Pagallo, G., & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine learning*, 5, 71–99.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. Michalski, J. Carbonnel, & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*, (pp. 464–482). Palo Alto, CA: Tioga.

Razzak, M. A., Hassan, T., & Pettipher, R. (1984). Extran-7: A Fortran-based software package for building expert systems. In M. A. Bramer (Ed.), *Research and development in expert systems* (pp. 23–30). Cambridge: Cambridge University Press.

Shapiro, A., & Niblett, T. (1982). Automatic induction of classification rules for a chess endgame. In M. R. B. Clarke (Ed.), *Advances in computer chess* (Vol. 3, pp. 73–91). Pergamon: Oxford.

Shapiro, A. D. (1987). *Structured Induction in expert systems*. Wokingham: Turing Institute Press with Addison Wesley.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Zhang, J., & Honavar, V. (2003). Learning decision tree classifiers from attribute value taxonomies and partially specified data. In *ICML-2003: Proceedings of the twentieth international conference on machine learning*, Menlo Park, CA: AAAI Press.

Zheng, Z. (1995). Constructing nominal X-of-N attributes. In *Proceedings of the fourteenth International joint conference on artificial intelligence (IJCAI, 95)* (pp. 1064–1070). Los Altos, CA: Morgan Kaufmann.

Zupan, B., Bohanec, M., Demsar, J., & Bratko, I. (1999). Learning by discovering concept hierarchies. *Artificial Intelligence, 109*, 211–242.

Subgroup Discovery

Definition

Subgroup discovery (Klösgen, 1996; Lavrač, Kavšek, Flach, & Todorovski, 2004) is an area of ►supervised descriptive rule induction. The subgroup discovery task is defined as given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically "most interesting," for example, are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest.

Recommended Reading

Klösgen, W., (1996). Explora: A multipattern and multistrategy discovery assistant. In *Advances in knowledge discovery and data mining* (pp. 249–271). Cambridge: MIT Press.

Lavrač, N., Kavšek, B., Flach, P. A., & Todorovski, L. (2004). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research, 5*, 153–188.

Sublinear Clustering

Artur Czumaj[1], Christian Sohler[2]
[1]University of Warwick, Coventry, UK,
[2]University of Paderborn,
Paderborn, Germany

Definition

Sublinear clustering describes the process of clustering a given set of input objects using only a small subset of the input set, which is typically selected by a random process. A solution computed by a sublinear clustering algorithm is an implicit description of the clustering (rather than a partition of the input objects), for example in the form of cluster centers. Sublinear clustering is usually applied when the input set is too large to be processed with standard clustering algorithms.

Motivation and Background

►Clustering is the process of partitioning a set of objects into subsets of similar objects. In machine learning, it is, for example, used in unsupervised learning to fit input data to a density model. In many modern applications of clustering, the input sets consist of billions of objects to be clustered. Typical examples include web search, analysis of web traffic, and spam detection. Therefore, even though many relatively efficient clustering algorithms are known, they are often too slow to cluster such huge inputs.

Since in some applications it is even not possible to cluster the entire input set, a new approach is needed to cope with very large data sets. The approach used in many different areas of science and engineering in this context is to *sample* a subset of the data and to analyze this sample instead of the whole data set. This is also the underlying method used in sublinear clustering. The main challenge and innovation in this area lies in the qualitative analysis of random sampling (in the form of approximation guarantees) and the design of *non uniform sampling* strategies that approximate the input set provably better than *uniform random sampling*.

Structure of the Learning System

In sublinear clustering a large input set of objects is to be partitioned into subsets of similar objects. Usually, this

input is a point set P either in the Euclidean space or in the metric space. The clustering problem is specified by an objective function that determines the quality or cost of every possible clustering. The goal is to find a clustering of minimum cost/maximum quality. For example, given a set P of points in the Euclidean space the objective of ▶*k*-means clustering is to find a set C of k centers that minimizes $\sum_{p\in P}(d(p,C))^2$, where $d(p,C)$ denotes the distance of p to the nearest center from C. Since usually the clustering problems are computationally hard ($\mathcal{NP}$-hard), one typically considers *approximate* solutions: instead of finding a clustering that optimizes the cost of the solution, one aims at a solution whose cost is close to the optimal one.

