A Feature Space View of Spectral Clustering

Ali Rahimi
Vision Interface Group, CSAIL
Massachusetts Institute of Technology
Cambridge, MA 02139
ali@mit.edu

Ben Recht
Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
brecht@media.mit.edu

Abstract

The transductive SVM is a semi-supervised learning algorithm that searches for a large margin hyperplane in feature space. By withholding the training labels and adding a constraint that favors balanced clusters, it can be turned into a clustering algorithm. The Normalized Cuts clustering algorithm of Shi and Malik, although originally presented as spectral relaxation of a graph cut problem, can be interpreted as a relaxation on clustering with transductive SVMs. Using this interpretation of Normalized Cuts, we identify some of its weaknesses, and propose a correction on it.

1 Introduction

An intuitive clustering idea is to find a hyperplane that passes through the data set with as great a distance to the data points as possible while separating the data into two even clusters. Support Vector Machines search for such a hyperplane when the data are labeled, and the transductive SVM [1] search for such a hyperplane when only some of the data are labelled. By withholding the training labels and adding a constraint that favors balanced clusters, the transductive SVM can be turned into a clustering algorithm. The Normalized Cuts clustering algorithm of Shi and Malik, although originally presented as spectral relaxation of a graph cut problem, can be interpreted as a relaxation on clustering with transductive SVMs. Using this interpretation of Normalized Cuts, we identify some of its weaknesses, and propose a correction on it.

Normalized Cuts views the data set as a graph, where nodes represent data points and edges are weighted according to the similarity, or "affinity", between data points. This is the starting point of many other graph-based clustering algorithms [2, 3]. The affinity matrix used in these algorithms is a Gram matrix of a possibly infinite-dimensional lifting of the data set. Based on this observation, we show that Normalized Cuts lifts the data set to an infinite-dimensional feature space and cuts the data by passing a hyperplane through a "gap" in the lifted data. It then labels points that fall on the same side of the hyperplane as belonging to the same cluster.

This new interpretation of Normalized Cuts reveals that it maximizes a gap that weights data points away from the mean of the data set more than those in the center of the data

set. This weighting causes Normalized Cuts to sometimes break elongated clusters and to be sensitive to outliers. By defining a new gap that gives equal weight to all data points, we derive a clustering algorithm (the Average Gap algorithm) that does not exhibit these problems. Finding labels under this new gap reduces to thresholding the top eigenvector of a matrix. We also derive a relaxation of the transductive SVM clustering problem which can be solved with semidefinite programming. This relaxation outperforms both the Normalized Cuts algorithm and its correction in experiments.

Lifting the vertices of a graph to a feature space is common in the graph cut literature [4, 3], though there is no search for a hyperplanar gap. Lifting appears in non-graph-based clustering as well. For example, [5] performs approximate K-Means clustering in feature space. Ben-Hur et. al [6] observed that when estimating the support of a lifted data set by fitting a hypersphere around it the resulting support functions form closed contours that can be used to label the data. By contrast to these methods, according to our interpretation, Normalized Cuts directly searches for a hyperplanar gap in the lifted data set.

This paper only presents 2-way clustering algorithms. If more clusters are sought, each 2-way cut can be further subdivided by running the clustering procedure recursively [7].

2 The Normalized Cuts Algorithm

This section provides a brief review of the Normalized Cuts algorithm. Given a set of data points $\mathbf{x} = \{x_i | x_i \in \mathcal{R}^d, i \in 1..N\}$, and an "affinity" measure $k(x, y)$, build the affinity matrix $\mathbf{K}$ with $K_{ij} = k(x_i, x_j)$. A common choice for k is the Gaussian kernel $k(x, y) = \exp\left(-\frac{\|x_i - y_i\|^2}{2\sigma^2}\right)$. The affinity matrix $\mathbf{K}$ defines the weights on a fully connected graph where each node corresponds to a data point x_i and K_{ij} is the weight of the edge between node i and node j. Assigning each x_i a label $y_i \in \{-1, +1\}$ cuts the graph into a set A of the vertices with label -1 and a set B of vertices with labels +1. The cost cut(A, B) is the sum of the weight of the edges between vertices in A and vertices in B. The goal of Normalized Cuts [7] is to find the cut that minimizes the following cost function:

