Spectral Clustering with Inconsistent Advice

Tom Coleman COLEMANT@CSSE.UNIMELB.EDU.AU
James Saunderson J.SAUNDERSON@UGRAD.UNIMELB.EDU.AU
Anthony Wirth AWIRTH@CSSE.UNIMELB.EDU.AU
The University of Melbourne, Victoria 3010 Australia

Abstract

Clustering with advice (often known as constrained clustering) has been a recent focus of the data mining community. Success has been achieved incorporating advice into the k-means and spectral clustering frameworks. Although the theory community has explored inconsistent advice, it has not yet been incorporated into spectral clustering. Extending work of De Bie and Cristianini, we set out a framework for finding minimum normalised cuts, subject to inconsistent advice.

1. Introduction

Clustering is an exploratory data analysis problem which asks us to form groups of related objects. Although humans have a good intuition for clustering in two dimensions, if the data is in a higher dimensional space, it can be hard to visualise. In this paper, we will focus on the problem of clustering data into *two* clusters, subject to a balance criterion and *advice*.

1.1. Clustering with advice

It is sensible for clustering algorithms to be able to incorporate *must-link* and *cannot-link* advice[1], as it is known in the constrained clustering community. For example, in biology, when experimentally clustering proteins (or genes etc), it is often practical to test associations of individual pairs. However, there is no guarantee that the advice we generate in this way will be correct. Additionally, it is well known that human and biological 'experiments' are often subject to noise. If we have enough noisy advice, that advice will be inconsistent—that is, there is no way to cluster the data which agrees with all the advice.

In that case, the objective naturally becomes to respect as much advice as possible. In fact, if we ignore all the data apart from the advice, we have the the *2-correlation clustering* problem (2CC), known to be NP-hard (Bansal et al., 2004). In this context, we can think of the *advice graph*—which has an edge for each piece of advice, labelled with a $+$ for a must-link, and a $-$ otherwise.

[1]Traditionally in the literature, what we call advice is referred to as *constraints*. We use the term advice here to avoid confusion with constraints that are introduced into the problem in later sections.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

1.2. Balanced clustering

A natural problem to solve when clustering in general is the *normalised cut* (Shi & Malik, 2000). Normalised cut (NCUT) asks us to find a cut which minimises inter-cluster affinity whilst maximising intra-cluster affinity in a sensible way. In the two cluster case, this is to minimise the quantity $\dfrac{\text{cut}(S,\bar{S})}{\text{vol}(S)\text{vol}(\bar{S})}$, where $\text{cut}(S,\bar{S})$ measures the affinity across the cut and $\text{vol}(S)$ is the total degree (out-affinity) of all the nodes within S.

Our aim for the paper is to attempt to optimise both the NCUT and 2CC criteria simultaneously. To do so we will need to relax the problems—we cannot hope to solve them combinatorially.

1.3. Relaxed versions of the clustering problems

Traditionally, spectral approaches were used for the NCUT problem, as they led to fast algorithms. Recently there has been a trend towards tighter, semidefinite programming (SDP)-based, relaxations. We will demonstrate how to alter the basic spectral clustering algorithm, and the SDP techniques, to integrate and deal sensibly with inconsistent advice.

Suppose that the advice *is* consistent (it is simple to check this fact). Once we know that this is the case, it is sensible to constrain the space of the solutions that we explore only to contain clusterings that are consistent with this advice.

1.4. Existing approaches

Spectral Clustering Spectral Clustering appeared first in the literature in the 1970s. Much of the recent popularity of the technique was instigated by the connection to NCUT as shown by Shi and Malik (2000).

Advice Advice (instance-level constraints) for clustering problems was introduced to the machine learning community in the work of Wagstaff and Cardie (2000) who developed a variant of the classic k-means algorithms to incorporate advice.

Kamvar, Klein and Manning (2003) integrated advice into the spectral formulation by directly changing entries of the Laplacian matrix. Xing et al. (2003) improve on this idea by essentially changing the Laplacian in a more consistent way. They do this by finding a metric that best agrees with the advice. These methods do not directly exploit the nature of the spectral algorithm.

