CS 1675 Machine Learning
Lecture 11

Linear models for classification

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

Supervised learning

Data: $D = \{d_1, d_2, .., d_n\}$ **a set of *n* examples**

$d_i = < \mathbf{x}_i, y_i >$

$\mathbf{x}_i$ is input vector, and *y* is desired output (given by a teacher)

Objective: learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(x_i)$ for all $i = 1, .., n$

Two types of problems:

- **Regression:** X discrete or continuous $\rightarrow$ Y is **continuous**
- **Classification:** X discrete or continuous $\rightarrow$ Y is **discrete**

Last lecture

Supervised learning examples

- **Classification:** Y is **discrete**

```
# ######
     ##
    ##
  ######
     ###
#     ##
# ##  ##
  #######
```

→ **Label "3"**

Handwritten digit (array of 0,1s)

Data:

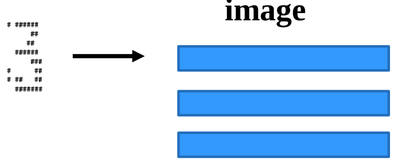


	image	**digit**
		3
		7
		5

….

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** $f : X \to Y$

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

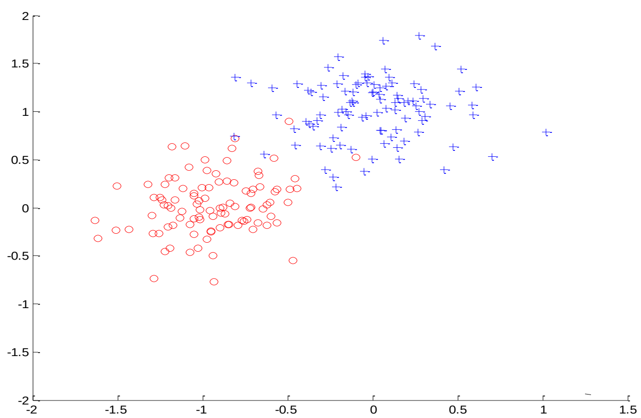
- **First step:**
  - we need to devise a model of the function f

Discriminant functions

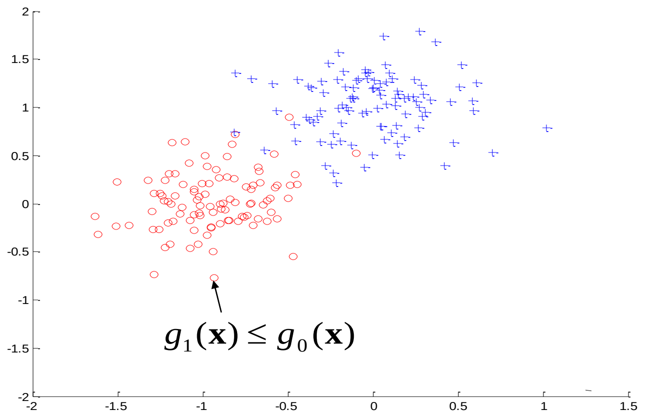
- A common way to represent a **classifier is by using**
  - **Discriminant functions**
- **Works for both the binary and multi-way classification**
- **Idea:**
  - For every class $i = 0,1, \ldots k$ define a function $g_i(\mathbf{x})$ mapping $X \rightarrow \Re$
  - When the decision on input **x**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