$$\text{cut}(A, B) \left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)} \right), \tag{1}$$

where Vol is the sum of the weights in a set. This cost function is designed to penalize cuts that are not well balanced. Finding the optimal Normalized Cut is NP hard, so the Normalized Cuts algorithm optimizes a relaxation of the above:

$$\begin{aligned} v^* = \quad & \arg\max_v \quad \frac{v^\top \mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}} v}{v^\top v} \\ & \text{s.t.} \qquad v^\top \mathbf{D}\mathbf{1} = 0 \end{aligned}$$

$\mathbf{D}$ is a diagonal matrix whose iith entry is the sum of the ith row of $\mathbf{K}$, and $\mathbf{1}$ is the column vector of all ones. The optimum v is the second eigenvector of $\mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}}$ (we casually refer to the nth eigenvector of a matrix in this paper as a shorthand for the eigenvector corresponding to the nth largest eigenvalue). The components of v^* are then thresholded to yield a vector in $\{-1, +1\}^N$:

$$\hat{y} \quad = \quad \text{sgn}(v^*). \tag{2}$$

This is the labeling as reported by Normalized Cuts. We refer to this algorithm as the Normalized Cuts algorithm (or just Normalized Cuts) and the unrelaxed cost function (1) as the Normalized Cut cost. Other relaxations for (1) are possible [8], but we do not provide a separating hyperplane interpretation for these relaxations here.

3 Hyperplane Cutting Methods for Clustering

In this section we formalize the notion of clustering with a hyperplane. To allow all possible separations of the data set, we lift the data to a high-dimensional space. Once a hyperplane is fitted, points that fall on the same side of the hyperplane will be labeled as being in the same cluster.

Any positive definite kernel $k(x, y)$ defines a lifting of a data point x in a compact subset of $\mathcal{R}^d$ to a possibly infinite-dimensional vector X in "feature space" via $X^j = \sqrt{\lambda_j}\phi_j(x)$, where $\phi_j(x)$ are the bases of the Mercer expansion $k(x, y) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x)\phi_j(y)$. The inner product in feature space is defined so that $< X_i, X_j >_k = k(x_i, x_j)$. With a slight abuse of notation, we will write $< X_i, X_j >_k = X_i^\top X_j$. For many kernels, including the Gaussian one, $\|X\|^2 = k(x, x) = 1$, so the kernel maps points x onto a sphere. Also, because $k(x, y)$ is always positive, the points in feature space must all lie in the same orthant.

We refer to the distance measure between the hyperplane and the data points as a "gap". The signed distance between a point X in feature space and a plane $\{X|\beta^\top X = 0\}$ that passes through the origin of feature space with normal β is $\beta^\top X$. The label of this point is the sign of this distance: $y = \text{sgn}(\beta^\top X)$.

We would like to find a hyperplane in feature space, parametrized by its (possibly infinite dimensional) normal vector β, to maximize some such a measure of gap $M(\beta)$. In subsequent sections, we will provide various examples of $M(\beta)$. To ensure that the plane passes through the data set, and that the clusters have roughly the same number of points, we additionally require that the average signed distance to the hyperplane be zero: $\sum_{i=1}^{N} \beta^\top X_i = 0$, or equivalently, $\beta^\top \bar{X} = 0$. The optimal β is found by solving

$$\begin{aligned} \beta^* = \quad & \arg\max_\beta \quad M(\beta) && (3) \\ & \text{s.t.} \qquad \|\beta\| = 1, \\ & \qquad\quad \beta^\top \bar{X} = 0 && (4) \end{aligned}$$

By assuming that $M(\beta)$ is only a function of the distances between the data points and the hyperplane, we can invoke the generalized representer theorem [9], which states that β^* is a linear combination of the data: $\beta^* = \mathbf{X}c^*$.

