CS 2750 Machine Learning
Lecture 10

# Linear models for classification

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

## 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 $X \rightarrow \Re$
  - When the decision on input **x** should be made choose the class with the highest value of $g_i(\mathbf{x})$

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

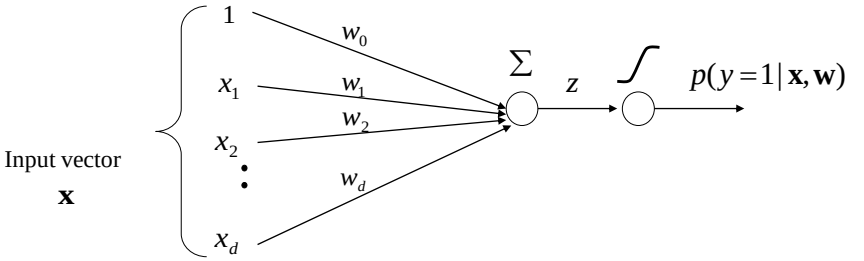
# Logistic regression model

- **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**

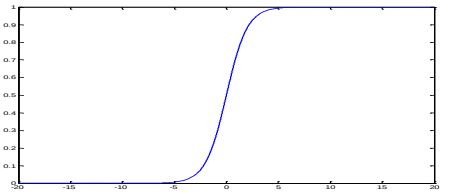
$$f(\mathbf{x},\mathbf{w}) = p(y=1|\mathbf{w},\mathbf{x}) = g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



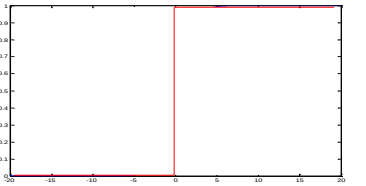

# Logistic function

**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

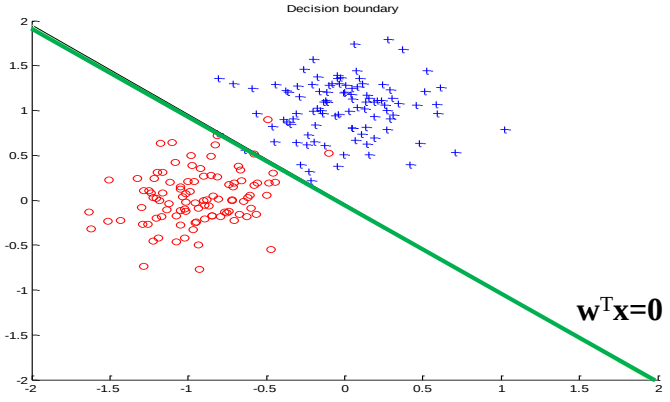


Threshold (hard) switching

## Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**

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




## 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:**

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

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

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

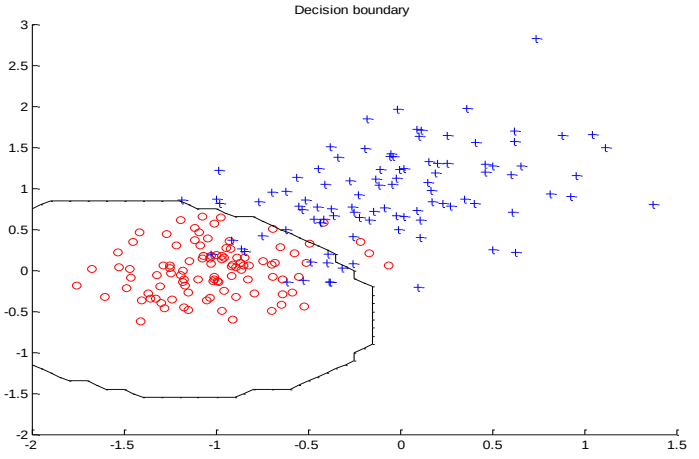
$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [J_{online}(D_k, \mathbf{w})]|_{\mathbf{w}^{(k-1)}}$$

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

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

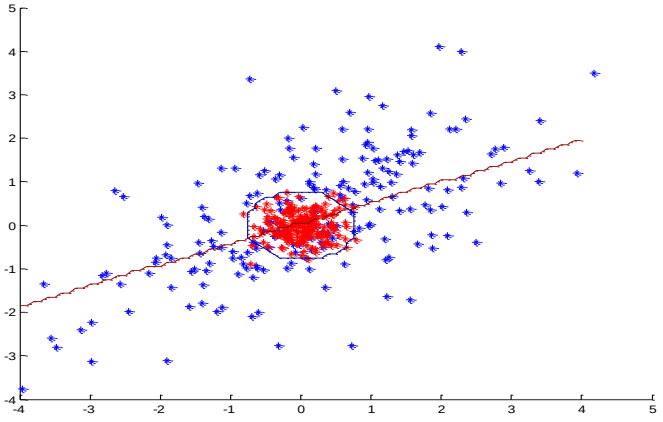
## When does the logistic regression fail?

