Approximation Algorithms for the Class Cover Problem

Adam Cannon and Lenore Cowen *
Department of Mathematical Sciences
Johns Hopkins University
Baltimore, MD 21218

Abstract

We introduce the class cover problem, a variant of disk cover with forbidden regions, with applications to classification and facility location problems. We prove similar hardness results to disk cover. We then present a polynomial-time approximation algorithm for class cover that performs within a $\ln n + 1$ factor of optimal, which is nearly tight under standard hardness assumptions. In the special case that the points lie in a d-dimensional space with Euclidean norm, for some fixed constant d, we obtain a polynomial time approximation scheme.

1 Introduction

Many different problems from the realm of classification and clustering, have been formulated as optimization problems. Examples of problems that have been recently studied include Schulman's new work [11] and others (see, for example, Bradley, Fayyad, and Mangasarian's survey [2]). Formulations such as [11] model what is essentially an unsupervised learning problem, where the clusters are determined entirely by optimization criteria that endeavor to capture how tightly the points clump together.

By contrast, this paper considers a clustering problem modeled by a much simpler, *supervised* learning problem: in fact, the simplest classification problem there is. We consider points of two types (for generalizations to multiple classes see Section 5), class one points (which we will refer to henceforth as *blue* points, or positive examples) and class two points (which we will refer to henceforth as *red* points, or negative examples). The goal is to pick a set of blue *centers* that separate the blue points from the red points in the following way: the distance from any of the blue points to its closest blue center is smaller than the distance from any of the red points to its closest blue center. This is equivalent to covering the blue points with a set of blue-centered balls of equal radius, such that no red points lie in these balls.

It is easy to observe that no matter what the set of blue points and red points are, such a separating set of blue centers always exists: simply pick all blue points as centers. This trivial separating set has every blue point at distance 0. However, it is not a very interesting separating set. The key question, is how many blue points are really needed: in particular, we seek a minimum cardinality set of blue centers that satisfies the above definition. Thus we define the *class cover* problem as follows: Let B be the set of points in class one, and R be the set of points in class two, with $|R| + |B| = n$. Then the class cover problem is,

Minimize k

*Supported in part by Office of Naval Research grant N00014-96-1-0829 and Defense Advanced Research Projects Agency as administered by the Air Force Office of Scientific Research under contract DOD F49620-99-1-0213

$$\text{subject to: } \max_{v \in B} d(v, S) < \min_{v \in R} d(v, S)$$

$$\text{where } S \subseteq B,\ |S| = k$$

Here the point to set distance $d(v, S)$ is defined as $\min_{s \in S} d(v, s)$. [1]

We suggest looking at the k for which a set of colored points has a solution to the class cover problem as some sort of measure of how well separated the two classes are: this was the point of view taken in the paper [6]. Our investigation of the class cover problem was originally motivated by our Approximate Distance Classification (see Cowen and Priebe [6] and Cannon, Cowen, and Priebe [4]) method for the classification of high dimensional data. In fact, a set of points whose solution to the class cover problem has size $\leq k$ are precisely the "k-separable in one-dimension" sets introduced in [6], and developed further as part of the ADC method. We discuss these connections further at the end of the paper.

The class cover problem also has a facility location interpretation. One can imagine the blue points as preferred or premium customers. Therefore, the objective becomes to locate the facilities in a way that puts the farthest preferred (blue) customer closer than the nearest regular (red) customer.

Our results. Our main result follows from a polynomial time reduction of the class cover problem to the special case of set cover called the *disk cover* problem, introduced by Hochbaum and Maass [7].This leads to a $\ln n + 1$ times optimal approximation algorithm for the class cover problem in general, that runs in cubic time. It also leads to a $(1 + \epsilon)$-times optimal polynomial-time approximation algorithm in the special case that the points lie in $\Re^d$ for some fixed d, but with a running time that's doubly exponential in $1/\epsilon$ and d. We remark that in the special case for achieving a constant factor approximation when the points lie in $\Re^2$, an approximation algorithm for disk cover of Brönnimann and Goodrich [3] can be used in place of the disk cover algorithm of [7] as a subroutine, to reduce the running time for class cover in this case to $O(n^5 \log n)$.

First we show the class cover problem is NP-Complete, and that approximating the class cover problem within a factor better than $O(\log n)$ is not possible unless NP $\subseteq \tilde{P}$. Then we give the transformation of our problem to a quadratic number of instances of the disk cover problem. The running times cited above then follow immediately from the greedy approximation algorithm for general set cover (including disk cover), and the better approximation factors for disk cover in fixed dimension cited above. Finally, we discuss applications to building classifiers, conclusions, and open problems.

