CS 2750 Machine Learning
Lecture 9

Classification learning II

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

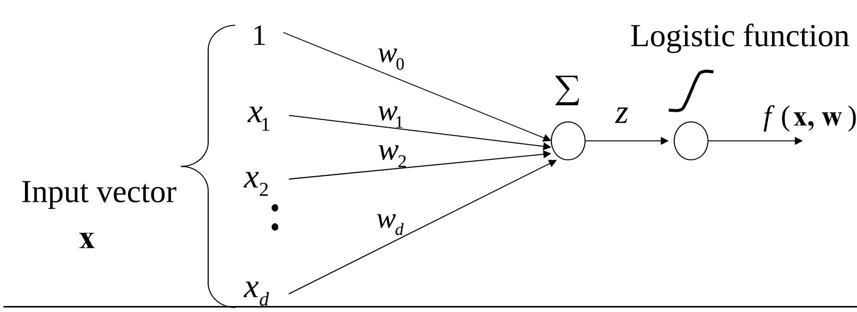
Logistic regression model

- **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

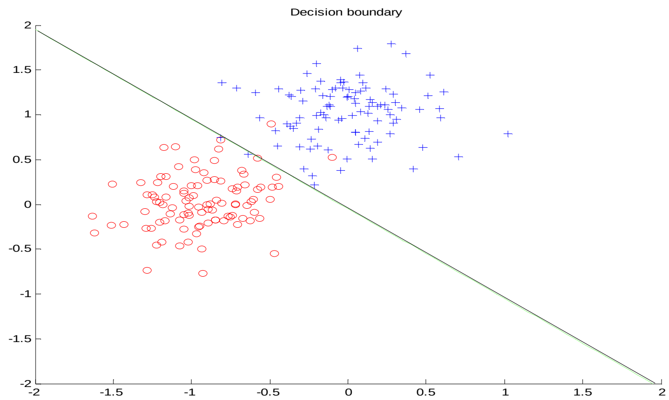
$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$



Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**

  Example: 2 classes (blue and red points)



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:**

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

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

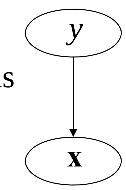
Generative approach to classification

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



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

  binary classification: two class-conditional distributions

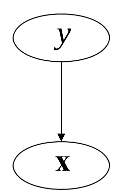
  $p(\mathbf{x} \mid y=0) \qquad p(\mathbf{x} \mid y=1)$
- $p(y)$ = **Priors on classes** **-** probability of class *y*

  binary classification: Bernoulli distribution

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

Quadratic discriminant analysis (QDA)

Model:



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

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

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

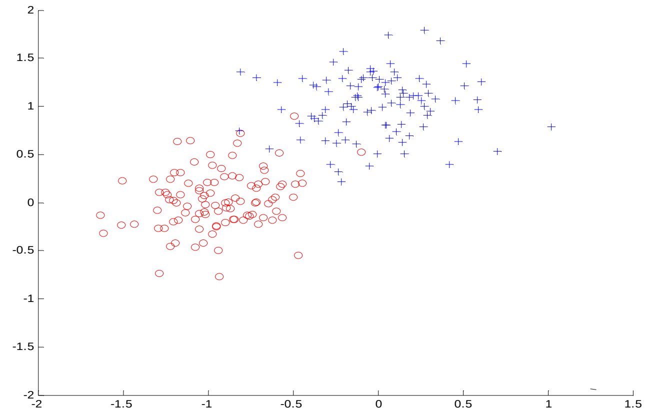
  Multivariate normal $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

