Multi-way classification

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

Multi-way classification

- **Binary classification** \( Y = \{0, 1\} \)
- **Multi-way classification**
  - **K classes** \( Y = \{0, 1, \ldots, K - 1\} \)
  - **Goal:** learn to classify correctly \( K \) classes
  - Or learn \( f : X \rightarrow \{0, 1, \ldots, K - 1\} \)
- **Errors:**
  - Zero-one (misclassification) error for an example:
    \[
    Error_1(x_i, y_i) = \begin{cases} 
    1 & f(x_i, w) \neq y_i \\
    0 & f(x_i, w) = y_i 
    \end{cases}
    \]
  - Mean misclassification error (for a dataset):
    \[
    \frac{1}{n} \sum_{i=1}^{n} Error_1(x_i, y_i)
    \]
Multi-way classification

Approaches:

• Generative model approach
  – Generative model of the distribution $p(x, y)$
  – Learns the parameters of the model through density estimation techniques
  – Discriminant functions are based on the model
    • “Indirect” learning of a classifier

• Discriminative approach
  – Parametric discriminant functions
  – Learns discriminant functions directly
    • A logistic regression model.

---

Generative model approach

Indirect:

1. Represent and learn the distribution $p(x, y)$
2. Define and use probabilistic discriminant functions
   
   $g_i(x) = \log p(y = i | x)$

Model

$p(x, y) = p(x | y) p(y)$

- $p(x | y)$ = Class-conditional distributions (densities)
  
  $p(x | y = i) \quad \forall i \quad 0 \leq i \leq K - 1$

- $p(y)$ = Priors on classes

- probability of class $y$
  
  $\sum_{i=1}^{K-1} p(y = i) = 1$
Multi-way classification. Example

Multi-way classification
Making class decision

Discriminant functions can be based on:
• Likelihood of data – choose the class (Gaussian) that explains the input data \((x)\) better (likelihood of the data)

> Choice: \( i = \text{arg max}_{i=0,\ldots,k-1} p(x | \theta_i) \)

\[ p(x | \theta_i) \approx p(x | \mu_i, \Sigma_i) \quad \text{For Gaussians} \]

• Posterior of a class – choose the class with higher posterior probability

> Choice: \( i = \text{arg max}_{i=0,\ldots,k-1} p(y = i | x, \Theta) \)

\[ p(y = i | x) = \frac{p(x | \Theta_i) p(y = i)}{\sum_{j=0}^{k-1} p(x | \Theta_j) p(y = j)} \]

---

Discriminative approach

• Parameteric model of discriminant functions
• Learns the discriminant functions directly

How to learn to classify multiple classes, say 0,1,2?

**Approach 1:**
– A binary logistic regression on every class versus the rest

```
1  0 vs. (1 or 2)
\(x_i\)  1 vs. (0 or 2)
\(x_d\)  2 vs. (0 or 1)
```
Multi-way classification. Example

Multi-way classification. Approach 1.
Multi-way classification. Approach 1.

Ambiguous region

Region of nobody

0 vs \{0,1\}
1 vs \{0,2\}
2 vs \{0,1\}

0 vs \{1,2\}

CS 2750 Machine Learning
**Discriminative approach.**

How to learn to classify multiple classes, say 0, 1, 2?

**Approach 2:**
- A binary logistic regression on all pairs

![Discriminative Approach Diagram](image)

---

**Multi-way classification. Example**

![Multi-way Classification Example](image)
Multi-way classification. Approach 2

- 0 vs 1
- 0 vs 2
- 1 vs 2

Ambiguous region
Multi-way classification. Approach 2

Multi-way classification with softmax

- A solution to the problem of having an ambiguous region

\[ p(y = i | x) = \mu_i = \frac{\exp(w_i^T x)}{\sum_j \exp(w_j^T x)} \quad \sum_i \mu_i = 1 \]
Multi-way classification with softmax

Learning of the softmax model

- Learning of parameters $w$: statistical view

$\mathbf{x} \xrightarrow{\text{Softmax network}} \mu_0 = P(y = 0 \mid \mathbf{x}) \quad \mu_{k-1} = P(y = k-1 \mid \mathbf{x}) \xrightarrow{\text{Multi-way Coin toss}} y$

Assume outputs $y$ are transformed as follows

$y \in \{0, 1, \ldots, k-1\} \Rightarrow y \in \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$
Learning of the softmax model

• Learning of the parameters $\mathbf{w}$: statistical view

**Likelihood of outputs**

$$L(D, \mathbf{w}) = p(\mathbf{Y} | \mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n} p(y_i | \mathbf{x}_i, \mathbf{w})$$

• We want parameters $\mathbf{w}$ that maximize the likelihood

**Log-likelihood trick**

– Optimize log likelihood of outputs instead:

$$l(D, \mathbf{w}) = \log \prod_{i=1}^{n} p(y_i | \mathbf{x}_i, \mathbf{w}) = \sum_{i=1}^{n} \log p(y_i | \mathbf{x}_i, \mathbf{w})$$

$$= \sum_{i=1}^{n} \sum_{q=0}^{k-1} \log \mu_{i,q} = \sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q}$$

