Clustering with Normalized Cuts is Clustering with a Hyperplane

Ali Rahimi[1] and Ben Recht[2]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
[2] MIT Media Laboratory,
Cambridge MA, 02139 USA
{ali,brecht}@mit.edu

Abstract. We present a set of clustering algorithms that identify cluster boundaries by searching for a hyperplanar gap in unlabeled data sets. It turns out that the Normalized Cuts algorithm of Shi and Malik [1], originally presented as a graph-theoretic algorithm, can be interpreted as such an algorithm. Viewing Normalized Cuts under this light reveals that it pays more attention to points away from the center of the data set than those near the center of the data set. As a result, it can sometimes split long clusters and display sensitivity to outliers. We derive a variant of Normalized Cuts that assigns uniform weight to all points, eliminating the sensitivity to outliers.

1 Introduction

The Normalized Cuts clustering algorithm of Shi and Malik [1] 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 spectral clustering algorithms [2]. The affinity matrix used in these algorithms is reminiscent of the Gram matrix that appears in kernel-based algorithms, such as the Support Vector Machine [3]. In this paper we show that Normalized Cuts lifts the data set to an infinite-dimensional feature space and cuts through 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.

A measure of gap can be any intuitive measure of distance between a plane and a set of points. The measure of gap implicitly used in Normalized Cuts focuses on the behavior of data points away from the mean of the data set, rather than heeding the behavior of points near the hyperplane itself. This weighting explains our observation that Normalized Cuts can sometimes break elongated clusters and why it is so sensitive to outliers.

By defining the gap as the average distance between the hyperplane and the data, we derive a clustering algorithm that does not exhibit these problems. Finding labels under this new gap reduces to thresholding the top eigenvector of a matrix.

Separating data with a hyperplane is a common technique in classification. Two examples are Support Vector Machines [3] and Fischer Linear Discriminants [4]. But classification differs from clustering in that the data are already labeled in classification, whereas in clustering, the data are unlabeled and the goal is to recover the labels. In section 7, we show how the SVM margin, typically used for classification, can be adapted to clustering.

Other clustering algorithms have been explicitly designed to operate in a lifted feature space. For example, [5, 6] perform approximate K-Means clustering in feature space. By fitting a hypersphere around the data set [7] or a plane between the data set and the origin [8], it is possible to learn a support function for the data set. This function is positive for test points that are similar to points in the training set and negative for test points that are not. These methods are not clustering algorithms per se because they do not recover the labels of the original data set. But Ben-Hur et. al [9] observed that the resulting support functions form closed contours around clusters of the data. They devised an algorithm for labeling the data based on these contours. By contrast, according to our interpretation, Normalized Cuts directly searches for a hyperplanar gap in the data set.

This paper only presents 2-way clustering algorithms. As with Normalized Cuts, if more clusters are sought, each 2-way cut can be further subdivided by running the clustering procedure recursively.

2 Normalized Cuts

Given a set of data points $\mathbf{x} = \{x_i | x_i \in \mathcal{R}^d, i \in 1..N\}$, and a distance measure $k(x, y)$, build the adjacency matrix $K_{ij} = k(x_i, x_j)$. This adjacency matrix is the starting point for many other affinity-based (aka spectral) clustering algorithms. A common choice for k is $k(x, y) = \exp\left(-\frac{\|x_i - y_i\|^2}{\sigma^2}\right)$. It is perhaps by accident that the distance measures used in spectral clustering are often postive definite function. But this choice allows us to invoke the kernel trick in section 3 and to interpret the operations of Normalized Cuts in a lifted space.

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 $\text{cut}(A, B)$ is the sum of the weight of the edges between vertices in A and vertices in B. The aim 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 cut is NP hard, so

Normalized Cuts resorts to a relaxation of the above:

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

$\mathbf{D}$ 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[1] of $\mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}}$. The components of v^* are then thresholded to yield a vector in $\{-1, 1\}^N$:

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

This is the labeling as reported by Normalized Cuts. Throughout this paper, we refer to this relaxation as the Normalized Cuts algorithm and interpret it as a separating hyperplane method. Other relaxations are possible [10], but we do not provide a separating hyperblane interpretation for these relaxations.

