A Voronoi Diagram Approach to Autonomous Clustering

Heidi Koivistoinen, Minna Ruuska, and Tapio Elomaa

Institute of Software Systems, Tampere University of Technology
P. O. Box 553, FI-33101 Tampere, Finland
firstname.lastname@tut.fi

Abstract. Clustering is a basic tool in unsupervised machine learning and data mining. Distance-based clustering algorithms rarely have the means to autonomously come up with the correct number of clusters from the data. A recent approach to identifying the natural clusters is to compare the point densities in different parts of the sample space.
In this paper we put forward an agglomerative clustering algorithm which accesses density information by constructing a Voronoi diagram for the input sample. The volumes of the point cells directly reflect the point density in the respective parts of the instance space. Scanning through the input points and their Voronoi cells once, we combine the densest parts of the instance space into clusters.
Our empirical experiments demonstrate the proposed algorithm is able to be come up with a high-accuracy clustering for many different types of data. The Voronoi approach clearly outperforms k-means algorithm on data conforming to its underlying assumptions.

1 Introduction

Clustering is the fundamental task of grouping together similar unlabeled objects in order to obtain some generalization through categorization. This need is encountered in a vast array of application. As similarity of objects varies from application to application, there cannot be an universally superior method of clustering instances. Thus, it should not come as a surprise that the number of different unsupervised clustering algorithms put forward in the literature is huge (see e.g., [1]).

There are two basic approaches to obtaining a clustering: one can use either the bottom-up *agglomerative* or the top-down *divisive* construction. The former begins with each instance in a singleton cluster and successively merges clusters together until a stopping criterion is met. In the latter approach one begins with all instances in a single cluster and partitions the clusters until satisfaction of a stopping criterion. Usually a *distance measure* is required to determine which clusters to merge together next or which cluster to divide because of the sparseness of the points in it.

Of course, it is also possible to choose an intermediate value for the number of clusters and start manipulating the clusters from there. A simple and widely used

clustering algorithm is k-means clustering [2–4]: Given k initial (e.g., random) cluster centers $C = \{ c_1, \ldots, c_k \}$ for the metric observations $S = \{ s_1, \ldots, s_n \}$, iterate the following until the cluster centers do not change.

1. For each observation point $s_i \in S$ determine the center $c_j \in C$ that is closest to it and associate s_i with the corresponding cluster.
2. Recompute the center locations (as the center of the mass of points associated with them).

The iterative k-means minimizes the squared error — the squared Euclidean distance between the cluster centers and the observations associated with them.

The obvious shortcomings of the basic k-means clustering are that the number of clusters needs to determined in advance and its computational cost with respect to the number of observations, clusters, and iterations. Though, k-means is efficient in the sense that only the distance between a point and the k cluster centers — not all other points — needs to be (re)computed in each iteration. Typically the number of observations n is much larger than k.

There have been many approaches trying to alleviate both of these problems. Variations of k-means clustering that are supposed to cope without prior knowledge of the number of cluster centers have been presented [5, 6]. Several proposals for scaling up clustering and, in particular, k-means for massive data sets have been proposed [7–10]. Quite often these studies assume that a particular distance measure is applied. Moore [10] and Elkan [11] have shown how the triangle inequality can be used for any distance function to reduce the number of distance calculations required during the execution of k-means by carrying information over from iteration to iteration.

While the proposed solutions to k-means clustering certainly improve its practical behavior, they do not overcome the fundamental problems associated with the algorithm. For example, the G-means algorithm of Hamerly and Elkan [6] makes k-means autonomously find approximately the right number of clusters at the expense of increased running time. However, the algorithm still errs consistently even on simple cases [12]. On the other hand, the speed of clustering alone, without exactness of the end result, is not of much use. Rather than stubbornly try to make distance-based clustering algorithms cope with the problem of identifying the right number of clusters, one needs to change the point of view and find an approach that is better suited to the task at hand. We propose density-based algorithms as a better alternative to identifying the correct number of clusters in the data.

