Mixture Models in Data Analysis
Naïve Bayes / Chow-Liu Tree Model
T-61.6020 Special Course in Computer and Information Science II P

Lasse Kärkkäinen
Helsinki University of Technology

2008-02-13
1 Introduction
   - Bayesian methods and Chow-Lee trees

2 Chow-Liu
   - Chow-Liu tree
   - Mutual information
   - Example

3 Material
   - Extra material and program code
Bayesian methods and Chow-Lee trees

Bayesian classifiers

- Popular in spam filters
- A simple statistical method for classifying content, based on Bayes’ theorem
  - Teaching the classifier (e.g. by words of documents)
    - A fairly small number of documents needed
    - Each classified to one of the classes
    - \( P(X_i|C) \) calculated for each word \( X_i \) and each class \( C \)
- Classification based on the words of the target document using the probabilities calculated during teaching
Naïve Bayes

- Naïve Bayes assumes fully independent variables $X_i$
- Seems to work well in practice, even when variables are strongly dependent
- Zhang04\(^1\) provides formal analysis
  - [...] no matter how strong the dependencies among attributes are, naive Bayes can still be optimal if the dependencies distribute evenly in classes, or if the dependencies cancel each other out.

\(^1\)H. Zhang (2004) *The Optimality of Naive Bayes*
Terminology

- Underlearning refers to the situation where the model does not sufficiently classify the data; some information is left unused.
- Overlearning is the opposite, when the model is too eager to explain things from the noise of the input data, rather than real information.
- Bayesian learning solves the trade-off between the two.
- The probability of event $X$, $P(X)$, is called the *prior* of $X$.
- A conditional probability, $P(X|C)$, is called the *posterior probability* because it is derived from or depends upon the specified value of $C$. 

Lasse Kärkkäinen

Helsinki University of Technology

Mixture Models
Bayesian methods and Chow-Lee trees

Likelihood function

- Likelihood function $L(Y|X)$ allows estimating unknown parameters $Y$ based on known outcomes $X$, so it is a sense the backwards of probability, which does the opposite with $P(X|Y)$.

- Bayes’ theorem formalizes this as follows:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \propto L(X|Y)P(X)$$

- The ratio $L(X|Y)/P(Y)$ is sometimes called *normalized likelihood* as it eliminates the normalization factor from the Bayes equation.
using the terms defined,

\[
    \text{posterior} = \frac{\text{likelihood} \ast \text{prior}}{\text{normalizing constant}}
    = \text{normalized likelihood} \ast \text{prior}
\]
Chow-Liu tree

- A method for approximating joint probability distributions of $n$ dependent random variables $X_1...X_n$
- E.g. $P(X_1, X_2, ..., X_6) \approx P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_2)P(X_5|X_2)P(X_6|X_5)$
- Can be represented as a tree of dependencies
- Only one new variable in each factor
The tree is constructed so that Kullback–Leibler divergence\(^2\) between the actual distribution and the approximation is minimized.

The paper shows that the divergence is
\[
D = - \sum I(X_i; X_{i-1}) + \sum H(X_i) - H(X_1, ..., X_n),
\]

where
- \(I(\ldots)\) is the mutual information between the variables, and
- \(H(\ldots)\) is the joint entropy of the variables.

Since only the first term depends on the ordering of the tree, maximizing the sum will provide the smallest divergence.
Introduction

Chow-Liu

Material

Chow-Liu tree

Construction of optimal dependency tree

- Chow & Liu algorithm works by finding the maximum information pair on each round and creating a link between them.
- The root node in this graph (maximum spanning tree) may be chosen arbitrarily.
  - Only the information between connected nodes, i.e. the sum of link weights, affects divergence.
  - A spanning tree is an undirected acyclic graph.
  - Once the root is chosen, the structure can be seen as a tree.