3 Lifting the Data Set

The kernel trick is a standard way of lifting the points of a data set to a high dimensional space [3]. Although there may be no way to separate N d-dimensional data points with a hyperplane in $\mathcal{R}^d$, it is often possible to do so in the lifted space.

Consider a positive definite function $k(x, y)$ of two vectors in $\mathcal{R}^d$. This is a Mercer kernel with expansion

$$k(x, y) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(y), \tag{3}$$

This expansion defines a lifting of a data point x in "input space" to a possibly infinite dimensional vector X in "feature space" via $X^j = \sqrt{\lambda_j} \phi_j(x)$. The inner product in feature space is defined so that $< X_i, X_j >_k = k(x_i, x_j)$.

A common choice for the kernel k is the Gaussian $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right)$. In many cases, including the Gausssian case, $\|X\|^2 = k(x, x) = 1$, 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. For analysis on the structure of feature space see [3].

Recall that the affinity matrix $\mathbf{K}$ of spectral clustering 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 $\mathbf{X} = \{X_i\}$.

[1] We casually refer to the nth eigenvector of a matrix in this paper. This is meant as a shorthand for the eigenvector of the matrix corresponding to the nth largest eigenvalue.

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 lie in a sphere in feature space, we think of D_{ii} as a distance between X_i and the average point.

We've shown that the $\mathbf{K}$ and $\mathbf{D}$ matrices of Normalized Cuts have geometric meaning in feature space. We now show how these quantities can be used cleave the data with a hyperplane in feature space.

4 Clustering with a Hyperplane

A simple clustering idea is to find a hyperplane that passes through the lifted data set with as great a distance to the data points as possible. We denote this distance as the "gap" because, ideally, a hyperplane lying in a large gap in the data set would have the largest distance to the points. Once a hyperplane is fitted, points that fall on the same side of the hyperplane will be labeled as being in the same cluster. See Figure 1.

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: $y = \mathrm{sgn}(\beta^\top X)$.

Let's find a hyperplane that maximizes the following measure of gap:

$$M_{NCUT}(\beta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\cos\theta_i} \left(\beta^\top X_i\right)^2 = \|\beta^\top \mathbf{X} \mathbf{D}^{-\frac{1}{2}}\|^2. \tag{4}$$

This is a weighted average of the distance between each point and the plane. The particular choice of the weight factor $1/\cos\theta_i$ is an implicit design choice in Normalized Cuts. Other weightings are possible. For example, in section 6 we examine the case when distances are weighed uniformly. For the time being, notice that this choice of weights gives greater weight to points away from the mean (remember that the lifted data set $\mathbf{X}$ lies on a unit sphere, so that angles between vectors are a good measure of distance). In section 5, we show that this particular weighting is actually detrimental to the performance of Normalized Cuts.

We would like to find a hyperplane in feature space, parametrized by its (possibly infinite dimensional) normal vector, to maximize $M_{NCUT}(\beta)$. To insure 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

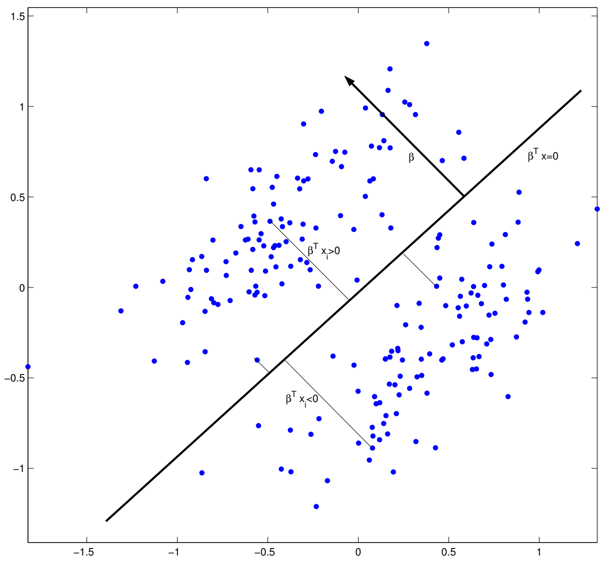

Fig. 1. A separating hyperplane cleaves the data set. The hyperplane is represented by its normal β. The hyperplane is chosen to maximize a gap, which, in the case of Normalized Cuts is defined as the weighted sum of the squared distances between the hyperplane and the data points.