In this paper we propose an agglomerative clustering algorithm that begins from the situation in which each instance makes up a cluster on its own. For singleton cluster selection we use *Voronoi diagrams* [15]. The initial clusters are merged together with neighboring cells as long as the cell volumes are below a user defined threshold. Observe that we do not need a distance measure, since only neighboring cells are candidates for merging and the Voronoi diagram readily provides us with the neighborhood information. On the other hand, we need to have the threshold volume to limit merging of cells. Our approach is efficient in the sense that there is no need for iterative processing.

To the best of our knowledge the only prior clustering algorithm taking directly advantage of Voronoi diagrams is that of Schreiber [16]. However, his algorithm only adjusts k-means clustering with the help of the Voronoi diagram. The Voronoi diagram is built over the clusters, not for the data points. Clusters with the largest error are halved in two. Voronoi diagrams and their duals Delaunay tesselations are, though, widely used in nearest neighbor methods [4]. Also clustering of graphs can use the Delaunay triangulation as the starting point of processing and then apply, e.g., the minimum spanning tree clustering to the graph at hand [1].

This paper is organized as follows. Section 2 motivates the algorithm introduced in this paper and discusses related work. Basics of Voronoi diagrams are recapitulated and the agglomerative clustering algorithm is introduced in Section 3. Finer points of the algorithm are further discussed in Section 4. Our empirical experiments in Section 5 show that clustering based on Voronoi diagrams significantly outperforms the k-means algorithm already on normally distributed spherical data, which suits the latter clustering algorithm well. The difference between the two is even higher on more complex (real-world) data. Section 6 contains the concluding remarks of this paper and outlines future work.

2 Motivation and Related Work

Clustering based on the regular-shaped classes easily runs into trouble when faced with data in which the existing clusters do not conform to the allowed shape of clusters. This is e.g. the case with k-means clustering and its spherical clusters. For example, Figure 2 demonstrates a data set with different geometric shapes that cannot be captured or approximated well by using only one shape of clusters.

Hence, a clear requirement for a general-purpose clustering algorithm is that it be able to discover clusters of arbitrary shape. Visually (non-overlapping) clusters are usually quite easy to detect because the density of data points within a cluster differs (significantly) from that of points surrounding it. There have been many approaches trying to utilize this intuition. For example, the DBSCAN algorithm of Ester, Kriegel, and Xu [17] searches for clusters that are maximal sets of density-connected points, that is, points that have at least a minimum number of neighbors in their immediate neighborhood form a cluster.

The subsequent algorithm DBCLASD by Xu et al. [18] explicitly approximates the cluster density by comparing the approximate subspace and overall volume of the point set. This tells us whether the subspace is denser than average in the given sample. Because the clusters may have arbitrary shape, it is not immediately clear how to compute the volume of the part of the instance space that a point set occupies. Xu et al. use a (hyper)rectangular grid division of the instance space in volume estimation. This makes volume approximation simple but, at the same time, highly dependent on the chosen granularity.

The elegant density-based clustering algorithm of Hinneburg and Keim [19] computes an influence function over the whole data set. *Density attractors* —

local maxima of the overall density — then determine the clusters. For a continuous and differentiable density function, a hill-climbing algorithm can be used to discover the clusters efficiently. On the downside, two (unintuitive) parameter values that heavily influence the quality of the obtained clustering need to be determined.

In addition to the practical clustering algorithms studied in data mining and machine learning literature, a lot of theoretical work has been devoted to this problem. Theoretically what we have at hand is an intractable optimization problem that can be efficiently solved only approximately. For example, k-means clustering is NP-hard already when $k = 2$. Recently the first linear-time $(1 + \varepsilon)$-approximation algorithm for fixed k and ε based on sampling was devised by Kumar, Sabharwal, and Sen [13]. Another recent theoretical work on k-means and k-median clustering is that of Har-Peled and Mazumdar [14].

Not only the different shapes of clusters, but also the different sizes of clusters can cause problems to some algorithms. For example, the k-means algorithm cannot cope well with different-sized clusters; i.e., clusters of different radius or with different numbers of points in them. Consider, e.g., two cluster centers distance d apart from each other. If one of the clusters has radius $2d/3$ and the other one radius $d/3$, then k-means will unavoidably place part of the points of the first cluster to the second cluster simply because its center is nearer to the points. Similarly, consider two clusters one containing 100 and the other one 1,000 points. Minimizing the squared distance in the denser cluster by placing both centers within it may be more profitable than letting both clusters contain one center.

