Linear models for classification

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

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 $i = 0, 1, ...k$ 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 $g_i(x)$

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

- LR defines a linear decision boundary

**Example:** 2 classes (blue and red points)

\[
D \leftarrow w^T x = 0
\]

Logistic regression: parameter learning

- **Notation:** 
  \[ \mu_i = p(y_i = 1 | x_i, w) = g(z_i) = g(w^T x_i) \]
- **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**
  \[
  \frac{\partial}{\partial w_j} l(D, w) = \sum_{i=1}^{n} x_{i,j} (y_i - g(z_i))
  \]
  \[
  \nabla_w l(D, w) = \sum_{i=1}^{n} x_i (y_i - f(w, x_i)) = \sum_{i=1}^{n} x_i (y_i - f(w, x_i))
  \]
- **Gradient descent:**
  \[
  w^{(k)} \leftarrow w^{(k-1)} - \alpha(k) \nabla_w [-l(D, w)] \Big|_{w^{(k-1)}}
  \]
  \[
  w^{(k)} \leftarrow w^{(k-1)} + \alpha(k) \sum_{i=1}^{n} [y_i - f(w^{(k-1)}, x_i)] x_i
  \]
Logistic regression. Online gradient descent

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

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

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

---

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

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

\( \phi_j(\mathbf{x}) \) - 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
- We can apply the same idea to the logistic regression:

\[ p(y=1|x) = g(w^T x) \]

\[ w_0, w_1, \ldots, w_k \] - parameters (weights)

\[ p(y=1|x) = g(w_0 x_0 + (w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_d x_d)) = g(w^T x) \]

Ridge (L2) penalty

Linear regression – Ridge penalty:

\[ J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \|w\|_{L2}^2 \]

Fit to data          Model complexity penalty

\[ \|w\|_{L2}^2 = \sum_{i=0}^{d} w_i^2 = w^T w \]

and \( \lambda \geq 0 \)

Logistic regression:

\[ J_n(w) = -\log P(D | w) + \lambda \|w\|_{L2}^2 \]

Fit to data          Model complexity penalty

\[ J_n(w) = -\left[ \sum_{i=1}^{n} y_i \log g(w^T x_i) + (1 - y_i) \log (1 - g(w^T x_i)) \right] + \lambda \|w\|_{L2} \]

Fit to data measured using the negative log likelihood
**Lasso (L1) penalty**

Linear regression – Lasso penalty:

\[
J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \|w\|_{L1}
\]

Fit to data \hspace{2cm} Model complexity penalty

\[\|w\|_{L1} = \sum_{i=0}^{d} |w_i| \quad \text{and} \quad \lambda \geq 0\]

Logistic regression:

\[
J_n(w) = -\log P(D | w) + \lambda \|w\|_{L1}
\]

Fit to data \hspace{2cm} Model complexity penalty

\[J_n(w) = -\left[\sum_{i=1}^{n} y_i \log g(w^T x_i) + (1 - y_i) \log (1 - g(w^T x_i))\right] + \lambda \|w\|_{L1}\]

Fit to data measured using the negative log likelihood

---

**Generative approach to classification**

Logistic regression:

- Represents and learns a model of \(p(y | x)\)
- An example of a **discriminative classification approach**
- Model is **unable** to sample (generate) data instances \((x, y)\)

Generative approach:

- Represents and learns a joint distribution \(p(x, y)\)
- Model is **able** to sample (generate) data instances \((x, y)\)
- The joint model defines probabilistic discriminant functions

**How?**

\[
g_1(x) = p(y = 1 | x) = \frac{p(x, y = 1)}{p(x)} = \frac{p(x | y = 1) p(y = 1)}{p(x)}
\]

\[
g_0(x) = p(y = 0 | x) = \frac{p(x, y = 0)}{p(x)} = \frac{p(x | y = 0) p(y = 0)}{p(x)}
\]

\[p(y = 0 | x) + p(y = 1 | x) = 1\]
Generative approach to classification

**Typical joint 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
    \[ p(x \mid y = 0) \quad p(x \mid y = 1) \]
- \( p(y) = \text{Priors on classes} \)
  - 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
    \[ x \sim N(\mu_0, \Sigma_0) \] 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
    \[ p(y, \theta) = \theta^y (1 - \theta)^{1-y} \quad y \in \{0,1\} \]
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?

