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

For any data set we use to test the classification model on we can build a confusion matrix:

- Counts of examples with:
  - class label \( \omega_j \) that are classified with a label \( \alpha_i \)

<table>
<thead>
<tr>
<th>predict</th>
<th>target</th>
<th>( \omega = 1 )</th>
<th>( \omega = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha = 1 )</td>
<td>140</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>( \alpha = 0 )</td>
<td>20</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation

For any data set we use to test the model we can build a confusion matrix:

<table>
<thead>
<tr>
<th>predict</th>
<th>target</th>
<th>$\omega = 1$</th>
<th>$\omega = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1$</td>
<td>140</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>$\alpha = 0$</td>
<td>20</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

Error: $\frac{37}{231}$

Accuracy $= 1 - \text{Error} = \frac{194}{231}$
# Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

<table>
<thead>
<tr>
<th>predict</th>
<th>( \omega = 1 )</th>
<th>( \omega = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha = 1 )</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>( \alpha = 0 )</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

- **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**

<table>
<thead>
<tr>
<th>predict</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>140</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>180</td>
</tr>
</tbody>
</table>

\[ SENS = \frac{140}{160} \quad SPEC = \frac{180}{190} \]

\[ PPV = \frac{140}{150} \quad NPV = \frac{180}{200} \]

Row and column quantities:
- Sensitivity (SENS)
- Specificity (SPEC)
- Positive predictive value (PPV)
- Negative predictive value (NPV)

Binary decisions: Receiver Operating Curves

- **Probabilities:**
  - \( SENS \quad p(x > x^* \mid x \in \omega_2) \)
  - \( SPEC \quad p(x < x^* \mid x \in \omega_1) \)
Receiver Operating Characteristic (ROC)

- ROC curve plots:
  \[ SN = p \left( x > x^* \mid x \in \omega_1 \right) \]
  \[ 1-\text{SP} = p \left( x > x^* \mid x \in \omega_2 \right) \]
  for different \( x^* \)

- SENS
  \[ p(x > x^* \mid x \in \omega_2) \]

- 1-SPEC
  \[ p(x > x^* \mid x \in \omega_1) \]

ROC curve

Case 1
Case 2
Case 3
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 $i = 0, 1, …, k$ 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

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

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:
  \[ g_1(x) = g(w^T x) \quad \text{and} \quad g_0(x) = 1 - g(w^T x) \]
- where \( g(z) = 1 / (1 + e^{-z}) \) - is a logistic function
  \[ f(x, w) = g_1(w^T x) = g(w^T x) \]

Logistic function

- 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. 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 \( p(x, y) \)
2. Use it to define probabilistic discriminant functions

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

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

Quadratic discriminant analysis (QDA)

Model:
- \( \text{Class-conditional distributions} \)
  - 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 \( x \sim N(\mu, \Sigma) \)
  \[
p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right]
  \]
- \( \text{Priors on classes (class 0,1)} \)
  - Bernoulli distribution
    \( y \sim \text{Bernoulli} \)
    \[
p(y, \theta) = \theta^y (1 - \theta)^{1-y} \quad y \in \{0,1\}
  \]
QDA

2 Gaussian class-conditional densities
QDA: Quadratic decision boundary

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

Contours of class-conditional densities

CS 2750 Machine Learning

LDA: linear decision boundary

Decision boundary

CS 2750 Machine Learning
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 !!!
    \[
    p(y = 1 \mid x, \mu_0, \mu_1, \Sigma) = g(w^T x)
    \]

When is the logistic regression model correct?

- Members of the exponential family can be often more naturally described as
  \[
  f(x \mid \theta, \phi) = h(x, \phi) \exp \left\{ \frac{\theta^T x - A(\theta)}{a(\phi)} \right\}
  \]
  \( \theta \) - A location parameter \( \phi \) - 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 \( \phi \)

- Very powerful result !!!
  - We can represent posteriors of many distributions with the same small network
Linear units

Linear regression
\[ f(x) = w^T x \]

Logistic regression
\[ f(x) = p(y = 1 \mid x, w) = g(w^T x) \]

Gradient update:
\[ w \leftarrow w + \alpha \sum_{i=1}^{n} (y_i - f(x_i)) x_i \]

The same
\[ w \leftarrow w + \alpha \sum_{i=1}^{n} (y_i - f(x_i)) x_i \]

Online:
\[ w \leftarrow w + \alpha (y - f(x)) x \]

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**
    \[ y = w^T x + \varepsilon \]
    \[ \varepsilon \sim N(0, \sigma^2) \]
  - **Logistic + Bernoulli**
    \[ y \sim \text{Bern}(\theta) \]
    \[ \theta = p(y = 1 \mid x) = g(w^T x) \]
Generalized linear models (GLM)

Assumptions:
• The conditional mean (expectation) is: \( \mu = f(w^T x) \)
  – \( f(.) \) is a response (or a link) function
• Output \( y \) is characterized by an exponential family distribution with mean \( \mu = f(w^T x) \)

Examples:
– Linear model + Gaussian noise
  \[ y = w^T x + \varepsilon \quad \varepsilon \sim N(0, \sigma^2) \]
  \[ y \sim N(w^T x, \sigma^2) \]
– Logistic + Bernoulli
  \[ y \sim \text{Bern}(\theta) \sim \text{Bern}(g(w^T x)) \]
  \[ \theta = g(w^T x) = \frac{1}{1 + e^{-w^T x}} \]

Generalized linear models (GLMs)

• A canonical response functions \( f(.) \):
  – encoded in the distribution
    \[ p(x \mid \theta, \phi) = h(x, \phi) \exp \left\{ \frac{\theta^T x - A(\theta)}{a(\phi)} \right\} \]
• Leads to a simple gradient form
• Example: Bernoulli distribution
  \[ p(x \mid \mu) = \mu^x (1 - \mu)^{1-x} = \exp \left\{ \log \left( \frac{\mu}{1 - \mu} \right) x + \log(1 - \mu) \right\} \]
  \[ \theta = \log \left( \frac{\mu}{1 - \mu} \right) \quad \mu = \frac{1}{1 + e^{-\theta}} \]
  – 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**

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

**Logistic regression**

\[ \hat{f}(x) = g\left(w_0 + \sum_{j=1}^{m} w_j \phi_j(x)\right) \]

\[ \phi_j(x) \quad - \text{an arbitrary function of } x \]

---

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

\[ \min_w \quad \text{Loss} \left(D, w\right) + Q\left(w\right) \]

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