In sublinear algorithms a solution is computed for a small representative subset of objects, for example a random sample. The solution is represented implicitly, for example, in the form of cluster centers and it can be easily extended to the whole input point set. The quality of the output is analyzed *with respect to the original point set*. The challenge is to *design and analyze fast (sublinear-time) algorithms* that select a subset of objects that very well represent the entire input, such that the solution computed for this subset will also be a good solution for the original point set. This can be achieved by uniform and non uniform *random sampling* and the computation of *core-sets*, i.e., small weighted subsets of the input that approximate the input with respect to a clustering objective function.

Theory/Solution

Clustering Problems

For any point p and any set Q in a metric space (X,d), let $d(p,Q) = \min_{q\in Q} d(p,q)$. A point set P is *weighted* if there is a function w assigning a positive weight to each point in P.

Radius *k*-Clustering: Given a weighted set P of points in a metric space (X,d), find a set $C \subseteq P$ of k centers minimizing $\max_{p\in P} d(p,C)$.

Diameter *k*-Clustering: Given a weighted set P of points in a metric space (X,d), find a partition of P into k subsets $P_1,\ldots,P_k$, such that $\max_{i=1}^{k} \max_{p,q\in P_i} d(p,q)$ is minimized.

***k*-Median:** Given a weighted set P of points in a metric space (X,d), find a set $C \subseteq P$ of k centers that minimizes $median(P,C) = \sum_{p\in P} w(p)\cdot d(p,C)$.

***k*-Means:** Given a weighted set of points P in a metric space (X,d), find a set $C \subseteq P$ of k centers that minimizes $mean(P,C) = \sum_{p\in P} w(p)\cdot (d(p,C))^2$.

Random Sampling and Sublinear-Time Algorithms

Random sampling is perhaps the most natural approach to design sublinear-time algorithms for clustering. For the input set P, random sampling algorithm follows the following scheme:

1. Pick a random sample S of points
2. Run an algorithm (possibly an approximation one) for (given kind of) clustering for S
3. Return the clustering induced by the solution for S

The running time and the quality of this algorithm depends on the size of the random sample S and of the running time and the quality of the algorithm for clustering of S. Because almost all interesting clustering problems are computationally intractable ($\mathcal{NP}$-hard), usually the second step of the sampling scheme uses an approximation algorithm. (An algorithm for a minimization problem is called a λ-approximation if it always returns a solution whose cost is at most λ times the optimum.)

While random sampling approach gives very simple algorithms, depending on the clustering problem at hand, it often finds a clustering of poor quality and it is usually difficult to analyze. Indeed, it is easy to see that random sampling has some serious limitations to obtain clustering of good quality. Even the standard assumption that the input is in metric space is not sufficient to obtain good quality of clustering because of the small clusters which are "hidden" for random sampling approach. (If there is a cluster of size $o(|P|/|S|)$ then with high probability the random sample set S will contain no point from that cluster.) Therefore, another important parameters used in the analysis is the *diameter* of the metric space Δ, which is $\Delta = \max_{p,q\in P} d(p,q)$.

Quality of Uniformly Random Sampling: The quality of random sampling for three basic clustering problems

(k-means, k-median, and min-sum k-clustering) have been analyzed in Ben-David (2004), Czumaj and Sohler (2007), and Mishra, Oblinger, and Pitt (2001). In these papers, generic methods of analyzing uniform random sampling have been obtained. These results assume that the input point sets are in a metric space and are unweighted (i.e., when the weight function w is always 1).

