Machine Learning, 56, 89–113, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Correlation Clustering*

NIKHIL BANSAL nikhil@cs.cmu.edu
AVRIM BLUM avrim@cs.cmu.edu
SHUCHI CHAWLA shuchi@cs.cmu.edu
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Editors: Nina Mishra and Rajeev Motwani

Abstract. We consider the following clustering problem: we have a complete graph on n vertices (items), where each edge (u, v) is labeled either $+$ or $-$ depending on whether u and v have been deemed to be similar or different. The goal is to produce a partition of the vertices (a clustering) that agrees as much as possible with the edge labels. That is, we want a clustering that maximizes the number of $+$ edges within clusters, plus the number of $-$ edges between clusters (equivalently, minimizes the number of disagreements: the number of $-$ edges inside clusters plus the number of $+$ edges between clusters). This formulation is motivated from a document clustering problem in which one has a pairwise similarity function f learned from past data, and the goal is to partition the current set of documents in a way that correlates with f as much as possible; it can also be viewed as a kind of "agnostic learning" problem.

An interesting feature of this clustering formulation is that one does not need to specify the number of clusters k as a separate parameter, as in measures such as k-median or min-sum or min-max clustering. Instead, in our formulation, the optimal number of clusters could be any value between 1 and n, depending on the edge labels. We look at approximation algorithms for both minimizing disagreements and for maximizing agreements. For minimizing disagreements, we give a constant factor approximation. For maximizing agreements we give a PTAS, building on ideas of Goldreich, Goldwasser, and Ron (1998) and de la Vega (1996). We also show how to extend some of these results to graphs with edge labels in $[-1, +1]$, and give some results for the case of random noise.

Keywords: clustering, approximation algorithm, document classification

1. Introduction

Suppose that you are given a set of n documents to cluster into topics. Unfortunately, you have no idea what a "topic" is. However, you have at your disposal a classifier $f(A, B)$ that given two documents A and B, outputs whether or not it believes A and B are similar to each other. For example, perhaps f was learned from some past training data. In this case, a natural approach to clustering is to apply f to every pair of documents in your set, and then to find the clustering that agrees as much as possible with the results.

Specifically, we consider the following problem. Given a fully-connected graph G with edges labeled "+" (similar) or "−" (different), find a partition of the vertices into clusters that agrees as much as possible with the edge labels. In particular, we can look at this in

*This research was supported in part by NSF grants CCR-0085982, CCR-0122581, CCR-0105488, and an IBM Graduate Fellowship.

terms of maximizing *agreements* (the number of $+$ edges inside clusters plus the number of $-$ edges between clusters) or in terms of minimizing *disagreements* (the number of $-$ edges inside clusters plus the number of $+$ edges between clusters). These two are equivalent at optimality but, as usual, differ from the point of view of approximation. In this paper we give a constant factor approximation to the problem of minimizing disagreements, and a PTAS[1] for maximizing agreements. We also extend some of our results to the case of real-valued edge weights.

This problem formulation is motivated in part by a set of clustering problems at Whizbang Labs (Cohen & McCallum, 2001; Cohen & Richman, 2001, 2002) in which learning algorithms were trained to help with various clustering tasks. An example of one such problem, studied by Cohen and Richman (2001, 2002) is clustering entity names. In this problem, items are entries taken from multiple databases (e.g., think of names/affiliations of researchers), and the goal is to do a "robust uniq"—collecting together the entries that correspond to the same entity (person). E.g., in the case of researchers, the same person might appear multiple times with different affiliations, or might appear once with a middle name and once without, etc. In practice, the classifier f typically would output a probability, in which case the natural edge label is log(Pr(same)/Pr(different)). This is 0 if the classifier is unsure, positive if the classifier believes the items are more likely in the same cluster, and negative if the classifier believes they are more likely in different clusters. The case of $\{+, -\}$ labels corresponds to the setting in which the classifier has equal confidence about each of its decisions.

What is interesting about the clustering problem defined here is that unlike most clustering formulations, we do not need to specify the number of clusters k as a separate parameter. For example, in min-sum clustering (Schulman, 2000) or min-max clustering (Hochbaum & Shmoys, 1986) or k-median (Charikar & Guha, 1999; Jain & Vazirani, 2001), one can always get a perfect score by putting each node into its own cluster—the question is how well one can do with only k clusters. In our clustering formulation, there is just a single objective, and the optimal clustering might have few or many clusters: it all depends on the edge labels.

To get a feel for this problem, notice that if there exists a perfect clustering, i.e., one that gets all the edges correct, then the optimal clustering is easy to find: just delete all "$-$" edges and output the connected components of the graph remaining. In Cohen and Richman (2002) this is called the "naive algorithm". Thus, the interesting case is when no clustering is perfect. Also, notice that for any graph G, it is trivial to produce a clustering that agrees with at least *half* of the edge labels: if there are more $+$ edges than $-$ edges, then simply put all vertices into one big cluster; otherwise, put each vertex into its own cluster. This observation means that for maximizing agreements, getting a 2-approximation is easy (note: we will show a PTAS). In general, finding the optimal clustering is NP-hard (shown in Section 3).