In addition to recovering the labels of the given data set, we can also find the splitting function $f(x)$ which returns the distance between any point to the hyperplane. As a function of an input point x_0, the signed distance between the corresponding feature point X_0 and the hyperplane defined by β is $f(x_0) = \beta^{*\top} X_0$. Due to the representer theorem, we can write

$$f(x_0) = \sum_{i=1}^{N} c_i k(x, x_i). \tag{5}$$

We show these separating functions in figures throughout this paper.

4 Normalized Cuts as Hyperplane Cutting

The kernel trick provides a geometric explanation for $\mathbf{K}$ and $\mathbf{D}$ as objects in a possibly infinite-dimensional space. The affinity matrix $\mathbf{K}$ has $k(x_i, x_j)$ as its ijth element. Since $K_{ij} = X_i^\top X_j$, we can write $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$, with $\mathbf{X} = \{X_i\}$.

The $\mathbf{D}$ matrix of Normalized Cuts is diagonal with

$$D_{ii} = \sum_{j=1}^{N} k(x_i, x_j) = X_i^\top \sum_{j=1}^{N} X_j.$$

Defining $\bar{X} = \sum_{j=1}^{N} X_j$, these entries are $D_{ii} = X_i^\top \bar{X} = \|\bar{X}\| \cos\theta_i$, where θ_i is the angle in feature space between the vector X_i and the mean data vector $\bar{X}/N$. Because points in feature space lie in a sphere, we think of D_{ii} as a distance between X_i and the average point.

Theorem 4.1. *Normalized Cuts finds a hyperplane that maximizes (3) with* $M(\beta) = M_{NCUT}(\beta) := \sum_{i=1}^{N} \frac{1}{\cos\theta_i} \left(\beta^\top X_i\right)^2$*, and assigns labels according to* $y_i = \operatorname{sgn}(\beta^\top X_i)$.

Proof. By the above discussion, we have $\sum_{i=1}^{N} \frac{1}{\cos\theta_i} \left(\beta^\top X_i\right)^2 = \|\beta^\top \mathbf{X}\mathbf{D}^{-\frac{1}{2}}\|^2$. With some algebra, it can be seen that $\beta^* = \mathbf{X}\mathbf{D}^{-\frac{1}{2}} v_2$, where v_2 is the second largest eigenvector of $\mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}$ [10]. From v_2, we can directly compute the signed distance between each point x_i and the hyperplane, without computing β^*, because:

$$\begin{aligned} \hat{y} &= \operatorname{sgn}\left(\beta^{*\top}\mathbf{X}\right) = \operatorname{sgn}\left(\beta^{*\top}\mathbf{X}\mathbf{D}^{-\frac{1}{2}}\right) = \operatorname{sgn}\left(v_2^\top \mathbf{D}^{-\frac{1}{2}}\mathbf{X}^\top \mathbf{X}\mathbf{D}^{-\frac{1}{2}}\right) \\ &= \operatorname{sgn}\left(v_2^\top \lambda_2\right) = \operatorname{sgn}\left(v_2^\top\right), \end{aligned}$$

where λ_2 is the second largest eigenvalue of $\mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}$ and v_2 is its corresponding eigenvector. This is identical to the Normalized Cuts labeling obtained from (2). □

The particular choice of the weight factor $1/\cos\theta_i$ in M_{NCUT} is an implicit design choice in the Normalized Cuts algorithm. It gives greater weight to points away from the lifted mean (remember that the lifted points X_i lie on a unit sphere, so that angles between vectors are a good measure of distance). This weighting appears to make Normalized Cuts sensitive to outliers, which is undesirable. The only benefit we see in this weighting is that finding a solution to equation (3) is simplified, because the second eigenvector of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$ automatically satisfies the balancing constraint of equation (4). Other weightings are possible and will be explored in later sections.