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

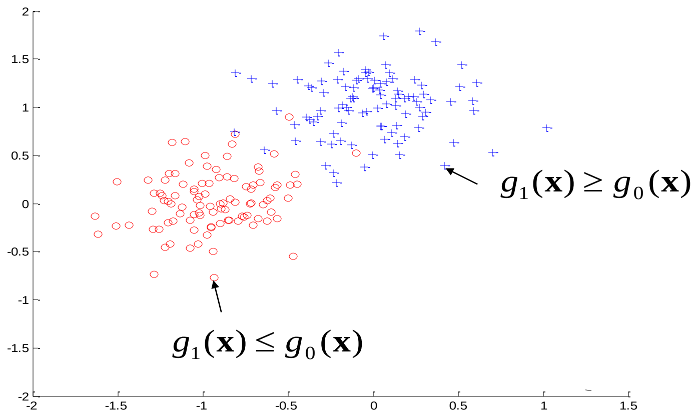
Discriminant functions



Discriminant functions

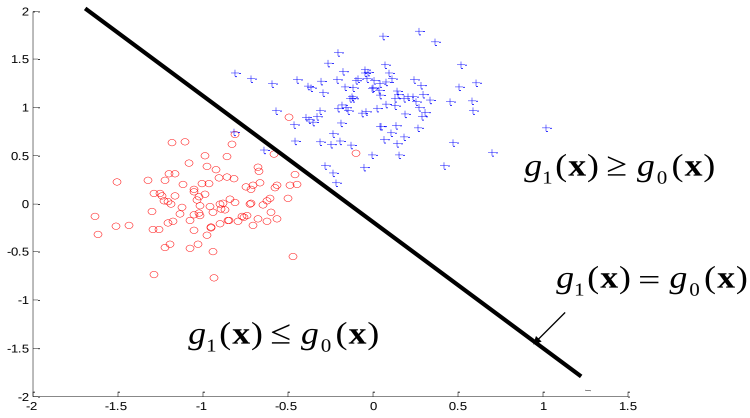


Discriminant functions

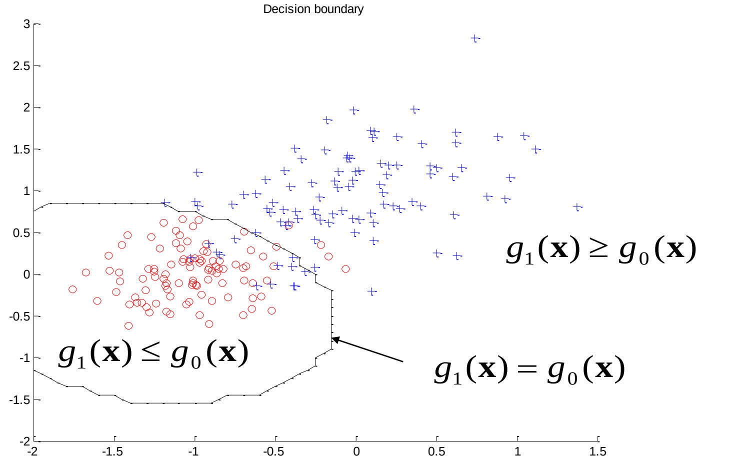


Discriminant functions

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



Quadratic decision boundary



How to design discriminant functions?

- Assume two linear models for classes 0, 1
  $g_1(\mathbf{x}) = \mathbf{w}_1{}^T \mathbf{x}$ $\quad g_0(\mathbf{x}) = \mathbf{w}_0{}^T \mathbf{x}$
- Class decision: $y^* = \arg\max_i \; g_i(\mathbf{x})$
- Training via regression:
  - if (x, 1)
    - Train $g_1(\mathbf{x}) = \mathbf{w}_1{}^T \mathbf{x}$ with y value 1
    - Train $g_0(\mathbf{x}) = \mathbf{w}_0{}^T \mathbf{x}$ with y value 0
  - if (x, 0)
    - Train $g_1(\mathbf{x}) = \mathbf{w}_1{}^T \mathbf{x}$ with y value 0
    - Train $g_0(\mathbf{x}) = \mathbf{w}_0{}^T \mathbf{x}$ with y value 1
- Use least squares error to find both
  $g_1(\mathbf{x}) = \mathbf{w}_1{}^T \mathbf{x}$ $\quad g_0(\mathbf{x}) = \mathbf{w}_0{}^T \mathbf{x}$

How to design discriminant functions?

- Previous design used two discriminant functions one for each class $g_1(\mathbf{x}) = \mathbf{w}_1{}^T \mathbf{x}$ $\quad g_0(\mathbf{x}) = \mathbf{w}_0{}^T \mathbf{x}$
- Binary classification is simpler:
  - We can use just one set of weights w
    $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ $\quad g_0(\mathbf{x}) = -\mathbf{w}^T \mathbf{x} \quad = -g_1(\mathbf{x})$
- Training via regression:
  $y^* = \arg\max_i \; g_i(\mathbf{x})$
  - if (x, 1)
    - Train $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ with y value 1
  - if (x, 0)
    - Train $g_1(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ with y value -1
- How to make a class decision?

How to design discriminant functions?