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

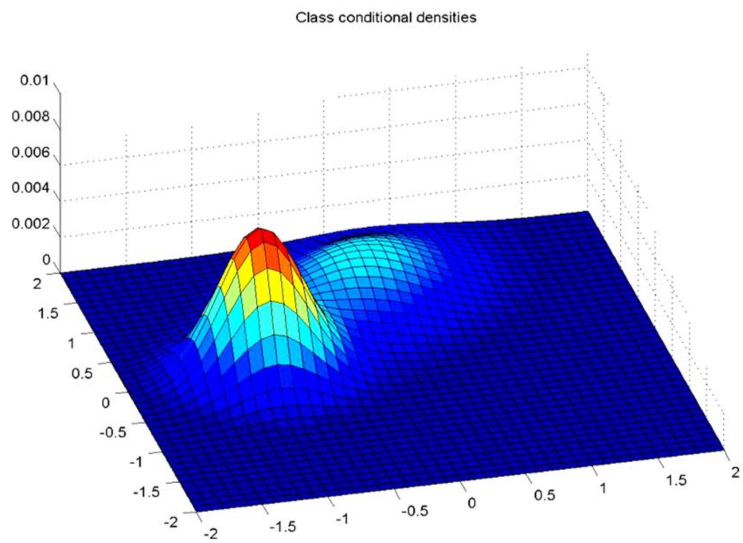
- **Priors on classes (class 0,1)** $y \sim Bernoulli$
  - **Bernoulli distribution**

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

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)

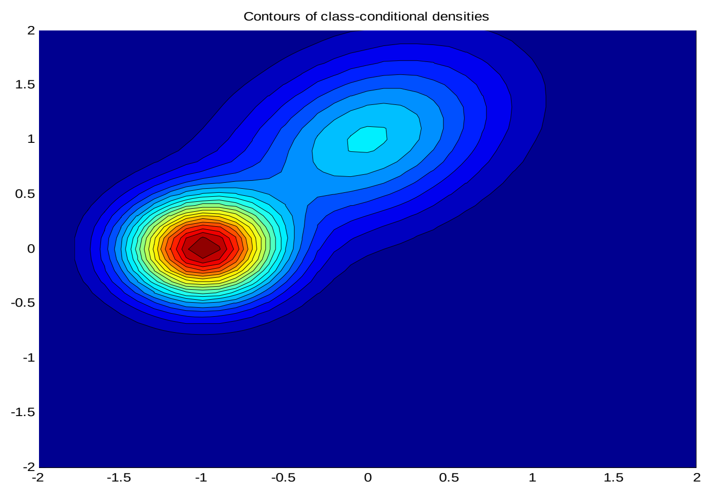
$$\underbrace{p(\mathbf{x} \mid \mu_1, \mathbf{\Sigma}_1)}_{g_1(\mathbf{x})} > \underbrace{p(\mathbf{x} \mid \mu_0, \mathbf{\Sigma}_0)}_{g_0(\mathbf{x})} \Rightarrow \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

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

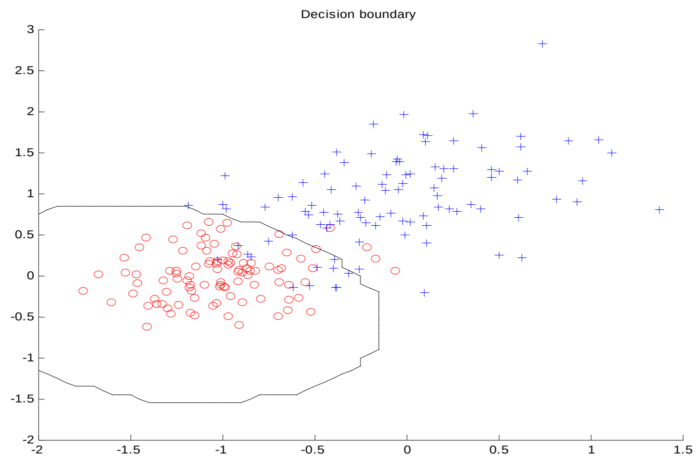
$$p(y=1 \mid \mathbf{x}) > p(y=0 \mid \mathbf{x}) \quad \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