optimal β is

$$\begin{aligned} \beta^* = \arg\max\nolimits_\beta \; & M_{NCUT}(\beta) \\ \text{s.t.} \quad & \|\beta\| = 1, \\ & \beta^\top \bar{X} = 0 \end{aligned} \tag{5}$$

This is the same as maximizing a Rayleigh quotient subject to a constraint:

$$\beta^* = \arg\max\nolimits_\beta \frac{\beta^\top \mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top \beta}{\beta^\top \beta} \tag{6}$$

$$\text{s.t.} \quad \beta^\top \bar{X} = 0. \tag{7}$$

The unconstrained Rayleigh quotient is maximized by the largest eigenvector of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$. It is easy to verify that this eigenvector is $\bar{X}$. But the constraint (7) requires us to choose a β perpendicular to this eigenvector. Since $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$ is positive semi-definite, the optimal β that satisfies the constraint is the eigenvector of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$ corresponding to its second largest eigenvalue.

Because $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$ is a potentially infinite-dimensional matrix, we can't compute its eigenvectors directly. But we can represent them in terms of the eigenvectors of the related matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{X}^\top\mathbf{X}\mathbf{D}^{-\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}$: If v_2 is the second eigenvector of $\mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}$, then $\beta^* = \mathbf{X}\mathbf{D}^{-\frac{1}{2}}v_2$ is the second eigenvector of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$.

From v_2, we can directly compute the signed distance between each point x_i and the hyperplane, without computing β^*:

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

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).

5 Discussion of Normalized Cuts

We have shown that Normalized Cuts and the hyperplane clustering method of the previous section perform identical operations. This means that any geometric intuition we might glean from the hyperplane algorithm carries over to Normalized Cuts.

A hyperplane in feature space corresponds to a function in input space. 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 $y(x_0) = \beta^* X_0$. Since $\beta^* = \mathbf{X}\mathbf{D}^{-\frac{1}{2}}v_2$, we can write

$$y(x_0) = \alpha^\top \mathbf{X}^\top X_0 = \sum_{i=1}^{N} \alpha_i k(x_i, x_0), \tag{8}$$

with $\alpha = v_2^\top \mathbf{D}^{-\frac{1}{2}}$.

So not only can we recover the labels of the given data set, we can also find the splitting function $y(x)$ which returns the distance to the hyperplane. We show these separating functions in figures throughout this paper.

One might suspect that clustering with a hyperplane might not always work. What if the data isn't even separable by a hyperplane? We've also stipulated that the hyperplane must go through the origin. Could this force certain clusters that straddle the origin to be split? Because we're operating in a high dimensional feature space, point clouds are easier to separate with a hyperplane. Even clouds that straddle the origin can be split by hyperplane. One way to see this is to observe that the splitting function in input space $y(x_0)$ is not a line.

Figure 2 shows that even clusters that straddle the origin in input space and require closed contours to be separated from other clusters can be isolated.

The presence of a weighting factor of $1/\cos\theta_i$ in equation (4) was surprising to us. 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 (5) is simplified, because the second eigenvector of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$ automatically satisfies the balancing constraint of equation (7). Figure 3 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

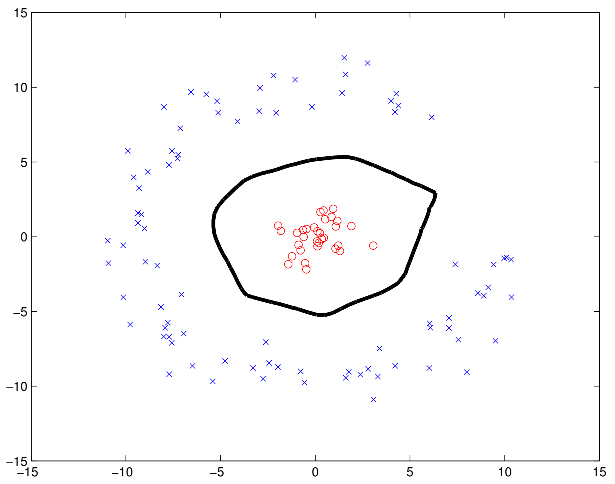

Fig. 2. A hyperplane containing the origin in feature space can separate data sets that are not separable by a hyperplane containing the origin in the input space. Normalized Cuts identifies the inner circle and the ring as two separate clusters despite them not being linearly separable in input space. The black line is the hyperplane back-projected into input space.