- Previous design used two discriminant functions one for each class
- Binary classification is simpler – only two classes:
  - We can use one set of shared weights w

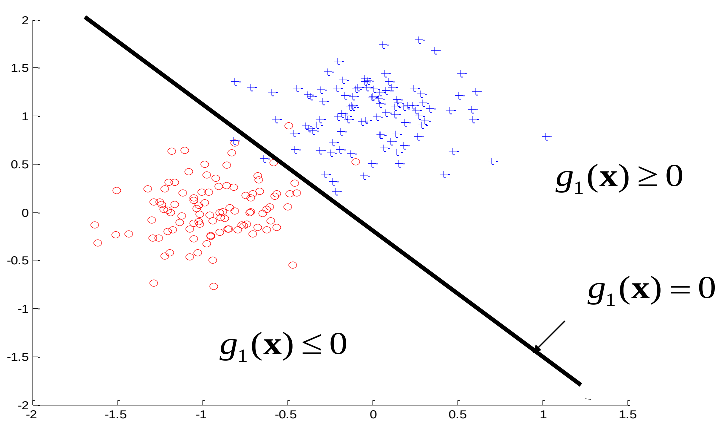
    $g_1(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ $\qquad g_0(\mathbf{x}) = -\mathbf{w}^T\mathbf{x} \quad = -g_1(\mathbf{x})$
- Training via regression:
  - if (x, 1)
    - Train $g_1(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with y value 1
  - if (x, 0)
    - Train $g_1(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with y value -1
- How to make a class decision?

  $g_1(\mathbf{x}) = \mathbf{w}^T\mathbf{x} > 0$ Class 1

  $g_1(\mathbf{x}) = \mathbf{w}^T\mathbf{x} < 0$ Class 0

Discriminant functions and decision boundary

- Linear decision boundary



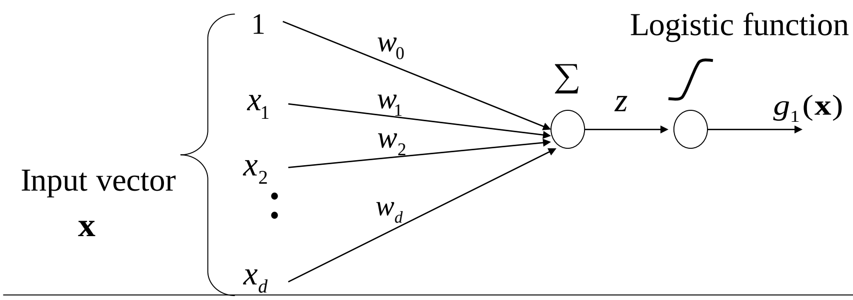
Logistic regression model

- **Discriminant functions:**

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

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

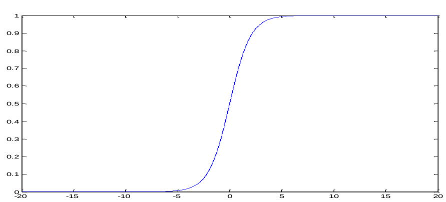
$$g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x}) = p(y=1 \mid \mathbf{x})$$



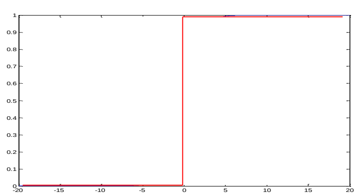
Logistic function

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

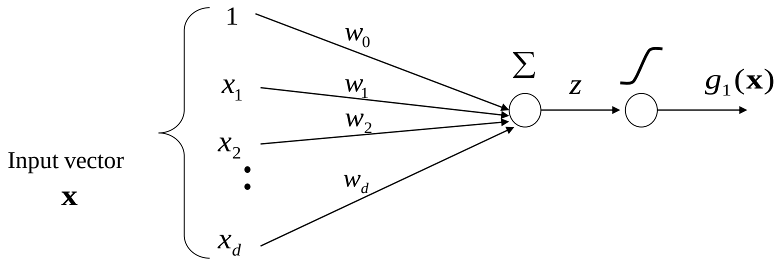
- **Discriminant functions:**

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

- **Values of discriminant functions vary in interval [0,1]**
  - **Probabilistic interpretation**

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

$$g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x}) = p(y=0 \mid \mathbf{x})$$