Theorem 1 *Let $\epsilon > 0$ be an approximation parameter. Suppose that an α-approximation algorithm for the k-median problem in a metric space is used in step* (2) *of the uniform sampling, where $\alpha \geq 1$* (Ben-David 2004; Czumaj & Sohler 2007, Mishra et al., 2001). *If we choose S to be of size at least*

$$c\alpha\left(k + \frac{\Delta}{\epsilon}\left(\alpha + k\ln(k\Delta\alpha/\epsilon)\right)\right)$$

for an appropriate constant c, then the uniform sampling algorithm returns a clustering C^* (of S) such that with probability at least 0.99 *the normalized cost of clustering for S satisfies*

$$\frac{median(S, C^*)}{|S|} \leq \frac{2(\alpha + 0.01)OPT(P)}{|P|} + \epsilon,$$

where $OPT(S) = \min_C median(P, C)$ is the minimum cost of a solution for k-median for P.

Similar results can be shown for the k-means problem, and also for min-sum k-clustering (cf. Czumaj & Sohler, 2007). For example, for k-means, with a sample S of size at least $c\alpha\left(k + (\Delta^2/\epsilon)\left(\alpha + k\ln(k\Delta^2\alpha/\epsilon)\right)\right)$, with probability at least 0.99 the normalized cost of k-means clustering for S satisfy

$$\frac{mean(S, C^*)}{|S|^2} \leq \frac{4(\alpha + 0.01)OPT(P)}{|P|^2} + \epsilon,$$

where $OPT(S) = \min_C mean(P, C)$ is the minimum cost of a solution for k-means for P.

Improvements of these bounds for the case when the input consists of points in Euclidean space are also discussed in Mishra et al. (2001), Czumaj and Sohler (2007) discuss also . For example, for k-median, if one takes S of size at least $c\alpha\,(k + \Delta k\ln(\Delta/\epsilon)/\epsilon)$, then with probability at least 0.99 the normalized cost of k-median clustering for S satisfies

$$\frac{median(S, C^*)}{|S|} \leq \frac{(\alpha + 0.01)OPT(P)}{|P|} + \epsilon\ ,$$

and hence the approximation ratio is better than that in Theorem 1 by factor of 2.

The results stated in Czumaj and Sohler (2007) allow to parameterize the constants 0.99 and 0.01 in the claims above.

Property Testing of the Quality of Clustering: Since most of the clustering problems are computationally quite expensive, in some situations it might be interesting not to find a clustering (or its succinct representation), but just to test if the input set has a good clustering.

Alon, Dar, Parnas, and Ron (2003) introduced the notion of approximate testing of good clustering. A point set P is *c-clusterable* if it has a clustering of the cost at most c, that is, $OPT(P) \leq c$. To formalize the notion of having no good clustering, one says a point set is *ϵ-far from $(1 + \beta)c$-clusterable*, if more than an ϵ-fraction of the points in P must be removed (or moved) to make the input set $(1 + \beta)c$-clusterable. With these definitions, the goal is to design fast algorithms that accept the input point sets P, which are c-clusterable, and reject with probability at least 2/3 inputs are ϵ-far from $(1 + \beta)c$-clusterable. If neither holds, then the algorithms may either accept or reject. The bounds for the testing algorithms are phrased in terms of *sample complexity*, that is, the number of sampled input points which are considered by the algorithm (e.g., by using random sampling).

Alon et al. (2003) consider two classical clustering problems in this setting: radius and diameter k-clusterings. If the inputs are drawn from an arbitrary metric space, then they show that to distinguish between input points sets that are c-clusterable and are ϵ-far from $(1 + \beta)c$-clusterable with $\beta < 1$, the sample complexity must be at least $\Omega(\sqrt{|P|/\epsilon})$. However, to distinguish between inputs that are c-clusterable and are ϵ-far from $2c$-clusterable, the sample complexity is only $O(\sqrt{k/\epsilon})$.

A more interesting situation is for the input points drawn from the Euclidean d-dimensional space. In that case, even a constant-time algorithms are possible.