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

QDA: Quadratic decision boundary



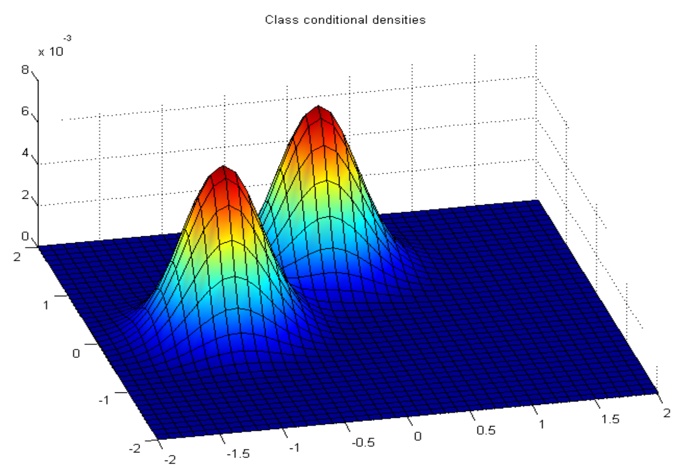
QDA: Quadratic decision boundary



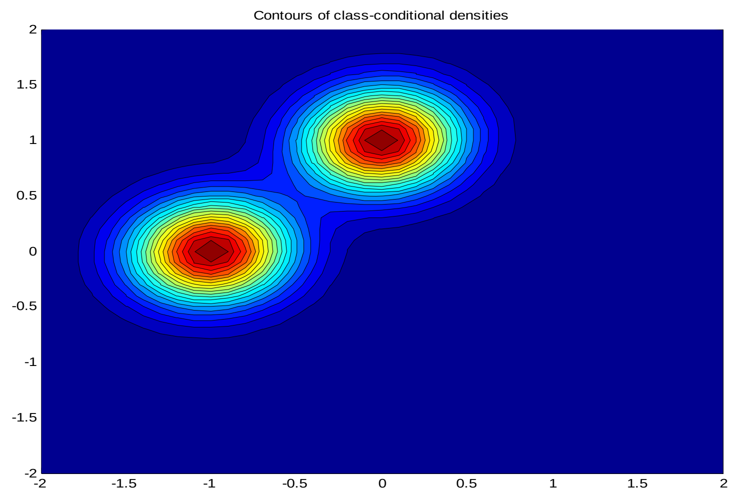
Linear discriminant analysis (LDA)

- When covariances are the same

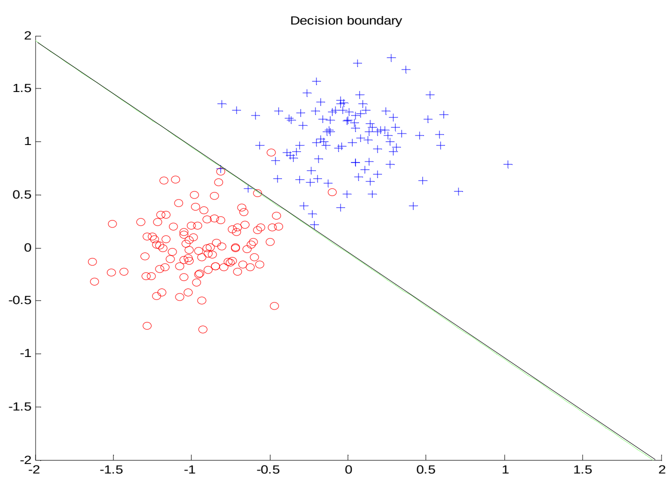
$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}),\ y = 0$$
$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}),\ y = 1$$



LDA: Linear decision boundary



LDA: linear decision boundary



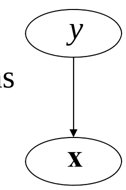
Generative classification models

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

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

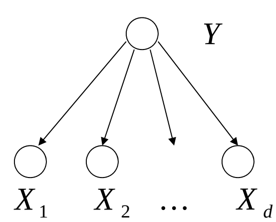
Naïve Bayes classifier

- **A generative classifier model with an additional simplifying assumption:**
  - **All input attributes are conditionally independent of each other given the class.**