2 Hardness results

Clearly the decision problem "Is there a solution to the class cover problem of value $\leq k$?" is in NP. It remains to show that the class cover problem is NP-hard. We show not only that it is NP-hard but that the dominating set problem is essentially a special case of the class cover problem. Thus hardness of approximation results for dominating set are inherited by class cover.

Recall the dominating set problem: Given a graph $G(V, E)$ to determine whether there is a set of vertices DS of size at most k such that all vertices are either in DS or adjacent to some element of DS. The optimization version of dominating set is to minimize the size of DS, that is, to minimize k. Now, given an instance of dominating set $G(V, E)$ we may construct an instance I of class cover in the following

[1] We remark that we are using the notation $d(v, s)$, which usually implies a metric, and in general most of our applications will be in a metric space, however, all of our definitions can also be made, and our general result also holds using a semi-metric, i.e. a cost structure $c(v, s)$ that does not satisfy the triangle inequality.

way. Create a blue point for each vertex $v \in V$. Create a single new red point r. Assign the distance between two blue points to be 1 if their corresponding vertices in G had an edge between them, and 2 otherwise. Assign the red point to be at distance 1.5 from all the blue vertices.

Proposition 2.1 *Any solution to the derived class cover instance I of size k corresponds directly to a dominating set of G of size k. Thus a polynomial time algorithm for class cover would imply a polynomial time algorithm for dominating set, and a poly-time approximation algorithm for class cover implies a poly-time approximation algorithm for dominating set with the same approximation factor.* □

The hardness results now follow from the fact that dominating set is a "class II" problem, according to the hardness results classification scheme (as described by Arora and Lund (see [1, 9])) and thus we have the following corollary:

Corollary 2.2 *There is no polynomial time approximation algorithm for class cover that approximates the optimal solution to class cover within a factor better than $(1-\delta)\log n$ unless $NP \subseteq \tilde{P}$.* □

3 An $\ln n + 1$-approximation algorithm

We first present the algorithm `Fixed-Radius`, which takes, in addition to the instance of class cover, an additional parameter $\alpha \in \Re$. We show that when we correctly "guess" $\alpha = \min_{v \in R} d(v, S^*)$, where S^* is an optimal solution to class cover, that `Fixed-Radius` returns a feasible solution to the class cover instance, of size at most $O(|S^*|(\ln+1))$, where $|S^*|$ is the size of the optimal solution to class cover. We then present the algorithm `Bubble-Grow`, which generates a polynomial number of guesses for α, and is guaranteed to call `Fixed-Radius` with the right value for α.

Algorithm Fixed-Radius

Input: An instance (B, R) of blue and red points with $|B| + |R| = n$ and a cost measure $c(i, j)$ defined on all pairs of points $i, j \in B \cup R$. An additional parameter $\alpha \in \Re$.

Output: S with $S \subseteq B$, or "α is infeasible".

1. For each $v \in B$,
   Let $B_v = \{u \in B \cup R | c(u, v) < \alpha\}$
2. For each v, if B_v contains any red points, place v in set X.
3. Output the collection of sets (balls) $\mathcal{C} = \bigcup_{v \in B \setminus X} B_v$.
4. If the union of all sets in $\mathcal{C}$ does not capture all points in B, then output "α is infeasible" and STOP.
5. Using the greedy approximation to set cover compute D, a collection of sets that cover all points in B from $\mathcal{C}$ of cardinality at most $|D^*|(\ln n + 1)$, where D^* is a minimum cardinality subset of sets $B_v \in \mathcal{C}$ whose union captures all points in B.
6. Output the set S whose elements are the centers of the balls belonging to D.

We make the following claims:

Proposition 3.1 *If algorithm* `Fixed-Radius` *returns a set S, then S is a feasible solution to the class cover problem.*

Proof: The set cover approximation in Step 5 does not consider balls that contain any red points (they are eliminated in Step 3). Since the elements of S are the centers of the balls in D output by the set cover approximation algorithm, there can be no red points within distance (cost) α of any point in S. Since D covers the blue points, every blue point must be within distance (cost) α of some element of S. Thus $\max_{v \in B} d(v, S) \leq \alpha < \min_{v \in R} d(v, S)$ and S is feasible. □

Proposition 3.2 *If S^*, with $|S^*| = k$ is an optimal solution to the class cover problem, then on input $\alpha = min_{v \in R} d(v, S^*)$, algorithm* `Fixed-Radius` *produces a set S with $|S| \leq k(\ln n + 1)$.*