Since $\beta^* = \mathbf{X}\mathbf{D}^{-\frac{1}{2}} v_2$, c in Equation (5) is $\mathbf{D}^{-\frac{1}{2}} v_2$ Figure 4(left) demonstrates Normalized Cuts' sensitivity to an outlier. By sliding one outlier along the x-axis, the clustering boundary can be arbitrarily shifted to the left or to the right. Figure 4(right) shows that Normalized Cuts will split elongated structures, because according to its weighting, it is favorable to have points on opposite ends of an elongated structure land on opposite sides of the separating plane.

5 Average Gap Algorithm

If equal weight is given to every point, the outlier and splitting problems are attenuated. Consider the new gap measure

$$M_{avg}(\beta) = \frac{1}{N} \sum_{i=1}^{N} \left(\beta^\top X_i\right)^2 = \frac{1}{N} \beta^\top \mathbf{X}\mathbf{X}^\top \beta.$$

The optimizer of M_{avg} subject to the balancing and norm constraints is the top eigenvalue of a related matrix. See [10] for a proof of the following theorem.

Theorem 5.1. *The label assignments from (3) with* $M(\beta) = M_{avg}(\beta)$ *are* $\operatorname{sgn}(v)$ *where* v *is the largest eigenvector of* $\mathbf{K} - \frac{\mathbf{K}\mathbf{1}\mathbf{1}^\top\mathbf{K}}{\mathbf{1}^\top\mathbf{K}\mathbf{1}}$.

Section 7 shows that this new hyperplane algorithm works as well as Normalized Cuts, and is less susceptible to outliers. We will refer to it as the Average Gap algorithm. Compare Figure 5 with Figure 4. The outlier does not affect the clustering boundary, no matter how

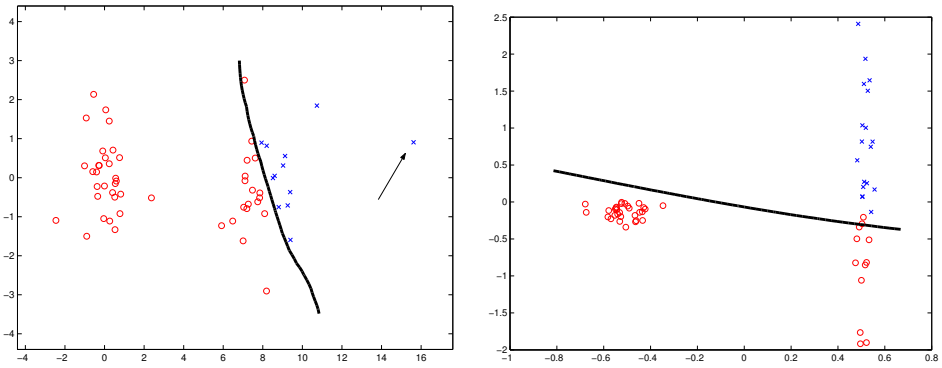

Figure 1: **Left:** In Normalized Cuts, an outlier can dwarf the influence of other points, because points away from the mean are heavily weighted. Sliding the outlier (indicated by the arrow) along the x-axis can shift the clustering boundary arbitrarily to the left or the right. Without the outlier, Normalized Cuts places the boundary between the two clusters. **Right:** Because Normalized Cuts puts more weight on points away from the mean, it prefers to have the ends of the elongated vertical cluster on opposite sides of the separating hyperplane.

far it is from the main body. Adding several outliers eventually does move the boundary (not shown). The elongated cluster is not split up. No stretching of the elongated cluster causes it to be split.

