Classification learning:

- Logistic regression
- Generative classification model

Classification

- **Data:** \( D = \{d_1, d_2, \ldots, d_n\} \)
  \[ d_i = \langle x_i, y_i \rangle \]
  - \( y_i \) 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 $g_i(x)$ mapping $X \rightarrow \mathbb{R}$
  - When the decision on input $x$ should be made choose the class with the highest value of $g_i(x)$

$$y^* = \arg \max_i g_i(x)$$

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

\[ g_1(x) \leq g_0(x) \]

Discriminant functions

\[ g_1(x) \geq g_0(x) \]

CS 2750 Machine Learning
Discriminant functions

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

\[
g_1(x) \geq g_0(x) \\
g_1(x) = g_0(x) \\
g_1(x) \leq g_0(x)
\]

Quadratic decision boundary

\[
g_1(x) \geq g_0(x) \\
g_1(x) = g_0(x) \\
g_1(x) \leq g_0(x)
\]
Logistic regression model

- Defines a linear decision boundary
- Discriminant functions:
  \[ g_1(x) = g(w^T x) \quad g_0(x) = 1 - g(w^T x) \]
- where \( g(z) = \frac{1}{1 + e^{-z}} \) - is a logistic function

\[ f(x, w) = g_1(w^T x) = g(w^T x) \]

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 regression model

- **Discriminant functions:**
  \[ g_1(x) = g(w^T x) \quad g_0(x) = 1 - g(w^T x) \]

- **Values of discriminant functions vary in interval \([0,1]\)**
  - **Probabilistic interpretation**
    \[ f(x, w) = p(y = 1 \mid w, x) = g_1(x) = g(w^T x) \]

Logistic regression

- We learn a probabilistic function
  \[ f : X \rightarrow [0,1] \]
  - where \( f \) describes the probability of class 1 given \( x \)
  \[ f(x, w) = g_1(w^T x) = p(y = 1 \mid x, w) \]

**Note that:**
\[ p(y = 0 \mid x, w) = 1 - p(y = 1 \mid x, w) \]

- 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 \( x \)
  
  \[ f(x, w) = g_1(w^T x) = p(y = 1 \mid x, w) \]

Note that:

\[ p(y = 0 \mid x, w) = 1 - p(y = 1 \mid x, w) \]

• Making decisions with the logistic regression model:

  If \( p(y = 1 \mid x) \geq 1/2 \) 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: \( g_1(x) = g_0(x) \)
• For the boundary it must hold:

  \[
  \log \frac{g_0(x)}{g_1(x)} = \log \frac{1 - g(w^T x)}{g(w^T x)} = 0
  \]

  \[
  \log \frac{g_0(x)}{g_1(x)} = \log \frac{\exp-(w^T x)}{1 + \exp-(w^T x)} = \log \exp-(w^T x) = w^T x = 0
  \]
Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**
  
  **Example:** 2 classes (blue and red points)

![Decision boundary image]

Logistic regression: parameter learning

**Likelihood of outputs**

- Let
  
  \[ D_i = \langle x_i, y_i \rangle \quad \mu_i = p(y_i = 1 | x_i, w) = g(z_i) = g(w^T x) \]

- Then
  
  \[ L(D, w) = \prod_{i=1}^{n} P(y = y_i | x_i, w) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i} \]

- **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

  \[ l(D, w) = \log \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i} = \sum_{i=1}^{n} \log \mu_i^{y_i} (1 - \mu_i)^{1-y_i} = \]
  
  \[ = \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i) \]
Logistic regression: parameter learning

- **Notation:**
  \[ \mu_i = p(y_i = 1 | x_i, w) = g(z_i) = g(w^T x) \]

- **Log likelihood**
  \[ l(D, w) = \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i) \]

- **Derivatives of the loglikelihood**
  \[- \frac{\partial}{\partial w_j} l(D, w) = \sum_{i=1}^{n} -x_{i,j} (y_i - g(z_i)) \]
  \[ \nabla_w l(D, w) = \sum_{i=1}^{n} -x_i (y_i - g(w^T x_i)) = \sum_{i=1}^{n} x_i (y_i - f(w, x_i)) \]

- **Gradient descent:**
  \[ w^{(k)} \leftarrow w^{(k-1)} - \alpha (k) \nabla_w [ -l(D, w) ] \mid_{w^{(k-1)}} \]
  \[ w^{(k)} \leftarrow w^{(k-1)} + \alpha (k) \sum_{i=1}^{n} [y_i - f( w^{(k-1)}, x_i ) ] x_i \]

---

Derivation of the gradient

- **Log likelihood**
  \[ l(D, w) = \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i) \]