- 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

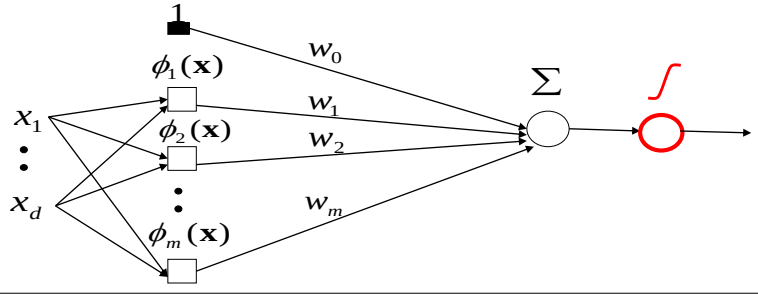
**Linear regression**

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

**Logistic regression**

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

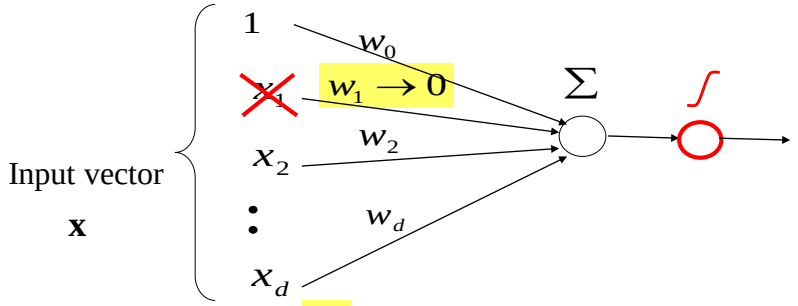
$\phi_j(\mathbf{x})$ - an arbitrary function of $\mathbf{x}$



## 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**
- **We can apply the same idea to the logistic regression:**

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




$$p(y=1|\mathbf{x}) = g(w_0x_0 + 0x_1 + w_2x_2 + w_3x_3 + \ldots w_dx_d) = g(\mathbf{w}^T\mathbf{x})$$

## Ridge (L2) penalty

**Linear regression – Ridge penalty:**

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

**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**

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

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

## Lasso (L1) penalty

**Linear regression – Lasso penalty:**

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

**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**

---

## Generative approach to classification

**Logistic regression:**

- **Represents and learns a model of** $p(y|\mathbf{x})$
- **An example of a discriminative classification approach**
- **Model is unable to sample (generate) data instances** $(\mathbf{x}, y)$

**Generative approach:**

- **Represents and learns a joint distribution** $p(\mathbf{x}, y)$
- **Model is able to sample (generate) data instances** $(\mathbf{x}, y)$
- **The joint model defines probabilistic discriminant functions**

**How?**

$$g_1(\mathbf{x}) = p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}, y=1)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x})}$$

$$g_o(\mathbf{x}) = p(y=0|\mathbf{x}) = \frac{p(\mathbf{x}, y=0)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x})}$$

$$p(y=0|\mathbf{x}) + p(y=1|\mathbf{x}) = 1$$

---

## Generative approach to classification

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

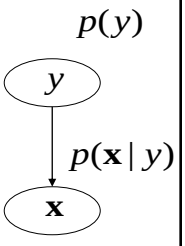
- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  binary classification: two class-conditional distributions

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

- $p(y)$ = **Priors on classes**
  - probability of class $y$
  - for binary classification: Bernoulli distribution

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



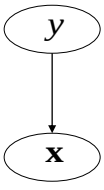

## Quadratic discriminant analysis (QDA)

**Model:**

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

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

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




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

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

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

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

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

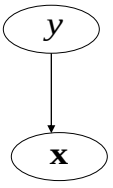
$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$$

- How about **class priors**?

## Learning Quadratic discriminant analysis (QDA)

- **Learning Class-conditional distributions**
  - **Learn parameters of 2 multivariate normal distributions**

