CS 2750 Machine Learning
Lecture 8

Classification learning II

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

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

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 model we can build a **confusion matrix:**

target

		$\omega = 1$	$\omega = 0$
predict	$\alpha = 1$	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

		$\omega = 1$	$\omega = 0$
predict	$\alpha = 1$	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

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

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

- **Specificity**

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

- **Positive predictive value (precision)**

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

- **Negative predictive value**

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

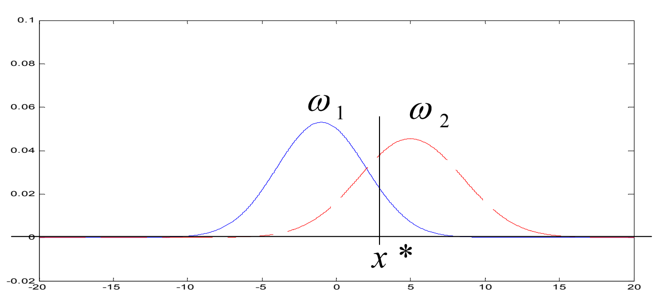
Binary classification: additional statistics

- **Confusion matrix**

		target		
		1	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 decisions: Receiver Operating Curves



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

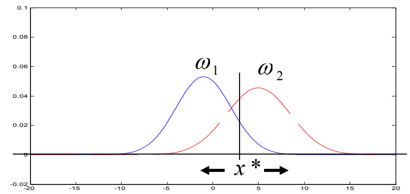
Receiver Operating Characteristic (ROC)

- **ROC curve plots :**

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

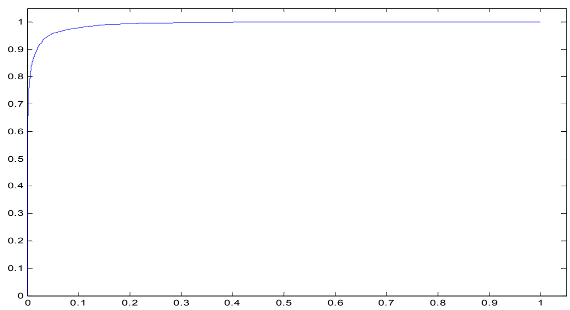
1-SP= $p(x > x^* \mid \mathbf{x} \in \omega_2)$