- **Derivatives of the loglikelihood**
  \[ \frac{\partial}{\partial w_j} l(D, w) = \sum_{i=1}^{n} \frac{\partial}{\partial z_i} [ y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i)] \frac{\partial z_i}{\partial w_j} \]

  **Derivative of a logistic function**
  \[ \frac{\partial g(z_i)}{\partial z_i} = g(z_i)(1 - g(z_i)) \]

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

  \[ = y_i (1 - g(z_i)) + (1 - y_i) (-g(z_i)) = y_i - g(z_i) \]

  \[ \nabla_w l(D, w) = \sum_{i=1}^{n} x_i (y_i - g(w^T x_i)) = \sum_{i=1}^{n} x_i (y_i - f(w, x_i)) \]
Logistic regression. Online gradient descent

- **On-line component of the loglikelihood**
  
  \[ J_{\text{online}}(D_i, w) = y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i) \]

- **On-line learning update for weight** \( w \)
  
  \[ w^{(k)} \leftarrow w^{(k-1)} - \alpha(k) \nabla_w [J_{\text{online}}(D_k, w)] \big|_{w^{(k-1)}} \]

- **ith update for the logistic regression** and \( D_k = \langle x_k, y_k \rangle \)
  
  \[ w^{(i)} \leftarrow w^{(k-1)} + \alpha(k) [y_i - f(w^{(k-1)}, x_k)]x_k \]

---

**Online logistic regression algorithm**

**Online-logistic-regression** \((D, \text{number of iterations})\)

**initialize** weights \( w = (w_0, w_1, w_2 \ldots w_d) \)

**for** \( i=1:1; \text{number of iterations} \)

**do**

- **select** a data point \( D_i = \langle x_i, y_i \rangle \) from \( D \)
- **set** \( \alpha = 1/i \)

**update** weights (in parallel)

\[ w \leftarrow w + \alpha(i) [y_i - f(w, x_i)]x_i \]

**end for**

**return** weights \( w \)
Online algorithm. Example.

\[ w_0 = 0.9177, \ w_1 = 6.297, \ \text{bias} = 0.9188 \]

Online algorithm. Example.

\[ w_0 = 3.9934, \ w_1 = 6.9135, \ \text{bias} = 3.6799 \]
Online algorithm. Example.

Generative approach to classification

Logistic regression: model \( p(y \mid x) \)

Generative approach:

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

E.g. \( g_0(x) = p(y = 0 \mid x) \quad g_1(x) = p(y = 1 \mid x) \)

Typical model \( p(x, y) = p(x \mid y) p(y) \)

- \( p(x \mid y) = \textbf{Class-conditional distributions (densities)} \)
  - binary classification: two class-conditional distributions
    \( p(x \mid y = 0) \quad p(x \mid y = 1) \)
- \( p(y) = \textbf{Priors on classes} - \text{probability of class } y \)
  - binary classification: Bernoulli distribution
    \( p(y = 0) + p(y = 1) = 1 \)
Quadratic discriminant analysis (QDA)

Model:
- Class-conditional distributions
  - Multivariate normal distributions
    \[ x \sim N(\mu_0, \Sigma_0) \quad \text{for} \quad y = 0 \]
    \[ x \sim N(\mu_1, \Sigma_1) \quad \text{for} \quad y = 1 \]
- Priors on classes (class 0,1)
  - 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)
  \[ p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right] \]
- ML estimate of parameters of a multivariate normal \( N(\mu, \Sigma) \) for a set of \( n \) examples of \( x \)
  Optimize log-likelihood: \( l(D, \mu, \Sigma) = \log \prod_{i=1}^{n} p(x_i \mid \mu, \Sigma) \)
  \[ \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \]
  \[ \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^T \]
- How about class priors?
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 \( x \) better (likelihood of the data)

\[
p(x \mid \mu_1, \Sigma_1) > p(x \mid \mu_0, \Sigma_0) \quad \text{then } y=1\quad \text{else } y=0
\]

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

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

\[
p(y = 1 \mid x) = \frac{p(x \mid \mu_1, \Sigma_1) p(y = 1)}{p(x \mid \mu_0, \Sigma_0) p(y = 0) + p(x \mid \mu_1, \Sigma_1) p(y = 1)}
\]
QDA: Quadratic decision boundary

Linear discriminant analysis (LDA)
- When covariances are the same
  \[ x \sim N(\mu_0, \Sigma), \ y = 0 \]
  \[ x \sim N(\mu_1, \Sigma), \ y = 1 \]
LDA: Linear decision boundary

Contours of class-conditional densities

LDA: linear decision boundary

Decision boundary