The starting point for our work is that one ought to be able to discover arbitrary-shaped clusters, like in the algorithms mentioned above. Density within and without a cluster is the measure that actually defines the clusters. Therefore, we change the point of view from distance-based to density-based clustering. Optimizing the correct measure is expected to lead to much better results than fixing algorithms attempting to optimize a measure that is only indirectly related to the task at hand. A lesson learned from prior work is that taking global characteristics of the data into account leads to better results than relying on local properties alone. Another design principle is that we do not want to restrict the number of clusters in advance like is required, e.g., in k-means. Moreover, we aim at as autonomous clustering as possible. That is, there should not be many parameters whose value the user needs to set. Any parameters that the user needs to set have to be intuitive, though not necessarily concrete.

3 Agglomerative Voronoi Clustering

3.1 Voronoi Diagrams and Delaunay Tessellations

A multidimensional Voronoi diagram [15] is a partitioning of the instance space $\mathbb{R}^d$ in regions R_j with the properties: Each center c_j lies exactly in one region R_j, which consists of all points $x \in \mathbb{R}^d$ that are closer to c_j than any other center

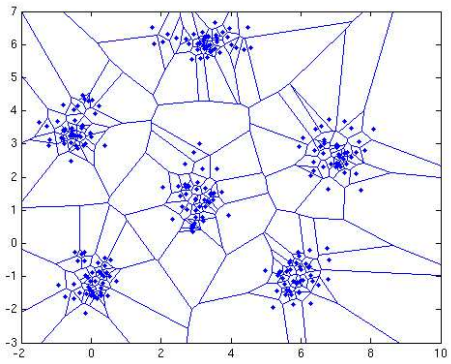

Fig. 1. A Voronoi diagram for data coming from six clusters.

c_k, $j \neq k$:

$$R_j = \left\{ x \in \mathbb{R}^d : \|x - c_j\| < \|x - c_k\|, \forall j \neq k \right\}.$$

The centers c_j are called *Voronoi points*. In the setting of unsupervised learning, where a sample $S = \{ s_1, \ldots, s_n \}$ is given, the sample points s_i, naturally, become the Voronoi points.

Let us call the region associated with a Voronoi point c_j its *cell*. Because a cell contains all those points that are closer to its center than any other center, the cell borders lie exactly in the middle of two centers. Figure 1 demonstrates the Voronoi diagram for a data set that clearly has six separate clusters.

The dual of the Voronoi diagram for a point set is its *Delaunay tessellation* (also Delaunay triangulation). It is a graph in which the vertices are the centers of the Voronoi cells (the initial data points) and edges connect any two vertices that have a common boundary in the Voronoi diagram. The same set of edges is obtained by connecting any two points p and q for which there exists a ball B that passes through p and q and does not contain any other point of S in its interior or boundary. The name Delaunay triangulation stems from the fact that in the plane the *triangulation* of a point set S is a planar graph with vertices S and a maximal set of straight line edges. Adding any further straight line edge would lead to crossing other edges. A subset of a edges of a triangulation a *tessellation* of S if it contains the edges of the convex hull and if each point of S has at least two adjacent edges.

Voronoi diagram is straightforward to calculate for a point set in the plane, but becomes more complicated in higher dimensions. Therefore, the uses of Voronoi diagrams are mostly limited to $\mathbb{R}^2$. However, the qhull algorithm of Barber, Dobkin, and Huhdanpaa [21] works for general dimension. The algorithm, among other things, is able to compute the Voronoi diagram for an arbitrary point set. We use qhull to compute the Voronoi diagrams that the clustering algorithm operates on.