for different x*



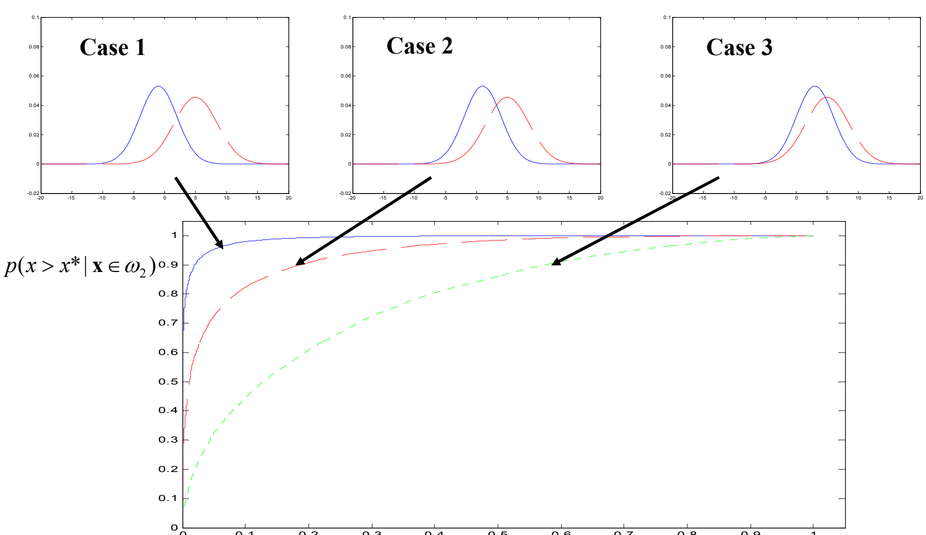
SENS

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



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

ROC curve



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

Receiver operating characteristic

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

Back to classification models

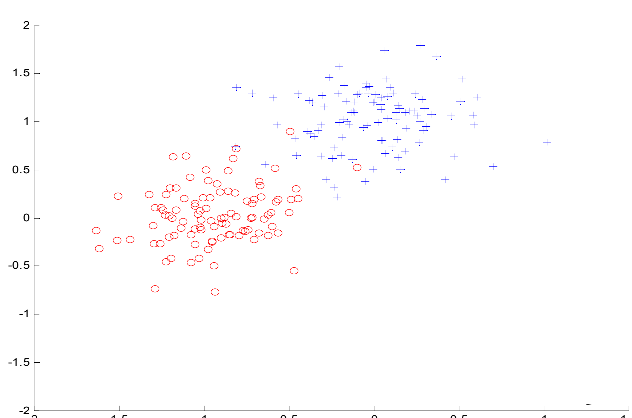
Discriminant functions

- One way to represent a **classifier is by using**
  – **Discriminant functions**
- **Works for binary and multi-way classification**

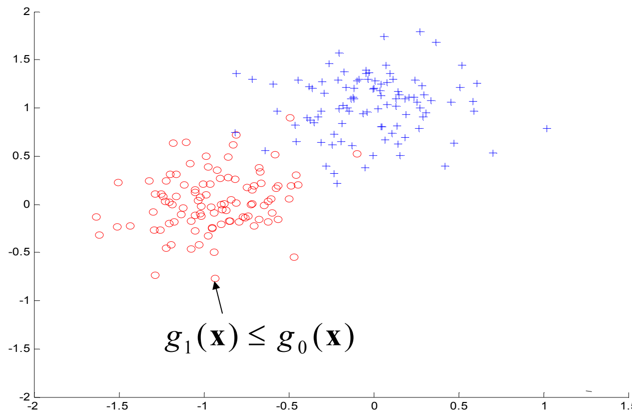
- **Idea:**
  – For every class $i = 0,1, \ldots k$ define a function $g_i(\mathbf{x})$ mapping $X \rightarrow \Re$
  – When the decision on input $\mathbf{x}$ should be made choose the class with the highest value of $g_i(\mathbf{x})$