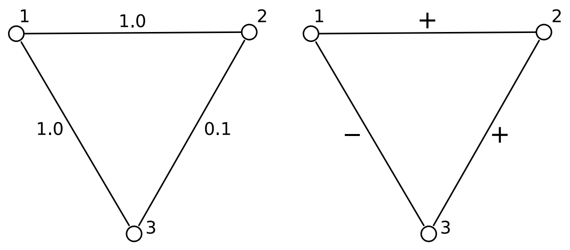
SDP relaxations for N-cut Xing and Jordan (2003) outline a SDP formulation for the NCUT problem for multiple clusters and highlight the connection to spectral clustering. De Bie and Cristianini (2006) provide an SDP which is easier to deal with, and demonstrate that the subspace trick can also be used to introduce advice to this problem.

The subspace trick De Bie, Suykens and De Moor (2004) outline a subspace trick to integrate advice into spectral clustering by constraining a solution to be within the subspace of solutions which agree with the advice. This approach was also previously mentioned in the work of Yu and Shi (2001). This technique leaves the spectral algorithm essentially unchanged; it now just searches for eigenvectors in a different subspace. However it is not necessarily apparent from their work how to extend this technique to inconsistent advice. *This is the key issue addressed in this paper.*

1.5. Addressing Inconsistency

So how can we apply the subspace trick when the advice we have is no longer consistent?

As a first approach (METHOD ONE), we could simply try to solve 2CC defined by the advice, and reject any advice that this solution fails to respect. A good solution to 2CC will ensure that we minimise the number of such edges that we will have to ignore. Then the advice that remains will be consistent, and we can then use the subspace trick. Or indeed, we could use any other constraint-based clustering algorithm in this way.



(a) N-cut problem (b) CC problem

Figure 1. A problem for which not all optimal solutions to 2CC are optimal for the accompanying NCUT problem.

However, this idea has some problems. A toy example of inconsistent advice in Figure 1 shows that deleting any one of the three edges will result in an optimal solution to 2CC. However, one specific cut (namely separating node 3 from nodes 1 and 2) has a much better NCUT cost. So, in forcing a particular optimal solution to 2CC, we are constraining our NCUT solution too much. A second approach (METHOD TWO) that solves this problem is to calculate the *cost* of an approximately optimal solution to 2CC. Rather than force our NCUT solution to be consistent with this 2CC solution, instead we simply require that our NCUT solution has the same correlation clustering cost. This approach will give the NCUT side of our algorithm some room to move, avoiding situations like Figure 1. This technique will be outlined in section 4.1.

A third approach (METHOD THREE) is to allow the algorithm to differ from the optimum correlation clustering cost, but only by some a given factor. So now the NCUT side of the problem has some breathing space in which to find a good solution, whilst we are still forcing a very good solution to 2CC. This approach is developed in section 4.2.

2. Relaxing the Problem

2.1. Normalised Cut

To define the NCUT problem, we begin with an edge-weighted affinity graph with associated affinity matrix A. Distant edges may have zero affinity—we can represent this by deleting the connecting edge, which will speed up the computation.

We represent a 2-clustering by a vector v, where each coordinate represents a datapoint, and is either $+1$ or -1 depending on cluster assignment. The NCUT value becomes:

$$\text{NCUT}(v) = \frac{v^T \mathcal{L}(A) v}{v^T \mathcal{L}(dd^T) v} \qquad (1)$$

where d is the vector of vertex degrees, and $\mathcal{L}(X)$ is the *Laplacian* of matrix X. Recall that if e is the vector consisting of all ones and $\text{diag}(x)$ is the matrix with vector x on the main diagonal and zeros elsewhere, then $\mathcal{L}(X) = \text{diag}(Xe) - X$.

2.2. Spectral Clustering

In the spectral relaxation, instead of assigning ± 1 to each vertex, we assign a real number v_i. If $v = (v_1, \ldots, v_n)$ then NCUT relaxes to

P1. *Spectral clustering*

$$\min \ \frac{v^T \mathcal{L}(A) v}{v^T D v} \qquad \text{s.t.} \qquad d^T v = 0$$

where $D = \text{diag}(d)$. This relaxation is correct because $v^T \mathcal{L}(A) v$ is invariant under translations of v so we can add the constraint $d^T v = 0$ without changing the optimum cost. With this constraint, the denominator of (1) can be simplified as in **P1**.

It turns out that $v = D^{\frac{1}{2}} u$ is an optimum solution of **P1**, where u is the eigenvector of $D^{-\frac{1}{2}} \mathcal{L}(A) D^{-\frac{1}{2}}$ corresponding to the smallest non-zero eigenvalue.

