Naïve Bayes classifier
& Evaluation framework

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

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 $p(x, y) = p(x \mid y)p(y)$

- $p(x \mid y)$ = Class-conditional distributions (densities)
  - binary classification: two class conditional distributions
    $p(x \mid y = 0)$  $p(x \mid y = 1)$
  
- $p(y)$ = 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. So we have:

\[
p(x, y) = p(x | y) p(y)
\]

\[
p(x | y) = \prod_{i=1}^{N} p(x_i | y)
\]

Learning of parameters of the model

Much simpler density estimation problems

- We need to learn:
  \[
p(x | y = 0) \quad \text{and} \quad p(x | y = 1) \quad \text{and} \quad p(y)
\]
- Because of the assumption of the conditional independence we need to learn:
  for every variable \( i \): \( p(x_i | y = 0) \) and \( p(x_i | y = 1) \)
- If the number of input attributes is large this much easier
- Also, the model gives us a flexibility to represent input attributes different of different forms !!!
- E.g. one attribute can be modeled using the Bernoulli, the other as Gaussian density, or as a Poisson distribution
Making a class decision for the Naïve Bayes

**Discriminant functions.**

- **Likelihood of data** – choose the class that explains the input data \((x)\) better (likelihood of the data)
  \[
  \prod_{i=1}^{N} p(x_i | \Theta_{1,i}) > \prod_{i=1}^{N} p(x_i | \Theta_{2,i})
  \]
  then \(y=1\)
  else \(y=0\)

- **Posterior of a class** – choose the class with better posterior probability
  \[
  p(y = 1 | x) > p(y = 0 | x)
  \]
  then \(y=1\)
  else \(y=0\)

\[
\begin{align*}
 p(y = 1 | x) &= \frac{\left( \prod_{i=1}^{N} p(x_i | \Theta_{1,i}) \right) p(y = 1)}{\left( \prod_{i=1}^{N} p(x_i | \Theta_{1,i}) \right) p(y = 0) + \left( \prod_{i=1}^{N} p(x_i | \Theta_{2,i}) \right) p(y = 1)} \\
 &\approx \frac{\sum_{i=1}^{N} p(x_i | \Theta_{1,i})}{\sum_{i=1}^{N} p(x_i | \Theta_{1,i}) + \sum_{i=1}^{N} p(x_i | \Theta_{2,i})}
\end{align*}
\]

Experimental evaluation

Dataset: a set of samples

Split the dataset to: **Training and testing data**

- Learn on the Training data
- Test on the Testing data
- Test errors give an honest assesment of the error for future cases (recall the overfit issue)
Prevent the train/test split bias

If we use only one train/test split we can be lucky or unlucky
A much better (less biased) option is to use multiple train/test
splits and average the test errors obtained on these splits
How to do the splits ?

• **Random subsampling:** choose the test and train set randomly
  k times

• **Cross-fold validation:** a more systematic approach
  – Split data to k equal partitions
  – Create a train data using k partitions, test data on the
    remaining partition
  – Gives us k different train test splits

Evaluation

For any data set we used to test the model 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>target $\omega$</th>
<th>$\alpha = 1$</th>
<th>$\alpha = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega = 1$</td>
<td>140</td>
<td>20</td>
</tr>
<tr>
<td>$\omega = 0$</td>
<td>17</td>
<td>54</td>
</tr>
</tbody>
</table>
## Evaluation

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

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

Error: ?

---

## Evaluation for the binary classification

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

<table>
<thead>
<tr>
<th>$\omega = 1$</th>
<th>$\omega = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1$</td>
<td>$TP$</td>
</tr>
<tr>
<td>$\alpha = 0$</td>
<td>$FN$</td>
</tr>
<tr>
<td>$\omega = 1$</td>
<td>$FP$</td>
</tr>
<tr>
<td>$\omega = 0$</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
  \[ SENS = \frac{TP}{TP + FN} \]

- Specificity
  \[ SPEC = \frac{TN}{TN + FP} \]

- Positive predictive value
  \[ PPT = \frac{TP}{TP + FP} \]

- Negative predictive value
  \[ NPV = \frac{TN}{TN + FN} \]

Binary classification. Additional quantities.

- Confusion matrix
  
  \[
  \begin{array}{ccc|c}
  target & 1 & 0 & \\
  \hline
  1 & 140 & 20 & SENS=140/160 \\
  0 & 10 & 180 & SPEC=180/190 \\
  \end{array}
  \]

  \[ PPV=140/150 \quad NPV=180/200 \]

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

ROC
• shows the discriminability between the two classes under different decision biases (types of errors we make matter)
• ROC curve is created by plotting:
  • the true positive rate against false positive rates
  • or sensitivity against (1-specificity)

Binary decisions: accuracy.

• Probabilities:
  – True positive (hit) \( p(x > x^* \mid x \in \omega_2) \)
  – False positive (false alarm) \( p(x > x^* \mid x \in \omega_1) \)
  – True negative (correct rejection) \( p(x < x^* \mid x \in \omega_1) \)
  – False negative (a miss) \( p(x < x^* \mid x \in \omega_2) \)
Decision threshold

- Movement of $x^*$ changes the probabilities:
  - True positive (hit) $p(x > x^* \mid x \in \omega_2)$
  - False positive (false alarm) $p(x > x^* \mid x \in \omega_1)$
  - True negative (correct rejection) $p(x < x^* \mid x \in \omega_1)$
  - False negative (a miss) $p(x < x^* \mid x \in \omega_2)$

Receiver Operating Characteristic (ROC)

- ROC curve plots: $p(x > x^* \mid x \in \omega_1)$ vs $p(x > x^* \mid x \in \omega_2)$ for different $x^*$
Bayesian decision theory

- Assume we want to incorporate our bias about the learning into the learning process
- Assume a multiway classification problem and more general confusion matrix
  - Counts of examples with:
    - class label $\omega_j$ that are classified with a label $\alpha_i$

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

agreement
**Zero-one loss function**

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

<table>
<thead>
<tr>
<th>ω</th>
<th>α = 0</th>
<th>α = 1</th>
<th>α = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>140</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>76</td>
</tr>
</tbody>
</table>

*agreement*

- What is the zero-ω loss for the confusion matrix?

---

**General loss function**

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

**Example:**

\[
\begin{array}{c|ccc}
\lambda(\alpha_i | \omega_j) & \alpha = 0 & \alpha = 1 & \alpha = 2 \\
\hline
\omega = 0 & 0 & 1 & 5 \\
\omega = 1 & 3 & 0 & 2 \\
\omega = 2 & 3 & 1 & 0 \\
\end{array}
\]
Bayesian decision theory

• More general loss function
  – Different misclassifications have different weight (loss)
    \[ \lambda_i (\omega_j | \omega_j) \]
• Expected loss for choice (action) \( \alpha_i \)
  \[ R(\alpha_i | \mathbf{x}) = \sum_j \lambda_i (\alpha_i | \omega_j) P(\omega_j | \mathbf{x}) \]
  – Also called conditional risk
• Decision rule: \( \alpha(\mathbf{x}) \)
  – Chooses label (action) according to the input
• Overall expected loss for the decision rule
  \[ R(\alpha) = \int R(\alpha(\mathbf{x}), \mathbf{x}) P(\mathbf{x}) d\mathbf{x} \]

Bayesian decision theory

• The optimal decision rule
  \[ \alpha^*(\mathbf{x}) = \arg \max_{\alpha_i} \sum_j \lambda_i (\alpha_i | \omega_j) P(\omega_j | \mathbf{x}) \]

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)