6 Relation to Transductive SVM

The transductive SVM [1] is a semi-supervised learning algorithm, but by withholding training labels and adding a constraint that favors balanced clusters, it provides a clustering criterion which we call the "clustering SVM". The transductive SVM problem is notoriously difficult to solve [11, 12, 13], as is the clustering SVM problem. We show that Normalized Cuts is a relaxation of the clustering SVM problem. However, solving the dual program for clustering SVM provides an even better relaxation.

The transductive SVM maximizes the distance to the point closest to the hyperplane, so its gap measure is $M_{CSVM}(\beta) = \min_i \beta^T X_i$. Substituting this gap into (3) yields the clustering SVM problem. Through the representer theorem we have $\beta = \mathbf{X}\mathbf{c}$, $\beta^\top X = \mathbf{c}^\top \mathbf{K}$, and $\|\beta\| = \mathbf{c}^\top \mathbf{K}\mathbf{c}$. With a standard manipulation, we obtain an equivalent formulation of the clustering SVM in terms of $\mathbf{c}$:

$$\begin{array}{ll} \min_{\mathbf{c}} & \mathbf{c}^\top \mathbf{K}\mathbf{c} \\ \text{s.t.} & (\mathbf{c}^\top K_i)^2 \geq 1, \qquad \mathbf{c}^\top \mathbf{K}\mathbf{1} = 0 \end{array} \tag{6}$$

This program is non-convex and is in general hard to solve exactly.

Proposition 6.1. *Normalized Cuts and the Average Gap algorithms are relaxations for the clustering SVM(6)*

Proof. The program (3) with the Normalized Cut gap M_{NCUT} and Average gap M_{avg} can be recast in a similar form similar to (6):

$$\begin{array}{ll} \min_{\mathbf{c}} & \mathbf{c}^\top \mathbf{K}\mathbf{c} \\ \text{s.t.} & \dfrac{1}{\sum_i w_i} \displaystyle\sum_{i=1}^{N} (\mathbf{c}^\top K_i)^2 w_i \geq 1, \qquad \mathbf{c}^\top \mathbf{K}\mathbf{1} = 0 \end{array} \tag{7}$$

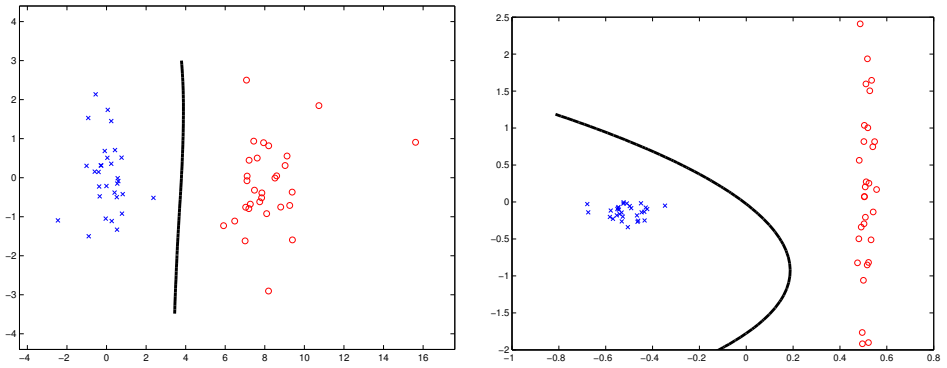

Figure 2: **Left:** The data set of Figure 4(left) is correctly segmented by weighting all points equally. The outlier point doesn't shift the clustering boundary significantly. **Right:** The data set of Figure 4(right) is correctly segmented by weighting all points equally.

where $w_i = 1$ for the Average Gap and $w_i = 1/\cos\theta_i$ for Normalized Cut gap. Since the constraints in (6) imply those of (7), algorithms that solve (7) are relaxations of (6). □

An alternative relaxation strategy is to solve the dual of (6), which is a semidefinite program. A priori it is not clear whether this dual strategy should provide a better relaxation for (6) than Normalized Cuts. The following theorem provides the necessary justification.