2.3. The SDP formulation

In the two cluster case, De Bie and Cristianini (2006) devised an efficient relaxation of NCUT to a semidefinite program. In this case, instead of assigning ± 1 to the vertices we assign *vectors* v_i of some common length. If X is the Gram matrix of these vectors (i.e. $X_{ij} = v_i^T v_j$) then the relaxation is

P2. *De Bie and Cristianini SDP*

$$\min_{X,q} \ \mathcal{L}(A) \bullet X$$

$$\text{s.t.} \quad \begin{array}{rrcll} & \mathcal{L}(dd^T) \bullet X & = & 1 & (2) \\ \forall i \in [n] & X_{ii} & = & q & (3) \\ & X & \succeq & 0 & (4) \end{array}$$

where $A \bullet B = \text{trace}(AB)$ for matrices of appropriate dimension. Here the free variable q is the common (squared) length of the v_i and (2) is a scaling constraint corresponding to the denominator of the NCUT objective function (1).

Importantly, given any $X \succeq 0$ we can find $v_1, \ldots, v_n$ such that $X_{ij} = v_i^T v_j$; we can thus convert a solution to **P2** to an assignment of vectors to the vertices of the graph.

Spectral clustering can be recovered from **P2** by removing the constraints of (3) (see Goemans (1997)).

2.4. The 'subspace trick'

The subspace trick of De Bie, Suykens and De Moor (2004) gives a method for incorporating consistent advice into spectral and SDP relaxations of NCUT. As an example, consider spectral clustering and suppose we have two 'blocks' of independent advice. The first that two vertices, say v_1 and v_2, should be in the same cluster and both should be in a different cluster to v_3, the second that vertices v_4 and v_5 should be in the same cluster. Then it makes sense to constrain the solution vector v so that $v_1 = v_2$ and $v_4 = v_5$ guaranteeing that these pairs of vertices end up in the same cluster after rounding. It also makes sense to constrain v so that $v_3 = -v_2 = -v_1$. This can be done by assuming v has the form

$$v = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & I_{n-5} \end{bmatrix} u = Yu$$

where $u \in \mathbb{R}^{n-3}$. The identity matrix corresponds to vertices for which we have no advice and so should not constrain.

3. Correlation Clustering

Given an advice graph, in the form of $+$ (must-link) or $-$ (cannot-link) edges between datapoints, the correlation clustering problem asks us to cluster the datapoints so that the number of pieces of advice (i.e. edge labels) that are disobeyed is minimised.

3.1. 2CC — the combinatorial problem

In general, unlike the affinity graph, the advice graph is not connected. So we can solve correlation clustering independently on each connected component. We call the vertices in a connected component an *advice block*. We assume without loss of generality that the order on the vertices ensures that the vertices within each block are consecutive. Here we will deal with the problem of solving 2CC for a single advice block $\mathcal{B}$ with m vertices. In later sections, we will consider multiple advice blocks.

If e is an edge within $\mathcal{B}$ let $w_e \in \pm 1$ correspond to the the sign of e. As for NCUT, we assign $v_i = \pm 1$ to each vertex depending on the cluster in which we place that vertex. For convenience, let E_{ij} be the matrix with a 1 in the (i,j) entry and zeros everywhere else.

Our immediate aim is to find, in terms of v, a simple expression for the number of constraints violated by the labelling.

Consider a single edge $e = \{i,j\}$ of $\mathcal{B}$ with label w_e. Define

$$M_e = (E_{ii} + E_{jj}) - w_e(E_{ij} + E_{ji})$$

and note that $M_e \succeq 0$ because its eigenvalues are 0 and 2. Now

$$\begin{aligned} v^T M_e v &= v_i^2 - 2w_e v_i v_j + v_j^2 && (5)\\ &= |v_i - w_e v_j|^2 && (6)\\ &= \begin{cases} 0 & \text{if } v \text{ respects the advice on } e \\ 4 & \text{otherwise.} \end{cases} && (7) \end{aligned}$$

So if we define $M_{\mathcal{B}} = \sum_e M_e$ it follows that

$$v^T M_{\mathcal{B}} v = 4 \times (\#\text{ pieces of advice violated by } v).$$

Thus 2CC is essentially

$$\min_{v \in \{-1,1\}^m} v^T M_{\mathcal{B}} v. \qquad (8)$$

Note that a clustering that satisfies all the advice in a block will have cost zero. Also observe that $v^T v = m$ is a constant so we could replace the objective function with $(v^T M_{\mathcal{B}} v)/(v^T v)$ without changing the optimum vector.

