Binary classification

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

Supervised learning

Data: \( D = \{D_1, D_2, \ldots, D_n\} \) \text{ a set of } n \text{ examples} 
\( D_i = \langle x_i, y_i \rangle \)
\( x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \) is an input vector of size \( d \)
\( y_i \) is the desired output (given by a teacher)

Objective: learn the mapping \( f : X \rightarrow Y \)
s.t. \( y_i \approx f(x_i) \) for all \( i = 1, \ldots, n \)

- **Regression:** \( Y \) is **continuous**
  Example: earnings, product orders \( \rightarrow \) company stock price
- **Classification:** \( Y \) is **discrete**
  Example: handwritten digit in binary form \( \rightarrow \) digit label
Linear regression: review

- **Function** \( f : X \rightarrow Y \) is a linear combination of input components

\[
f(x) = w_0 + w_1x_1 + w_2x_2 + \ldots w_dx_d = w_0 + \sum_{j=1}^{d} w_jx_j
\]

- **Bias term** \( w_0 \)
- **Input vector** \( x = [x_1, x_2, \ldots, x_d] \)
- **Error function** measures how much our predictions deviate from the desired answers

\[
J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
\]

- **Learning**
  
  We want to find the weights minimizing the error!
Solving linear regression: review

• The optimal set of weights satisfies:

\[ \nabla_w (J_n(w)) = - \frac{2}{n} \sum_{i=1}^{n} (y_i - w^T x_i) x_i = 0 \]

Leads to a system of linear equations (SLE) with \( d+1 \) unknowns of the form

\[ A w = b \]

\[ w_0 \sum_{i=1}^{n} x_{i,0} x_{i,j} + w_1 \sum_{i=1}^{n} x_{i,1} x_{i,j} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,j} = \sum_{i=1}^{n} y_i x_{i,j} \]

Solutions to SLE:
• e.g. matrix inversion (if the matrix is singular)

\[ w = A^{-1} b \]

---

Linear regression. Example

• 1 dimensional input \( x = (x_i) \)
Linear regression. Example.

• 2 dimensional input \( \mathbf{x} = (x_1, x_2) \)

---

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 \( f : X \rightarrow \{0, 1\} \)
• **Misclassification error function**
  \[
  \text{Error}_1(\mathbf{x}_i, y_i) = \begin{cases} 
  1 & f(\mathbf{x}_i, \mathbf{w}) \neq y_i \\
  0 & f(\mathbf{x}_i, \mathbf{w}) = y_i 
\end{cases}
  \]
• Error we would like to minimize: \( E_{(x,y)}(\text{Error}_1(\mathbf{x}, y)) \)
• **First step:** we need to devise a model of the function
Discriminant functions

• One 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 $\mathcal{X} \rightarrow \mathbb{R}$
  – 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**

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

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 in the input space
- Discriminant functions:
  
  \[ g_1(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) \quad \text{and} \quad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T \mathbf{x}) \]

  where \( g(z) = \frac{1}{1 + e^{-z}} \) - 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:**
  \[ g_1(x) = g(w^T x) \quad g_0(x) = 1 - g(w^T x) \]

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

![Logistic regression model diagram](image)

Logistic regression

- We learn a probabilistic function
  \[ f : X \to [0,1] \]
  - 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:**
\[ p(y = 0 \mid x, w) = 1 - p(y = 1 \mid x, w) \]

- Transformation to binary class values:
  
  | If \( p(y = 1 \mid x) \geq 1/2 \) | then choose 1 |
  | Else | choose 0 |

CS 2750 Machine Learning
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:

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

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

- Let
  \[ D_i = \langle x_i, y_i \rangle \]
  \[ \mu_i = p(y_i = 1 \mid x_i, w) = g(z_i) = g(w^T x) \]
- Then
  \[ L(D, w) = \prod_{i=1}^{n} P(y = y_i \mid x_i, w) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i} \]
- 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

- 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
  \[ - \frac{\partial}{\partial w_j} l(D, w) = \sum_{i=1}^{n} -x_{i,j} (y_i - g(z_i)) \]
  \[ \nabla_w l(D, w) = \sum_{i=1}^{n} -x_i (y_i - g(w^T x_i)) = \sum_{i=1}^{n} -x_i (y_i - f(w, x_i)) \]
  - Nonlinear in weights!!
- Gradient descent:
  \[ w^{(k)} \leftarrow w^{(k-1)} - \alpha(k) \nabla_w [ -l(D, w) ] \bigg|_{w^{(k-1)}} \]
  \[ w^{(k)} \leftarrow w^{(k-1)} + \alpha(k) \sum_{i=1}^{n} [y_i - f(w^{(k-1)}, x_i)] x_i \]
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** \( w \)
  \[ J_{\text{online}} (D_k, w) \]
  \[ w^{(k)} \leftarrow w^{(k-1)} - \alpha(k) \nabla_w [J_{\text{online}} (D_k, w)] \big|_{w^{(k-1)}} \]

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

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

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

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

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

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

CS 2750 Machine Learning
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
    $p(x \mid y = 0)$, $p(x \mid y = 1)$
  
  - $p(y) = \text{Priors on classes}$ - probability of class $y$
  - binary classification: Bernoulli distribution
    $p(y = 0) + p(y = 1) = 1$

---

Naïve Bayes classifier

- A generative classifier model with an additional simplifying assumption:
  - All input attributes are conditionally independent of each other given the class. So we have:

  $$p(x, y) = p(y) p(x \mid y) = p(y) \prod_{i=1}^{d} p(x_i \mid y)$$
Learning of the Naïve Bayes classifier

Model:
\[ p(x, y) = p(y) p(x | y) = p(y) \prod_{i=1}^{d} p(x_i | y) \]

Learning
- learning of parameters of the BBN:
  - Class prior
    \[ p(y) \]
  - class conditional distributions:
    \[ p(x_i | y) \text{ for all } i = 1, \ldots, d \]

- In practice class conditional distribution can have different models: e.g. one attribute can be modeled using the Bernoulli, the other as Gaussian density, or as a Poisson distribution

Making class decision for the Naïve Bayes

Discriminant functions
- Posterior of a class – choose the class with better posterior probability
  \[ p(y = 1 | x) > p(y = 0 | x) \]
  then \( y = 1 \)
  else \( y = 0 \)
  \[ p(y = 1 | x) = \frac{\left( \prod_{i=1}^{d} p(x_i | y = 1) \right) p(y = 1)}{\left( \prod_{i=1}^{d} p(x_i | y = 0) \right) p(y = 0) + \left( \prod_{i=1}^{d} p(x_i | y = 1) \right) p(y = 1)} \]

- Likelihood of data – choose the class that explains the input data (x) better (likelihood of the data)
  \[ \prod_{i=1}^{d} p(x_i | y = 1) > \prod_{i=1}^{d} p(x_i | y = 0) \]
  then \( y = 1 \)
  else \( y = 0 \)