Theorem 6.1. *The dual of (6) achieves a higher cost than the relaxation (7).*

Proof. Since (6) is a non-convex quadratically constrained quadratic program its dual is a semidefinite program [14]:

$$\begin{aligned} \max_{\alpha \geq 0} \quad & \sum_{i=1}^{N} \alpha_i \\ \text{s.t.} \quad & \mathbf{c}^\top (\mathbf{K} - \mathbf{K}\mathrm{diag}\,(\alpha)\mathbf{K})\,\mathbf{c} \geq 0, \qquad \forall \mathbf{1}^\top \mathbf{K}\mathbf{c} = 0 \end{aligned} \tag{8}$$

The dual of (7) is:

$$\begin{aligned} \max_{\lambda > 0} \quad & \lambda \\ \text{s.t.} \quad & \mathbf{c}^\top (\mathbf{K} - \mathbf{K}\lambda\mathrm{diag}\,(\{w_i\})\mathbf{K})\,\mathbf{c} \geq 0 \quad \forall \mathbf{1}^\top \mathbf{K}\mathbf{c} = 0 \end{aligned} \tag{9}$$

There is no duality between (7) and this dual (9). The dual (8) optimizes over an arbitrary diagonal matrix, whereas the dual (9) optimizes over a more constrained diagonal matrix. Therefore (7) is a lower bound on the dual (8) of the clustering SVM problem. Since both of these are lower bounds on the primal clustering SVM problem (6), solving (8) yields a value closer to the optimum of (6) than does solving (7). □

$\mathbf{c}$ can be extracted in the dual from the null-space of $\mathbf{K} - \mathbf{K}\mathrm{diag}(\alpha^*)\mathbf{K}$ [4, 14]. The labeling becomes $\mathrm{sgn}(\mathbf{K}\mathbf{c})$. The theorem implies that extracting $\mathbf{c}$ from (8) is better than extracting $\mathbf{c}$ from Normalized Cuts or the Average Gap algorithms.

7 Numerical Experiments

We compared Normalized Cuts, the Average Gap algorithm, and the dual relaxation of clustering SVM on datasets from the UCI repository. The Wisconsin breast cancer data set

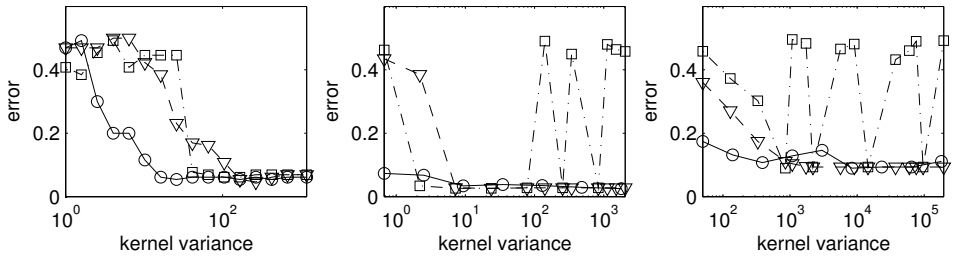

Figure 3: A plot of error as measured against given labels against the log of the variance of the gaussian kernel used for clustering. $\square$ represent the Normalized Cuts algorithm, $\triangledown$ represent the Average Gap algorithm, and $\circ$ represent the clustering SVM. Figure (left) is the `wine` data set, (middle) is the `cancer1` data set and (right) is the `cancer2` dataset.

dataset	n	σ^2	CSVM	NCUT	AVG
`wine`	130	4.90e3	0.939	0.931	0.931
`cancer1`	683	1.20e5	0.962	0.973	0.973
`cancer2`	569	4.16e6	0.907	N/A	0.907
`ionosphere`	351	2.49e2	0.692	0.704	0.704
`MNIST`	1000	4.82e9	0.854	0.748	0.748

Table 1: Clustering performance.

