Naïve Bayes Classification

Nickolai Riabov, Kenneth Tiong

Brown University

Fall 2013
Structure of the Talk

- Theory of Naïve Bayes classification
- Naive Bayes in SQL
Notation

- $X$ – Set of features of the data
- $Y$ – Set of classes of the data
Bayes’ Theorem

\[ P(y|x) = \frac{P(x|y)P(y)}{P(x)} \]

- \( P(y) \) – Prior probability of being in class \( y \)
- \( P(x) \) – Probability of features \( x \)
- \( P(x|y) \) – Likelihood of features \( x \) given class \( y \)
- \( P(y|x) \) – Posterior probability of \( y \)
Based on Bayes’ theorem, we can compute which of the classes $y$ maximizes the posterior probability

$$y^* = \arg \max_{y \in Y} P(y|x)$$

$$= \arg \max_{y \in Y} \frac{P(x|y)P(y)}{P(x)}$$

$$= \arg \max_{y \in Y} P(x|y)P(y)$$

(Note: we can drop $P(x)$ since it is common to all posteriors)
Commonality with maximum likelihood

- Assume that all classes are equally likely a priori:

\[ P(y) = \frac{1}{\text{# of elements in } Y} \quad \forall \ y \in Y \]

- Then,

\[ y^* = \text{arg max}_{y \in Y} P(x|y) \]

- That is, \( y^* \) is the \( y \) that maximizes the likelihood function.
Desirable Properties of the Bayes Classifier

- Incrementality: Each new element of the training set leads to an update in the likelihood function. This makes the estimator robust.
- Combines Prior Knowledge and Observed Data.
- Outputs a probability distribution in addition to a classification.
Bayes Classifier

- Assumption: Training set consists of instances of different classes $y$ that are functions of features $x$ (In this case, assume each point has $k$ features, and there are $n$ points in the training set)

- Task: Classify a new point $x_{n+1}$ as belonging to a class $y_{n+1} \in Y$ on the basis of its features by using a MAP classifier

$$y^* \in \arg \max_{y_{n+1} \in Y} P(x_{1,n+1}, x_{2,n+1}, \ldots, x_{k,n+1} | y_{n+1}) P(y_{n+1})$$
Bayes Classifier

- $P(y)$ can either be externally specified (i.e. it can actually be a prior), or can be estimated as the frequency of classes in the training set
- $P(x_1, x_2, \cdots, x_k|y)$ has $O(|X|^k|Y|)$ parameters – can only be estimated with a very large number of data points
Bayes Classifier

- Can reduce the dimensionality of the problem by assuming that features are conditionally independent given the class (this is the Naïve Bayes Assumption)

\[
P(x_1, x_2, \cdots, x_k | y) = \prod_{i=1}^{k} P(x_i | y)
\]

- Now, there’s only \(O(|X||Y|)\) parameters to estimate
- If the distribution of \(x_1, \cdots x_n | y\) is continuous, this result is even more important
  - \(P(x_1, x_2, \cdots, x_k | y)\) has to be estimated nonparametrically; this method is very sensitive to high-dimensional problems
Bayes Classifier

- Learning step consists of estimating $P(x_i|y)$
  \[ \forall i \in \{1, 2, \cdots, k\} \]
- Data with unknown class is classified by computing the $y^*$ that maximizes the posterior

\[
y^* \in \arg \max_{y_{n+1} \in Y} P(y_{n+1}) \prod_{i=1}^{k} P(x_{n+1,i}|y_{n+1})
\]

- Note: Due to underflow, the above is usually replaced with the numerically tractable expression

\[
y^* \in \arg \max_{y_{n+1} \in Y} \ln(P(y_{n+1})) + \sum_{i=1}^{k} \ln(P(x_{n+1,i}|y_{n+1}))
\]
Example

- Classifying emails into spam or ham
- Training set: \( n \) tuples that contain the text of the email and its class

\[ x_{i,j} = \begin{cases} 1 & \text{if word } i \text{ in email } j \\ 0 & \text{otherwise} \end{cases}; \ y_j = \begin{cases} 1 & \text{if ham} \\ 0 & \text{if spam} \end{cases} \]

- Calculate likelihood of each word by class:

\[
P(x_i|y = 1) = \frac{\sum_{j=1}^{n} x_{i,j} \cdot y_j}{\sum_{j=1}^{n} y_j}
\]

\[
P(x_i|y = 0) = \frac{\sum_{j=1}^{n} x_{i,j} \cdot (1 - y_j)}{\sum_{j=1}^{n} (1 - y_j)}
\]
Example