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

- 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:

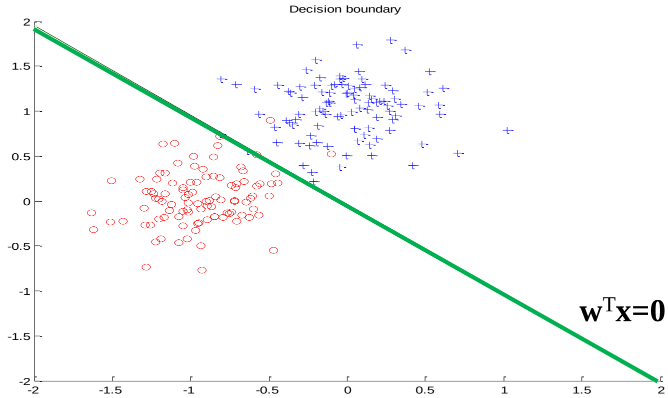
$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^T\mathbf{x})}{g(\mathbf{w}^T\mathbf{x})} = 0$$

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

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**

$$D_i = < \mathbf{x}_i, y_i > \quad \mu_i = p(y_i = 1 | \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x}_i)$$

- **Then**

$$L(D, \mathbf{w}) = \prod_{i=1}^{n} P(y = y_i | \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1 - y_i}$$

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

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

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

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**

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

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

Derivative of a logistic function

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

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

$$= y_i(1-g(z_i)) + (1-y_i)(-g(z_i)) = y_i - g(z_i)$$

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

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

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

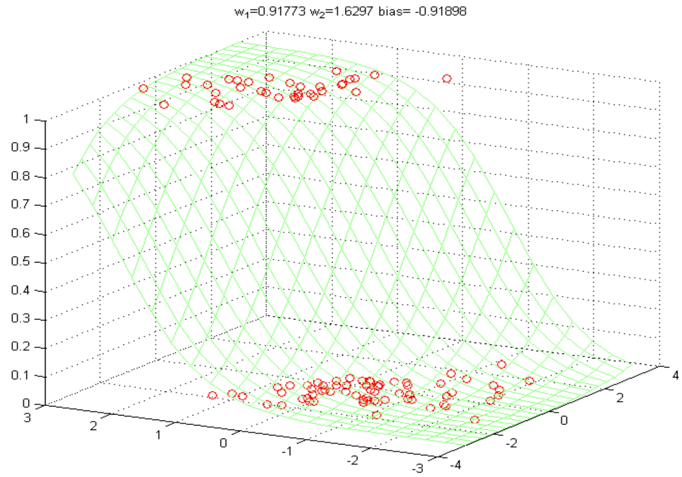
- **ith update for the logistic regression** and $D_k = < \mathbf{x}_k, y_k >$

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

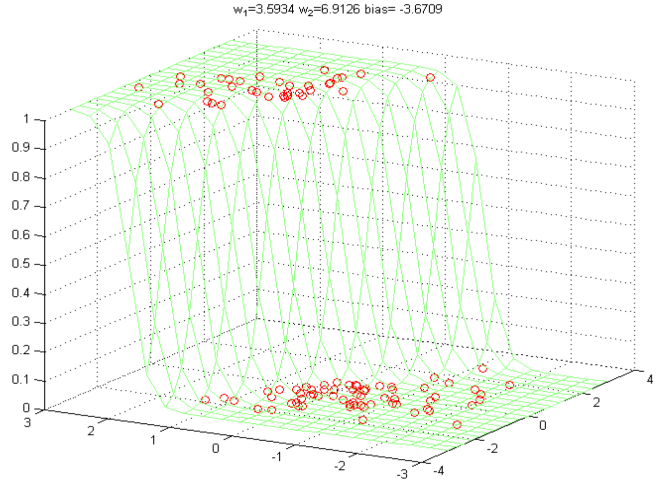
Online logistic regression algorithm

Online-logistic-regression (*stopping_criterion*)
  initialize weights $\mathbf{w} = (w_0, w_1, w_2 \dots w_d)$
  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 $\mathbf{w}$