(`cancer1`) and the new diagnostic dataset (`cancer2`) were originally obtained from the University of Wisconsin Hospitals, Madison. We also used the first two classes in the wine recognition dataset (`wine`) and the ionosphere dataset (`ionosphere`). We retain the first 100 instances for each of the 10 classes in the `MNIST` data set.

In all of data sets, except for `MNIST`, clustering performance is the number of correctly assigned labels to each class. In the `MNIST` experiment, we evaluate the performance of the clustering algorithms by how much they split clusters. We count the percentage of disagreeing label assignments for each of the 10 classes, and averaged this number over the 10 classes. Because of the balancing constraint (4), none of the algorithms assigned the same label to the entire data set.

The semidefinite program for the clustering SVM was solved using the SeDuMi solver [15] and the YALMIP parser [16]. We were able to run sets of one hundred points in a few seconds and sets of one thousand points in about an hour on a dual Xeon 2.8 GHZ machine.

The clustering algorithms were all compared using a gaussian kernel. Figure 3 plots clustering performance against various choices of σ^2.

All three algorithms are sensitive to the choice of the kernel variance over a wide range. The semidefinite relaxation of the clustering SVM performs better over a wider range of σ^2. Furthermore, we note that the Normalized Cuts algorithm oscillates wildly for some datasets as σ^2 is varied. The labeling produced by Normalized Cuts in high error regions appeared erratic. High error rates were never due to separating outliers from the dataset. The clustering SVM and the Average Gap algorithm do not exhibit this problem.

Table 7 reports the performance of the algorithms for σ^2 in regions where the output of the algorithms did not change when changing σ^2. We found no such region for `cancer2` using Normalized Cuts.

8 Conclusion

We have provided a new interpretation for the Normalized Cuts relaxation of Shi and Malik and showed that it can be thought of as searching for a maximum gap hyperplane in a data set. In fitting this hyperplane, Normalized Cuts pays more attention to outliers, and so fails to recover sensible clusters in some cases. We showed how to avoid this pitfall by weighting all data points equally. In experiments, the correction of Normalized Cuts is more robust to changes in the kernel variance.

We showed that Normalized Cuts and the proposed correction are relaxations of a clustering version of the transductive SVM. We derived a semidefinite relaxation of this so-called clustering SVM and found that it slightly outperforms the spectral methods.

Our interpretation of Normalized Cuts can also be used to justify semi-supervised versions of it as well, although we did not explore this possibility in this paper.

References

[1] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[2] D. Verma and M. Meila. A comparison of spectral clustering algorithms. In *http://www.cs.washington.edu/research/spectral*, 2003.

[3] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *ACM Symposium on Theory of Computing*, 2004.

[4] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, (42):1115–1145, 1995.

[5] M. Girolani. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, May 2002.

[6] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[7] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[8] E.P. Xing and M.I. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report CSD-03-1265, Division of Computer Science, University of California, Berkeley, 2003.

[9] B. Schölkopf, R. Herbrich, A.J. Smola, and R.C. Williamson. A generalized representer theorem. Technical Report 81, NeuroCOLT, 2000.

[10] A. Rahimi and B. Recht. Clustering with normalized cuts is clustering with a hyperplane. In *Statistical Learning in Computer Vision workshop in ECCV*, 2004.

[11] T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, pages 200–209, 1999.

[12] K.P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 368–374, 1998.

[13] T. De Bie and N. Cristianini. Convex methods for transduction. In *Neural Information Processing Systems (NIPS)*, 2003.

[14] Yuri Nesterov, Henry Wolkowicz, and Yinyu Ye. Nonconvex quadratic optimization. In Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe, editors, *Handbook of Semidefinite Programming*, International series in operations research and management science, pages 361–420, Boston, MA, 2000. Kluwer Academic Publishers.

[15] J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. 11-12:625–653, 1999.

[16] J. Löfberg. YALMIP 3. Technical report, http://control.ee.ethz.ch/ joloef/yalmip.msql, 2004.