- Define prior, calculate numerator of posterior probability:

\[
P(y_{n+1} = 1 \mid x_{1,n+1}, x_{2,n+1}, \ldots, x_{k,n+1})
\]

\[
\propto P(y_{n+1} = 1) \prod_{i=1}^{k} P(x_{i,n+1} \mid y_{n+1} = 1)
\]

\[
P(y_{n+1} = 0 \mid x_{1,n+1}, x_{2,n+1}, \ldots, x_{k,n+1})
\]

\[
\propto P(y_{n+1} = 0) \prod_{i=1}^{k} P(x_{i,n+1} \mid y_{n+1} = 0)
\]

- If \( P(y_{n+1} = 1 \mid \vec{x}_{n+1}) > P(y_{n+1} = 0 \mid \vec{x}_{n+1}) \), classify as ham.
- If \( P(y_{n+1} = 1 \mid \vec{x}_{n+1}) < P(y_{n+1} = 0 \mid \vec{x}_{n+1}) \), classify as spam.
Naive Bayes in SQL

- **Why SQL?**
  - Standard language in a DBMS
  - Eliminates need to understand and modify internal source

- **Drawbacks**
  - Limitations in manipulating vectors and matrices
  - More overhead than systems languages (e.g. C)
Efficient SQL implementations of Naïve Bayes

Numeric attributes
- Binning is required (create k uniform intervals between min and max, or take intervals around the mean based on multiples of std dev)
- Two passes over the data set to transform numerical attributes to discrete ones
  - First pass for minimum, maximum and mean
  - Second pass for variance (due to numerical issues)

Discrete attributes
- We can compute histograms on each attribute with SQL aggregations
Bayesian K-means (BKM) is a generalisation of Naïve Bayes (NB)

- NB has 1 cluster per class, BKM has $k > 1$ clusters per class
- The class decomposition is found by K-Means algorithm
K-Means algorithm finds $k$ clusters by choosing $k$ data points at random as initial cluster centers. Each data point is then assigned to the cluster with center that is closest to that point.

Each cluster center is then replaced by the mean of all data points that have been assigned to that cluster.

This process is iterated until no data point is reassigned to a different cluster.
### Tables needed for Bayesian K-means

<table>
<thead>
<tr>
<th>Table</th>
<th>Content</th>
<th>PK</th>
<th>non-key columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>XH</td>
<td>Normalized data</td>
<td>$i$</td>
<td>$g, X_1, \ldots, X_d$</td>
</tr>
<tr>
<td>CH</td>
<td>Centroids</td>
<td>$g$</td>
<td>$C_{11}, C_{12}, \ldots, C_{dk}$</td>
</tr>
<tr>
<td>XD</td>
<td>distances</td>
<td>$i, g$</td>
<td>$d_1, \ldots, d_k$</td>
</tr>
<tr>
<td>XN</td>
<td>nearest cluster</td>
<td>$i, g$</td>
<td>$j$</td>
</tr>
<tr>
<td>NLQ</td>
<td>Suff. stats.</td>
<td>$g, j$</td>
<td>$N_g, L_1, \ldots, L_d, Q_1, \ldots, Q_d$</td>
</tr>
<tr>
<td>WCR</td>
<td>mixture model</td>
<td>$g, j$</td>
<td>prior,$C_1, \ldots, C_d, R_1, \ldots, R_d$</td>
</tr>
<tr>
<td>XP</td>
<td>prob. per cluster</td>
<td>$i, g$</td>
<td>$p_1, \ldots, p_k$</td>
</tr>
<tr>
<td>XC</td>
<td>predicted class</td>
<td>$i$</td>
<td>$g$</td>
</tr>
<tr>
<td>MQ</td>
<td>model quality, $N_g$</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
The following SQL statement computes k distances for each point, corresponding to the g\textsuperscript{th} class.

```
INSERT INTO XD
SELECT i, XH.g, (X1-C1_X1)**2 + .. + (X4-C1_X4)**2,
     .., (X1-C3_X1)**2 + .. + (X4-C3_X4)**2
FROM XH, CH WHERE XH.g=CH.g;
```
Results

- Experiment with 4 real data sets, comparing NB, BKM, and decision trees (DT)
- Numeric and discrete versions of Naïve Bayes had similar accuracy
- BKM was more accurate than NB and similar to decision trees in global accuracy. However BKM is more accurate when computing a breakdown of accuracy per class
Results

- Low numbers of clusters produced good results
- Equivalent implementation of NB in SQL and C++: SQL is four times slower
- SQL queries were faster than User-Defined functions (SQL optimisations are important!)
- NB and BKM exhibited linear scalability in data set size and dimensionality.