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

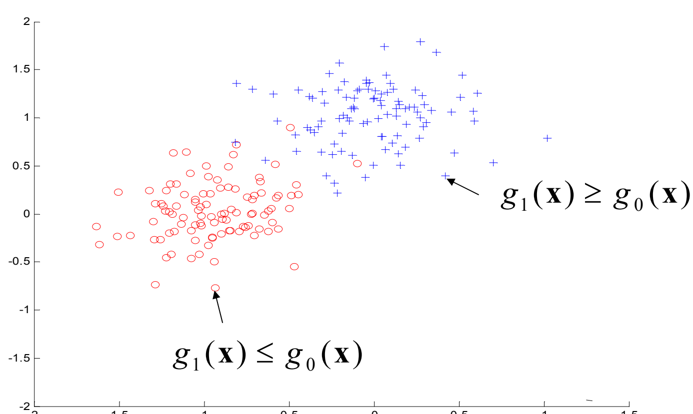
Discriminant functions



Discriminant functions

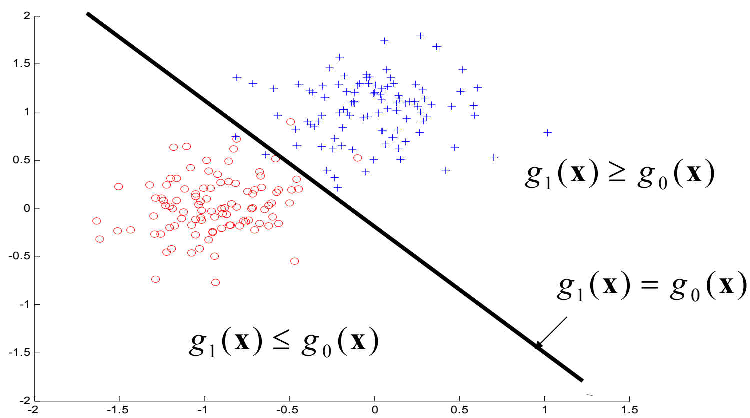


Discriminant functions

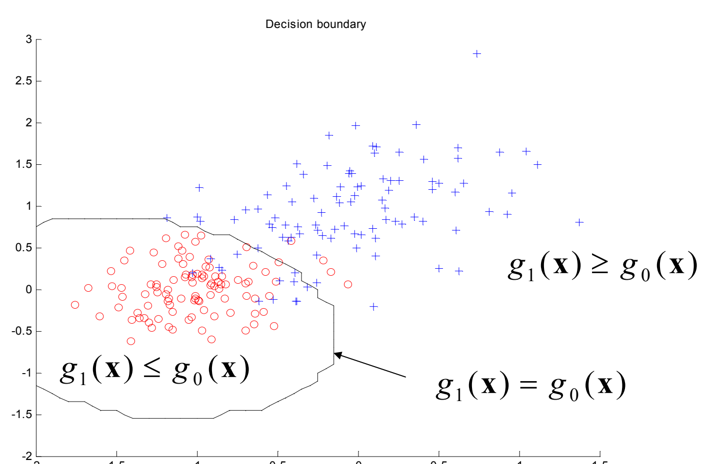


Discriminant functions

- Define **decision boundary**



Quadratic decision boundary



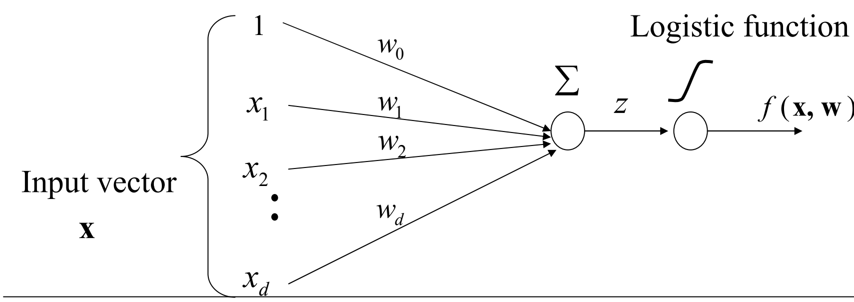
Logistic regression model

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

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

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

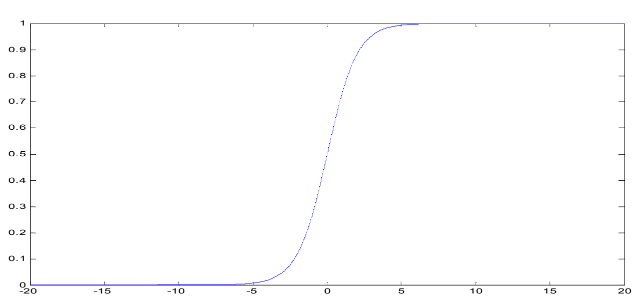
$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$



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]



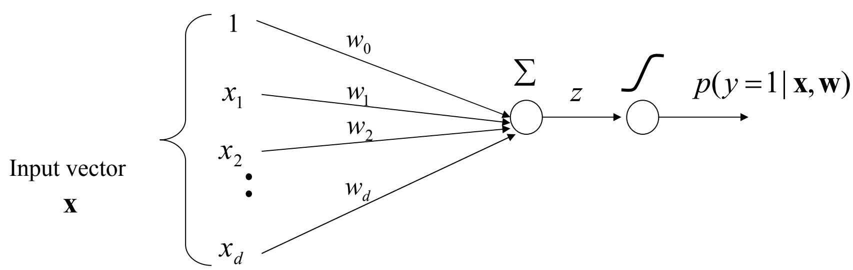
Logistic regression model

- **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 [0,1]**
  - **Probabilistic interpretation**