• **Objective to optimize**

$$J(D_t, \mathbf{w}) = -\sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q}$$

---

Learning of the softmax model

• **Error to optimize:**

$$J(D_t, \mathbf{w}) = -\sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q}$$

• **Gradient**

$$\frac{\partial}{\partial \mathbf{w}_{jk}} J(D_t, \mathbf{w}) = \sum_{i=1}^{n} -x_{i,j} (y_{i,j} - \mu_{i,j})$$

• The same very easy **gradient update** as used for the binary logistic regression

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \alpha \sum_{i=1}^{n} (y_{i,j} - \mu_{i,j}) \mathbf{x}_i$$

• But now we have to update the weights of $k$ networks
Multi-way classification

- When is the softmax the right model?

\[ x \xrightarrow{\text{Softmax network}} \mu_0 = P(y = 0 | x), \mu_{k-1} = P(y = k - 1 | x) \]

Assume:

\[ p(x | y = i) = h(x, \varphi) \exp \left\{ \frac{(\theta_i^T x - A(\theta_i))}{a(\varphi)} \right\} \]

\( \theta_i \) - location parameter for class conditional i

\( \varphi \) - scaling parameter (the same for all classes)
Multi-way classification

- **Softmax model is an accurate model** when class conditional densities are represented with densities from the exponential family with the same scaling parameter.
- For two classes it reduces to the **logistic regression model**

\[
\begin{align*}
\mu_0 &= P(y = 0 | x) \\
\mu_{k-1} &= P(y = k - 1 | x) \\
p(x | y = i) &= \exp \left\{ \left( \theta_i^T x - b(\theta_i) \right) \right\} + c(x, \varphi)
\end{align*}
\]

- $\theta_i$ - location parameter for class conditional $i$
- $\varphi$ - scaling parameter (the same for all classes)

Bayesian decision theory
Confusion matrix

Results of classification are recorded in:

- **Confusion matrix:**
  - Counts of examples with:
    - class label $\omega_j$ that are classified with a label $\alpha_i$

<table>
<thead>
<tr>
<th></th>
<th>$\omega = 0$</th>
<th>$\omega = 1$</th>
<th>$\omega = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0$</td>
<td>140</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>$\alpha = 1$</td>
<td>17</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>$\alpha = 2$</td>
<td>12</td>
<td>4</td>
<td>76</td>
</tr>
</tbody>
</table>

agreement

Zero-one loss function

- **Misclassification error**
  - Based on the zero-one loss function
    - Any misclassified example counts as 1
    - Correctly classified example counts as 0

<table>
<thead>
<tr>
<th></th>
<th>$\omega = 0$</th>
<th>$\omega = 1$</th>
<th>$\omega = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0$</td>
<td>140</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>$\alpha = 1$</td>
<td>17</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>$\alpha = 2$</td>
<td>12</td>
<td>4</td>
<td>76</td>
</tr>
</tbody>
</table>

agreement

- What is the zero-one loss for the confusion matrix?
General loss function

- Error function based on a more general loss function
  - Different misclassifications have different weight (loss)
  - $\alpha_i$ our choice
  - $\omega_j$ true label
  - $\lambda(\alpha_i \mid \omega_j)$ loss for classification

Example:

\[
\begin{array}{c|ccc}
\omega = 0 & \omega = 1 & \omega = 2 \\
\hline
\alpha = 0 & 0 & 1 & 5 \\
\alpha = 1 & 3 & 0 & 2 \\
\alpha = 2 & 3 & 1 & 0 \\
\end{array}
\]

Bayesian decision theory

- More general loss function
  - Different misclassifications have different weight (loss)
    \[
    \lambda(\alpha_i \mid \omega_j)
    \]
- Expected loss for the classification choice $\alpha_i$
  \[
  R(\alpha_i \mid x) = \sum_j \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid x) \\
  \]
  - Also called conditional risk
  - Posterior of the class
- Decision rule: $\alpha(x)$
  - Chooses label (action) according to the input
- The optimal decision rule
  \[
  \alpha^*(x) = \arg \min_{\alpha_i} \sum_j \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid x)
  \]
Bayesian decision theory

- The optimal decision rule
  \[ \alpha^*(x) = \arg \min_{\alpha_i} \sum_j \lambda(\alpha_i \mid \omega_j)P(y = \omega_j \mid x) \]

  How to modify classifiers to handle different loss?
  - **Discriminative models:**
    - Directly optimize the parameters according to the new loss function
  - **Generative models:**
    - Learn probabilities as before
    - Decisions about classes are biased to minimize the empirical loss (as seen above)

Calculating the loss for data

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

  \[
  \begin{array}{ccc}
  \hline
  \omega & \omega = 0 & \omega = 1 & \omega = 2 \\
  \hline
  \alpha = 0 & 140 & 20 & 22 \\
  \alpha = 1 & 17 & 54 & 8 \\
  \alpha = 2 & 12 & 4 & 76 \\
  \hline
  \end{array}
  \]

  - Agreement

- **Loss**
  \[ \frac{1}{N} \sum_i \sum_j \lambda(\alpha_i \mid \omega_j)N(\alpha_j \mid \omega_j) \]