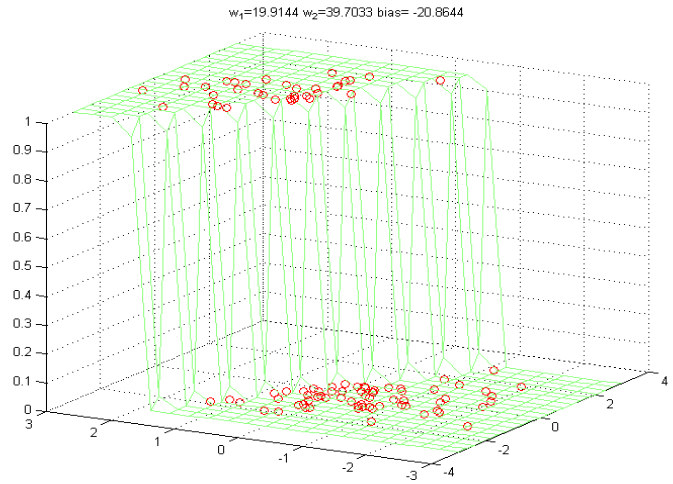
Online algorithm. Example.



Online algorithm. Example.



Online algorithm. Example.



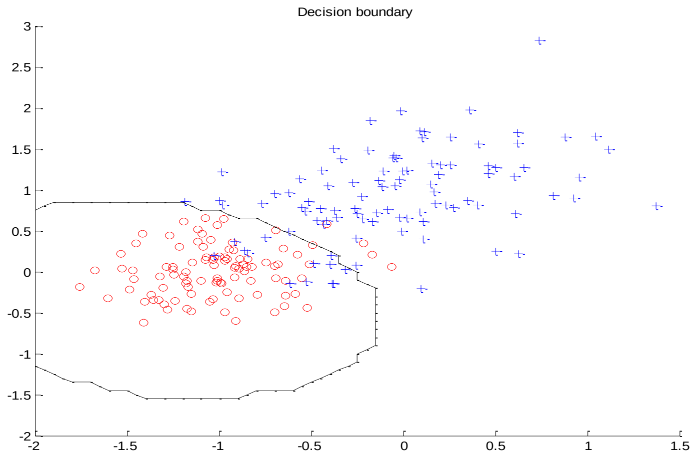
Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**

  Example: 2 classes (blue and red points)

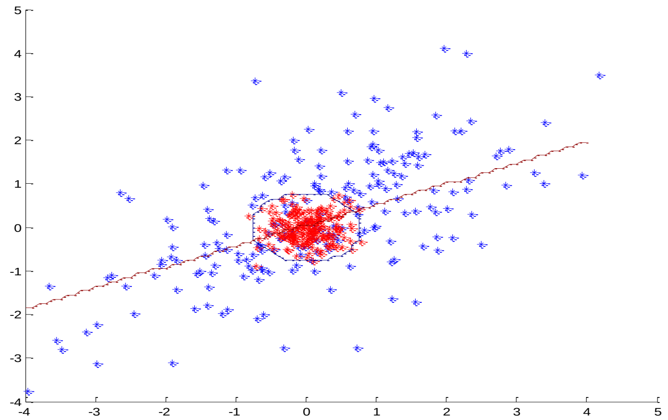
When does the logistic regression fail?

- 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

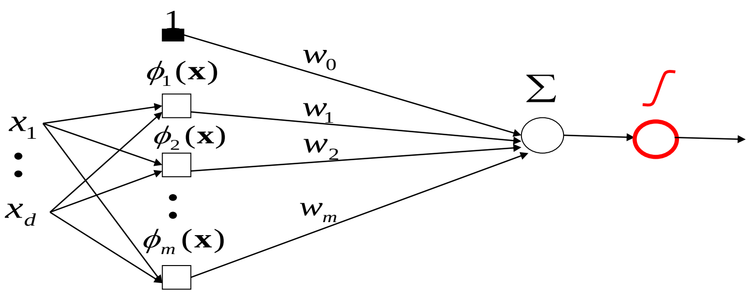
Linear regression

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

Logistic regression

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

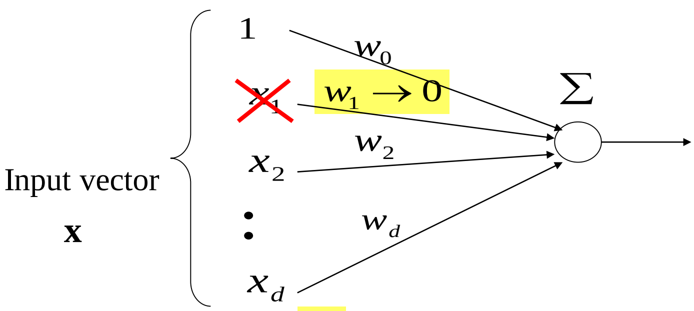
$\phi_j(\mathbf{x})$ - an arbitrary function of $\mathbf{x}$