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

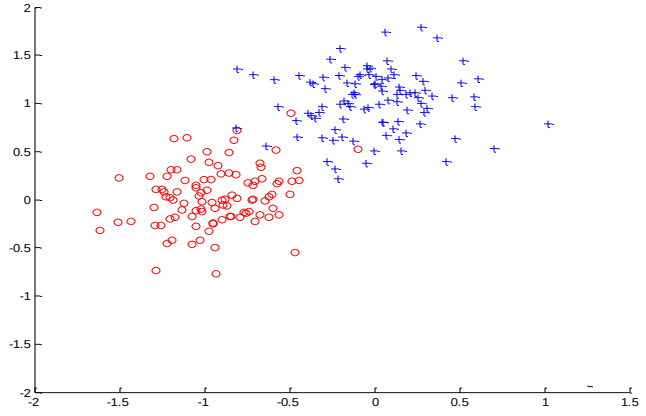



  - **Use the density estimation methods**

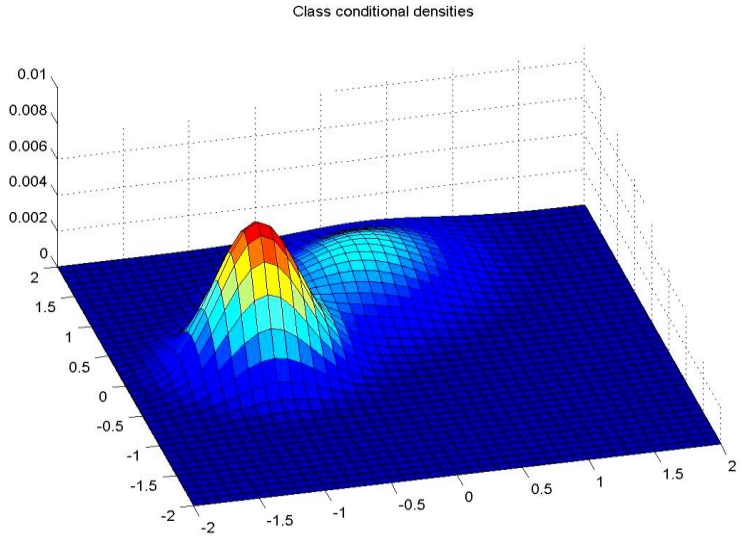
- **Learning Priors on classes (class 0,1)** $y \sim Bernoulli$
  - **Learn the parameter of the Bernoulli distribution**
  - **Again use the density estimation methods**

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

# QDA




# 2 Gaussian class-conditional densities



# QDA: Making class decision

Basically we need to design discriminant functions

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

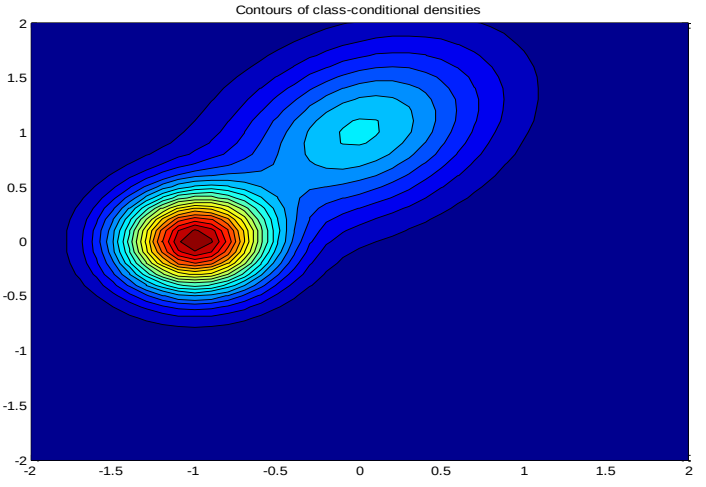
$$\underbrace{p(y=1\,|\,\mathbf{x})}_{g_1(\mathbf{x})} > \underbrace{p(y=0\,|\,\mathbf{x})}_{g_0(\mathbf{x})} \quad \Rightarrow \quad \begin{array}{l}\text{then } y=1\\ \text{else } y=0\end{array}$$

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

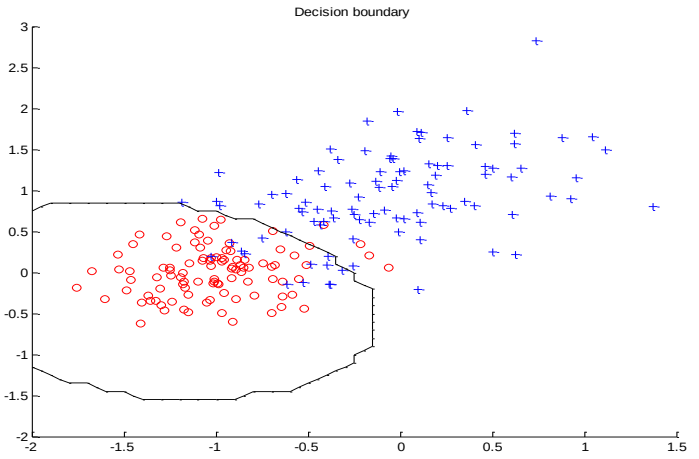
- **Notice it is sufficient to compare:**

$$p(\mathbf{x}\,|\,\mu_1,\boldsymbol{\Sigma}_1)\,p(y=1) > p(\mathbf{x}\,|\,\mu_0,\boldsymbol{\Sigma}_0)\,p(y=0)$$