  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 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 different of different forms !!!**
- E.g. one attribute can be modeled using the Bernoulli, the other as Gaussian density, or as a Poisson distribution

Making a class decision for the Naïve Bayes

Discriminant functions

- **Likelihood of data** – choose the class that explains the input data (**x**) better (likelihood of the data)

$$\underbrace{\prod_{i=1}^{d} p(x_i \mid \Theta_{1,i})}_{g_1(\mathbf{x})} > \underbrace{\prod_{i=1}^{d} p(x_i \mid \Theta_{2,i})}_{g_0(\mathbf{x})} \Rightarrow \text{ then } y=1 \text{ else } y=0$$

- **Posterior of a class** – choose the class with better posterior probability $p(y=1 \mid \mathbf{x}) > p(y=0 \mid \mathbf{x})$ then $y=1$ else $y=0$

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

Back to logistic regression

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

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

- **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 **!!!**

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

When is the logistic regression model correct?

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

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

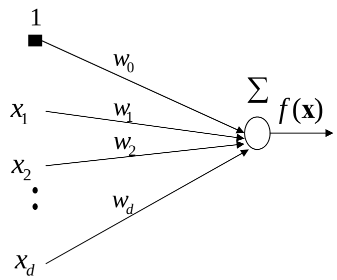
$\boldsymbol{\theta}$ - A location parameter $\quad$ $\boldsymbol{\varphi}$ - A scale parameter

- **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 network**

Linear units

Linear regression

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



Gradient update:

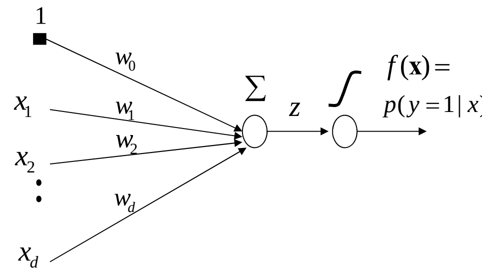
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))\mathbf{x}_i$$

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

The same

Logistic regression

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



Gradient update:

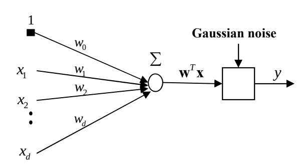
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))\mathbf{x}_i$$

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

Gradient-based learning

- 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**

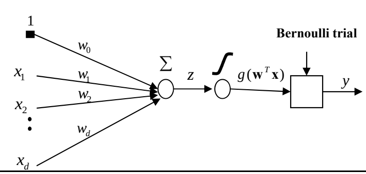
    $$y = \mathbf{w}^T \mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2)$$



  - **Logistic + Bernoulli**

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

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



Generalized linear models (GLIM)

Assumptions:

- The conditional mean (expectation) is:

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

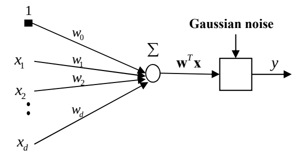
  – Where $f(.)$ is a **response function**

- Output y is characterized by an exponential family distribution with a conditional mean μ

Examples:

- **Linear model + Gaussian noise**

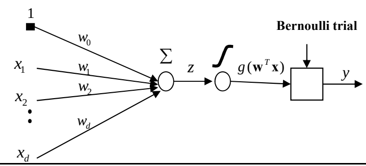
$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2)$$