Learning Quadratic discriminant analysis (QDA)

- Learning Class-conditional distributions
  - Learn parameters of 2 multivariate normal distributions
    \[ x \sim N(\mu_0, \Sigma_0) \quad \text{for} \quad y = 0 \]
    \[ x \sim N(\mu_1, \Sigma_1) \quad \text{for} \quad y = 1 \]
  - 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

\[ p(y, \theta) = \theta^y (1 - \theta)^{1-y} \quad y \in \{0,1\} \]
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

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

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

\[
p(y = 1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mu_1, \Sigma_1) p(y = 1)}{p(\mathbf{x} \mid \mu_0, \Sigma_0) p(y = 0) + p(\mathbf{x} \mid \mu_1, \Sigma_1) p(y = 1)}
\]

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

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

---

QDA: Quadratic decision boundary

Contours of class-conditional densities

---
QDA: Quadratic decision boundary

Linear discriminant analysis (LDA)
• Assumes 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
Generative classification models

Idea:
1. Represent and learn the distribution \( p(x, y) \)
2. Model is able to sample (generate) data instances \((x, y)\)
3. The model is used to get probabilistic discriminant functions \( g_o(x) = p(y = 0 | x) \quad g_1(x) = p(y = 1 | x) \)

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

- \( p(x \mid y) \) = Class-conditional distributions (densities)
  - binary classification: two class-conditional distributions
    \[ p(x \mid y = 0) \quad p(x \mid y = 1) \]
- \( p(y) \) = 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.
- One of the basic ML classification models (often performs very well in practice)
  
  So we have:
  \[ p(x, y) = p(x \mid y) p(y) \]
  \[ p(x \mid y) = \prod_{i=1}^{d} p(x_i \mid y) \]
Learning parameters of the model

Much simpler density estimation problems
• We need to learn:
  \[ p(x \mid y = 0) \quad \text{and} \quad p(x \mid y = 1) \quad \text{and} \quad p(y) \]
• Because of the assumption of the conditional independence we need to learn:
  for every input variable i: \( p(x_i \mid y = 0) \) and \( p(x_i \mid y = 1) \)
• Much easier if the number of input attributes is large
• Also, the model gives us a flexibility to represent input attributes of different forms !!!
• E.g. one attribute can be modeled using the Bernoulli, the other using Gaussian density, or a Poisson distribution

Making a class decision for the Naïve Bayes

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

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

\[
p(y = 1 \mid x) = \frac{\left( \prod_{i=1}^{d} p(x_i \mid \Theta_{1i}) \right) p(y = 1)}{\left( \prod_{i=1}^{d} p(x_i \mid \Theta_{1i}) \right) p(y = 0) + \left( \prod_{i=1}^{d} p(x_i \mid \Theta_{2i}) \right) p(y = 1)}
\]
Next: two interesting questions

(1) Two probabilistic models with linear decision boundaries:
  - Logistic regression
  - LDA model (2 Gaussians with the same covariance matrices)
    \[ x \sim N(\mu_0, \Sigma) \text{ for } y = 0 \]
    \[ x \sim N(\mu_1, \Sigma) \text{ for } y = 1 \]
  - Question: Is there any relation between the two models?

(2) Two models with the same gradient:
  - Linear model for regression
  - Logistic regression model for classification
  have the same gradient update
    \[ \mathbf{w} \leftarrow \mathbf{w} + \alpha \sum_{i=1}^{n} (y_i - f(x_i)) \mathbf{x}_i \]
  - Question: Why is the gradient the same?

---

Logistic regression and generative models

- Two models with linear decision boundaries:
  - Logistic regression
  - Generative model with 2 Gaussians with the same covariance matrices
    \[ x \sim N(\mu_0, \Sigma) \text{ for } y = 0 \]
    \[ x \sim N(\mu_1, \Sigma) \text{ for } y = 1 \]
  - Question: Is there any relation between the two models?
  - Answer: Yes, the 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 !!!

\[
p(y = 1 | x, \mu_0, \mu_1, \Sigma) = g(w^T x)
\]
Logistic regression and generative models

- Members of the exponential family can be often more naturally described as

\[ f(x \mid \theta, \phi) = h(x, \phi) \exp \left\{ \theta^T x - A(\theta) \right\} \frac{1}{\alpha(\phi)} \]

\( \theta \) - 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 logistic regression model

The gradient puzzle …

**Linear regression**

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

Gradient update:

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

Online:

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

**Logistic regression**

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

Gradient update:

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

Online:

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

The same
The gradient puzzle …

• 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**
    
    \[ y = \mathbf{w}^{T}\mathbf{x} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2) \]
  
  - **Logistic + Bernoulli**
    
    \[ y = \text{Bernoulli}(\theta) \]
    
    \[ \theta = p(y = 1 | \mathbf{x}) = g(\mathbf{w}^{T}\mathbf{x}) \]

---

**Generalized linear models (GLIMs)**

**Assumptions:**

• The conditional mean (expectation) is:
  
  \[ \mu = f(\mathbf{w}^{T}\mathbf{x}) \]
  