# QDA: Quadratic decision boundary



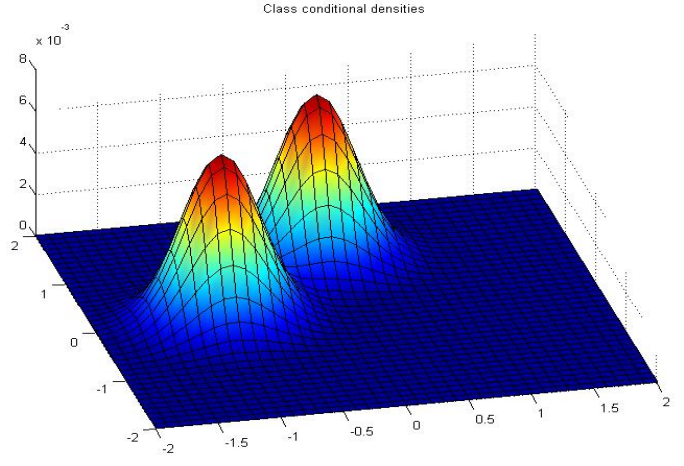
# QDA: Quadratic decision boundary



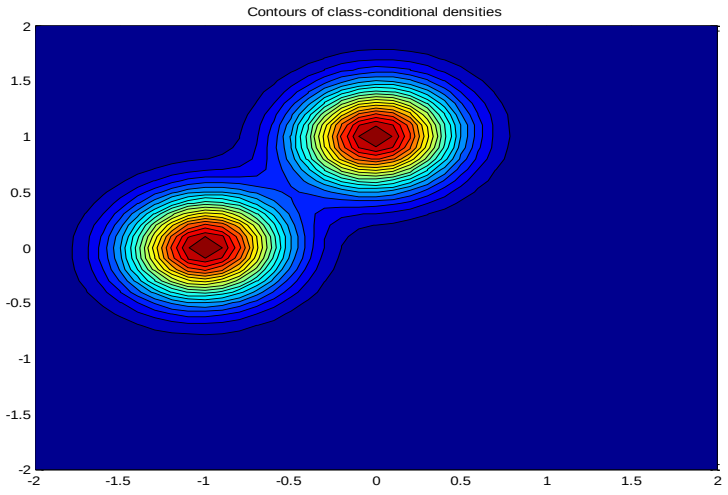

# Linear discriminant analysis (LDA)

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

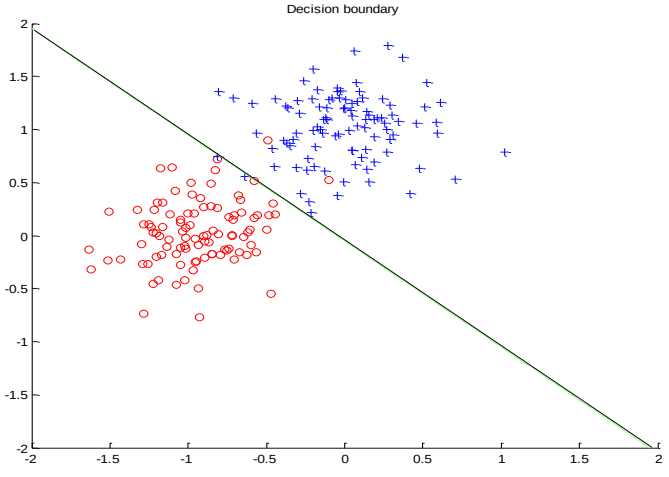
  $\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}),\ y = 1$



# LDA: Linear decision boundary




# LDA: linear decision boundary



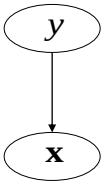
## Generative classification models

**Idea:**

1. **Represent and learn the distribution** $p(\mathbf{x}, y)$
2. **Model is <u>able</u> to sample (generate) data instances** $(\mathbf{x}, y)$
3. **The model is used to get probabilistic discriminant functions** $g_o(\mathbf{x}) = p(y=0 \mid \mathbf{x}) \quad g_1(\mathbf{x}) = p(y=1 \mid \mathbf{x})$

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

- $p(\mathbf{x} \mid y)$ = **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
  $p(y=0) + p(y=1) = 1$




## Naïve Bayes classifier

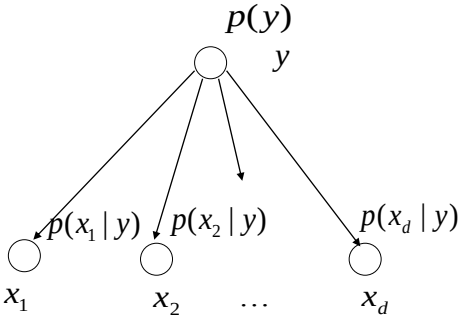
**A generative classifier model with an additional simplifying assumption:**

