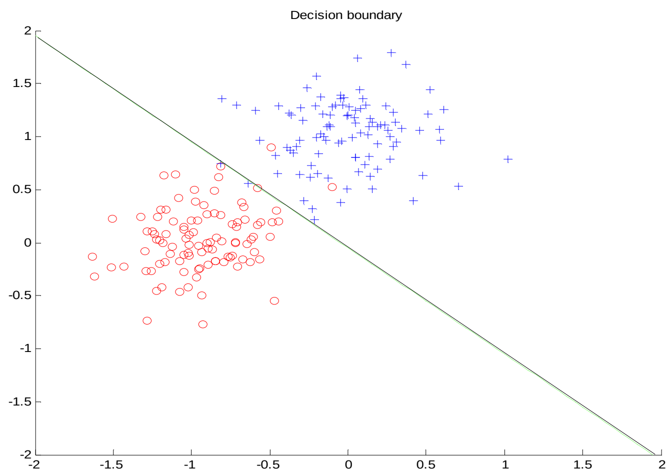
$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}),\ y = 0$$
$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}),\ y = 1$$



LDA: Linear decision boundary



LDA: linear decision boundary



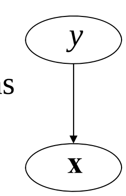
Generative classification models

Idea:

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

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

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



- $p(\mathbf{x} \mid y) =$ **Class-conditional distributions (densities)**

  binary classification: two class-conditional distributions

  $p(\mathbf{x} \mid y = 0) \qquad p(\mathbf{x} \mid y = 1)$

- $p(y)$ = **Priors on classes** **-** probability of class y

  binary classification: Bernoulli distribution

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