Proof: It is sufficient to show that when $\alpha = min_{v \in R} d(v, S^*)$ is input to `Fixed-Radius`, $|S^*| = |D^*|$, where D^* is the optimal solution to the set cover problem used in Step 5. Since we can build a feasible solution S from D^* with $|S| = |D^*|$, we have by optimality of S^*, that $|S^*| \leq |D^*|$. Suppose $|S^*| = k < |D^*|$, then we grow balls of radius α around the k elements of S^*. Since there is a red point at distance (cost) α from some element of S^* and since S^* is feasible, all of the blue points must be covered by the balls. Therefore these balls are a cover of size k in the set cover problem of Step 5, violating optimality of D^*. Hence $|S^*| = |D^*|$ as desired. □

3.1 The main algorithm

Based on the propositions of the previous section, it is clear we could produce an $O(\log n)$-approximation to class cover, if we could run algorithm `Fixed-Radius` for all $\alpha \in \Re$, $0 < \alpha \leq \max_{i,j \in B \cup R} c(i, j)$, and output the minimum-sized feasible solution.

In particular, we visualize all blue points growing simultaneously balls of radius α; initially $\alpha = 0$. If we imagine increasing α continuously, we achieve blue-centered balls of radius α: if these balls capture any red (forbidden) point, they "pop" (i.e. are removed from the collection $\mathcal{C}$ of active sets.) Algorithm `Fixed-Radius` runs on the "snapshot" for each fixed α; clearly if α is increased continuously there are too many snapshots and this is not polynomial time. Notice that some sort of binary search on α won't work either, because the behavior of the cover is not monotonic in α. However, we notice that the sets in $\mathcal{C}$ do not change continuously as α increases, rather the sets that comprise $\mathcal{C}$ only change each time a new point is captured by an existing ball. The number of radii α for which this occurs can be upper bounded by n^2, the maximum number of different pairwise distances; thus checking only n^2 different α (and choosing the solution of minimum cardinality among them) can be shown to suffice. In fact, a little thought shows we can reduce the number of α considered to linear, because it only *helps* the cover when more blue points are captured by a blue-centered ball. Thus for each blue point, a particular α of interest is the one that leads to an open ball with radius equal to its closest red point (i.e. the value for alpha just before the ball "pops"). This is the idea behind our main algorithm, `Bubble-Grow`.

Algorithm Bubble-Grow

Input: An instance (B, R) of n_b blue and n_r red points with $n_b + n_r = n$ and a cost measure $c(i, j)$ defined on all pairs of points $i, j \in B \cup R$.

Output: S with $S \subseteq B$.

1. For each $v_i \in B$,

- Create associated list $L_i = (l_i^1, \ldots l_i^n)$, consisting of the elements x, stored as $(x, c(x, v_i)$, and sorted by increasing $c(x, v_i)$.
- $\alpha_i \longleftarrow min_{x \in R} c(x, v_i)$
- $h_i \longleftarrow 1$
- $t_i \longleftarrow 1$

2. Sort the lists L_i, according to increasing α_i, let $L_{(i)} = (l_{(i)}^1, \ldots, l_{(i)}^n)$ denote the ith list under this ordering.

3. For $i = 1$ to n_b DO

   $B_{(i-1)} \longleftarrow \emptyset$

   For $j = i$ to n_b

   $t_j \longleftarrow$ { the largest index k, so that $l_{(j)}^k$ has $c(x, v_{(j)}) < \alpha_{(i)}$}

   $B_{(j)} \longleftarrow B_{(j)} \cup \{l_{(j)}^{h_j} \ldots l_{(j)}^{t_j}\}$

   $h_j \longleftarrow t_j + 1$

   Output $\mathcal{C} = \{B_{(i)} : 1 \leq i \leq n_b\}$

   If the union of all sets in $\mathcal{C}$ covers all points in B

   Use the greedy approximation to set cover to output D_i, a collection of sets (balls) that capture all of the points in B.

4. Let D' be the minimum cardinality solution from among the sets D_i output in step 3. Return the set S whose elements are the centers of the balls belonging to D'.

Remark: We note that here the `Fixed-Radius` routine is integrated into step three and not actually called as a separate routine.

