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

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 *priori* 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

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

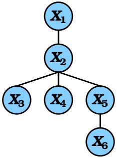
Bayes' theorem

- Using the terms defined,

$$\begin{aligned} \textit{posterior} &= \frac{\textit{likelyhood} * \textit{prior}}{\textit{normalizing constant}} \\ &= \textit{normalized likelyhood} * \textit{prior} \end{aligned}$$

Chow-Liu tree

- A method for approximating joint probability distributions of n dependent random variables $X_1 \dots X_n$
- E.g. $P(X_1, X_2, \dots, X_6) \approx P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_2)P(X_5|X_2)P(X_6|X_5)$



- Can be represented as a tree of dependencies
- Only one new variable in each factor

Construction of optimal dependency tree

- 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 $D = -\sum I(X_i; X_{i-1}) + \sum H(X_i) - H(X_1, \ldots, X_n)$, 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

[2] the difference between two probability distributions

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

Calculating 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

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) \\
&= H(Y) - H(Y|X) \\
&= H(X) + H(Y) - H(X,Y)
\end{aligned}
$$

- $H(Var)$ is the entropy of a variable
- $H(VarA|VarB)$ is the entropy of $VarA$ on the condition $VarB$
- $H(VarA, VarB)$ is the joint entropy of the variables

Entropy 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:
  $p_1(1) = 0.53,\ p_2(1) = 0.42,\ p_3(1) = 0.39$

$x_1x_2x_3$	$P(x_1, x_2, x_3)$	$p_1(x_1)p_2(x_2)p_3(x_3)$
000	0.15	0.166
001	0.14	0.106
010	0.00	0.120
011	0.18	0.077
100	0.29	0.188
101	0.00	0.120
110	0.17	0.135
111	0.07	0.087

- Mutual information matrix:

nan	0.004	0.242
0.004	nan	0.093
0.242	0.093	nan

chowlee.cpp 1/2

```
#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) ~ 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]];
        }
    }
```

chowlee.cpp 2/2

```
    cout << bitfreq[0] << " " << bitfreq[1] << " " << bitfreq[2] << endl;
    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*bi
        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]+jointf
                information += tmp*log(tmp/(p(1e-2*bitfreq[i], u)*p(1e-2*bitfreq[j], v))
            }
            cout << information << '\t';
        }
        cout << endl;
    }
}
```

Further reading

- Zhang04 – http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf
- Chow-Liu – http://ieeexplore.ieee.org/iel5/18/22639/01054142.pdf (free access from TKK)
- http://en.wikipedia.org/wiki/Chow-Liu_tree