Learning the k in k-means

Greg Hamerly **Charles Elkan**
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0114
$\{$*ghamerly,elkan*$\}$@*cs.ucsd.edu*

Abstract

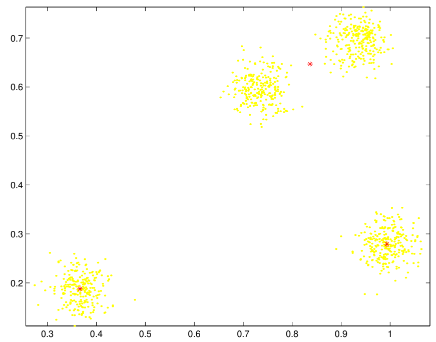
When clustering a dataset, the right number k of clusters to use is often not obvious, and choosing k automatically is a hard algorithmic problem. In this paper we present a new algorithm for choosing k that is based on a new statistical test for the hypothesis that a subset of data follows a Gaussian distribution. The algorithm runs k-means with increasing k until the test fails to reject the hypothesis that the data assigned to each k-means center are Gaussian. We present results from experiments on synthetic and real-world data showing that the algorithm works well, and better than a recent method based on the BIC penalty for model complexity.

1 Introduction

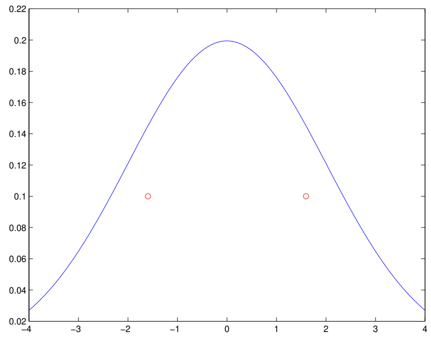
Clustering is the task of dividing a set of data points into several subsets, where the points in each subset are similar to each other, and different from points in other subsets. Clustering algorithms are useful tools for data mining, compression, unsupervised learning, probability estimation, and other tasks in machine learning and statistics. However, most clustering algorithms require the user to provide the number of clusters (called k), and it is not always clear what the best value for k is. Figure 1 shows examples where k has been improperly chosen. In general, choosing k is often an *ad hoc* decision based on prior knowledge, assumptions, and practical experience.

The most common center-based clustering algorithms (in particular k-means and Gaussian expectation-maximization) are based on the assumption that true clusters have Gaussian distributions. With these methods, only one center should be used to describe each subset of data that follows a Gaussian distribution. If multiple centers are used to describe a Gaussian subset, the centers are a needlessly complex description of the subset (see Figure 1(b)), and in fact the multiple centers capture the truth about the subset less well than one center.

In this paper we present a new statistical test for deciding whether to split a k-means center into two centers, and a simple algorithm called G-means based on the test for discovering an appropriate k. We describe examples and present experimental results that show that the new algorithm is successful.



(a) Three k-means centers in red (dark) are used to cluster four true clusters in yellow (light). The top center should be split to cover the two clusters nearest it.



(b) Two k-means centers (red) are used to cluster one true cluster (distribution shown in blue). One center only should be used.

Figure 1: Two poor-quality final clustering solutions. On the left, k is too small, while on the right k is too big. One center should be used to represent one Gaussian cluster.

2 Related work

Several algorithms have been proposed previously to determine k automatically. Like our method, most previous methods are wrappers around k-means or some other clustering algorithm for fixed k. Wrapper methods use splitting and/or merging rules for centers to increase or decrease k as the algorithm proceeds. Other research on agglomerative clustering suggests choosing k based on the "stability" of the merging tree (dendrogram) of distances between points. Doing this usually requires prior knowledge about the data, and using these methods often no easier than choosing k directly.

Pelleg and Moore [5] proposed a regularization framework for learning k, which they call X-means. The algorithm scores each clustering model using the so-called Bayesian Information Criterion

$$BIC(M) = \mathcal{L}(D) - \frac{p}{2} \log R$$

where $\mathcal{L}(D)$ is the log-likelihood of the dataset according to model M, and p and R are the model complexity parameters [3]. Then they choose the model with the best BIC score.

Bischof *et al.* [1] use a minimum description length (MDL) framework, where the description length is a measure of how well the data is fit by the model. Their algorithm starts with a large value for k and removes centers (reduces k) whenever that choice reduces the description length. Between steps of reducing k, they use the k-means algorithm to optimize the model fit to the data.

