CS 1675 Intro to Machine Learning
Lecture 8

Logistic regression

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 $g_i(\mathbf{x})$

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

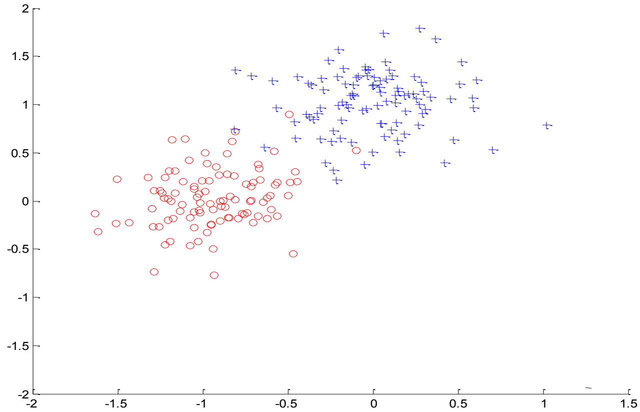
Discriminant functions

- 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 $g_i(\mathbf{x})$

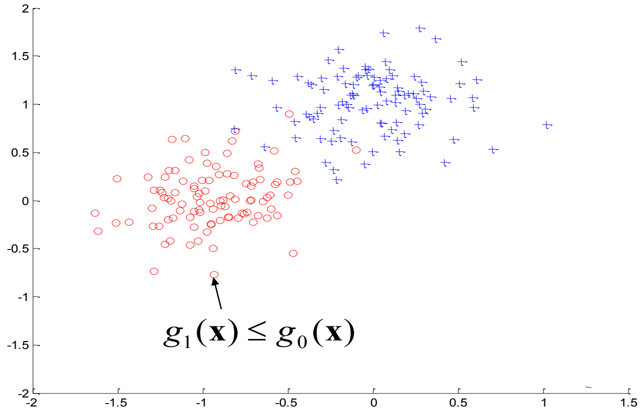
$$y^* = \arg\max_i \; g_i(\mathbf{x})$$

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

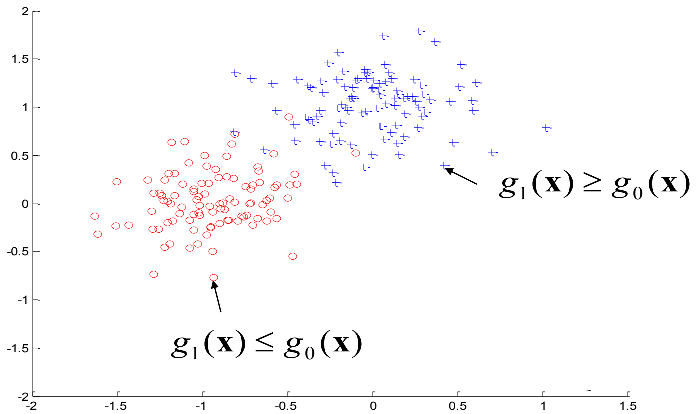
Discriminant functions



Discriminant functions

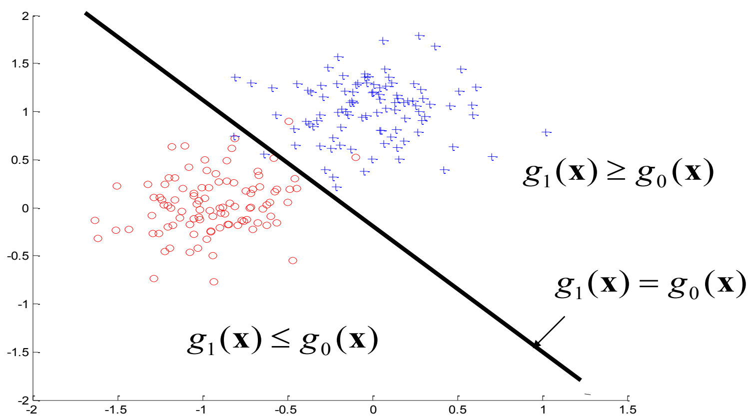


Discriminant functions

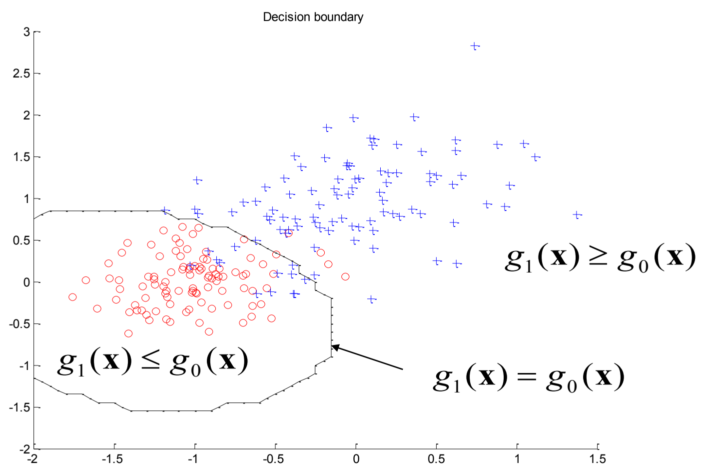


Discriminant functions

- **Decision boundary:** **discriminant functions are equal**



Quadratic decision boundary



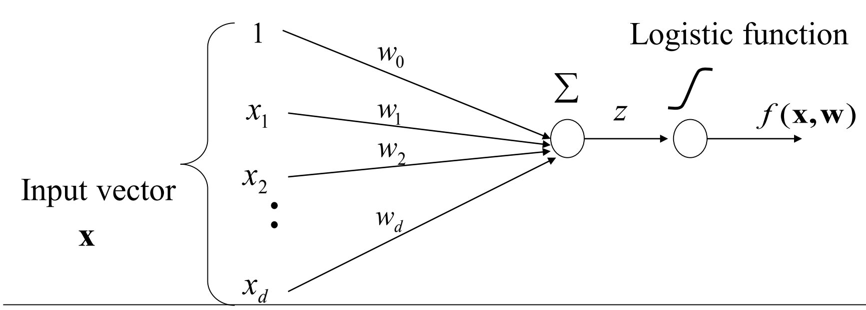
Logistic regression model

- **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

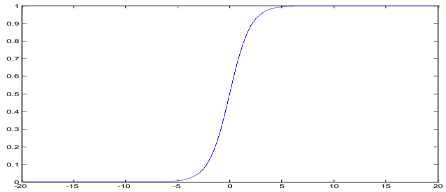
$$f(\mathbf{x},\mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



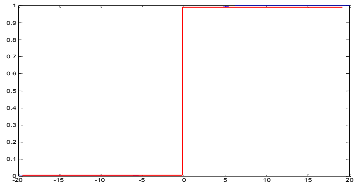
Logistic function

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

- 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



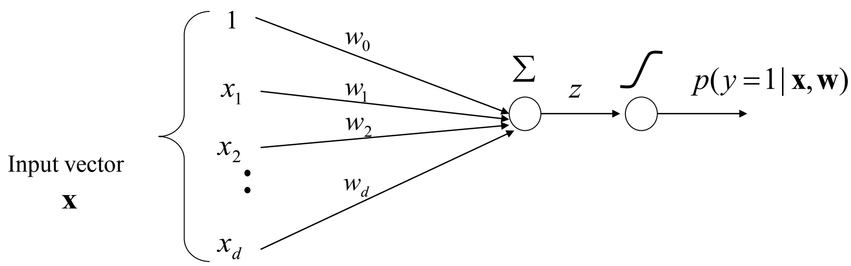
Logistic (smooth) switching



Threshold (hard) switching

Logistic regression model

- **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 regression

- We learn **a probabilistic function**

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

  – where *f* describes the probability of class 1 given $\mathbf{x}$

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