Theorem 2 *For the radius k-clustering, one can distinguish between points sets in R^d that are c-clusterable from those ϵ-far from c-clusterable with the sample complexity $\widetilde{O}(dk/\epsilon)$* (Alon et al., 2003) *(The $\widetilde{O}$-notation ignores logarithms in the largest occurrence of a variable; $\widetilde{O}(f(n)) = O(f(n) \cdot (\log f(n))^{O(1)})$.)*

Furthermore, for any $\beta > 0$, one can distinguish between points sets in R^d that are c-clusterable from those ϵ-far from $(1+\beta)$c-clusterable with the sample complexity $\widetilde{O}(k^2/(\beta^2\epsilon))$.

Theorem 3 *For the diameter k-clustering, one can distinguish between points sets in R^d that are c-clusterable from those ϵ-far from $(1+\beta)$c-clusterable with the sample complexity $\widetilde{O}(k^2 d(2/\beta)^{2d}/\epsilon)$* (Alon et al., 2003).

Core-Sets: Sublinear Space Representations with Applications

A *core-set* is a small weighted set of points S that provably approximates another point set P with respect to a given clustering problem (Bădoiu, Har-Peled, & Indyk, 2002). The precise definition of a core-set depends on the clustering objective function and the notion of approximation. For example, a coreset for the k-median problem can be defined as follows:

Definition 4 *A weighted point set S is called* ϵ-coreset *for a point set P for the k-median problem, if for every set C of k centers, we have $(1-\epsilon) \cdot median(P,C) \leq median(S,C) \leq (1+\epsilon) \cdot median(P,C)$* (Har-Peled & Mazumdar, 2004).

A core-set as defined above is also sometimes called a *strong* core-set, because the cost of the objective function is approximately preserved for any set of cluster centers. In some cases it can be helpful to only require a weaker notion of approximation. For example, for some applications it is sufficient that the cost is preserved for a certain discrete set of candidate solutions. Such a core-set is usually called a *weak* core-set. In high-dimensional applications it is sometimes sufficient that the solution is contained in the low-dimensional subspace spanned by the core-set points.

Constructing a Core-Set: There are deterministic and randomized constructions for core-sets of an unweighted set P of n points in the R^d. Deterministic core-set constructions are usually based on the *movement paradigm*. The input points are moved to a set of few locations such that the overall movement is at most ϵ times the cost of an optimal solution. Then the set of points at the same location are replaced by a single point whose weight equals the number of these points. Since for the k-median problem the cost of any solution changes by at most the overall movement, this immediately implies that the constructed weighted set is an ϵ-core-set. For other similar problems more involved arguments can be used to prove the core-set property. Based on the movement paradigm, for k-median a core-set of size $O(k \log n/\epsilon^d)$ can be constructed efficiently (Har-Peled & Mazumdar, 2004).

Randomized core-set constructions are based on non uniform sampling. The challenge is to define a randomized process for which the resulting weighted point set is with high probability a core-set. Most randomized coreset constructions first compute a bi-criteria approximation C'. Then every point is sampled with probability proportional to its distance to the nearest center of C'. A point q is assigned a weight proportional to $1/p_q$, where p_q is the probability that p is sampled. For every fixed set C of k centers, the resulting sample is an unbiased estimator for $median(P,C)$. If the sample set is large enough, it approximates the cost of every possible set of k centers within a factor of $(1 \pm \epsilon)$. The above approach can be used to obtain a weak core-set of size independent of the size of the input point set and the dimension of the input space (Feldman, Monemizadeh, & Sohler, 2007). A related construction has been previously used to obtain a strong core-set of size $\widetilde{O}(k^2 \cdot d \cdot \log n/\epsilon^2)$. Both constructions have constant success probability that can be improved by increasing the size of the core-set.

Applications Core-sets have been used to obtain improved approximation algorithms for different variants of clustering problems. Since the core-sets are of sublinear size and they can be constructed in sublinear time, they can be used to obtain sublinear-time approximation algorithms for a number of clustering problems.