In hierarchical clustering algorithms, "cluster analysis" is used to determine the best number of clusters for a data set. One heuristic is to build a dendrogram of the data based on a cluster distance metric, and to search for areas of the tree that are relatively stable with respect to inter-cluster and intra-cluster distances. However, this method of finding the number of clusters is best applied with domain-specific knowledge and human intuition.

3 The Gaussian-means (G-means) algorithm

The G-means algorithm starts with a small number of k-means centers, and grows the number of centers. Each iteration of the algorithm splits into two those centers whose data appears not to come from a Gaussian distribution. Between each round of splitting, we run k-means on the entire data set and all the centers to refine the current solution. We can initialize with just $k = 1$ or $k = 2$, or we can choose some larger value of k if we have some prior knowledge about the range of k.

G-means algorithm

1. Run k-means on all centers and data.
2. For each k-means center, use a statistical test to detect if the data assigned to the center shows a Gaussian distribution.
3. If the data look Gaussian, keep the center. Otherwise, split the center into two centers.
4. Repeat from step 1 until no more centers are added.

This algorithm repeatedly makes decisions based on a statistical test for the data assigned to each center. If the data currently assigned to a k-means center appear to be Gaussian, then we want to represent that data with only one center. However, if the same data does not appear to be Gaussian, then we want to use multiple centers to model it properly.

The k-means algorithm implicitly assumes that the data points in each cluster are distributed spherically around the center. Less restrictively, the Gaussian expectation-maximization algorithm assumes that the data points in each cluster have a multidimensional Gaussian distribution with a covariance matrix that may or may not be a unit matrix. The Gaussian distribution test that we present below is valid for any covariance matrix. The test also accounts for the number of data points involved, which prevents the G-means algorithm from making bad decisions about small clusters.

4 A distortion-based test for normality

To specify the G-means algorithm fully we need a test to detect whether the data assigned to a center are sampled from a Gaussian. The alternative hypotheses are:

- H_0: The data around the center are sampled from a Gaussian.
- H_1: The data around the center are not sampled from a Gaussian.

If we accept the null hypothesis H_0, then we believe that that center is sufficient to model its data, and we should not split the cluster into two subclusters. If we reject H_0 and accept H_1, then we want to split the cluster.

We actually perform two tests of the null hypothesis H_0. For each test we must compute a statistic and fix a significance level α. We use statistics that measure the distortion of the data around two centers. Under H_0 the distribution of these *distortion statistics* is a scaled version of the standard χ^2 distribution, which can therefore be used to perform the hypothesis tests.

Let $X = \{x_i\}$ for $1 \leq i \leq |X|$ be any set of w-dimensional data points and let c be a center. The distortion $r(X, c)$ is the sum of the squared Euclidean distances between x_i

and c:

$$r(X, c) = \sum_{i=1}^{|X|} ||x_i - c||^2.$$

The test of H_0 for center c and the data X which is assigned to c proceeds as follows:

- Find the covariance Σ of X, and the eigenvectors v_j and eigenvalues λ_j of Σ, where $1 \leq j \leq w$. Let $p = \arg\max_j \lambda_j$.
- Create two centers c_1 and c_2 with initial locations

  $$c_1 = c - v_p\sqrt{\frac{2\lambda_p}{\pi}} \qquad c_2 = c + v_p\sqrt{\frac{2\lambda_p}{\pi}}$$

  Run k-means on X with centers c_1 and c_2. Let X_1 (X_2) be the partition of X that is assigned to c_1 (c_2).
- Calculate the distortions $r_1 = r(X_1, c_1)$ and $r_2 = r(X_2, c_2)$. Then r_1 and r_2 are the two distortion statistics.
- Formulate the χ^2 statistics $m_1 = f(r_1, \{\lambda_j\})$ and $m_2 = f(r_2, \{\lambda_j\})$. The function f is defined below.
- Perform two hypothesis tests: if $m_1 < \chi^2_{\alpha,d_1}$ or $m_2 < \chi^2_{\alpha,d_2}$, then reject H_0 and keep the split centers c_1 and c_2 in place of c. Otherwise, we accept H_0 and keep the center c. Here $d_1 = |X_1| - 1$ and similarly for d_2.

The initial center locations for c_1 and c_2 are the locations that give the minimum variance when H_0 is true, which is along the axis (eigenvector v_p) of the largest spread (the largest eigenvalue λ_p). We omit the proof due to the limited space. Placing the initial center locations at the location of minimum variance means that k-means has to use very few update iterations, since the initial locations are near the correct locations (assuming H_0 is true and the data is sampled from a Gaussian).