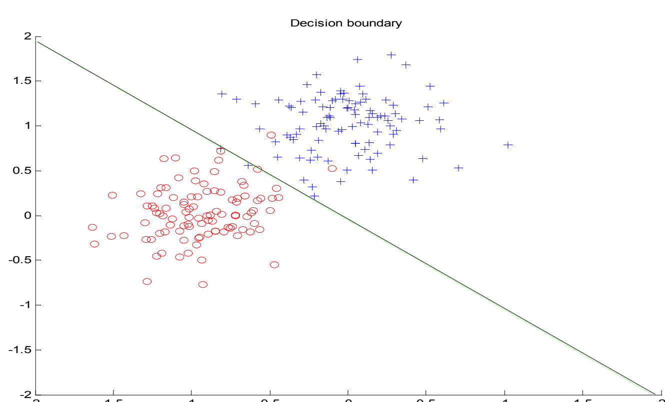
  $$f(\mathbf{x},\mathbf{w}) = p(y=1\,|\,\mathbf{w},\mathbf{x}) = g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**

  Example: 2 classes (blue and red points)



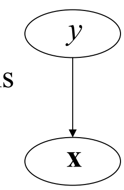
Generative approach to classification

Idea:

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

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

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

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

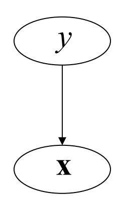
- $p(y)$ = **Priors on classes** **-** probability of class y

  binary classification: Bernoulli distribution

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

Quadratic discriminant analysis (QDA)

Model:



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

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

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

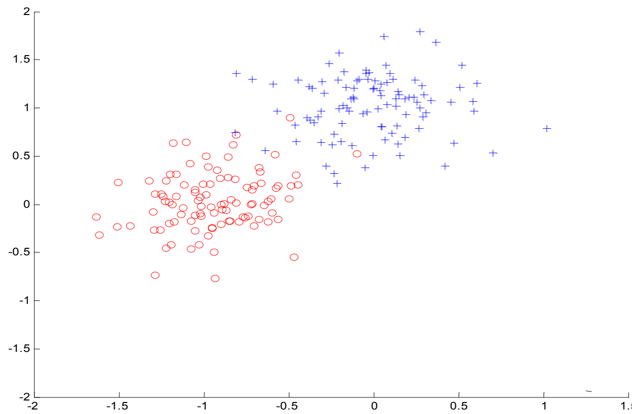
Multivariate normal $\quad \mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

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

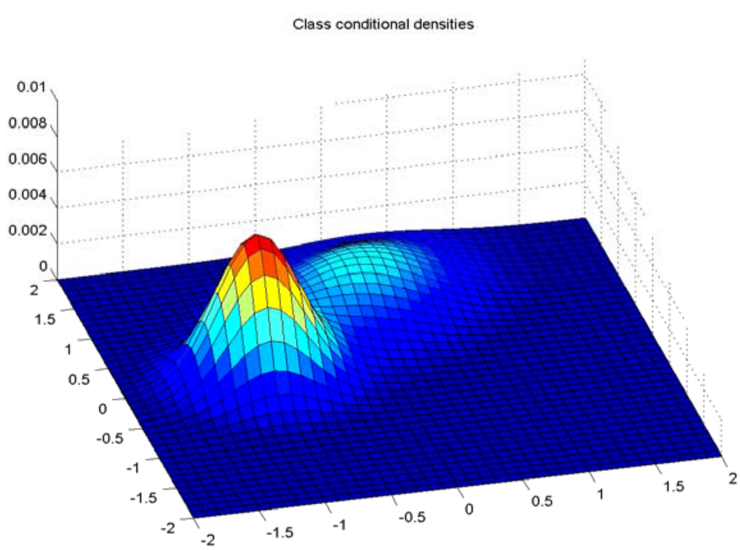
- **Priors on classes (class 0,1)** $\quad y \sim \mathit{Bernoulli}$
  - **Bernoulli distribution**