  Where \( f(.) \) is a **response function**

• Output \( y \) is characterized by an exponential family distribution with a conditional mean \( \mu \)

**Examples:**

- **Linear model + Gaussian noise**
  
  \[ y = \mathbf{w}^{T}\mathbf{x} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2) \]

- **Logistic + Bernoulli**
  
  \[ y \approx \text{Bernoulli}(\theta) \]
  
  \[ \theta = g(\mathbf{w}^{T}\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^{T}\mathbf{x}}} \]
Generalized linear models (GLIMs)

- A canonical response function $f(.)$:
  - encoded in the sampling distribution

$$p(x | \theta, \varphi) = h(x, \varphi) \exp \left\{ \frac{\theta^T x - A(\theta)}{\alpha(\varphi)} \right\}$$

- Leads to a simple gradient form

- Example: Bernoulli distribution

$$p(x | \mu) = \mu^x (1 - \mu)^{1-x} = \exp \left\{ \log \left( \frac{\mu}{1 - \mu} \right) x + \log(1 - \mu) \right\}$$

$$\theta = \log \left( \frac{\mu}{1 - \mu} \right) \quad \mu = \frac{1}{1 + e^{-\theta}}$$

- Logistic function matches the Bernoulli

Evaluation of classifiers
Classification model learning

Learning:
• Many different ways and objective criteria used to learn the classification models. Examples:
  – Mean squared errors to learn the discriminant functions
  – Negative log likelihood (logistic regression)

Evaluation:
• One possibility: Use the same error criteria as used during the learning (apply to train & test data). Problems:
  – May work for discriminative models
  – Harder to interpret for humans.
• Question: how to more naturally evaluate the classifier performance?

Evaluation of classification models

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 $\omega = 1$</th>
<th>$\omega = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1$</td>
<td>140</td>
<td>17</td>
</tr>
<tr>
<td>$\alpha = 0$</td>
<td>20</td>
<td>54</td>
</tr>
</tbody>
</table>
Evaluation of classification models

Confusion matrix entries are often normalized with respect to the number of examples $N$ to get proportions of the different agreements and disagreements among predicted and target values.

<table>
<thead>
<tr>
<th>target</th>
<th>$\omega = 1$</th>
<th>$\omega = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1$</td>
<td>140/231</td>
<td>17/231</td>
</tr>
<tr>
<td>$\alpha = 0$</td>
<td>20/231</td>
<td>54/231</td>
</tr>
</tbody>
</table>

Basic evaluation statistics

Basic statistics calculated from the confusion matrix:

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>$\alpha = 0$</td>
<td>20</td>
<td>54</td>
</tr>
</tbody>
</table>

Classification Accuracy $= \frac{194}{231}$
Basic evaluation statistics

Basic statistics calculated from the confusion matrix:

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

Classification Accuracy = $\frac{194}{231}$

Misclassification Error = $\frac{37}{231} = 1 - \text{Accuracy}$

Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

<table>
<thead>
<tr>
<th>Target</th>
<th>$\omega = 1$</th>
<th>$\omega = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>predict</td>
<td>$\alpha = 1$</td>
<td>$TP$</td>
</tr>
<tr>
<td></td>
<td>$\alpha = 0$</td>
<td>$FN$</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>target</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>140</td>
<td>10</td>
<td>(PPV = 140/150)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>180</td>
<td>(NPV = 180/200)</td>
<td></td>
</tr>
</tbody>
</table>

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

Often project data points to one dimensional space:

**Defined for example by:** \( w^T x + w_0 \) or \( p(y=1|x, w) \)
**Binary classification models**

Often project data points to one dimensional space:

**Defined for example by:** $w^T x + w_0$ or $p(y=1|x,w)$

![Image](image_url)

**Question:** how good is the model with parameters $w$ in terms of class discriminability at different decision thresholds?

**Receiver Operating Characteristic (ROC)**

- **Probabilities:**
  - $SENS$
  - $SPEC$

  $p(x > x^* | x \in \omega_2)$
  $p(x < x^* | x \in \omega_1)$
Receiver Operating Characteristic (ROC)

- ROC curve plots:
  
  \[ SN = p(x > x^* | x \in \omega_2) \]
  
  \[ 1 - SP = p(x > x^* | x \in \omega_1) \]

  for different \( x^* \)

Case 1

Case 2

Case 3
Receiver operating characteristic

- **ROC**
  - shows the discriminability between the two classes under different thresholds representing different decision biases
- **Decision bias**
  - can be changed using the different loss function

- **Quality of a classification model:**
  - Area under the ROC
  - Best value 1, worst (no discriminability): 0.5