- **All input attributes are conditionally independent of each other given the class**.
- One of the basic ML classification models (often performs very well in practice)

So we have:

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

$$p(\mathbf{x} \mid y) = \prod_{i=1}^{d} p(x_i \mid y)$$



## Learning parameters of the model

**Much simpler density estimation problems**

- We need to learn:

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

- Because of the assumption of the conditional independence we need to learn:

  for every input variable i: $p(x_i \mid y=0)$ and $p(x_i \mid y=1)$

- **Much easier if the number of input attributes is large**
- **Also, the model gives us a flexibility to represent input attributes of different forms !!!**
- E.g. one attribute can be modeled using the Bernoulli, the other using Gaussian density, or a Poisson distribution

## Making a class decision for the Naïve Bayes

**Discriminant functions**

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

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

$$p(y=1 \mid \mathbf{x}) = \frac{\left(\prod_{i=1}^{d} p(x_i \mid \Theta_{1,i})\right) p(y=1)}{\left(\prod_{i=1}^{d} p(x_i \mid \Theta_{1,i})\right) p(y=0) + \left(\prod_{i=1}^{d} p(x_i \mid \Theta_{2,i})\right) p(y=1)}$$

## Next: two interesting questions

**(1) Two probabilistic models with linear decision boundaries:**

- **Logistic regression**
- **LDA model (2 Gaussians with the same covariance matrices**

$$x \sim N(\mu_0, \Sigma) \quad \text{for} \quad y = 0$$
$$x \sim N(\mu_1, \Sigma) \quad \text{for} \quad y = 1$$

- **Question: Is there any relation between the two models?**

**(2) Two models with the same gradient:**

- **Linear model for regression**
- **Logistic regression model for classification**

have the same gradient update

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

- **Question: Why is the gradient the same?**

---

## Logistic regression and generative models

- **Two models with linear decision boundaries:**
  - **Logistic regression**
  - **Generative model with 2 Gaussians with the same covariance matrices**

$$x \sim N(\mu_0, \Sigma) \quad \text{for} \quad y = 0$$
$$x \sim N(\mu_1, \Sigma) \quad \text{for} \quad y = 1$$

**Question: Is there any relation between the two models?**

**Answer: Yes, the two models are related !!!**

- When we have **2 Gaussians with the same covariance matrix** the probability of y given **x** has the form of a logistic regression model **!!!**

$$p(y = 1 | \mathbf{x}, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = g(\mathbf{w}^T \mathbf{x})$$

# Logistic regression and generative models

- **Members of the exponential family can be often more naturally described as**

$$f(\mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\varphi}) = h(x, \boldsymbol{\varphi}) \exp\left\{ \frac{\boldsymbol{\theta}^T \mathbf{x} - A(\boldsymbol{\theta})}{a(\boldsymbol{\varphi})} \right\}$$

$\boldsymbol{\theta}$ - A location parameter $\boldsymbol{\varphi}$ - A scale parameter

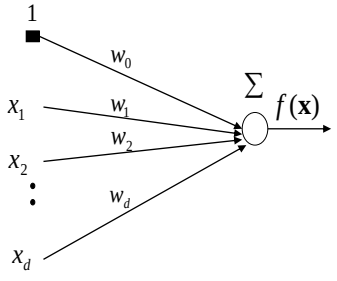
- **Claim:** A **logistic regression** is a correct model when class conditional densities are from the same distribution in the exponential family and have **the same scale factor** $\boldsymbol{\varphi}$
- **Very powerful result !!!!**
  - **We can represent posteriors of many distributions with the same small logistic regression model**



# The gradient puzzle …

**Linear regression**

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



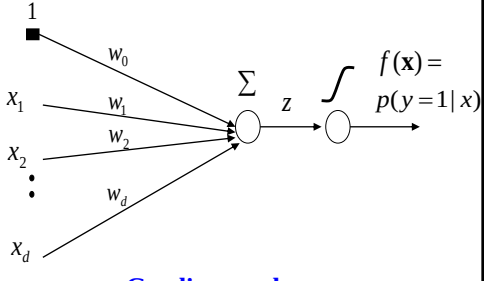

**Gradient update:**

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

**Online:** $\mathbf{w} \leftarrow \mathbf{w} + \alpha (y - f(\mathbf{x})) \mathbf{x}$

**Logistic regression**

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




**Gradient update:**

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

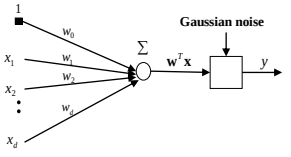
**Online:** $\mathbf{w} \leftarrow \mathbf{w} + \alpha (y - f(\mathbf{x})) \mathbf{x}$

