Clustering to Minimize Entropy — Probabilistic Measure (Climetropy)

Vladimír Šišma

Department of Theoretical Computer Science and Mathematical Logic,
Charles University in Prague

June 2009

Contents

Motivational Example 1

- we deal a specific clustering problem of Data Mining
- customers $c_1, \ldots, c_6$ are watching movies m_1, m_2, m_3 and they rate them by mark 1, 2, 3, 4 or 5
- 5 for best movies, 1 for the worst movies
- example of sequences of ratings:
  - $c_1 : (m_1, 2), (m_3, 2)$ - two ratings of movies m_1, m_3
  - $c_2 : (m_1, 2), (m_2, 4)$
  - $c_3 : (m_2, 5), (m_3, 4)$
  - $c_4 : (m_3, 5), (m_1, 2)$
  - $c_5 : (m_2, 4), (m_1, 1)$
  - $c_6 : (m_3, 1), (m_2, 4)$
- rating distribution of movies in all sequences:
  - m_1: 1 - 1x, 2 - 3x (good entropy)
  - m_2: 4 - 3x, 5 - 1x (good entropy)
  - m_3: 1 - 1x, 2 - 1x, 4 - 1x, 5 - 1x (bad entropy)

Example 1

- let's analyze following cluster of two respective three groups
- clustering group G_1: c_1, c_6
- rating distribution of movies:
  - m_1: 2 - 1x
  - m_2: 4 - 1x
  - m_3: 1 - 1x, 2 - 1x
- clustering group G_2: c_3, c_4
- rating distribution of movies:
  - m_1: 2 - 1x
  - m_2: 5 - 1x
  - m_3: 4 - 1x, 5 - 1x
- rating distributions of all movies in both groups have good entropy
- remaining sequences c_2, c_5 can be included into groups G_1, G_2 or own new group G_3 arbitrarily without increasing of entropy of any rating distribution

Example 1 - Conclusion

- What we didn't improve by clustering:
  - entropy of rating distribution of movies m_1, m_2 in groups G_1, G_2 and their entropy in all sequences didn't change and its value is still relative low
- What we improved by clustering:
  - much better entropy of rating distribution of movie m_3 in groups G_1, G_2 then its entropy in all sequences
  - entropy of rating distribution of all movies in all groups G_1, G_2 (and eventually G_3) are similar and relatively low

Introduction

- we deal a specific problem of Data Mining
- let's have several sequences of k-tuples from the same domain. Let the sequences be called c-sequences
- the c-sequences are related to each other because
  - they can contain a few similar tuples
  - two tuples are similar when they are equal in one or more items
- the amount of differences in the set of c-sequences is **entropy** of the entire set - entropy of rating (in general called counting) distribution of elements in tuples
- the main goal of the presented paper is to cluster the set of c-sequences to limited number of disjoint subsets so that the entropy in each subset is minimal. The limit is given before the clustering process starts. Let a subset of c-sequences of any clustering be called a c-group.

Introduction

- we introduce a function for measuring entropy of any subset of c-sequences. The function computes the measure of relevance between a c-sequence and a c-group (a set of c-sequences). The function is called the c-reputation. The computation of the c-reputation is based on the theory of probability and mainly on the "Kullback-Leibler divergence". The c-reputation is normalized. It means that values of several c-reputations can be compared for all c-sequence-c-group combinations without loss of meaning.
- we describe one simple clustering algorithm that is based mainly on the c-reputation. It tries to find a clustering of the set of the input c-sequences such that all other clusterings that differs in only one c-sequence are worse. A worse clustering means that the c-reputation of the only one shifted c-sequence and its new c-group is not better than the c-reputation of the c-sequence and the original c-group. The algorithm is similar to famous k-mean algorithm.

Formal Definitions

- Let's define the c-sequence. The c-sequence is sequence of k-tuples of length n $[[s_{11}, \ldots, s_{1k}], \ldots, [s_{n1}, \ldots, s_{nk}]]$ where k, n are positive integer, for all $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$ $s_{ij} \in S_j$. The sets $S_1, \ldots, S_k$ are called domains of k-tuples, related to the specific problem.
- in the Example 1:
  - $k = 2$,
  - various *c-sequences* can have various lengths n
  - $S_1 = \{m1, m2, m3\}$, $S_2 = \{1, 2, 3, 4, 5\}$

Formal Definitions

- Let's define the c-distribution. The *c-distribution* is computed by an algorithm, related to the specific problem. Input of the algorithm is any set of *c-sequences*. Output of the algorithm and *c-distribution* is probability distribution defined on sets of domains $\{S_1, \ldots, S_k\}$.
- in the Example 1:
  - there are three *c-distributions:* probability distributions of ratings for movies m_1, m_2, m_3. One distribution for each movie
  - the domain for all *c-distributions* is set S_2

Formal Definitions

- Let's define the c-group. The *c-group* contains a set of *c-sequences* and several fixed *c-distributions*. *A c-distributions* in a *c-group* is typically evaluated on the *c-sequences* in the same *c-group*.
- in the Example 1:
  - group G_1 from the Example 1 with the three *c-distributions* described above is *c-group*
  - groups G_2, G_3 from the Example 1 with the *c-distributions* are *c-groups* analogically, of course

Formal Definitions

- Let's define c-cluster. Let's have a set (called input set) of *c-sequences* that have the same positive integer k and the same domains $S_1, \ldots, S_k$. Let's have one cluster of input set of *c-sequences*, ie. several disjoint subsets of input *c-sequences*. Let's have fixed set of *c-distributions*. The *c-cluster* contains the set of *c-groups*. Each *c-group* contains one such subset. Each *c-group* contain the same given set of *c-distributions* that are evaluated on the *c-group*'s subset of *c-sequences*.
- in the Example 1:
  - set of *c-groups* for groups G_1, G_2, G_3 from the Example 1 (described above) forms *c-cluster*

