Automatically Determining Number of Clusters

Hua Yu (hyu@cs.cmu.edu)

Abstract

Automatically determining number of clusters in the data is an unsolved/unexplored problem. First I'll show why we need to do this, and whether this is a reasonable problem in text clustering in particular. Then starting from simple 1-d/2-d study, I find BIC (Bayesian Information Criterion) is a useful measure, which by penalizing model fitness by model complexity, usually tells the right number of clusters. Experiments on EM clustering of 1-d/2-d data are presented. In text clustering, I find inter-document similarity matrix, when correctly organized & visualized, a very good representation of the collection. Based on that I tried a Similarity-matrix based clustering algorithm, which gives visually appealing results. However, BIC measure can't be easily extended to text clustering case.

1 Motivation

1.1 Why do we need to know the number of clusters

Given a large collection of documents, you don't have time to read through them one by one. Here naturally arises the need to cluster them and present them in a structured way. But before starting out, most clustering system will ask you for a number: how many clusters would you like to cluster them into? It's a magic number, how can the user know in advance?

Hierarchical clustering schemes tries to avoid the problem by presenting a tree structure (dendrogram) of the collection, generally a binary tree. They argue that the user can get whatever number of clusters out by cutting the tree at suitable points. Again the problem is: how can the user know where to cut in advance?

I tried GAC clustering on a subset of SWB corpus. As said before, the package asks the user for the number of clusters wanted before starting out. Obviously supplying the wrong number gets back meaningless clusters.

Not knowing the right number can hurt in some cases. In speech recognition, acoustic events are modelled by mixture Gaussian. It's widely known that using too many, as well as too few mixture components results in degradation of recognition performance. But how to set the right number is an unsolved problem.

So here's a niche in clustering research: how to discover the right number of clusters automatically?

1.2 Is it possible or reasonable a goal?

For example when you look up at the sky in a clear summer night, you'll see a lot of stars, and you'll have no difficulty saying that here's a group of them, there's another group ... The grouping might be wrong, might be fuzzy, but it helps. So I define my goal as to discover the rough number of "natural" clusters there in the data, to get a starting point.

For document clustering, again, is it possible? Here're some nice pictures (Appendix pp. 13). It's baiscally visualization of inter-document similarity matrix for 100 documents randomly sampled from TDT corpus. Brighter point means higher similarity value, except the diagonal, where the value is one (maximum) but the color is sort of dark (because of the way Matlab manipulates colors). I'm quite excited when I first got them, even if I can't automatically derive the magic number, simply presenting the picture will help the user a lot. Experiment on SWB corpus shows similar results.

1.3 Literature Survey

There're few paper dedicated to this topic. The only one I found is in [2], a paper on clustering from a physics point of view. This paper gives a clean theoretical treatment of model complexity vs. model fitness relationship, though it doesn't come up with any experiment or workable solution. [3] introduced the BIC (Bayesian Information Criterion), and discussed how it can be used in audio segmentation and audio segments clustering (with a different focus). Since he only cares about discriminating between single Gaussian and mixture Gaussian with 2 components, he avoids much headaches which general purpose clustering has to face. [4] is not directly relevant to my purpose, though he gives a rather comprehensive review of popular clustering methods used in text clustering, and some ideas on testing cluster validity

As there's little work done before, I decide to start from some simulated study of 1-d/2-d clustering, get some experience before I move on to text clustering.

2 BIC criterion in speech segmentation

BIC criterion arises in speech segmentation study, where people want to detect drastic change in audio stream, like speaker change, music starts, etc. The objective is to segment the audio stream into some homogeneous pieces.

Suppose a homogeneneous segment can be modelled with a Gaussian, which is usually reasonable with appropriate feature selection. If there's a speaker change in a segment, then it can't be modelled well with a single Gaussian; instead we'd like to use 2 Gaussians to model it. So the problem seems to reduce to model complexity question: which model fits data better, single Gaussian or 2 Gaussians?