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

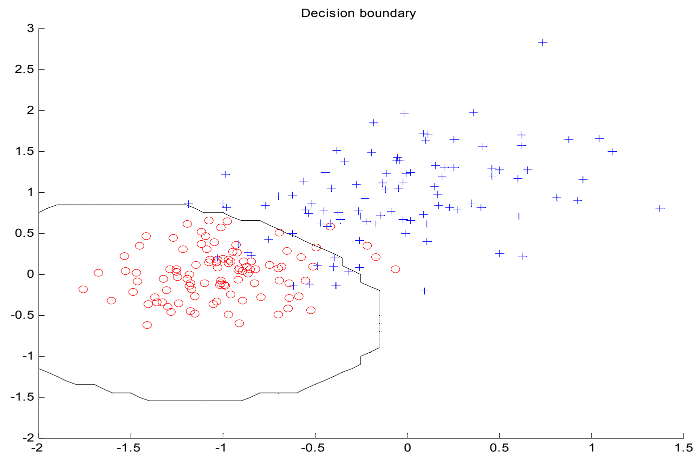
QDA



2 Gaussian class-conditional densities



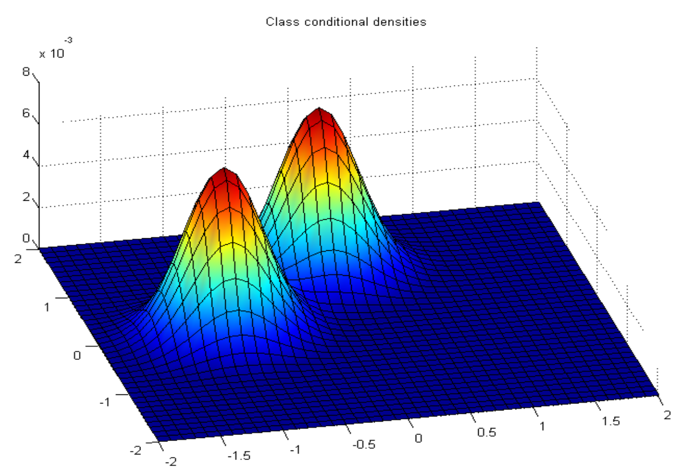
QDA: Quadratic decision boundary



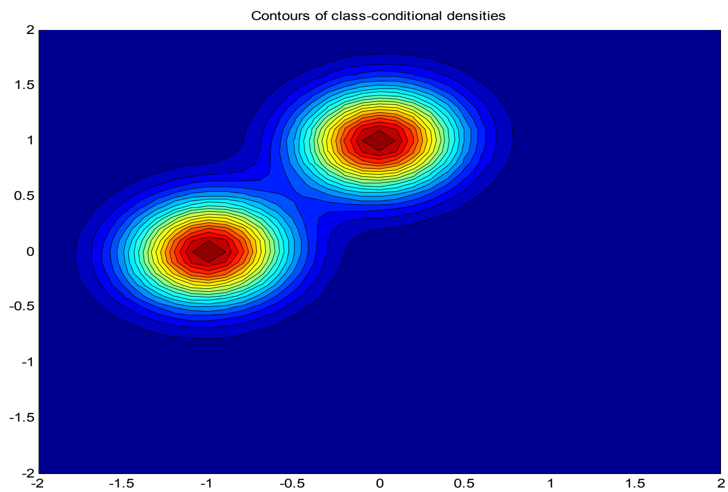
Linear discriminant analysis (LDA)

- 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

Logistic regression vs LDA

- **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 **!!!**

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

When is the logistic regression model correct?