Another simple fact to notice is that if the graph contains a triangle in which two edges are labeled $+$ and one is labeled $-$, then no clustering can be perfect. More generally, the number of edge-disjoint triangles of this form gives a lower bound on the number of disagreements of the optimal clustering. This fact is used in our constant-factor approximation algorithm.

For maximizing agreements, our PTAS is quite similar to the PTAS developed by de la Vega (1996) for MAXCUT on dense graphs, and related to PTASs of Arora, Karger, and Karpinski (1999) and Arora, Frieze, and Kaplan (2002). Notice that since there must exist a clustering with at least $n(n-1)/4$ agreements, this means it suffices to approximate agreements to within an additive factor of εn^2. This problem is also closely related to work on testing graph properties of Goldreich, Goldwasser, and Ron (1998), Parnas and Ron (2002), and Alon et al. (2000). In fact, we show how we can use the General Partition Property Tester of Goldreich, Goldwasser, and Ron (1998) as a subroutine to get a PTAS with running time $O(ne^{O((\frac{1}{\varepsilon})^{\frac{1}{\varepsilon}})})$. Unfortunately, this is doubly exponential in $\frac{1}{\varepsilon}$, so we also present an alternative direct algorithm (based more closely on the approach of de la Vega (1996)) that takes only $O(n^2 e^{O(\frac{1}{\varepsilon})})$ time.

Relation to agnostic learning. One way to view this clustering problem is that edges are "examples" (labeled as positive or negative) and we are trying to represent the target function f using a hypothesis class of vertex clusters. This hypothesis class has limited representational power: if we want to say (u, v) and (v, w) are positive in this language, then we have to say (u, w) is positive too. So, we might not be able to represent f perfectly. This sort of problem—trying to find the (nearly) best representation of some arbitrary target f in a given limited hypothesis language—is sometimes called *agnostic* learning (Kearns, Schapire, & Sellie, 1994; Ben-David, Long, & Mansour, 2001). The observation that one can trivially agree with at least half the edge labels is equivalent to the standard machine learning fact that one can always achieve error at most $1/2$ using either the *all positive* or *all negative* hypothesis.

Our PTAS for approximating the number of agreements means that if the optimal clustering has error rate ν, then we can find one of error rate at most $\nu + \varepsilon$. Our running time is exponential in $1/\varepsilon$, but this means that we can achieve any constant error gap in polynomial time. What makes this interesting from the point of view of agnostic learning is that there are very few problems where agnostic learning can be done in polynomial time.[2] Even for simple classes such as conjunctions and disjunctions, no polynomial-time algorithms are known that give even an error gap of $1/2 - \varepsilon$.

Organization of this paper. We begin by describing notation in Section 2. In Section 3 we prove that the clustering problem defined here is NP complete. Then we describe a constant factor approximation algorithm for minimizing disagreements in Section 4. In Section 5, we describe a PTAS for maximizing agreements. In Section 6, we present simple algorithms and motivation for the random noise model. Section 7 extends some of our results to the case of real-valued edge labels. Finally, subsequent work by others is briefly described in Section 8.

2. Notation and definitions

Let $G = (V, E)$ be a complete graph on n vertices, and let $e(u, v)$ denote the label ($+$ or $-$) of the edge (u, v). Let $N^+(u) = \{u\} \cup \{v : e(u, v) = +\}$ and $N^-(u) = \{v : e(u, v) = -\}$ denote the positive and negative neighbors of u respectively.

We let OPT denote an optimal clustering on this graph. In general, for a clustering $\mathcal{C}$, let $\mathcal{C}(v)$ be the set of vertices in the same cluster as v. We will use A to denote the clustering produced by our algorithms.

In a clustering $\mathcal{C}$, we call an edge (u, v) a mistake if either $e(u, v) = +$ and yet $u \notin \mathcal{C}(v)$, or $e(u, v) = -$ and $u \in \mathcal{C}(v)$. When $e(u, v) = +$, we call the mistake a *positive mistake*, otherwise it is called a *negative mistake*. We denote the total number of mistakes made by a clustering $\mathcal{C}$ by $m_{\mathcal{C}}$, and use m_{OPT} to denote the number of mistakes made by OPT.

For positive real numbers x, y and z, we use $x \in y \pm z$ to denote $x \in [y - z, y + z]$. Finally, let $\bar{X}$ for $X \subseteq V$ denote the complement $(V \setminus X)$.

3. NP-completeness

In this section, we will prove that the problem of minimizing disagreements, or equivalently, maximizing agreements, is NP-complete. It is easy to see that the decision version of this problem (viz. is there a clustering with at most z disagreements?) is in NP since we can easily check the number of disagreements given a clustering. Also, if we allow arbitrary weights on edges with the goal of minimizing *weighted* disagreements, then a simple reduction from the Multiway Cut problem proves NP-hardness—simply put a $-\infty$-weight edge between every pair of terminals, then the value of the multiway cut is equal to the value of weighted disagreements. We use this reduction to give a hardness of approximation result for the weighted case in Section 7.