Under the null hypothesis H_0, r_1 and r_2 are each distributed according to a scaled χ^2 distribution. The mean and variance of the random variable r_1 (similarly for r_2) are:

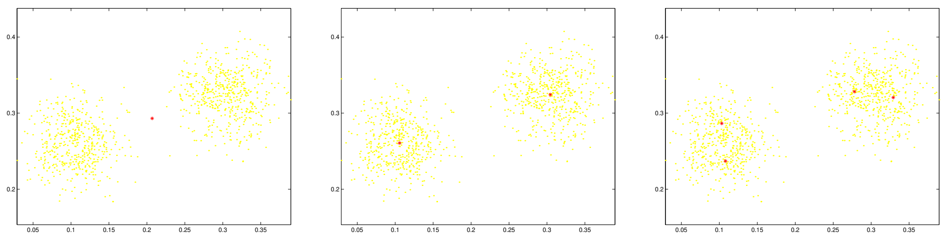
$$\begin{aligned} \mu_1 &= E\left[r_1 | H_0, \{\lambda_j\}\right] \\ &= d_1 \left(\sum_{j=1}^{d} \lambda_j - \frac{2}{\pi}\lambda_p \right) \qquad (1) \\ \sigma_1^2 &= Var\left[r_1 | H_0, \{\lambda_j\}\right] \\ &= d_1 \left(2\sum_{j=1}^{d} \lambda_j^2 + \lambda_p^2 \left(\left(\frac{2}{\pi}\right)^2 - 2 \right) \right) \qquad (2) \end{aligned}$$

where λ_j is the jth eigenvalue of the covariance matrix of X, λ_p is the largest eigenvalue, and $d_1 = |X_1| - 1$ is the degrees of freedom. We omit the proofs for these properties due to space limitations; they will appear in a longer version of this paper.

The probability density function of the distortion statistic is linearly related to the PDF of the χ^2 distribution. We transform from the distortion statistic to the χ^2 statistic to perform the significance test with the function f, which is defined as

$$f(r_1, \{\lambda_j\}) = \frac{r_1 - \mu_1}{\sqrt{\sigma_1^2}}\sqrt{2d_1} + d_1.$$

Here $2d_1$ is the variance and d_1 is the mean of the χ^2 distribution with d_1 degrees of freedom.



(a) Initialization. Before running G-means, $k = 1$ and the center is located at the centroid of the data. The distortion around the center is $r = 12.71$.

(b) Iteration 1. After one split, $k = 2$. The distortions around each center $r = (0.97, 0.87)$ are significant, so we reject H_0. This will be the final solution for this data.

(c) Iteration 2. After two more splits, $k = 4$. The distortions $r = (0.36, 0.27, 0.26, 0.31)$. are *not* significant, so we do not accept the new splits.

Figure 2: An example of running G-means for three iterations on a 2-dimensional dataset with two true clusters and $n = 1000$ points.

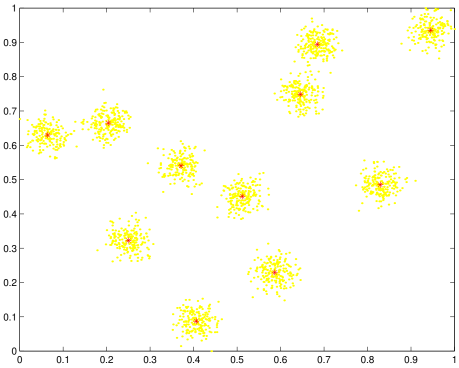
The test we perform is one-tailed since the goal of k-means is to minimize within-cluster distortion. Using a one-tailed test checks if either subcluster improves the distortion significantly. If it does, then we reject H_0 and accept the split centers.

We must choose the significance level of the test, α, which is the probability with which we will make a type 2 error (i.e. reject H_0 when we should not reject it). Since the algorithm performs many statistical tests while learning k, it is appropriate to use a Bonferroni adjustment to reduce the chance of making type 2 errors over multiple tests. For example, if we want a 0.01 chance of making a type 2 error in 10 tests, we should apply a Bonferroni adjustment to make each test use $\alpha = 0.05/10 = 0.005$. The G-means algorithm makes $2(2k - 1)$ statistical tests if it finds k final centers, which is not a large number of tests if k is small, so the Bonferroni correction does not need to be extreme. In the experiments below we use $\alpha = 0.0001$, which corresponds for example to a 0.3% chance of an incorrect split overall when $k = 8$.

