Logistic regression

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
  \[
  Error_1(x_i, y_i) = \begin{cases} 
  1 & f(x_i, w) \neq y_i \\
  0 & f(x_i, w) = y_i 
  \end{cases}
  \]
• Error we would like to minimize: \( E_{(x,y)}(Error_1(x, y)) \)
• First step: we need to devise a model of the function
Discriminant functions

• One convenient way to represent classifiers is through
  – Discriminant functions

• Works for 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 \( g_i(x) \)

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

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

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

- Define **decision boundary**.

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

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

Quadratic decision boundary
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 \( g(z) = 1/(1 + e^{-z}) \) is a logistic function

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

![Logistic regression model diagram](image)

Logistic function

function

\[ g(z) = \frac{1}{1 + e^{-z}} \]

- 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 function graph](image)
Logistic regression model

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

\[ \sum z \int p(y=1 \mid x, w) \]

Input vector \( x \)

\[ \begin{align*}
  1 & \quad \begin{cases} w_0 \\ w_1 \\ w_2 \end{cases} \\
  x_1 & \quad \begin{cases} x_1 \\ x_2 \end{cases} \\
  x_2 & \quad \begin{cases} \cdot \\ \cdot \end{cases} \\
  x_d & \quad \begin{cases} \cdot \\ \cdot \end{cases}
\end{align*} \]

Logistic regression

- Instead of learning the mapping to discrete values 0,1
  \[ f : X \rightarrow \{0,1\} \]
- we learn a **probabilistic function**
  \[ f : X \rightarrow [0,1] \]
  - where \( f \) describes the probability of class 1 given \( x \)
    \[ f(x, w) = p(y = 1 \mid x, w) \]

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

- Transformation to discrete class values:
  - 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:

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

\[
\log \frac{g_\alpha(x)}{g_1(x)} = \log \frac{\exp(-w^T x)}{1 + \exp(-w^T x)} = \log \exp(-(w^T x)) = w^T x = 0
\]

Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**
  **Example:** 2 classes (blue and red points)
Logistic regression: parameter learning.

Likelihood of outputs
- Let
  \[ D_i = \langle \mathbf{x}_i, y_i \rangle \]
  \[ \mu_i = p(y_i = 1 | \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x}) \]
- Then
  \[ L(D, \mathbf{w}) = \prod_{i=1}^{n} P(y = y_i | \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i} \]
- Find weights \( \mathbf{w} \) 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

- 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
  \[ \frac{\partial}{\partial \mathbf{w}_j} l(D, \mathbf{w}) = \sum_{i=1}^{n} -x_{i,j} (y_i - g(z_i)) \]
  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:
  \[ \mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [-l(D, \mathbf{w})] \bigg|_{\mathbf{w}^{(k-1)}} \]
  \[ \mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} + \alpha(k) \sum_{i=1}^{n} [y_i - f(\mathbf{w}^{(k-1)}, \mathbf{x}_i)] \mathbf{x}_i \]
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** \( w \)
  \[ J_{\text{online}} (D_k, w) \]
  \[ w^{(k)} \leftarrow w^{(k-1)} - \alpha(k) \nabla_w [J_{\text{online}} (D_k, w)] \bigg|_{w^{(k-1)}} \]

- **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**

```python
Online-logistic-regression (D, number of iterations)
initialize weights \( w = (w_0, w_1, w_2 \ldots w_d) \)
for i = 1: 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.

CS 2750 Machine Learning
Online algorithm. Example.

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

  **Derivative of a logistic function**
  \[
  \frac{\partial}{\partial z_i} \left[ y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i) \right] = y_i \frac{1}{g(z_i)} \frac{\partial g(z_i)}{\partial z_i} + (1 - y_i) \frac{-1}{1 - g(z_i)} \frac{\partial g(z_i)}{\partial z_i} 
  \]
  \[
  = y_i (1 - g(z_i)) + (1 - y_i) (-g(z_i)) = y_i - g(z_i) 
  \]

\[
\nabla_w l(D, w) = \sum_{i=1}^{n} -x_i (y_i - g(w^T x_i)) = \sum_{i=1}^{n} -x_i (y_i - f(w, x_i)) 
\]
Generative approach to classification

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

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

Typical model $p(x, y) = p(x \mid y) p(y)$
- $p(x \mid y) = \text{Class-conditional distributions (densities)}$
  - Binary classification: two class-conditional distributions
    $p(x \mid y = 0)$ $p(x \mid y = 1)$
- $p(y) = \text{Priors on classes} - \text{probability of class } y$
  - Binary classification: Bernoulli distribution
    $p(y = 0) + p(y = 1) = 1$

Example:
- Class-conditional distributions
  - Multivariate normal distributions
    $x \sim N(\mu_0, \Sigma_0)$ for $y = 0$
    $x \sim N(\mu_1, \Sigma_1)$ for $y = 1$
  - Multivariate normal $x \sim N(\mu, \Sigma)$
    $p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right]$
- Priors on classes (class 0,1) $y \sim \text{Bernoulli}$
  - Bernoulli distribution
    $p(y, \theta) = \theta^y (1 - \theta)^{1-y} \quad y \in \{0,1\}$
2 Gaussian class-conditional densities

2 Gaussians: Quadratic decision boundary
Learning of parameters of the model

Density estimation in statistics
• We see examples – we do not know the parameters of Gaussians (class-conditional densities)
\[ p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right] \]

• **ML estimate of parameters** of a multivariate normal \( N(\mu, \Sigma) \) for a set of \( n \) examples of \( x \)
  - Optimize log-likelihood: \( l(D, \mu, \Sigma) = \log \prod_{i=1}^{n} p(x_i | \mu, \Sigma) \)
  \[
  \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i, \quad \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^T
  \]
• How about **class priors**?

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)
  \[
  \frac{p(x | \mu_1, \Sigma_1)}{g_1(x)} > \frac{p(x | \mu_0, \Sigma_0)}{g_0(x)} \quad \text{then} \quad y=1, \quad \text{else} \quad y=0
  \]
• **Posterior of a class** – choose the class with better posterior probability
  \[
  p(y = 1 | x) > p(y = 0 | x) \quad \text{then} \quad y=1, \quad \text{else} \quad y=0
  \]
  \[
  p(y = 1 | x) = \frac{p(x | \mu_1, \Sigma_1) p(y = 1)}{p(x | \mu_0, \Sigma_0) p(y = 0) + p(x | \mu_1, \Sigma_1) p(y = 1)}
  \]