This is a tricky question: generally with more parameters in the model, you'll get better likelihood, this is widely known as overfitting problem. BIC is a likelihood criterion penalized by model complexity: number of free parameters in the model. The idea is, for our case, if the data is simple enough to be well modelled with single Gaussian, then we'll go with this simple model; 2 Gaussian model might give a little better likelihood, but that doesn't justify for the additional complexity. This is also known as Minimum Description Length principle.

$$\text{BIC(M)} = \text{logL(X,M)} - \text{lambda} * 1/2 * \#\text{(M)} * \log\text{(N)}$$

where M is model, X is data set, N is the number of samples in X, lambda is a constant, usually 1.

or,

$$\text{AvgBIC(M)} = \text{BIC(M)/N} = \text{logL(X,M)/N} - \text{lambda} * 1/2 * \#\text{(M)} * \log\text{(N)/N}$$

The detail of why using this penalty goes a long way into the statistics literature, involving some assumption about the data and generally hard to interpret. The general idea is we don't penalize much when there's enough data, in which case the measured likelihood converges to the true likelihood. But this is a good starting point where we can experimentally determine its goodness.

Here's a nice plot of BIC for 1-d case (see Appendix pp.1,2).

The idea is to search through all x-values, see if somewhere

$$\text{BIC(2 Gaussian)} > \text{BIC(single Gaussian)}$$

and choose the point with the largest distance to be the breaking point.

3 Simulated Study of 1-d/2-d clustering

In the 1-d/2-d cases, we model each cluster as a Gaussian, so choosing number of clusters amounts to determine number of components in mixture Gaussian modelling.

Note to get the likelihood measurement for k-mixture modelling, we need to actually find the optimal k-mixture model that fits the data. This is of course an EM task. EM formular for mixture Gaussian estimation can be found in [1].

I did a series of experiments, enclosed in the report are:

- modelling of data generated from a single Gaussian model
- modelling of data generated from 2 well-seperated Gaussian model

- modelling of data generated from 2 not-well-seperated Gaussian model
- modelling of 2-d data generated from 8 Gaussian model

together with some brief discussion of interesting observations. (see Appendix pp.3-12)
The conclusion I've got from these experiments is:

- BIC criterion works pretty well in these experiments
- EM convergence rate / optimality is an important factor in carrying out experiments. When the mixture components are not well seperated, EM is very likely to get stuck in some local maximal trap. This is especially the case when we're trying to model a k-cluster data with (k+1) Gaussians: basically we're modelling some sampling variance.

  I tried annealed EM, an EM variant that's said to be robust w.r.t. local maximum, but there's no guarantee and in fact, it didn't work well in my problem.

  EM convergence problem is well discussed in [1].

Another non-parametric method for choosing the right model size is cross-validation, i.e. we keep an eye on the performance on an independent validaton set while fitting the training set with different models. I also examined its usefulness in the simulated study. In general it's useful, and easy to apply, though it's not panacea. There's one counter-example in my experiments enclosed in this report, and one can easily think of other counterexamples where the cross-validation set is a subset of training set. Empirically it's used a lot, but we should bear in mind it's just another rule-of-the-thumb techniques.

4 How to determine number of clusters

Now we have a method for determining number of clusters, i.e. start from a reasonably small number, we monitor the growth of likelihood function as the model complexity increases, until a point when BIC criterion says "Stop".

There's another way we can go, as described in [2]. They call it "simulated annealing" or "stachastic relaxation". Translated into my terms, what they're doing is that they fix the number of clusters to be a number large enough, and each time they set a different likelihood value as target, adjust model parameters to achieve that likelihood. When gradually increasing the likelihood, they watch how the clusters behavior changes over time. They argue that at each step there's only a small number of distinct clusters, as the likelihood we're asking for goes higher, existing clusters will split, result in more distinct clusters. They argue that splitting happens at some critical points in time, it's not a smooth process. So at the end we should be able to get a natural hierarchical clustering.

Of course I find the argument there is kind of handwaving, how to define "distinctive" may turn out to be as difficult as the original problem. Also how to gradually increase the likelihood, how to set the stepsize is also not obvious.