**The same**

## The gradient puzzle …

- The **same simple gradient update rule** derived for both the linear and logistic regression models
- Where the magic comes from?
- Under the **log-likelihood** measure the function models and the models for the output selection fit together:
  - **Linear model + Gaussian noise**

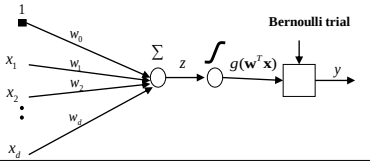
$$y = \mathbf{w}^T\mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2)$$




  - **Logistic + Bernoulli**

$$y = \text{Bernoulli}(\theta)$$

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




## Generalized linear models (GLIMs)

**Assumptions:**

- The conditional mean (expectation) is:

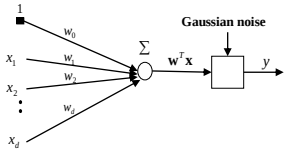
$$\mu = f(\mathbf{w}^T\mathbf{x})$$

  - Where $f(.)$ is a **response function**
- Output y is characterized by an exponential family distribution with a conditional mean $\mu$

**Examples:**

  - **Linear model + Gaussian noise**

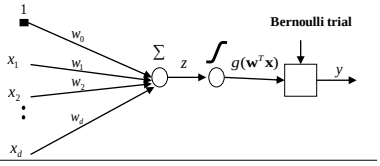
$$y = \mathbf{w}^T\mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2)$$




  - **Logistic + Bernoulli**

$$y \approx \text{Bernoulli}(\theta)$$

$$\theta = g(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$



## Generalized linear models (GLIMs)

- **A canonical response functions $f(.)$ :**
  - **encoded in the sampling distribution**

$$p(\mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\varphi}) = h(x, \boldsymbol{\varphi}) \exp\left\{ \frac{\boldsymbol{\theta}^T \mathbf{x} - A(\boldsymbol{\theta})}{a(\boldsymbol{\varphi})} \right\}$$

- **Leads to a simple gradient form**
- **Example: Bernoulli distribution**

$$p(x \mid \mu) = \mu^x (1-\mu)^{1-x} = \exp\left\{ \log\left( \frac{\mu}{1-\mu} \right) x + \log(1-\mu) \right\}$$

$$\theta = \log\left( \frac{\mu}{1-\mu} \right) \qquad \mu = \frac{1}{1+e^{-\theta}}$$

  - **Logistic function matches the Bernoulli**

## Evaluation of classifiers

## Classification model learning

**Learning:**

- Many different ways and objective criteria used to learn the classification models. Examples:
  - Mean squared errors to learn the discriminant functions
  - Negative log likelihood (logistic regression)

**Evaluation:**

- One possibility: Use the same error criteria as used during the learning (apply to train & test data). Problems:
  - May work for discriminative models
  - Harder to interpret for humans.
- **Question:** how to more naturally evaluate the classifier performance?

## Evaluation of classification models

For any data set we use to test the classification model on we can build a **confusion matrix:**

- Counts of examples with:
- class label $\omega_j$ that are classified with a label $\alpha_i$

| | | target | |
|---|---|---|---|
| | | $\omega = 1$ | $\omega = 0$ |
| **predict** | $\alpha = 1$ | 140 | 17 |
| | $\alpha = 0$ | 20 | 54 |

## Evaluation of classification models

**Confusion matrix entries are often normalized with respect to the number of examples N to get proportions of the different agreements and disagreements among predicted and target values**

| | | target $\omega = 1$ | target $\omega = 0$ |
|---|---|---|---|
| predict | $\alpha = 1$ | $140/231$ | $17/231$ |
| | $\alpha = 0$ | $20/231$ | $54/231$ |

## Basic evaluation statistics

Basic statistics calculated from the confusion matrix:

| | | target $\omega = 1$ | target $\omega = 0$ |
|---|---|---|---|
| predict | $\alpha = 1$ | 140 | 17 |
| | $\alpha = 0$ | 20 | 54 |

**Classification Accuracy** = 194/231

## Basic evaluation statistics

Basic statistics calculated from the confusion matrix:

| | | target $\omega = 1$ | target $\omega = 0$ |
|---|---|---|---|
| **predict** | $\alpha = 1$ | 140 | 17 |
| | $\alpha = 0$ | 20 | 54 |

**Classification Accuracy** = 194/231

**Misclassificion Error** = 37/231 = 1 - **Accuracy**

## Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

| | | target $\omega = 1$ | target $\omega = 0$ |
|---|---|---|---|
| **predict** | $\alpha = 1$ | $TP$ | $FP$ |
| | $\alpha = 0$ | $FN$ | $TN$ |

*TP: True positive (hit)*
*FP: False positive (false alarm)*
*TN: True negative (correct rejection)*
*FN: False negative (a miss)*