Proposition 3.3 *Suppose that the optimal solution to class cover has cardinality k. Then the above algorithm finds a class cover of cardinality at most $k(\ln n + 1)$.*

Proof: Let α^* be the distance to the nearest red vertex in a minimum size separating set, D^*. Since the list of α_i computed in Step 2 includes all minimum blue-to-red distances, $\alpha^* = \alpha_i$ for some $1 \leq i \leq n_b$. Consequently, by the argument in Propositions 3.1 and 3.2, one of the set cover $\ln n + 1$-approximate solutions computed in Step 3 of the algorithm must correspond to a $\ln n + 1$-approximation of the cardinality of D^*. Since the algorithm chooses the globally minimum size set cover solution to build its ultimate class cover solution, it must return a class separating set with cardinality less than or equal to $k(\ln n + 1)$. The algorithm always outputs some cover since the first time through the loop at step 3 produces at worst the trivial cover (all of B) as a solution. □

Proposition 3.4 *Algorithm* `Bubble-Grow` *with input n points with n_b blue points, runs in $O(n_b^3 + n_b n \log n) = O(n^3)$ time.*

Proof: The amount of time needed for step 1 is dominated by the amount of time to sort n_b lists of n element lists, for $O(n_b n \log n)$ time. The second step just sorts n indices, and takes $O(n \log n)$ time. For a fixed i, the construction of the set B_i over *all* the different values of α takes only linear time, since we are keeping a pointer to the sorted list of points which have been captured so far in B_i's ball, and we just

scan forward in the list to add new points to the set, when α is increased. Over n_b balls, this is a total of $O(n_b n)$ time. Finally, the greedy algorithm for set cover, can be implemented to run in $O(n_b^2)$ time, and we are running n_b different set covers, for a total of $O(n_b^3)$ time. □

We remark that the analysis above, and our description of the algorithm both assume that the pairwise costs $c(i,j)$ are given as input to the algorithm. In the case that the points are in a metric space, for example, if they are points in R^d with a Euclidean norm, it is more natural to assume the points themselves are simply given as input, and as "step 0" the algorithm would first have to compute all pairwise distances, for a cost of $O(n^2 d)$.

Thus we have proved the following theorem:

Theorem 3.5 *There is a* $\ln n + 1$*-approximation algorithm for class cover that runs in time* $O(n^3 + n^2 d)$.

3.2 Special Case of Fixed Dimension.

We now consider the special case where our red and blue points lie in $\Re^d$, under the Euclidean metric. Suppose S is a set of points in $\Re^d$, and $\mathcal{D}$ is a set of balls of equal radius. The *disk cover* problem, as defined by Hochbaum and Maass [7] is to find a minimum number of balls in $\mathcal{D}$ that cover the points in S.

Hochbaum and Maass prove the following theorem:

Theorem 3.6 *[7] Let* $d \geq 1$ *be some finite dimension and* $\epsilon > 0$ *be fixed. Let* $\delta = (\epsilon^{-1}\sqrt{d})^d$. *Then there is a polynomial times approximation scheme* $H(d,\epsilon)$ *that outputs a cover of* n *points in a* $d-$*dimensional Euclidean space by* d*-dimensional balls of radius* R *in* $O(\epsilon - d\delta(2n)^{d\delta+1})$ *steps that is* $(1+\epsilon)^d$ *times the size of the optimal cover.*

It just remains to observe that the collection of balls produced by algorithm `Fixed-Radius` in step 3, if it satisfies the condition of step 4, is a legal formulation of the disk cover problem when $S = B$ is a set of points in $\Re^d$. Thus the Hochbaum-Maass algorithm can be substituted for step 5 of algorithm `Fixed-Radius` to output the approximate cover. Algorithm `Bubble-Grow` which calls `Fixed-Radius` at most n^2 times will produce a solution with the same approximation factor as the approximate disk cover to the class cover problem; the running time blows up by a factor of n^2 since the running time of `Fixed-Radius` will be dominated by the running time of the Hochbaum-Maass disk cover algorithm.

In the special case of $\Re^2$, the running time can be further improved by an algorithm of Brönnimann and Goodrich, who present a constant factor approximation of the 2-dimensional disk cover algorithm that runs in $O(n^3 \log n)$ time. Substituting this in for the Hochbaum-Maass algorithm above gives us an $O(1)$ approximation to the class cover problem with a running time of $O(n^5 \log n)$ (see [3]).