3.2. Relaxations of 2CC

Recall that our overall aim is to constrain any algorithm we have for (approximately) solving Ncut to produce clusterings which are, in terms of 2CC cost, not much worse than the optimum.

Since we cannot hope to solve (8) exactly, we will instead solve a relaxed version of it. In this paper we consider two relaxations which arise in much the same way as the spectral and SDP relaxations of Ncut.

P3. *Spectral relaxation of correlation clustering*

$$\min_v \frac{v^T M_{\mathcal{B}} v}{v^T v}$$

Observe that the solution of **P3** is given by any non-zero vector in the λ_{min}-eigenspace of $M_{\mathcal{B}}$.

P4. *SDP relaxation of correlation clustering*

$$\begin{aligned} &\min_X \quad M_{\mathcal{B}} \bullet X \\ &\text{s.t.} \quad \forall i \in [m] \quad X_{ii} = 1 \qquad (9)\\ &\qquad\qquad\qquad\quad X \succeq 0 \end{aligned}$$

For either relaxation, if the advice is consistent, the relaxation produces a solution of the same cost (zero) as the optimal solution to the combinatorial problem (8). This is because any solution of the original problem is a feasible point of the relaxed problem, and the relaxed problem has non-negative cost as $M_{\mathcal{B}} \succeq 0$.

4. Clustering with inconsistent advice

In this section, we give the details of Method Two and Method Three, introduced in Section 1.5, for both the spectral and SDP relaxations of Ncut.

Throughout, let $\mathcal{B}_1, \ldots, \mathcal{B}_p$ be the advice blocks of the advice graph. Let $v_{\mathcal{B}}$ denote the projection of v onto the coordinates involved in advice block $\mathcal{B}$. Along similar lines, if $u_{\mathcal{B}}$ is a vector of length $|\mathcal{B}| \leq n$ associated with the advice block $\mathcal{B}$, define $\widetilde{u_{\mathcal{B}}}$ to be the length n vector that agrees with $u_{\mathcal{B}}$ in the appropriate coordinates and has zeros elsewhere. For a $|\mathcal{B}| \times |\mathcal{B}|$ matrix $M_{\mathcal{B}}$ we also define $\widetilde{M_{\mathcal{B}}}$ in a similar fashion.

4.1. Combining 2CC and Ncut: Method Two

Let OPT_j denote the optimum cost of the SDP relaxation of 2CC (**P4**) for block j. For the SDP relaxation, we can add the constraint that for each advice block, the 2CC cost of point X is at most $q \cdot \text{OPT}_j$. (The scaling by q is necessary because in **P4** the variables satisfy $X_{ii} = 1$ whereas in **P2** the variables satisfy $X_{ii} = q$.) This forces the new SDP (**P5**) only to consider points of minimum SDP-relaxed 2CC cost.

P5. Method Two*(SDP version)*

$$\begin{aligned} &\min_{X,q} \quad \mathcal{L}(A) \bullet X \\ &\qquad\qquad \mathcal{L}(dd^T) \bullet X = \text{vol}(\mathcal{V}) \\ &\text{s.t.} \qquad \forall i \in [n] \quad X_{ii} = q \\ &\qquad\quad \forall j \in [p] \quad \widetilde{M_{\mathcal{B}_j}} \bullet X \leq q \cdot \text{OPT}_j \qquad (10)\\ &\qquad\qquad\qquad\qquad X \succeq 0 \end{aligned}$$

In the spectral case, the analogous thing to do would be to add the following constraints to the spectral relaxation of Ncut.

$$\forall j \in [p] \quad \frac{v_{\mathcal{B}_j}^T M_{B_j} v_{\mathcal{B}_j}}{v_{\mathcal{B}_j}^T v_{\mathcal{B}_j}} \leq \lambda_{min}(M_{\mathcal{B}_j}) \qquad (11)$$

But doing so would mean the problem would no longer be an eigenvalue problem—in fact it would be an SDP—which would undermine the main strength of spectral clustering, its speed.

Luckily, the condition in (11) is equivalent to the condition that each $v_{\mathcal{B}_j}$ is in the λ_{min}-eigenspace of $M_{\mathcal{B}_j}$, resulting in **P6**.