Kullback-Leibler Divergence

Kullback-Leibler divergence is denoted and defined, lets U, V are probability distributions:

$$D_{KL}\left(U|V\right) = \sum_i U(i) \ln \frac{U(i)}{V(i)}$$

where holds (by a limit) $0 \cdot \ln \frac{0}{x} = 0$.
Facts:

- it measures divergence of two probability distributions - increases with more divergent distributions
- $D_{KL}\left(U|V\right) \geqq 0$, $D_{KL}\left(U|V\right) = 0 \iff U = V$
- it holds: if $\forall i, U(i) > 0 \Rightarrow V(i) > 0$ then $D_{KL}\left(U|V\right) < \infty$ (1)

A proof:

- based on Jensen inequality: ln is concave function

The Function c-reputation

Denotation:

- let Q be a c-sequence, E be a c-distribution, G be a c-group
- let Φ be a set of c-sequences, Ψ be a set of c-distributions, Ω be a set of c-groups (or c-cluster too)
- for $E \in \Psi$ let's denote the domain of c-distribution E by E_D
- let $G + Q$ be c-group such that G with additional c-sequence Q
- let $G_E(s)$ be the probability of item $s \in E_D$ in counting distribution evaluated by an algorithm related to a c-distribution E in a c-group G (on c-group's set of c-sequences). Then G_E is a mapping $E_D \longrightarrow \mathbb{R}$
  - in the Example 1: let G be group G_1 in Example 1, let E be c-distribution that counts rating distribution of movie m_1. Then $G_E : S_2 \longrightarrow \mathbb{R} : 1, 2 \longmapsto \frac{1}{2}; 3, 4, 5 \longmapsto 0$
- let $\tilde{Q}$ be auxiliary c-group containing only a c-sequence Q

The Function c-reputation

- the function c-reputation of the c-sequence $Q \in \Phi$ according to c-group $G \in \Omega$ is denoted and defined:

$$R_G(Q) = \exp\left(-\sum_{E\in\Psi} D_{KL}(\tilde{Q}_E|G_E)\right)$$

- c-reputation quantifies the measure of similarity between a c-sequence Q and a c-group G
- $R_G(Q) \in [0, 1]$. Decreases with more divergent Q, G. Maximum: $\forall E \in \Psi, \tilde{Q}_E = G_E \Leftrightarrow R_G(Q) = 1$
- from (1) on page 12 holds: if G contains Q then $R_G(Q) > 0$
- $R_G(Q)$ can be considered as "average probability"

Clustering Algorithm

- we introduce the simple clustering algorithm.
- it tries to find the best clustering of the set of the input c-sequences.
- it works iteratively and in each iteration it tries to find better clustering. It stops when it finds such clustering that each c-sequence has better c-reputation in its c-group then in other c-group.
- first iteration begins with initial cluster. It can be chose:
  - randomly
  - $\{\Phi, \{\}, \ldots, \{\}\}$
  - result cluster of the last running of the algorithm
  - others
- the input of algorithm is the input set of c-sequences Φ and limitative constants GroupMax, IterMax. GroupMax limits the number of groups and IterMax limits the number of iterations. The output of algorithm is the found set of c-groups Ω.

Scheme of the Algorithm

- set Ω to initial cluster of Φ, $|\Omega| = GroupMax$
- **do** //do iterations
  - **for** $Q \in \Phi$ //find shifts and do shifts immediately
    - denote G' such that Q is in G'
    - **delete** Q **from** G'
    - **find** $G \in \Omega$ such that $R_{G+Q}(Q)$ is maximum
    - **insert** Q **into** G, let's denote this by **shift** of Q if $G' \neq G$
- **while** the number of iterations are less then $IterMax$ and at least one **shift** of any c-sequence occurred

Clustering Algorithm - Properties

- the complexity of algorithm is:

$$O\left(\left(\sum_{Q \in \Phi} |Q|\right) \cdot GroupMax \cdot IterMax\right)$$

  It is "multilinear" complexity, it means it can process a quite large input.
- the algorithm doesn't guarantee to find *ideal* cluster
  - different initial clusters can lead to different result clusters
- unless defining maximum of iterations the algorithm doesn't guarantee finite number of iterations

Clustering Algorithm - Fast Convergency

- the algorithm usually converges very fast
  - number of shifts in following iterations decrease exponentially with base around 2 (usually)
  - reason is under research

"Bad" alternative algorithm:

- set Ω to initial cluster of Φ, $|\Omega| = GroupMax$
- **do** //do iterations
  - **for** $Q \in \Phi$ //find shifts
    - **find** $G \in \Omega$ such that $R_{G+Q}(Q)$ is maximum, denote $G_Q := G$
  - **for** $Q \in \Phi$ //do shifts
    - denote G' such that Q is in G'
    - **move** Q **from** G' **into** G_Q, let's call this by **shift** of Q if $G' \neq G_Q$
- **while** the number of iterations are less then $IterMax$ and at least one **shift** of any c-sequence occurred

The alternative algorithm usually diverges. It means that the number of shifts in consecutive iterations usually diverges for alternative algorithm.

Tests

- I processed several tests with production commercial data
  - one test is similar to Example 1, but with 500 000 costumers, 17 700 movies, 100 000 000 ratings
- alone climetropy didn't show sufficient performance till today
  - compared with methods specialized to specific problem
- but it is possible to use Climetropy as preprocessor for other efficient methods that can be used for each group computed by Climetropy separately