the right. Figure 4 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.

In the next sections, we provide two alternative definitions for $M(\beta)$. The one in the following section gives equal weights to all points while retaining the balancing constraint of equation (7). The one in section 7 uses the SVM margin.

6 Average Cost Gap

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

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

We can maximize this gap subject to the same constraints as before:

$$\begin{aligned} \max\nolimits_\beta \ & M_{avg}(\beta) \\ \text{s.t.} \ \ & \|\beta\| = 1, \\ & \beta^\top \bar{X} = 0 \end{aligned}$$

This time however, the top eigenvector will not be $\bar{X}$, so the balancing constraint will not automatically be satisfied by the second eigenvector.

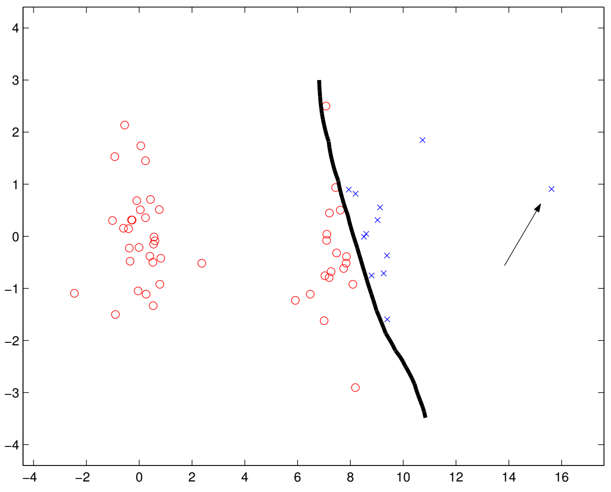

Fig. 3. In Normalized Cuts, an outlier can dwarf the influence of other points, because points away from the mean are heavily weighted. Sliding the outlier (indicated by the arrow) along the x-axis can shift the clustering boundary arbitrarily to the left or the right. Without the outlier, Normalized Cuts places the boundary between the two clusters.

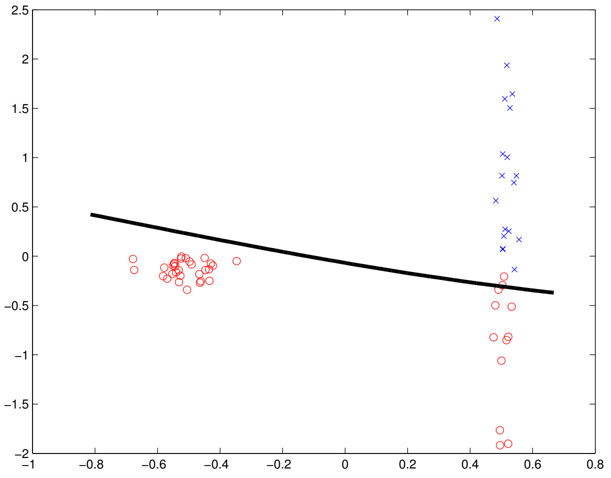

Fig. 4. Because Normalized Cuts puts more weight on points away from the mean, it prefers to have the ends of the elongated vertical cluster on opposite sides of the separating hyperplane.