Lasse Kärkkäinen

Helsinki University of Technology

Mixture Models
Mutual information

- Before tree construction, the mutual information of each node pair must be calculated
- Mutual information is a quantity that measures the mutual dependence of two variables
- For continuous variables
  \[ I(X; Y) := \int_Y \int_X p(x, y) \log_2 \left( \frac{p(x, y)}{p_1(x)p_2(y)} \right) \, dx \, dy \]
- For discrete variables, replace the integrals with sums
- \( p(x, y) \) is the joint probability (density) function of the variables and \( p_1 \) & \( p_2 \) are marginal probability functions of \( x \) and \( y \), respectively
- Base 2 logarithm gives the result in bits, but using different base only adds a constant multiplier
Calculating mutual information

- Alternatively, entropies may be used in place of probability functions

\[
I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X, Y)
\]

- \( H(\text{Var}) \) is the entropy of a variable
- \( H(\text{VarA}|\text{VarB}) \) is the entropy of \( \text{VarA} \) on the condition \( \text{VarB} \)
- \( H(\text{VarA}, \text{VarB}) \) is the joint entropy of the variables
Entropic vs. probabilities

The previous equivalency comes from the definition of entropy

$$H(X) := \int_X p(x) \log_2(p(x)) \, dx$$

$$H(X_1, X_2) := \int_{X_2} \int_{X_1} p(x_1, x_2) \log_2(p(x_1, x_2)) \, dx_1 \, dx_2$$

$$H(X_1|X_2) := \int_{X_2} \int_{X_1} p(x_1, x_2) \log_2(p(x_1|x_2)) \, dx_1 \, dx_2$$

$$= H(X_1, X_2) - H(X_2)$$
**Example**

- **Input data:**
  \[
p_1(1) = 0.53, \quad p_2(1) = 0.42, \quad p_3(1) = 0.39
\]

<table>
<thead>
<tr>
<th>(x_1x_2x_3)</th>
<th>(P(x_1, x_2, x_3))</th>
<th>(p_1(x_1)p_2(x_2)p_3(x_3))</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0.15</td>
<td>0.166</td>
</tr>
<tr>
<td>001</td>
<td>0.14</td>
<td>0.106</td>
</tr>
<tr>
<td>010</td>
<td>0.00</td>
<td>0.120</td>
</tr>
<tr>
<td>011</td>
<td>0.18</td>
<td>0.077</td>
</tr>
<tr>
<td>100</td>
<td>0.29</td>
<td>0.188</td>
</tr>
<tr>
<td>101</td>
<td>0.00</td>
<td>0.120</td>
</tr>
<tr>
<td>110</td>
<td>0.17</td>
<td>0.135</td>
</tr>
<tr>
<td>111</td>
<td>0.07</td>
<td>0.087</td>
</tr>
</tbody>
</table>

- **Mutual information matrix:**

<table>
<thead>
<tr>
<th></th>
<th>0.004</th>
<th>0.242</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.004</td>
<td>nan</td>
<td>0.093</td>
</tr>
<tr>
<td>0.242</td>
<td>0.093</td>
<td>nan</td>
</tr>
</tbody>
</table>
```cpp
#include <cmath>
#include <cstdlib>
#include <iostream>

double p(double prob, bool test) { return test ? prob : 1.0 - prob; }

int main(void) {
    using namespace std;
    int freq[8] = {0};
    int bitfreq[3] = {0};
    // Joint frequencies n_{uv}(i,j) \sim jointfreq[i][j][u][v]
    int jointfreq[3][3][2][2] = {{0}};
    for (int round = 0; round < 100; ++round) {
        int val = rand() % 7;
        if (val > 2) ++val;
        if (val % 3 == 2) val ^= 1;
        ++freq[val];
        bool bits[3];
        bits[0] = val & 4;
        bits[1] = val & 2;
        bits[2] = val & 1;
        for (int i = 0; i < 3; ++i) {
            if (bits[i]) ++bitfreq[i];
            for (int j = 0; j < 3; ++j) ++jointfreq[i][j][bits[i]][bits[j]];
        }
    }
}```
double prob[8], naiveprob[8];
for (int i = 0; i < 8; ++i) {
    prob[i] = 1e-2 * freq[i];
    naiveprob[i] = p(1e-2*bitfreq[0], i & 4) * p(1e-2*bitfreq[1], i & 2) * p(1e-2*bitfreq[2], i & 1);
    cout << prob[i] << " " << naiveprob[i] << endl;
}
for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 3; ++j) {
        double information = 0.0;
        for (int k = 0; k < 4; ++k) {
            bool u = k/2, v = k%2;
            double tmp = double(jointfreq[i][j][u][v]) / (jointfreq[i][j][0][0] + jointfreq[i][j][1][0] + jointfreq[i][j][1][1]) ;
            information += tmp*log(tmp/(p(1e-2*bitfreq[i], u)*p(1e-2*bitfreq[j], v)))/log(2.0);
        }
        cout << information << '\t';
    }
    cout << endl;
}
Further reading