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

- Making decisions with the logistic regression model:

?

Logistic regression

- We learn **a probabilistic function**

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

  – where *f* describes the probability of class 1 given $\mathbf{x}$

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

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

- Making decisions with the logistic regression model:

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:

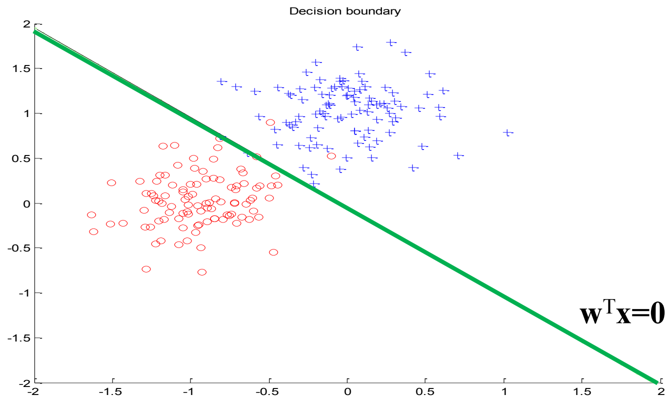
$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^{\mathbf{T}}\mathbf{x})}{g(\mathbf{w}^{\mathbf{T}}\mathbf{x})} = 0$$

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

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

- **Notation:** $\mu_i = p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$
- **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:**

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

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

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**

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

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

Derivative of a logistic function

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

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

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

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

Logistic regression. Online gradient descent

- **On-line component of the loglikelihood**

$$J_{\text{online}}(D_i,\mathbf{w}) = -\left[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)\right]$$