## Additional statistics

- **Sensitivity (recall)**

$$SENS = \frac{TP}{TP + FN}$$

- **Specificity**

$$SPEC = \frac{TN}{TN + FP}$$

- **Positive predictive value (precision)**

$$PPT = \frac{TP}{TP + FP}$$

- **Negative predictive value**

$$NPV = \frac{TN}{TN + FN}$$

## Binary classification: additional statistics

- **Confusion matrix**

| | | target 1 | target 0 | |
|---|---|---|---|---|
| predict | 1 | 140 | 10 | $PPV = 140/150$ |
| | 0 | 20 | 180 | $NPV = 180/200$ |
| | | $SENS = 140/160$ | $SPEC = 180/190$ | |

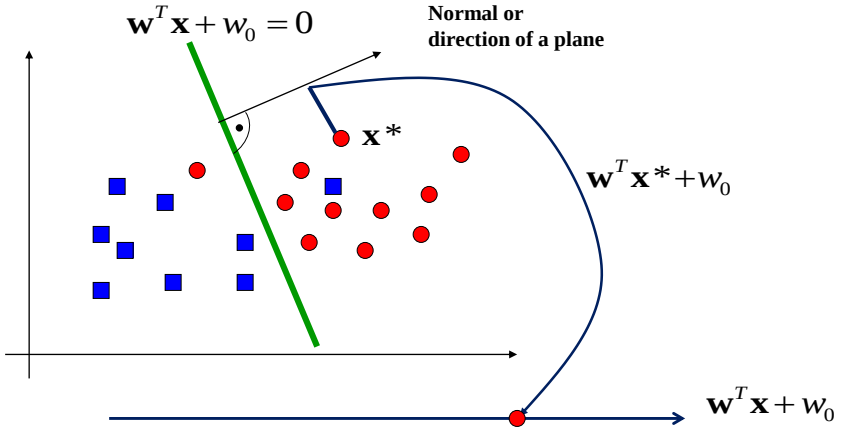
**Row and column quantities:**

- Sensitivity (SENS)
- Specificity (SPEC)
- Positive predictive value (PPV)
- Negative predictive value (NPV)

## Binary classification models

Often project data points to one dimensional space:

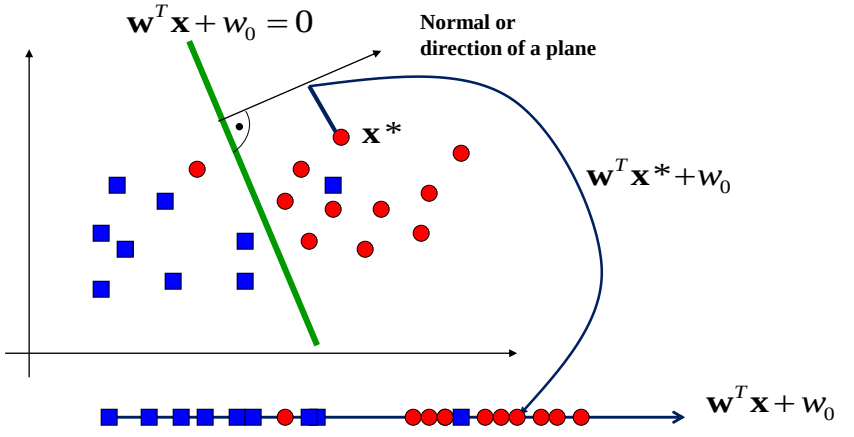
**Defined for example by:** $\mathbf{w^T x}+w_0$ or $p(y=1|\mathbf{x},\mathbf{w})$




## Binary classification models

Often project data points to one dimensional space:

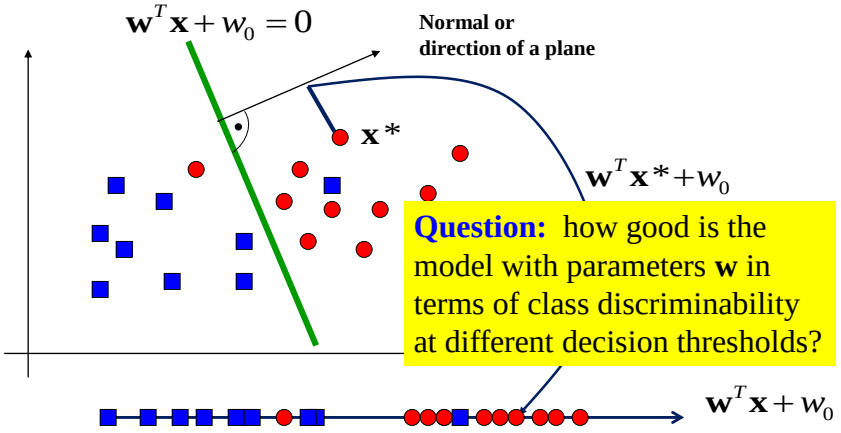
**Defined for example by:** $\mathbf{w^T x}+w_0$ or $p(y=1|\mathbf{x},\mathbf{w})$