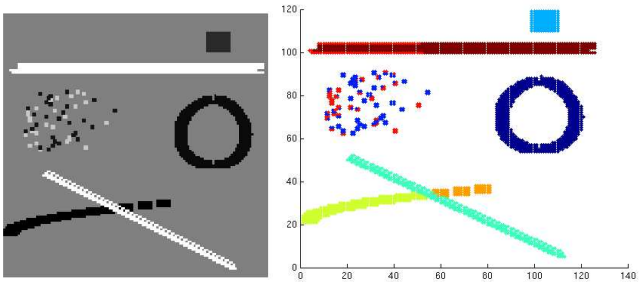

Fig. 2. An example data set with different geometric shapes (left) and the result of applying Voronoi clustering to it (right). The colors denote clusters. Intuitively, there are classification errors only in the cloud of points in the middle left and in the fringe points of the horizontal line on top.

3.2 The Method

Given a data set of n points $S = \{ s_1, \ldots, s_n \} \subseteq \mathbb{R}^d$ to be clustered we begin by constructing the Voronoi diagram for S. The computational complexity of Voronoi diagram construction in the general case is $\Theta(n \log n)$ [15], but requires only linear time in some restricted cases [20]. For the ease of illustration, let us consider the two-dimensional plane.

The algorithm is given as input parameter a threshold value max indicating the maximum volume allowed to a cell that still can be combined into an evolving cluster. We approximate local instance space density by cell volume, so only it matters in deciding whether the density is high enough for further combination into a cluster. Hence, individual clusters may grow to any size as long as the local density is sufficient. After all cells of at most volume max have been taken care of, we only have relatively large cells remaining.

In the Voronoi diagram for the data each point makes up its own cell. The volumes (areas in case of the plane) of the cells are computed (or approximated) next. The cell volumes are needed, because we go through the data points in the increasing order of the cell volume associated with the point. Our current implementation computes the exact cell area in the plane as the area of a polygon, but in higher dimensions d we simply approximate the cell to be a hyperball with volume $(\pi^{\lfloor n/2 \rfloor} r^d)/\Gamma(d/2 + 1)$, where $\Gamma(z + 1) = z!$ for integer values and the radius r is the average distance of cell corners from its center. This approximation is not very accurate in small dimensions, but it improves as the number of dimensions grows. Each cell is associated with a class label. We just assign increasing integer values as class labels.

The point having the smallest Voronoi cell is handled first. It has no known neighboring cells, so we just assign the class label 1 to this cell. The cell considered next might be a neighbor of the first cell, which would be detected by the two sharing a corner point. In that case, if the volume of the latter cell is below the threshold value max, they get combined together and assume the class label of the first cell. Otherwise, the second cell may turn out not to be a neighbor

Table 1. The Voronoi clustering algorithm.

```
Algorithm Voronoi_Clustering( S, max )

1. Construct the Voronoi diagram for the sample S = { s_1, ..., s_n }.
2. Approximate the Voronoi cell volumes and order the points
   accordingly. Without loss of generality, let the obtained order be
   s_1, ..., s_n.
3. For i = 1 to n do
   If the volume of the cell R_i associated with s_i is at most max
   then
       merge R_i with an adjacent cluster with the smallest class number
       if one exists, otherwise assign a new class number to the cell
   else
       assign R_i to the closest neighboring cluster
```

of the first one, in which case it gets a class label of its own. However, it is still possible that the two points belong to the same cluster and later get combined into the same cell due to having a common neighboring cell.

In the general case, a cell has many neighboring cells with different class labels and some that have not yet been labeled (those that are larger in volume than the cell at hand). The known neighbors are processed in the order of their size (i.e., in order of their class values). The cell under consideration is merged to its neighbor with the smallest class label, and its labeled neighbors also assume the same class. Cells are combined as long as their volume stays under the max value. If the max value is ever reached, there is no need to go through the remaining neighbor cells.

When all points have been considered once, all the cluster combinations have been executed. There is no need to iterate the process. However, we still need to do a simple post processing of the clustering obtained to make it sensible. Consider for instance the example of Figure 1. It is easy to see that the cell combining procedure described above will detect the six clusters with ease for a wide range of values for the threshold max. However, it is as clear that the outermost points of the clusters cannot get combined, because then the clusters would grow together.