P6. Method Two *(spectral version)*

$$\begin{array}{lrcl} \min\limits_{v} & \dfrac{v^T \mathcal{L}(A) v}{v^T D v} & & \\ \text{s.t.} & d^T v & = & 0 \qquad (12) \\ & \forall j \in [p] \quad v_{\mathcal{B}_j} & \in & \lambda_{min}\text{-eigenspace of } M_{\mathcal{B}_j} \qquad (13) \end{array}$$

The constraints (12) and (13) are forcing v to be in some linear subspace of $\mathbb{R}^n$. So the problem can then be solved using the subspace trick. Details of how to do this are in Appendix A.

4.2. Combining 2CC and Ncut: Method Three

The main drawback of Method Two is that it does not give the algorithm much freedom to balance the trade-off between the 2CC and the Ncut problem. If the advice is quite inconsistent, then forcing the algorithm to follow solutions of a relaxation of 2CC too closely will result in poor performance.

Above, we forced the algorithm to produce a (relaxation of) a clustering that had cost at most the minimum cost of the appropriate relaxation of 2CC. Now we introduce a parameter $f \geq 1$ which tells us the factor by which we are willing to exceed the 2CC cost.

This is straightforward to introduce to the SDP formulation. We simply replace the constraints (11) of **P5** with

$$\forall j \in [p] \quad \widetilde{M_{\mathcal{B}_j}} \bullet X \leq f \cdot q \cdot \text{OPT}_j. \qquad (14)$$

In the spectral formulation, the constraints we actually want to add are

$$\forall j \in [p] \quad \frac{v_{\mathcal{B}_j}^T M_{B_j} v_{\mathcal{B}_j}}{v_{\mathcal{B}_j}^T v_{\mathcal{B}_j}} \leq f \cdot \lambda_{min}(M_{\mathcal{B}_j}) \qquad (15)$$

but, again we cannot add these and still have an eigenvalue problem. Unfortunately in this case we cannot get a constraint equivalent to (15) by the subspace trick. So, in the interests of producing a practical algorithm, we approximate (15) by

$$\forall j \in [p] \quad v_{\mathcal{B}_j} \in (\leq f \cdot \lambda_{min})\text{-eigenspace of } M_{B_j} \quad (16)$$

where the $(\leq f \cdot \lambda_{min})$-eigenspace of M_{B_j} is the span of all eigenvectors of $M_{\mathcal{B}_j}$ with eigenvalue at most $f \cdot \lambda_{min}$. If v satisfies (16) then it satisfies (15), but the converse does not necessarily hold.

Replacing the constraints in (13) of **P6** with the constraints in (16) gives our final spectral algorithm for clustering with inconsistent advice. It can again be solved with the subspace trick, using the techniques outlined in Appendix A, because all the constraints simply force v to be in some linear subspace of $\mathbb{R}^n$.

5. Experimental Investigations

5.1. Experiment Setup

In order to test the performance of the algorithms on real world datasets, we used six of the UCI repository datasets (Asuncion & Newman, 2007). All datasets are multi-dimensional binary classification problems.

Both datasets were stripped of incomplete records, and in one case (the Spambase dataset), sampled down to 500 datapoints. In each case, the two clusters were of different sizes. This contributed to the mediocre performance of the pure spectral algorithm. This gives us reason to believe that adding advice will help the situation.

For reasons of speed, our experiments primarily use the spectral version of each of the algorithms. Relaxed solutions are rounded to clusterings by cutting at zero. This ensures that advice respected in the relaxed solution is respected in the final clustering.

Advice We generated two different 'types' of synthetic advice for these problems to get a sense of how the algorithms perform. The first we call Dense—here we are generating around n pieces of advice. We are generating that advice in a dense fashion—we concentrate all advice within 5 separate groups of 20 datapoints. This simulates a few sets of experiments done on some small subset of the total dataspace. Each piece of advice agrees with the actual classification independently with some probability p.

The second type of advice is the Complete case—here we are simulating pairwise comparisons that are relatively cheap, but quite noisy. So we generate a piece of advice for each pair of datapoints, and thus our advice graph is complete.

2CC Algorithms In order to test Method One, we need to solve 2CC on each advice block. In the Dense case, we use a tight, strongly performing SDP relaxation (Agarwal et al., 2005). In the complete case, we use the simple 3-approximation algorithm of Bansal, Blum and Chalwa (2004) with a final local search step (see also our other paper (Coleman et al., 2008)).