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

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

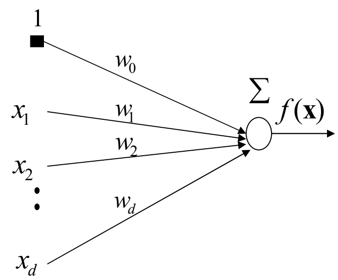
$\boldsymbol{\theta}$ - A location parameter $\quad \boldsymbol{\varphi}$ - A scale parameter

- **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

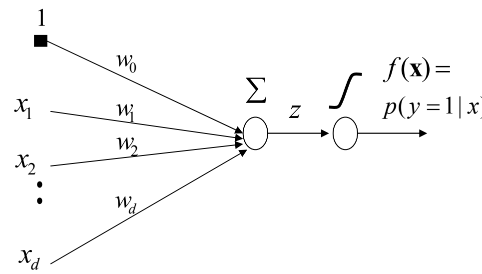
Linear regression

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



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

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**

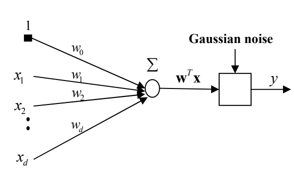
    $$y = \mathbf{w}^T \mathbf{x} + \varepsilon$$

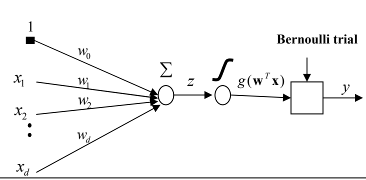
    $$\varepsilon \sim N(0, \sigma^2)$$

  - **Logistic + Bernoulli**

    $$y \sim \text{Bern}(\theta)$$

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





Generalized linear models (GLIM)

Assumptions:

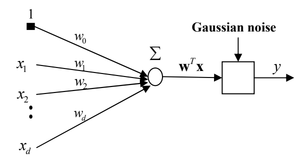
- The conditional mean (expectation) is: $\mu = f(\mathbf{w}^T \mathbf{x})$
  - $f(.)$ is a **response (or a link) function**
- Output y is characterized by an exponential family distribution with mean $\mu = f(\mathbf{w}^T \mathbf{x})$

Examples:

- **Linear model + Gaussian noise**

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

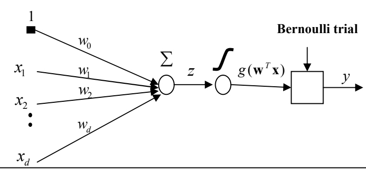
  $y \sim N(\mathbf{w}^T \mathbf{x}, \sigma^2)$



- **Logistic + Bernoulli**

  $y \sim \text{Bern}(\theta) \sim \text{Bern}(\text{g}(\mathbf{w}^{\text{T}} \mathbf{x}))$

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



Generalized linear models (GLMs)

- **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**

Non-linear extension of logistic regression

- use **feature (basis) functions** to model **nonlinearities**
  - the same trick as used for the linear regression

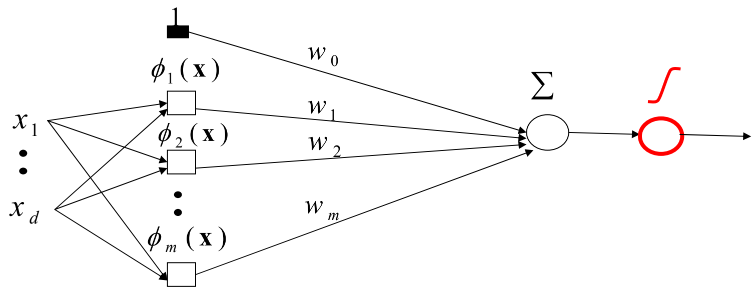
Linear regression

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

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 $\mathbf{x}$



Regularization

Similarly to the linear regression we can penalize the logistic regression or other GLM models for their complexity

- **L1 (lasso) regularization penalty**
- **L2 (ridge) regularization penalty**

- Typically: the optimization of weights **w** looks as follows

$$\min_{\mathbf{w}} \quad \underbrace{Loss(D, \mathbf{w})}_{\text{fit}} + \underbrace{Q(\mathbf{w})}_{\text{Complexity penalty}}$$

- $Loss(D, \mathbf{w})$ functions:
  - Mean squared error
  - Negative log-likelihood
- Regularization penalty $Q(\mathbf{w})$: L1, L2 or a combination