4 Applications to Classification

Given a solution S to the class cover problem, it can be incorporated into a simple classifier as follows: declare a new unclassified point to be in the positive class if it lies within distance $\max_{v \in B} d(v,S)$ from some blue center, and in the negative class if its distance is at least $\min_{v \in R} d(v,S)$. (For those points whose distance is between $\max_{v \in B} d(v,S)$ and $\min_{v \in R} d(v,S)$, what the classifier should ideally answer will be application dependent, either picking its class arbitrarily, randomly, as some function of its relative proximity to each of these two measures, or alternately it could answer "no decision").

The relationship between the simple classifier defined above, and the degenerate one-dimensional case of the Cowen/Priebe ADC dimension reduction system is worth noting. Cowen and Priebe [4, 6] consider an orthogonal question in pattern recognition: namely, they look for maps of high-dimensional

data into low-dimensional space, that preserve *inter*--class distance in the supervised learning setting, and *inter*-cluster distance in the unsupervised setting. These maps are chosen from a set of maps into j-dimensional space that are indexed by j subsets of the orginal data $w_1, \ldots w_j$, termed the *witness sets.* The ADC map associated with the sets $w_1, \ldots w_j$ maps the point x to the point in j-dimensional space whose ith coordinate is $\min_{v \in w_i} d(v, x)$. It was shown by Linial et. al.[8] that random ADC maps to $O(\log^2 n)$ dimensions under the L_p norms, with appropriately chosen witness set sizes, approximately preserves within a $O(\log n)$ factor ALL interpoint distances (not just interclass distances) of an n-point data set, independent of the dimensionality of the original data, so this is a rich and interesting set of maps to consider.

In one dimension, the ADC map picks a single witness set W, and maps each point x to the scalar quantity that is $\min_{w \in W} d(w, x)$. Clearly, a set W of size k that is feasible for the class cover problem, yields a map into one dimension that places all the blue points to the left of all the red points. Such W are precisely what Cowen and Priebe call the *k-separable sets* (see [6]). Why do we want to minimize k? There are two reasons. The first is computational; we simply need to compute fewer distances if we have a small witness set. The second reason is more qualitative. Suppose we have n_b training observations from the blue class. We believe there is a qualitative difference in class structure between instances in which there are small separating sets and instances in which the only separating set has size close to or equal to n_b.

5 Conclusions and Open Problems

There is an asymmetry in the way the class cover problem treats blue and red points; this asymmetry allows us to model so-called "one class" problems (where the goal is to separate class "B" from "the rest" or "noise", the class $\overline{B}$). To generalize to a symmetric two class problem or to three or more classes, one might give an alternative definition where there is a witness set and a covering of balls required for each separate color class, so that no point of a different color lies within these balls.

A second issue is that the class cover problem as defined is very sensitive to misclassified points, or mistakes. Any point mislabeled as red in a blue neighborhood, or visa versa, can hugely blow up the size of the optimal solution, and cause the essential structure of the separating set to be missed. A relaxation of class cover would ask for all but some small subset of the points to satisfy the condition; where we fix the size, k, of the class separating set and the approximation is on the feasibility of the solution. With the appropriate definitions (for example, a reasonable way to count violations of feasibility would be to ask, given a fixed k, for the solution that has the minimum number of pairs (b, r) such that red point r lies in blue ball b, over all blue balls), we conjecture that some sort of approximation algorithm is achievable.

Another area for further research, that we alluded to in the previous section, would be in using the class cover problem to somehow characterize the difficulty of a particular classification instance. We may generalize the notion of k-separability in one dimension to j-dimensions by using j-dimensional ADC maps, and requiring separation of classes by a hyperplane in j dimensions.

Finally, we ask if the running time for approximate class cover in the general case, can be reduced to quadratic.

Acknowledgments

We thank Samir Kuller for his early encouragement and enthusiasm for the problem formulation, and Carey Priebe for interesting discussions.

References

[1] S. Arora and C. Lund. Hardness of approximations. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[2] P.S. Bradley, Usama M. Fayyad, and O.L. Mangasarian. Mathematical programming for data mining: Formulatons and challenges. Technical report, January 1998.

[3] H. Bronnimann and M. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete Comput. Geom.*, 14:463–479, 1995.

[4] A.H. Cannon, L.J. Cowen, and C.E. Priebe. Approximate distance classification. In *Computer Science and Statistics, Proceedings of the 30th Symposium on the Interface"*, 1998.

[5] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[6] L.J. Cowen and C.E. Priebe. Randomized non-linear projectons uncover high-dimensional structure. *Advances in Applied Math*, 19:319–331, 1997.

[7] D.S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985.

[8] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[9] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1996.

[10] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[11] Leonard Schulman. Clustering for edge cost minimization. Unpublished Manuscript, 1999.