In this and all other imaginable applications the following heuristics takes care of the points at the fringe of a cluster (of any shape). Center points with a cell of volume larger than the threshold max simply get combined to the closest neighboring class. Table 1 represents the algorithm in more compact form.

4 Discussion

After initialization the algorithm goes through the data points once doing only relatively simple calculations. Hence, the approach is efficient in practice. There are two $O(n \log n)$ phases in the algorithm: Voronoi diagram construction and

sorting of the Voronoi points according to their associated cell volumes. In our current implementation neither of these tasks, however, constitutes the most significant phase in time consumption. The most time intensive task in the algorithm is adjacency information processing because of the heavy data structures chosen. Our current implementation uses corner point representation to identify neighboring Voronoi cells. The number of corner points, unfortunately, grows with increasing number of instance space dimensions. Thus, this implementation becomes inefficient for domains with many attributes. It is our intention to turn to using Delaunay tesselations instead to record neighborhood information. The graph provides directly provides cell adjacency information as more efficient to handle.

In the Voronoi diagram of a point set the cell boundaries are by definition equally distant from the relevant points. Thus, the Voronoi diagram naturally gives raise to kind of *maximum margin separation* of the data. Our algorithm combines existing cells instead of creating artificial center points — like, e.g., k-means does. Hence, the maximal margins are maintained also in the clustered data. Since there are no reasons for placing the cluster border closer to one or other cluster, maximal margin clustering seems a natural choice. Moreover, results on boosting and kernel methods have shown maximal margin separation to be a good strategy [22].

One can view Voronoi clustering as a change of paradigm from the k-means, whose Euclidean distance minimization can be made autonomous with respect to the number of clusters only by optimizing a parameter that is too hard to learn [6, 12]. In the future we will study whether the one input parameter in our clustering algorithm can be deduced without user intervention.

5 Empirical Evaluation

To evaluate the proposed algorithm empirically, we test it on generated and real-world data. As the comparison algorithm we use the basic k-means algorithm randomly choosing the initial cluster centers (as implemented by Matlab). For k-means, the correct number of clusters is given as a parameter value. Moreover, to favor it more, we mostly use generated data that ought to, by default, suit k-means well. Let us describe the data sets:

2DEqualDiscs: In the first experiment 10 cluster centers were drawn randomly from the instance space and 150 points were uniformly drawn to each circular cluster of equal radius. The instance space height and width is 80 units and the radius of a cluster is 3 units. Because of the randomly drawn cluster centers, the clusters do sometimes partially overlap. To better account for the effects of overlapping clusters, we report the results of 5 independent runs in this setting, each consisting of 10 repetitions.

2DEqualDiscs — long distances: To see what is the effect of cluster overlapping, we changed the above situation by drawing the cluster centers from an instance space of height and width 150 units. Thus, the clusters are not as probable to overlap as in the previous experiment.

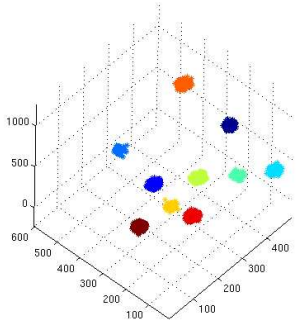

Fig. 3. An example of the 3DEqualBalls data: 10 randomly drawn clusters of equal size.

2DUnequalDiscs: This experiment is the same as the first one, except that this time the cluster sizes vary from one hundred to one thousand points (random selection, possible sizes multiples of one hundred).

3DEqualBalls: This experiment corresponds to the first one, except that the instance space now has three dimensions. See Figure 3 for an example situation.

3DUnequalBalls: As the second experiment, but in three dimensions.

3DObjects: Classification of different 3D objects which are shown as a 2D projection in Figure 2.

The results of our empirical comparison are given in Table 2. A general remark that can be made concerning these results is the fact that standard deviation is quite high in all the test domains. That partially accounts for the fact that in all the experiments Voronoi clustering was found to be statistically significantly more accurate than k-means clustering (as determined by t-test at 95% significance level).

k-means fares quite well in comparison when 2D spherical data is processed, however Voronoi clustering is clearly more accurate on even this data conforming to the underlying assumptions of k-means. When the cluster distances are increased, the relative advantage of Voronoi clustering becomes even clearer. The same effect can be observed when the clusters have unequal numbers of points in them. On 3D balls the superiority of Voronoi clustering is further enhanced. The fact that it is not more clearly better than k-means on the 3D object data is somewhat surprising and will need further analysis.