But the idea is clear: we're examining model goodness vs. model complexity, and we can go along either dimension, while keeping an eye on the other. [2] is monitoring model complexity while requiring higher and higher model accuracy, the way I'm going is to monitor model accuracy while gradually increasing model complexity. The advantage of going this way is model complexity is easier to control, in our case it's just the number of clusters, so we don't need to worry about how to set the stepsize.

While BIC criterion gives stop criterion in the simulated study, in real life it might not work well. For example, BIC penalty is proportional to number of parameters of a cluster, in text clustering, how many parameters should one cluster have? It depends on our representation of a cluster, in other words, dimension of feature space. Clearly if we use the original Vector Space Model without dimensional reduction, i.e. use word as indexing unit, the penalty will be unrealisticly high. Also, using EM to find the maximum likelihood for each k (# clusters) is not trivial. It's time consuming and may very likely fail to find the global maximum, detering the use of BIC criterion.

5 Text Clustering – a tentative study

As shown above, I get some quite nice pictures of inter-document similarity matrix. Sim(i,j) is simply the cosine measurement between document i and document j, using Smart ltc term weighting. This is all straightforward, the question is how to order documents correctly to unveil the structure in the corpus. Without the ordering information, the plot just looks random (Appendix pp. 14,15).

To get the ordering information is essentially a clustering task. Of course we can do the traditional clustering, but once we have the Sim matrix, we can just as well do clustering on Sim, with the goal being permute it to a meaningful form.

So I invented a "new" clustering algorithm based on Sim matrix. Our target is to get those bright square sub-matrices, "bright" means heavily weighted elements. In other words, our goal is to group documents in a way that similarity between documents of the same group is maximized, and we don't care about the similarity between documents from different group. In a sense this is like minimizing within-cluster scatter.

The algorithm has an EM flavor, though monotonicity is not guaranteed. Assuming we want k clusters,

0) randomly assign each document to a cluster

1) for each document, compare it with every current cluster, assign it to the one that's closest. The closeness between document i and cluster j is measured as the average similarity between i and every document in cluster j.

It's very similar to k-means algorithm, except that I didn't calculate any cluster statistics explicitly. Cluster statistics is implicit in the sub-Sim-matrix corresponding to that cluster.

The results are visually appealing (Appendix pp.16). Though it suffers the same problem as EM type algorithm: local maximum. In an effort to examine the effect of k on clustering, I tried to calculate the "likelihood" as I increase the value of k. "likelihood" is defined as the overall within-cluster similarity, i.e. for each cluster j, we sum up similarities between all documents in that cluster, then add it up for all k clusters. I run the experiment on 800 documents from TDT corpus, covering 20 human assigned categories. The "likelihood" curve should be monotonically increasing, but due to local maximum traps, I can't get 2 of the true maximum, after trying various different starting points (I've even tried the "Oracle" initial point).

In the mixture Gaussian modelling case we can use BIC criterion to set the stop point, here we can't. Also the likelihood curve doesn't show any interesting shape to give some indication of the right number of clusters.

6 Conclusion

Determining number of clusters is difficult. In simple scenario like 1-d/2-d case, BIC criterion is very useful and usually gives the right answer, though actually carrying out EM algorithm to achieve global maximum with a certain k is not easy: we may run into local maximum. BIC criterion can't be easily extended to text clustering, due to the "dimensionality curse". However, the notion of natural clusters does exist, and it's easy to find them by using a simple Sim-matrix based clustering algorithm, and I still believe it's possible to get a rough estimate of the magic number.

7 Acknowledgements

The author would like to thank Tom Pierce for help on the GAC software package.

References

[1] Lei Xu, On Convergence Properties of the EM algorithm for Gaussian Mixtures, Neural Computation, 8, 129-151, 1996

[2] Kenneth Rose, et al., Statistical Mechanics and Phase Transitions in Clustering, Physical Review Letters, Vol.65, 8, 1990

[3] S.S.Chen et al., Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion, Hub4 Workshop, 1998

[4] P.Willett, Recent Trends in Hierarchic Document Clustering: a Critical Review

[5] T.Hofmann, et al., Statistical Models for Co-occurance Data, M.I.T. A.I. Memo No.1625, 1998

[6] T.Hofmann, Structuring Document Databases by Automated Clustering and Abstraction