For each advice type on each dataset, we ran spectral clustering with no advice (as a baseline), Method One (as a second baseline), and then spectral clustering with every different meaningful f value from 1 upwards. That is, every increment in f that added one additional eigenvector to a single block, until all eigenvectors were added (which is exactly the same as the no advice case).

5.2. Results

Dense advice Figure 2 displays the results of the Dense advice problem on the Heart Disease dataset with $p = 0.75$. We can see that the advice here is sufficiently inconsistent that algorithms which follow it closely (i.e. Method One and Method Two) perform far worse than the algorithm that ignores it completely (that is, spectral alone). But we can see that by increasing f and striking a balance between ignoring advice and respecting it too strongly, we can achieve results that outperform either extreme. We also note that two other datasets, Congressional Voting Records and Australian, perform similarly.

Figure 3 shows the results of running very similar advice (again $p = 0.75$) on the Spambase dataset. Here we can see that algorithms that strictly follow the advice outperform algorithms that ignore it, quite significantly. It is perhaps unsurprising then that when we allow the algorithm more and more freedom to ignore the advice we move toward the baseline no advice score. This highlights the fact that these algorithms are not always of use—there needs to be enough inaccuracy in the advice that attempting to follow it is not a great idea.

However, if we lower p to be 0.65, the situation changes, and we get a scenario as demonstrated by Figure 4. Here as for the Heart Disease case, using only the 2CC solution is worse than using no advice at all, and for a large range of f values, the compromise of using some advice is better than either extreme. Here the Haberman dataset performs similarly. The difference in this case is that for high f values, very poor performance is exhibited. We will discuss this in the next section.

Finally, we consider the Hepatitis dataset (Figure 5). Here we see new behaviour, as our algorithms only begin to perform well for high f values.

Complete Advice Figures 6 and 7 show the results of the experiments on the two datasets with Complete advice. We first notice that in order to get meaningful experiments, we needed to set p extraordinarily low—all the way down to $p = 0.53$. If p is much higher than this, advice is *so* complete that any incorrect edges will be vastly overshadowed by correct ones, and simply solving 2CC on the instance will give 100% accuracy.

However, with $p = 0.53$ and the problem interesting, we can see that things are similar to the Dense case. Again, when f is low, we start at the 2CC-baseline,

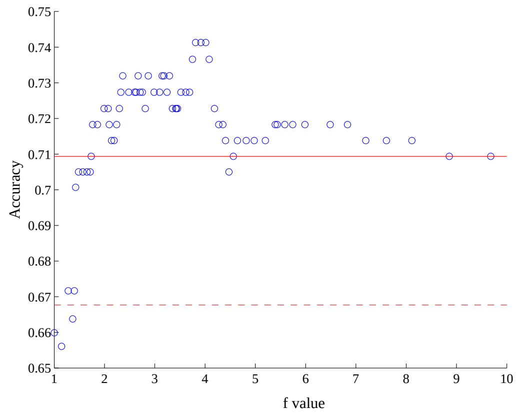


Figure 2. Heart Disease dataset, Dense advice, $p = 0.75$. The unbroken line is the baseline no advice accuracy; the dashed line is the correlation clustering based algorithm (Method One).