Our final experiment applies the two clustering algorithms to a text recognition task. Figure 4 depicts the original figure and results of k-means clustering into 4, 5, and 25 clusters. The obvious problem is that squared distance is minimized by dividing the input space into the requested number of more or less equal-sized clusters. Figure 5 shows the result of Voronoi clustering using two different max values. This time the hand-written words stand out better or worse, because there is no need to partition the instance space into equal-sized clusters.

Table 2. Average accuracies over 10 repetitions in the test domains.

Data Set		k-means	Voronoi
2DEqualDiscs	A	71.6 ±8.8	76.0 ±8.2
	B	77.4 ±8.8	79.1 ±14.1
	C	78.0 ±9.7	81.6 ±15.3
	D	74.6 ±13.4	81.7 ±12.4
	E	74.4 ±13.0	84.6 ±18.0
2DEqualDiscs — long distances	A	77.0 ±7.3	100.0 ±0.0
	B	71.6 ±12.5	98.9 ±3.3
	C	76.7 ±7.7	93.8 ±11.1
2DUnequalDiscs	A	76.8 ±9.2	90.7 ±5.8
	B	76.8 ±5.9	92.9 ±6.6
	C	75.0 ±11.0	90.0 ±8.6
	D	76.8 ±8.8	92.3 ±6.7
	E	83.3 ±10.5	93.1 ±8.4
3DEqualBalls	A	71.1 ±16.0	91.0 ±28.5
	B	70.6 ±12.7	99.0 ±3.1
	C	71.6 ±13.0	99.0 ±3.2
	D	75.1 ±9.8	95.0 ±10.8
	E	74.9 ±12.8	100.0 ±0.0
3DUnequalBalls	A	73.6 ±7.3	95.6 ±8.5
	B	79.6 ±7.4	94.9 ±9.5
	C	72.0 ±12.7	96.0 ±9.2
3DObjects	A	74.4 ±11.8	80.2 ±10.6
	B	78.1 ±16.1	81.6 ±7.9
	C	69.9 ±15.0	73.5 ±7.7

6 Conclusion and Further Work

Clustering algorithms are widely used and useful tools in unsupervised learning. The user is usually required to provide the value for at least one parameter of the algorithm. Ideally, she should be liberated from this task and the algorithm be able to autonomously adapt to the characteristic properties of the given data. This being too idealistic, the parameter values requested from the user should be as easy to determine as possible.

In this paper we proposed an autonomous clustering algorithm that initially builds the Voronoi diagram for the data supplied. It only requires the user to provide one, quite intuitive, parameter. Our empirical experiments demonstrated that Voronoi clustering is an effective algorithm over a wide range of data. It outperforms k-means clustering significantly on all our test domains.

There are many details and implementation issues that still need our attention. For instance, the current implementation suffers from being dependent on the number of corner points, which limits its efficiency in high-dimensional domains. Delaunay tesselation is expected to relive us from this inconvenience.

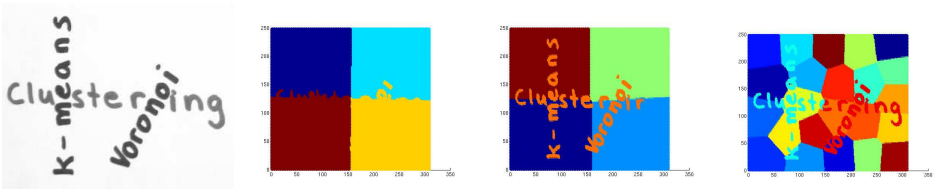

Fig. 4. The original text data and results by k-means with 4, 5, and 25 clusters.

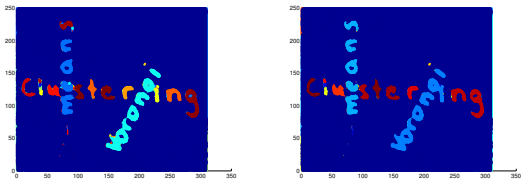

Fig. 5. Text recognition with Voronoi clustering using parameter value 50 and 70.