- **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

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

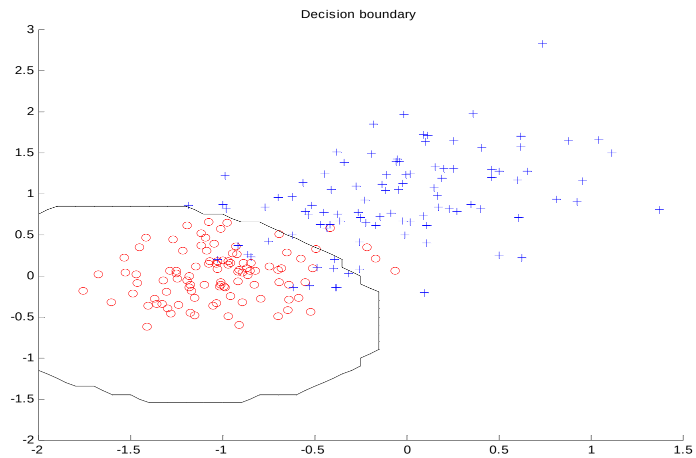
$$p(x \mid \mu) = \mu^x (1-\mu)^{1-x} = \exp\left\{ \log\left(\frac{\mu}{1-\mu}\right) x + \log(1-\mu) \right\}$$

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

  – **Logistic function matches the Bernoulli**

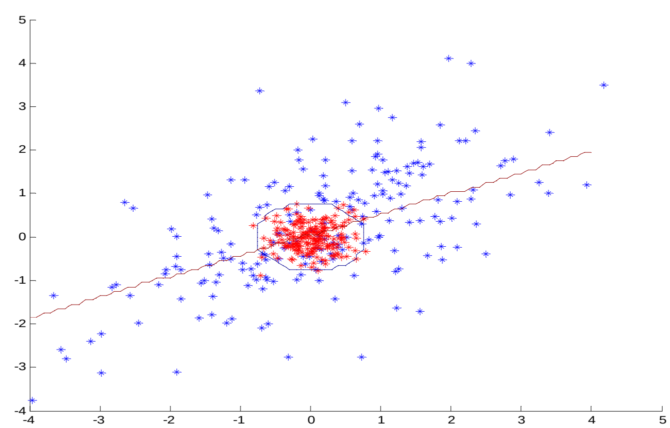
When does the logistic regression fail?

- Quadratic decision boundary is needed



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

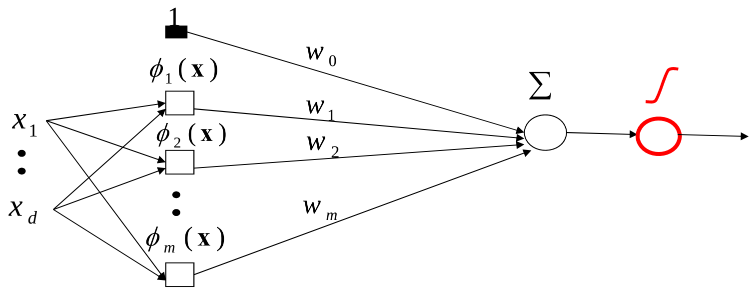
Linear regression

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

Logistic regression

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

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



Evaluation of classifiers

Evaluation

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

- Counts of examples with:
- class label ω_j that are classified with a label α_i

target

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

Evaluation

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

target

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

agreement

Error: ?

Evaluation

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

target

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

agreement

Error: = 37/231

Accuracy = 1- Error = 194/231

Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

target

predict		$\omega = 1$	$\omega = 0$
	$\alpha = 1$	*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	$NPV = 180/200$
		$SENS = 140/160$	$SPEC = 180/190$	

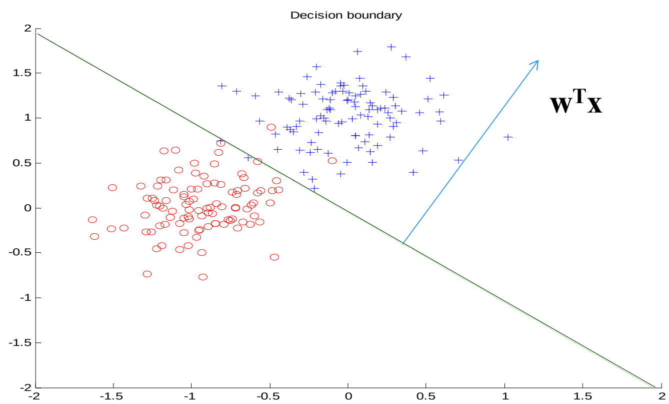
Row and column quantities:

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

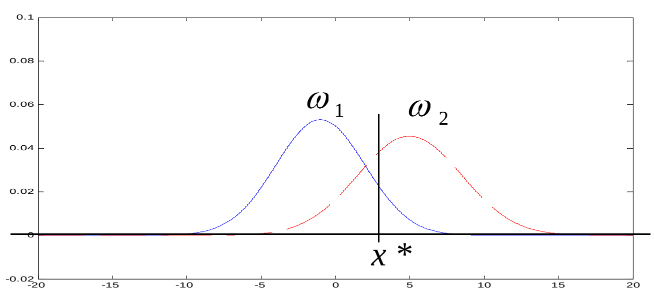
Classifiers

Project datapoints to one dimensional space:

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



Binary decisions: Receiver Operating Curves



- **Probabilities:**
  - *SENS* $p(x > x^* \mid \mathbf{x} \in \omega_2)$
  - *SPEC* $p(x < x^* \mid \mathbf{x} \in \omega_1)$

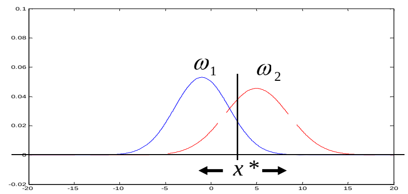
Receiver Operating Characteristic (ROC)

- **ROC curve plots :**

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

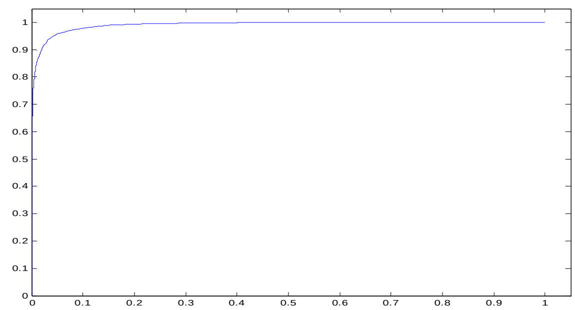
  1-SP= $p(x > x^* \mid \mathbf{x} \in \omega_1)$

  for different x*



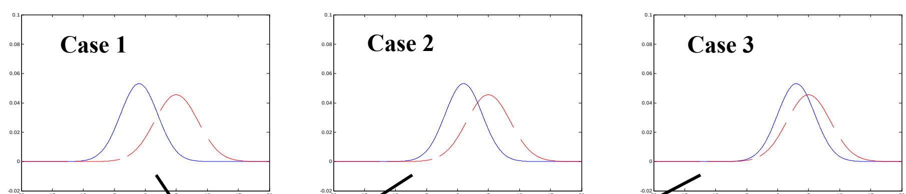
SENS

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

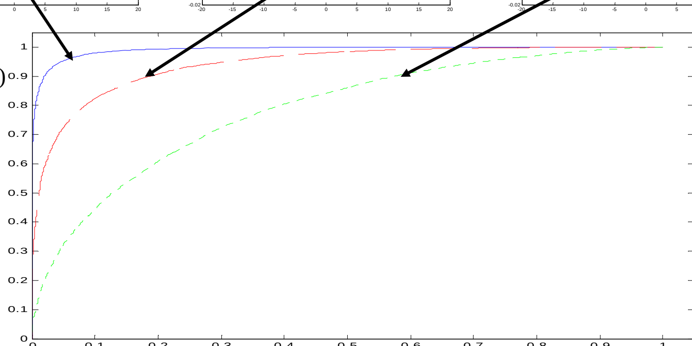


1-SPEC $p(x > x^* \mid \mathbf{x} \in \omega_1)$

ROC curve



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



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

Receiver operating characteristic

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

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