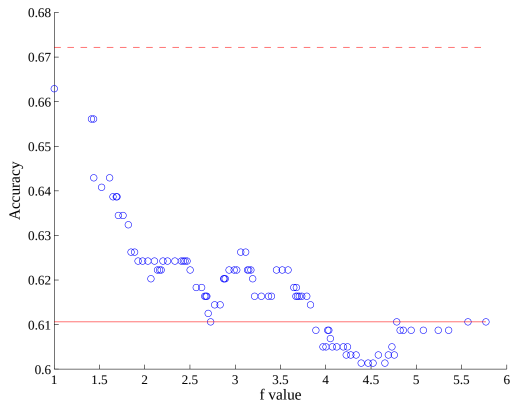


Figure 3. Spambase dataset, Dense Advice, $p = 0.75$

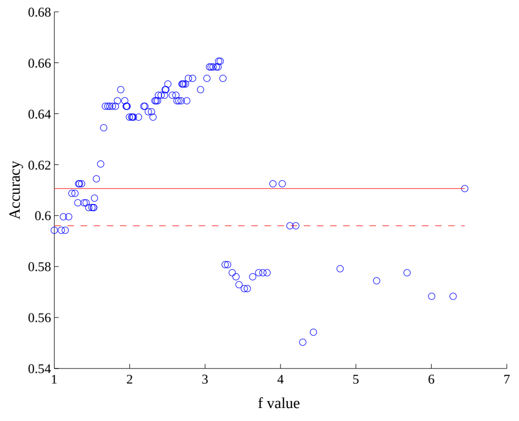


Figure 4. Spambase dataset, Dense advice, $p = 0.65$

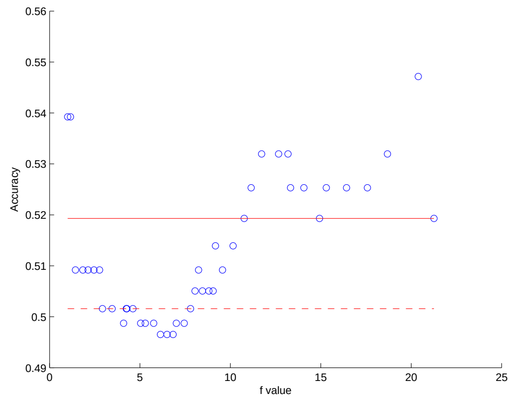


Figure 5. HEPATITIS dataset, DENSE advice, $p = 0.6$

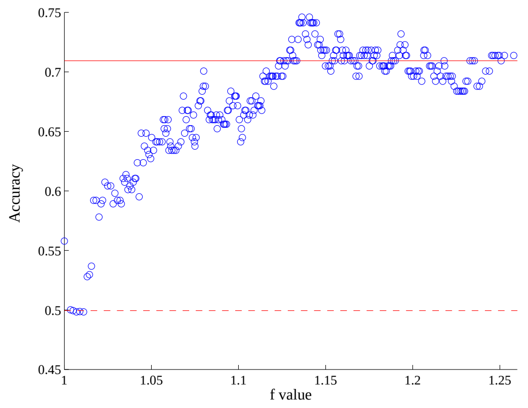


Figure 6. HEART DISEASE dataset, COMPLETE advice, $p = 0.53$

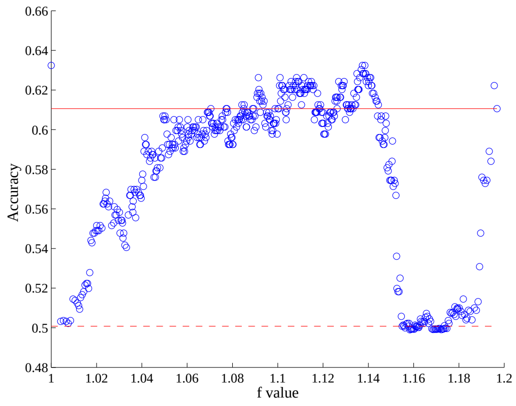


Figure 7. SPAMBASE dataset, COMPLETE advice, $p = 0.53$

and as f increases we move towards and above the no-advice baseline. An interesting point is that the 2CC baseline is around 0.5 in both cases. Note that this is an extremely low score—the advice alone is useless for solving the problem, yet it is still a useful addendum for the spectral method.

As we saw in the DENSE case, one interesting difference between the two datasets is the way the performance drops off as f increases. For HEART DISEASE, the performance seems to asymptote to the no-advice case as we increase f (as we would expect). However, in both cases for the SPAMBASE data, there is a huge dropoff in performance for high end f values. We have no explanation currently for this phenomenon.

6. Conclusions and further work

We have presented a new algorithm that uses inconsistent advice in spectral clustering. This paper is the first to do so. Initial experiments indicate that in many situations our methods are successful, however further theoretical and experimental work is needed. For example, given a clustering problem with inconsistent advice, how do we know when to use METHOD THREE rather than METHOD TWO? And if we are to use METHOD THREE, how do we decide which value of f to choose?

This paper was intended as a largely theoretical work—experiments were performed to give preliminary evidence that the techniques work. Certainly a more thorough comparison to existing work is needed—a technique similar to METHOD ONE could be used in order to compare our algorithms to other approaches that can only deal with consistent constraints. These will be tested in the full version of the paper.

Additionally, in this paper we focused on clustering into two clusters. This is for two reasons. First, there is no obvious way to express cannot-link advice when we have more than two clusters—the approach used here does not generalise nicely. Furthermore, we do not know of an SDP relaxation of the problem which fits into the framework of this paper for the case of more than two clusters. Future work will try to address these problems.