We give a proof of NP hardness for the *unweighted* case by reducing the problem of Partition into Triangles GT11 in Garey and Johnson (2000) to the problem of minimizing disagreements. The reader who is not especially interested in NP-completeness proofs should feel free to skip this section.

The Partition into Triangles problem is described as follows: Given a graph G with $n = 3k$ vertices, does there exist a partition of the vertices into k sets $V_1, \ldots, V_k$, such that for all i, $|V_i| = 3$ and the vertices in V_i form a triangle.

Given a graph $G = (V, E)$, we first transform it into a complete graph G' on the same vertex set V. An edge in G' is weighted $+1$ if it is an edge in G and -1 otherwise.

Let A be an algorithm that given a graph outputs a clustering that minimizes the number of mistakes. First notice that if we impose the additional constraint that all clusters produced by A should be of size at most 3, then given the graph G', the algorithm will produce a partition into triangles if the graph admits one. This is because if the graph admits a partition into triangles, then the clustering corresponding to this triangulation has no negative mistakes, and any other clustering with clusters of size at most 3 has more positive mistakes than this clustering. Thus we could use such an algorithm to solve the Partition into Triangles problem.

We will now design a gadget that forces the optimal clustering to contain at most 3 vertices in each cluster. In particular, we will augment the graph G' to a larger complete graph H, such that in the optimal clustering on H, each cluster contains at most 3 vertices from G'.

The construction of H is as follows: In addition to the vertices and edges of G', for every 3-tuple $\{u, v, w\} \subset G'$, H contains a clique $C_{u,v,w}$ containing n^6 vertices. All edges inside these cliques have weight $+1$. Edges between vertices belonging to two different cliques have weight -1. Furthermore, for all $u, v, w \in G'$ each vertex in $C_{u,v,w}$ has a positive edge to u, v and w, and a negative edge to all other vertices in G'.

Now assume that G admits a triangulation and let us examine the behavior of algorithm A on graph H. Let $N = n^6\binom{n}{3}$.

Lemma 1. *Given H as input, in any clustering that A outputs, every cluster contains at most three vertices of G'.*

Proof: First consider a clustering $\mathcal{C}$ of the following form:

1. There are $\binom{n}{3}$ clusters.
2. Each cluster contains exactly one clique $C_{u,v,w}$ and some vertices of G'.
3. Every vertex $u \in G'$ is in the same cluster as $C_{u,v,w}$ for some v and w.

In any such clustering, there are no mistakes among edges between cliques. The only mistakes are between vertices of G' and the cliques, and those between the vertices of G'. The number of mistakes of this clustering is at most $n^7(\binom{n}{2} - 1) + \binom{n}{2}$ because each vertex in G' has n^6 positive edges to $\binom{n}{2}$ cliques and is clustered with only one of them.

Now consider a clustering in which some cluster has four vertices in G', say, u, v, w and y. We show that this clustering has at least $n^7(\binom{n}{2} - 1) + \frac{n^6}{2}$ mistakes. Call this clustering X. Firstly, without loss of generality we can assume that each cluster in X has size at most $n^6 + n^4$, otherwise there are at least $\Omega(n^{10})$ negative mistakes within a cluster. This implies that each vertex in G' makes at least $\binom{n}{2}n^6 - (n^6 + n^4)$ positive mistakes. Hence the total number of positive mistakes is at least $n^7(\binom{n}{2} - 1) - n^5$. Let X_u be the cluster containing vertices $u, v, w, y \in G'$. Since X_u has at most $n^6 + n^4$ vertices, at least one of u, v, w, y will have at most n^4 positive edges inside X_u and hence will contribute at least an additional $n^6 - n^4$ negative mistakes to the clustering. Thus the total number of mistakes is at least $(\binom{n}{2} - 1)n^7 - n^5 + n^6 - n^4 \geq n^7(\binom{n}{2} - 1) + n^6/2$. Thus the result follows. $\square$

The above lemma shows that the clustering produced by A will have at most 3 vertices of G in each cluster. Thus we can use the algorithm A to solve the Partition into Triangles problem and the reduction is complete.

4. A constant factor approximation for minimizing disagreements

As a warm-up to the general case, we begin by giving a very simple 3-approximation to the best clustering containing two clusters. That is, if the best two-cluster partition of the graph has x mistakes, then the following algorithm will produce one with at most $3x$ mistakes.

Let OPT(2) be the best clustering containing two clusters, and let the corresponding clusters be $\mathcal{C}_1$ and $\mathcal{C}_2$. Our algorithm simply considers all clusters of the form $\{N^+(v), N^-(v)\}$ for $v \in V$. Of these, it outputs the one that minimizes the number of mistakes.

Theorem 2. *The number of mistakes of the clustering output by the algorithm stated above is at most $m_A \leq 3m_{\mathrm{OPT}(2)}$.*

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.