Another important application is clustering of data streams. A data stream is a long sequence of data that typically does not fit into main memory, for example, a sequence of packet headers in IP traffic monitoring. To analyze data streams we need

algorithms that extract information from a stream without storing all of the observed data. Therefore, in the data streaming model algorithms are required to use $\log^{O(1)} n$ bits of memory. For core-sets, a simple but rather general technique is known, which turns a given construction of a strong core-set into a data streaming algorithm, i.e., an algorithm that processes the input points sequentially and uses only $\log^{O(1)}$ space (for constant k and ϵ) and computes a $(1+\epsilon)$-approximation for the optimal set of centers of the k-median clustering (Har-Peled & Mazumdar, 2004). Core-sets can also be used to improve the running time and stability of clustering algorithms like the k-means algorithm (Frahling & Sohler, 2006).

Recommended Reading

Alon, N., Dar, S., Parnas, M., & Ron, D. (2003). Testing of clustering. *SIAM Journal on Discrete Mathematics, 16*(3), 393–417.

Bădoiu, M., Har-Peled, S., & Indyk, P. (2002). Approximate clustering via core-sets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, (pp. 250–257).

Ben-David, S. (2004). A framework for statistical clustering with a constant time approximation algorithms for k-median clustering. In *Proceedings of the 17th Annual Conference on Learning Theory (COLT)*, (pp. 415–426).

Chen, K. (2006). On k-median clustering in high dimensions. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, (pp. 1177–1185).

Czumaj, A., & Sohler, C. (2007). Sublinear-time approximation for clustering via random sampling. *Random Structures & Algorithms, 30*(1–2), 226–256.

Feldman, D., Monemizadeh, M., & Sohler, C. (2007). A PTAS for k-means clustering based on weak coresets. In *Proceedings of the 23rd Annual ACM Symposium on Computational Geometry (SoCG)*, (pp. 11–18).

Frahling, G., & Sohler, C. (2006). A fast k-means implementation using coresets. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SoCG)*, (pp. 135–143).

Har-Peled, S. & Kushal, A. (2005). Smaller coresets for k-median and k-means clustering. In *Proceedings of the 21st Annual ACM Symposium on Computational Geometry (SoCG)*, (pp. 126–134).

Har-Peled, S., & Mazumdar, S. (2004). On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, (pp. 291–300).

Meyerson, A., O'Callaghan, L., & Plotkin S.(July 2004). A k-median algorithm with running time independent of data size. *Machine Learning, 56*(1–3), (pp. 61–87).

Mishra, N., Oblinger, D., & Pitt, L. (2001). Sublinear time approximate clustering. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, (pp. 439–447).

Subspace Clustering

►Projective Clustering

Subsumption

Claude Sammut
The University of New South Wales,
Sydney NSW, Australia

Subsumption provides a syntactic notion of generality. Generality can simply be defined in terms of the cover of a concept. That is, a concept, C, is more general than a concept, C', if C covers at least as many examples as C' (see ►Learning as Search). However, this does not tell us how to determine, from their syntax, if one sentence in a concept description language is more general than another. When we define a *subsumption* relation for a language, we provide a syntactic analogue of generality (Lavrač & Džeroski, 1994). For example, *θ-subsumption* (Plotkin, 1970) is the basis for constructing generalization lattices in ►inductive logic programming (Shapiro, 1981). See ►Generality of Logic for a definition of *θ-subsumption*. An example of defining a subsumption relation for a domain specific language is in the LEX program (Mitchell, Utgoff, & Banerji, 1983), where an ordering on mathematical expressions is given.

Cross References

►Generalization
►Induction
►Learning as Search
►Logic of Generality

Recommended Reading

Lavrač, N., & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and applications*. Chichester: Ellis Horwood.

Mitchell, T. M., Utgoff, P. E., & Banerji, R. B. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. Palo Alto: Tioga.

Plotkin, G. D. (1970). A note on inductive generalization. In B. Meltzer & D. Michie (Eds.), *Machine intelligence* (Vol. 5, pp. 153–163). Edinburgh University Press.