Acknowledgments

Thanks to Ian Davidson for encouraging us to pursue this problem, and to the anonymous reviewers of the draft version. This work was supported by the Australian Research Council through Discovery Project

Grant DP0663979.

References

Agarwal, A., Charikar, M., Makarychev, K., & Makarychev, Y. (2005). $O(\sqrt{\log n})$ approximation algorithms for MIN UNCUT, MIN 2CNF DELETION, and DIRECTED CUT problems. *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 573–581.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Bansal, N., Blum, A., & Chawla, S. (2004). Correlation clustering. *Machine Learning*, *56*, 89–113.

Coleman, T., Saunderson, J., & Wirth, A. (2008). A local-search 2-approximation for 2-correlation clustering. Submitted.

De Bie, T., & Cristianini, N. (2006). Fast SDP Relaxations of Graph Cut Clustering, Transduction, and Other Combinatorial Problems. *The Journal of Machine Learning Research*, *7*, 1409–1436.

De Bie, T., Suykens, J., & De Moor, B. (2004). Learning from general label constraints. *Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*, 671–679.

Goemans, M. (1997). Semidefinite programming in combinatorial optimization. *Mathematical Programming*, *79*, 143–161.

Kamvar, S., Klein, D., & Manning, C. (2003). Spectral learning. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 561–566.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Wagstaff, K., & Cardie, C. (2000). Clustering with instance-level constraints. *Proceedings of the Seventeenth International Conference on Machine Learning*, *1103*, 1110.

Xing, E., & Jordan, M. (2003). *On Semidefinite Relaxation for Normalized K-cut and Connections to Spectral Clustering.* Computer Science Division, University of California.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*, *15*, 505–512.

Yu, S., & Shi, J. (2001). *Grouping with Bias.* Carnegie Mellon University, the Robotics Institute.

A. Implementing spectral clustering with inconsistent advice

In this section we explain how to implement the spectral version of METHOD TWO and METHOD THREE using the subspace trick.

Let $W_{\mathcal{B}_j}$ be a matrix whose columns are a basis for the $(\leq f \cdot \lambda_{min})$-eigenspace of $M_{\mathcal{B}_j}$. Let $\mathcal{N}(X)$ and $\mathcal{R}(X)$ respectively denote the nullspace and range space of a matrix X. Then the problem can be written as follows:

P7. *Spectral clustering with inconsistent advice*

$$\min_v \quad v^T \mathcal{L}(A) v$$

$$\text{s.t.} \quad \begin{array}{rrcl} & v^T D v & = & 1 \\ & v & \in & \mathcal{N}(d^T) \qquad (17) \\ \forall j \in [p] & v_{\mathcal{B}_j} & \in & \mathcal{R}(W_{\mathcal{B}_j}) \qquad (18) \end{array}$$

Let

$$W = \begin{bmatrix} W_{\mathcal{B}_1} & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & W_{\mathcal{B}_p} & 0 \\ 0 & \cdots & 0 & I \end{bmatrix}$$

where the dimension of I is the number of vertices not involved in any advice. Then we can replace the constraints (17) and (18) with

$$v \in \mathcal{N}(d^T) \cap \mathcal{R}(W) = \mathcal{C}$$

Suppose Y is a matrix satisfying $\mathcal{R}(Y) = \mathcal{C}$ and $Y^T D Y = I$. Then if we let $v = Yu$ it is clear that $v \in \mathcal{R}(Y)$ which is what we want. So **P7** becomes

$$\min\ (u^T Y^T)\mathcal{L}(A)(Yu) \quad \text{s.t.} \quad u^T u = 1 \qquad (19)$$

A solution of (19) is given by taking u to be an eigenvector corresponding to the smallest eigenvalue of $Y^T \mathcal{L}(A) Y$. The solution to the original problem is then $v = Yu$.

Elementary linear algebra shows Y generated thus is satisfactory:

1. Let N be a matrix whose columns are an orthonormal basis for $\mathcal{N}(d^T W)$ with respect to the inner product $\langle x, y \rangle = y^T x$.
2. Let R be a matrix whose columns are an orthonormal basis for $\mathcal{R}(W)$ with respect to the inner product $\langle x, y \rangle_D = y^T D x$.
3. Set $Y = RN$.