# Binary classification models

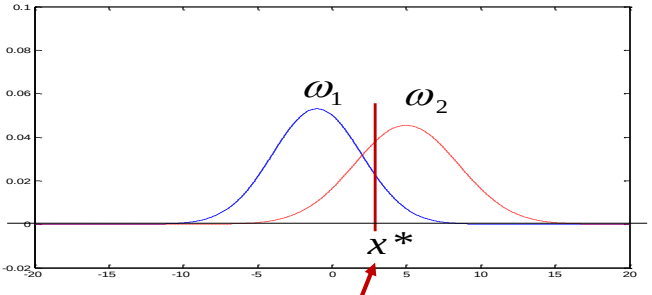
Often project data points to one dimensional space:

**Defined for example by:** $\mathbf{w}^T\mathbf{x}+w_0$ or $p(y=1|\mathbf{x},\mathbf{w})$




**Question:** how good is the model with parameters **w** in terms of class discriminability at different decision thresholds?

# Receiver Operating Characteristic (ROC)




- **Probabilities:**
  - *SENS*
  - *SPEC*

**threshold**

$p(x > x^* | \mathbf{x} \in \omega_2)$

$p(x < x^* | \mathbf{x} \in \omega_1)$

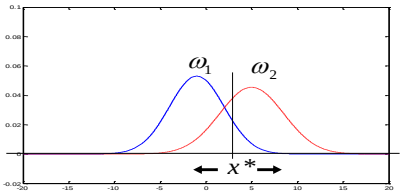
# Receiver Operating Characteristic (ROC)

- **ROC curve plots :**

SN= $p(x > x^* \mid \mathbf{x} \in \omega_2)$

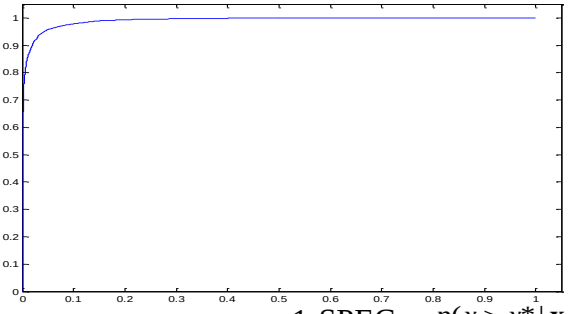
1-SP= $p(x > x^* \mid \mathbf{x} \in \omega_1)$

**for different x***



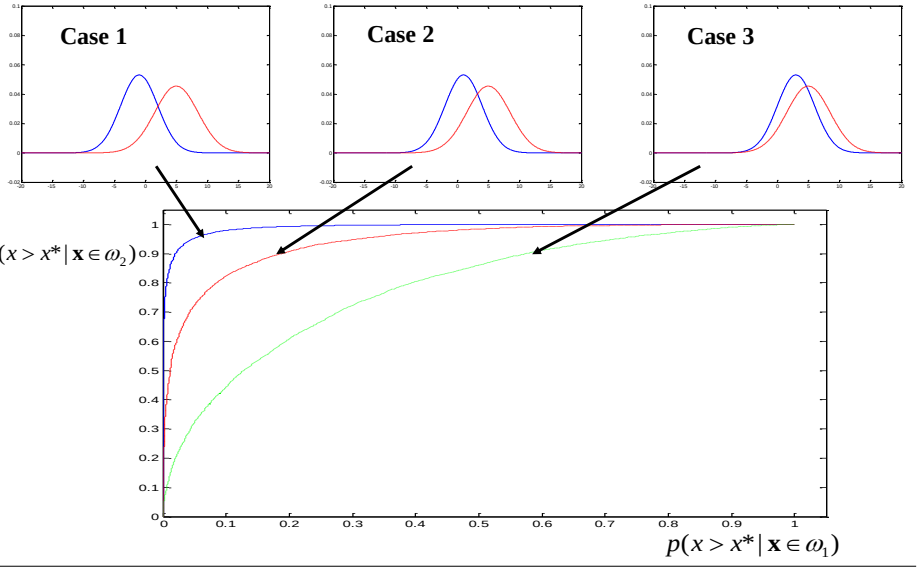

SENS

$p(x > x^* \mid \mathbf{x} \in \omega_2)$



1-SPEC $p(x > x^* \mid \mathbf{x} \in \omega_1)$

# ROC curve



## Receiver operating characteristic

- **ROC**
  - shows the discriminability between the two classes under different thresholds representing different decision biases
- **Decision bias**
  - can be changed using the different loss function

- **Quality of a classification model:**
  - Area under the ROC
  - Best value 1, worst (no discriminability): 0.5