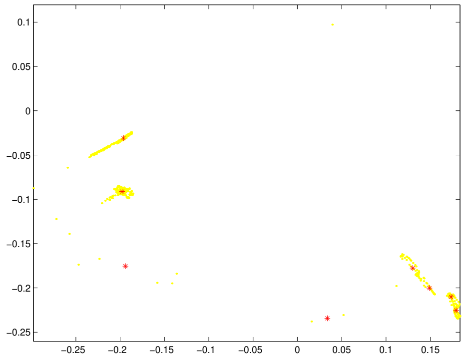
A benefit of using the statistical test above (over simply comparing the values of two variances) is that when the number of sample points is small, the test tends to accept the null hypothesis. In other words, the test can find cluster structure at many levels of resolution in the data, but when a cluster is too small (too few data points), the test becomes less confident. Therefore it will not split clusters with a small number of points, which would happen if we were simply comparing two variances.

An example with two true clusters. Figure 2 shows a run of the G-means algorithm on a synthetic dataset with two true clusters and 1000 points, using $\alpha = 0.0001$. The distortion of the data around the initial center at the middle of the dataset is $r = 12.71$. After one iteration of G-means, we have $k = 2$ centers and statistics of:

	c_1	c_2
n	509	491
r	0.97	0.87
m	195.7	176.5
$\chi^2_{\alpha,n-1}$	397.9	382.0



(a) G-means finds the correct number of centers $k = 11$ and places them in the right locations in this synthetic dataset. Search started from $k = 1$, with $\alpha = 0.0001$.



(b) G-means finds $k = 8$ clusters here, in a real-world dataset with 2 dimensions and 1037 data points. Search started from $k = 1$, with $\alpha = 0.0001$.

Figure 3: G-means finds the correct number of clusters in the synthetic dataset on the left, and discovers 8 genuine clusters in the real-world `gzip` dataset on the right.

Here r represents the distortion statistic, and $m = f(r, \{\lambda_j\})$ is the equivalent scaled statistic to compare with the χ^2 distribution. Both values of m are smaller than their corresponding $\chi^2_{\alpha,n-1}$ values, so we reject H_0 and accept the split.

Next we split each of the previous two centers, giving $k = 4$ total centers. Now the statistics are:

	c_{11}	c_{12}	c_{21}	c_{22}
n	263	246	235	256
r	0.36	0.27	0.26	0.31
m	289.8	198.9	216.5	264.9
$\chi^2_{\alpha,n-1}$	185.3	171.1	162.0	179.4

Here, the empirical values m are all larger than their corresponding $\chi^2_{\alpha,n-1}$ values, so we accept H_0 and reject both splits.

5 Experimental results

Figure 3(a) shows an example of the G-means algorithm correctly finding 11 true clusters, starting with one center and using $\alpha = 0.0001$.

Figure 3(b) shows the results on a dataset of 1,037 data points in 2 dimensions which represents a trace of the execution of the program `gzip`. The original dataset had 8,163 dimensions. Each dimension represents a basic block of execution from the program's binary image. A value x in dimension y means that basic block y was executed y times in a given time-slice of execution. Each data point represents a time-slice of 100 million executed instructions. This dataset is an important one used in current research on optimizing processor architecture [**?**].

We reduce the dimension of the `gzip` data drastically by using a random linear projection to 2 dimensions, as in [2]. After this random projection, we use the G-means algorithm to

num_splits	1	2	3	4	5	6	7	8	9	10
k found by X-means	1	2	3	5	9	14	24	36	50	50

Table 1: Number of centers found by X-means in the 2-dimensional dataset `gzip`, as a function of the parameter `num_splits`. $k = 50$ appears to be a hard limit in the X-means software. It is not clear how to set `num_splits` based on this table.

α	0.00001	0.0001	0.001	0.01
k found by G-means	6	8	8	11

Table 2: Number of centers found by G-means in the same dataset. Results are robust over a wide range of values for α. See the text for why $\alpha = 0.0001$ is a reasonable *a priori* choice.

dimension	2	6	10	14	18	22
average k	9.0	12.3	14.1	15.8	15.1	15.2
standard deviation of k	3.4	3.6	3.8	4.2	3.2	4.5

Table 3: Mean and standard deviation of the number of clusters found by G-means on the `gzip` dataset. For each test we randomly project the data down to the dimension shown and run G-means on the projected data with $\alpha = 0.0001$. We do this 30 times for each dimension.

cluster the data and to find k.