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**
- **Recall the linear model:**

$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots w_d x_d = \mathbf{w}^T \mathbf{x}$$

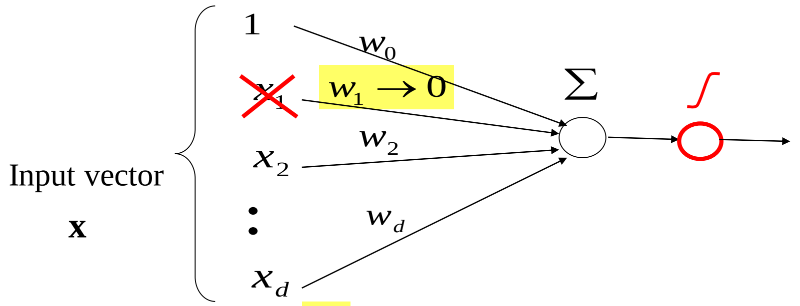


$$f(\mathbf{x}) = w_0 x_0 + 0 x_1 + w_2 x_2 + w_3 x_3 + \ldots w_d x_d = \mathbf{w}^T \mathbf{x}$$

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:**

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



$$p(y=1|\mathbf{x}) = g(w_0 x_0 + 0 x_1 + w_2 x_2 + w_3 x_3 + \dots w_d x_d) = g(\mathbf{w}^T\mathbf{x})$$

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**

$$\|\mathbf{w}\|_{L2}^2 = \sum_{i=0}^{d} w_i^2 = \mathbf{w}^T\mathbf{w} \qquad \text{and} \qquad \lambda \geq 0$$

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:

$$J_n(\mathbf{w}) = -\log P(D \mid \mathbf{w}) + \lambda\|\mathbf{w}\|_{L1}$$

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** $p(y \mid \mathbf{x})$
- **An example of a discriminative classification approach**
- **Model is <u>unable</u> to sample (generate) data instances** $(\mathbf{x}, y)$

Generative approach:

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

How?

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

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

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

Generative approach to classification

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

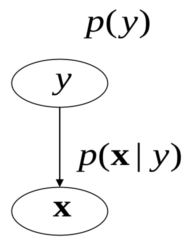
- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  binary classification: two class-conditional distributions

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

- $p(y)$ = **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**

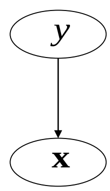
$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad \text{for} \quad y = 0$$
$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{for} \quad y = 1$$

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**

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



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 $\mathbf{x}$

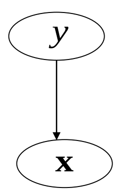
  Optimize log-likelihood: $l(D, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$

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

- How about **class priors**?

Learning Quadratic discriminant analysis (QDA)

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



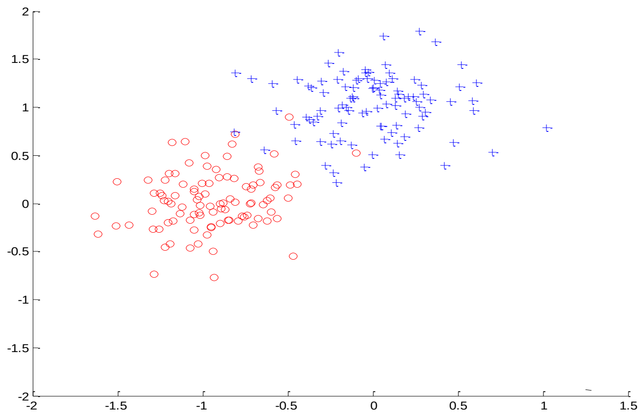
$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad \text{for} \quad y = 0$$
$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{for} \quad y = 1$$