Acknowledgements

Work supported by Academy of Finland project "𝔸𝕃𝔼𝔸: Approximation and Learning Algorithms". In addition the work of the first author is financially supported by the graduate school of Tampere University of Technology.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31** (1999) 264–323
2. MacQueen, J.B.: On convergence of k-means and partitions with minimum average variance (abstract). Annals of Mathematical Statistics **36** (1965) 1084
3. Forgy, E.: Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. Biometrics **21** (1965) 768
4. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Wiley & Sons, New York, NY (1973)
5. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In Langley, P., ed.: Proc. 17th Intl. Conf. on Machine Learning, San Francisco, CA, Morgan Kaufmann (2000) 727–734
6. Hamerly, G., Elkan, C.: Learning the k in k-means. In Thrun, S., Saul, L.K., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA (2004) 281–288
7. Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Intl. Conf. on Very Large Data Bases, San Francisco, CA, Morgan Kaufmann (1994) 144–155
8. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, New York, NY, ACM Press (1995) 103–114

9. Guha, S., Rastogi, R., Shim, K.: Cure: An efficient clustering algorithm for large datasets. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, New York, NY, ACM Press (1998) 73–84
10. Moore, A.W.: The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In Boutilier, C., Goldszmidt, M., eds.: Proc. 16th Conf. on Uncertainty in Artificial Intelligence, San Francisco, CA, Morgan Kaufmann (2000) 397–405
11. Elkan, C.: Using the triangle inequality to accelerate k-means. In Fawcett, T., Mishra, N., eds.: Proc. 20th Intl. Conf. on Machine Learning, Menlo Park, AAAI Press (2003) 147–153
12. Elomaa, T., Koivistoinen, H.: On autonomous k-means clustering. In Hacid, M.S., Murray, N., Raś, Z.W., Tsumoto, S., eds.: Foundations of Intelligent Systems, Proc. Fifteenth International Symposium, ISMIS'05. Volume 3488 of Lecture Notes in Artificial Intelligence., Berlin Heidelberg New York, Springer (2005) 228–236
13. Kumar, A., Sabharwal, Y., Sen, S.: A simple linear time $(1 + \varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In: Proc. 45th Annual IEEE Symposium on Foundations on Computer Science, Los Alamitos, CA, IEEE Press (2004) 454–462
14. Har-Peled, S., Mazumdar, S.: On coresets for k-means and k-median clustering. In: Proc. 36th Annual ACM Symposium on Theory of Computing, New York, NY, ACM Press (2004) 291–300
15. Aurenhammer, F., Klein, R.: Voronoi diagrams. In Sack, J., Urrutia, G., eds.: Handbook of Computational Geometry. North-Holland, Amsterdam, The Netherlands (2000) 201–290
16. Schreiber, T.: A Voronoi diagram based adaptive k-means-type clustering algorithm for multidimensional weighted data. In Bieri, H., Noltemeier, H., eds.: Computational Geometry — Methods, Algorithms and Applications. Volume 553 of Lecture Notes in Computer Science., Berlin Heidelberg New York, Springer (1991) 265–275
17. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Menlo Park, CA, AAAI Press (1996) 226–231
18. Xu, X., Ester, M., Kriegel, H.P., Sander, J.: A distribution-based clustering algorithm for mining in large spatial databases. In: Proceedings of the Fourteenth International Conference on Data Engineering, Los Alamitos, CA, IEEE Computer Society Press (1998) 324–331
19. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In Agrawal, R., Stolorz, P.E., Piatetsky-Shapiro, G., eds.: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, Menlo Park, CA, AAAI Press (1998) 58–65
20. Aggarwal, A., Guibas, L.J., Saxe, J.B., Shor, P.W.: A linear-time algorithm for computing the voronoi diagram of a convex polygon. Discrete & Computational Geometry **4** (1989) 591–604
21. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.T.: The Quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software **22** (1996) 469–483
22. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge, UK (2004)