In Tables 1 and 2 we see that depending on the parameter given to the algorithms (for X-means it is `num_splits`, for G-means it is α), we can obtain a different number for k. However, it is far more intuitive how to choose α than how to choose `num_splits`, since the former is the standard statistical test parameter, while the latter is a heuristic which must reflect some notion of previous knowledge of the data. As for most real-world datasets, for this dataset there is no objectively correct value of k. However the plot in Figure 3(b) indicates that any number of clusters much smaller or larger than 8 is implausible.

Table 3 shows the average and standard deviation of the number of clusters G-means found in the `gzip` dataset as a function of the data dimension. For each test we randomly project the data down to the dimension shown in the table and run G-means with $\alpha = 0.0001$. These results show that G-means works well in higher-dimensional data, and is relatively stable at finding the same number of clusters despite the dimension of the data. Extreme dimension reduction (e.g. down to 2 dimensions) can collapse clusters together, thus we expect to see the trend that G-means finds fewer clusters in lower dimensions. However, at 14 dimensions the average number of clusters found levels off, indicating that there are 15 genuine clusters in this dataset which are revealed in $>=$ 14 dimensions.

6 Discussion and conclusions

We have introduced a new statistical test for determining whether data points are a random sample from a Gaussian distribution with arbitrary dimension and covariance matrix. The new test uses a distortion statistic with a scaled χ^2 distribution, making the test easy to implement. Our new G-means algorithm uses this statistical test as a wrapper around k-means to discover the number of clusters k automatically. The only parameter supplied to the algorithm is α, the significance level of the statistical test, which can easily be set in a standard way. Empirically, the G-means algorithm works well at finding the correct number of clusters and the locations of genuine cluster centers, and we have shown it works well in moderately high dimensions. The statistical test used by G-means could also be used as a test to *merge* centers rather than split them.

The G-means algorithm takes linear time and space in the number of data points n and the dimension w, since k-means is $O(nw)$ in time and space. The computation of the covariance matrix Σ is also $O(nw)$, and the computation of the eigenvectors and eigenvalues of Σ is $O(w^3)$, which is reasonable if the dimension w is small.

The G-means algorithm is described above as a wrapper around standard k-means. However the same idea can be applied with other center-based clustering algorithms, including expectation-maximization for mixtures of Gaussians, and harmonic k-means [**?**, 8].

Clustering in high dimensions has been an open problem for many years. Recent research has shown that it may be preferable to use dimensionality reduction techniques before clustering, and then use a low-dimensional clustering algorithm such as k-means, rather than clustering in the high dimension directly. In [2] the author shows that using a simple, inexpensive linear projection preserves much of the properties of data (such as cluster distances), while making it easier to find the clusters. Thus there is a need for good-quality, fast clustering algorithms for low-dimensional data. The work reported here is a step in this direction.

Additionally, recent image segmentation algorithms such as normalized cut [7, 4] are based on eigenvector computations on distance matrices. These "spectral" clustering algorithms still use k-means as a post-processing step to find the actual segmentation (usually in a lower-dimensional space than the original input) and they require k to be specified externally. Thus the G-means algorithm will be useful in combination with spectral clustering.

Thanks to Sameer Agarwal and Kristin Branson for fruitful discussions, and to Brad Calder for the use of the `gzip` dataset.

References

[1] Horst Bischof, Aleš Leonardis, and Alexander Selb. MDL principle for robust vector quantisation. *Pattern analysis and applications*, 2:59–72, 1999.

[2] Sanjoy Dasgupta. Experiments with random projection. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)*, pages 143–151, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[3] Robert E. Kass and Larry Wasserman. A reference Bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the American Statistical Association*, 90(431):928–934, 1995.

[4] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Neural Information Processing Systems*, 14, 2002.

[5] Dan Pelleg and Andrew Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann, San Francisco, CA, 2000.

[6] Peter Sand and Andrew Moore. Repairing faulty mixture models using density estimation. In *Proceedings of the 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2001.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[8] Bin Zhang. Generalized k-harmonic means – boosting in unsupervised learning. Technical Report HPL-2000-137, Hewlett-Packard Labs, 2000.