  - **Use the density estimation methods**

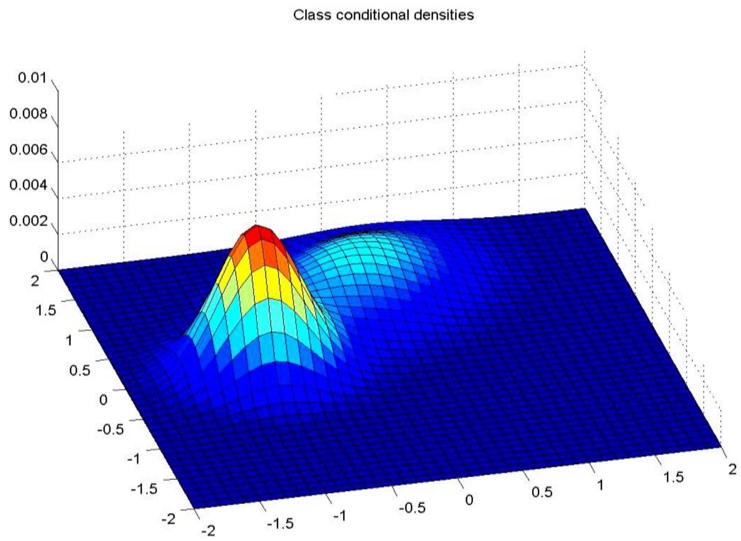
- **Learning Priors on classes (class 0,1)** $y \sim Bernoulli$
  - **Learn the parameter of the Bernoulli distribution**
  - **Again use the density estimation methods**

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

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

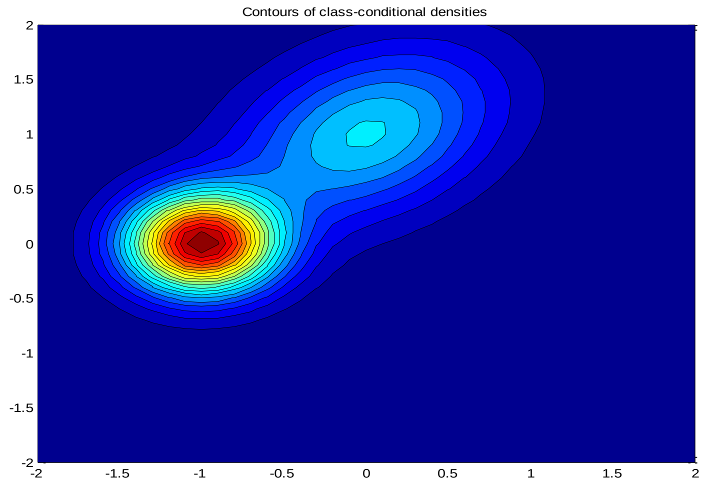
$$\underbrace{p(y=1\,|\,\mathbf{x})}_{g_1(\mathbf{x})} > \underbrace{p(y=0\,|\,\mathbf{x})}_{g_0(\mathbf{x})} \quad \Rightarrow \quad \text{then } y=1 \text{ else } y=0$$

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

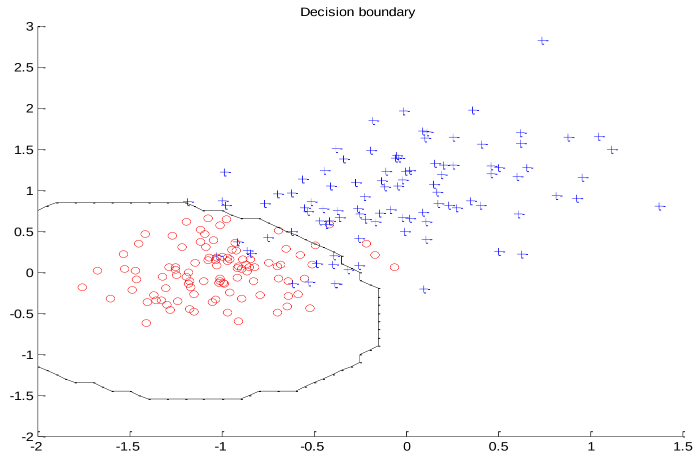
- **Notice it is sufficient to compare:**

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

QDA: Quadratic decision boundary



QDA: Quadratic decision boundary



Linear discriminant analysis (LDA)

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

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