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

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

Online logistic regression algorithm

Online-logistic-regression (*stopping_criterion*)

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

while *stopping_criterion = FALSE*

do **select** next data point $D_i = < \mathbf{x}_i, y_i >$

set $\alpha(i)$

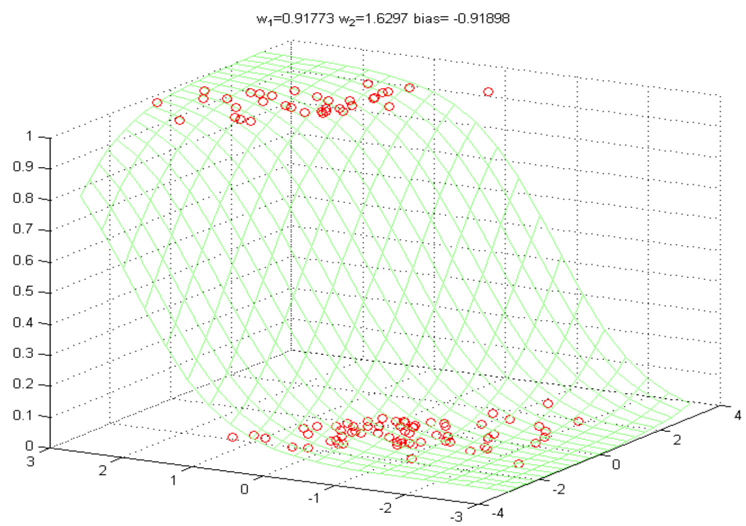
update weights (in parallel)

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

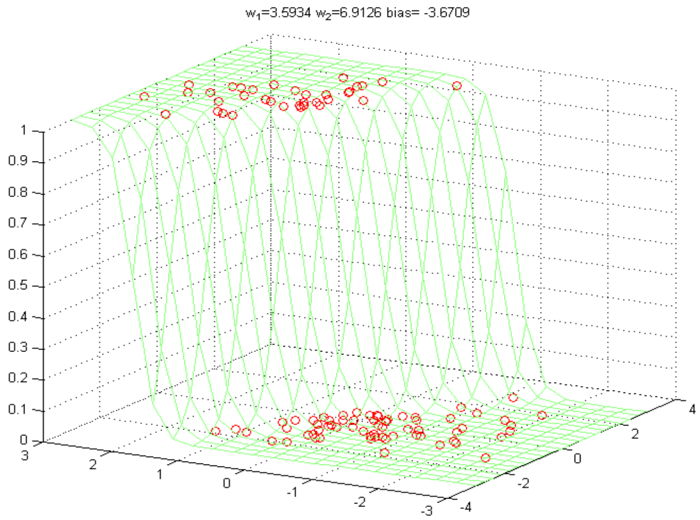
end

return weights $\mathbf{w}$

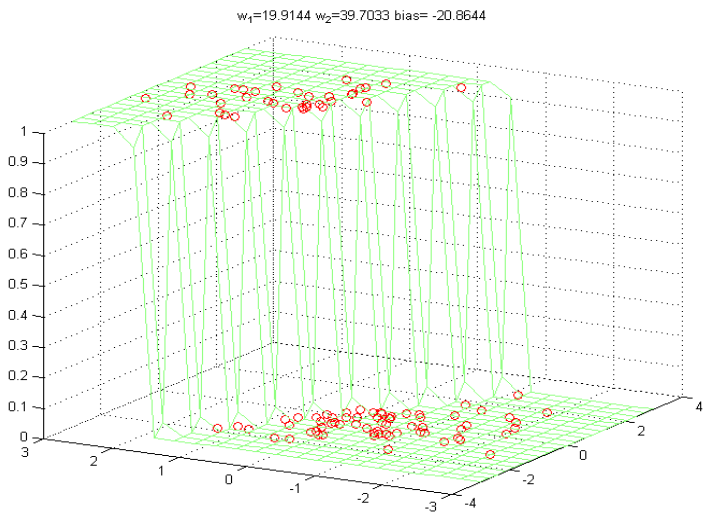
Online algorithm. Example.



Online algorithm. Example.



Online algorithm. Example.



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

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

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

Evaluation

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

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

agreement

Accuracy: ?

Evaluation

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

target

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

agreement

Accuracy = 194/231

Evaluation

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

target

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

Accuracy = 194/231

Error = ?

Evaluation

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

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

Accuracy = 194/231

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

Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

		target	
		$\omega = 1$	$\omega = 0$
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)**

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

target

		$\omega = 1$	$\omega = 0$
predict	$\alpha = 1$	TP	FP
	$\alpha = 0$	FN	TN

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)