The balancing constraint forces us to search for a β orthogonal to $\bar{X}$. The symmetric projection matrix $\mathrm{Z} = \mathbf{I} - \bar{X}\bar{X}^\top / \|\bar{X}\|^2$ spans the space of vectors orthogonal to $\bar{X}$, and has $\bar{X}$ in its null-space. Because $\mathbf{Z}$ is a projection matrix, it is idempotent, so $\mathbf{ZZ} = \mathbf{Z}$. We can eliminate the balancing constraint by forcing β to lie in the span of $\mathbf{Z}$. We can do this by premultiplying β by $\mathbf{Z}$ wherever it appears in the optimization problem:

$$\begin{aligned} \max\nolimits_\beta \; & \beta^\top \mathbf{Z}^\top \mathbf{X}\mathbf{X}^\top \mathbf{Z}\beta \\ \text{s.t.} \; & \beta^\top \mathbf{Z}^\top \mathbf{Z}\beta = 1. \end{aligned}$$

This is the same as finding an eigenvector corresponding to the largest λ in the generalized eigenvalue problem:

$$\mathbf{Z}^\top \mathbf{X}\mathbf{X}^\top \mathbf{Z}\beta = \lambda \mathbf{Z}\beta, \tag{9}$$

where we have taken advantage of the symmetry and idempotency of $\mathbf{Z}$. If v is the largest eigenvector of

$$\mathbf{X}^\top \mathbf{Z}\mathbf{X} = \mathbf{K} - \frac{\mathbf{K}\mathbf{1}\mathbf{1}^\top \mathbf{K}}{\mathbf{1}^\top \mathbf{K}\mathbf{1}},$$

then $\beta^* = \mathbf{Z}^\top \mathbf{X} v$ is the largest generalized eigenvector in equation (9).

To label the data points, we can just threshold v:

$$\hat{y} = \mathrm{sgn}(\beta^{*\top}\mathbf{X}) = \mathrm{sgn}(v^\top \mathbf{X}^\top \mathbf{Z}\mathbf{X}) = \mathrm{sgn}(v^\top)$$

.

Operationally, whereas Normalized Cuts thresholds the second largest eigenvector of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}^\top$, using an unweighted gap results in an algorithm that thresholds the largest eigenvector of $\mathbf{K} - \frac{\mathbf{K}\mathbf{1}\mathbf{1}^\top \mathbf{K}}{\mathbf{1}^\top \mathbf{K}\mathbf{1}}$. This new hyperplane algorithm works as well as Normalized Cuts, and is less susceptible to outliers. Compare Figure 5 with Figure 3. The outlier does not affect the clustering boundary, no matter how far it is from the main body. Adding several outliers eventually does move the boundary (not shown). Also compare Figure 6 with Figure 4. The elongated cluster is not split up. No stretching of the elongated cluster causes it to be split.

7 A Better Margin?

We have been using a somewhat simplistic definition of gap by averaging the distance to every data point. We could instead measure the distance only to the points nearest the hyperplane. Using such a gap would provide greater robustness to outliers and provide a concise summary of the data set in terms of support vectors.

Just as with an SVM, we would like to maximize the margin C:

$$\begin{aligned} \max\nolimits_\beta \; & C \\ \text{s.t.} \; & |\beta^\top X_i| \geq C, \\ & \|\beta\| = 1 \end{aligned}$$

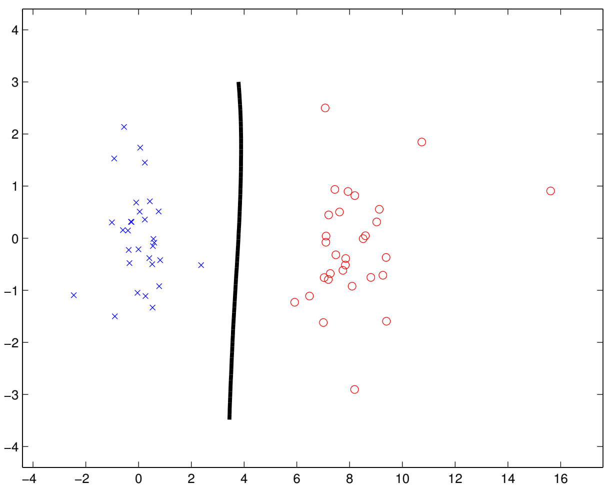

Fig. 5. The data set of Figure 3 is correctly segmented by weighting all points equally. The outlier point doesn't shift the clustering boundary significantly.

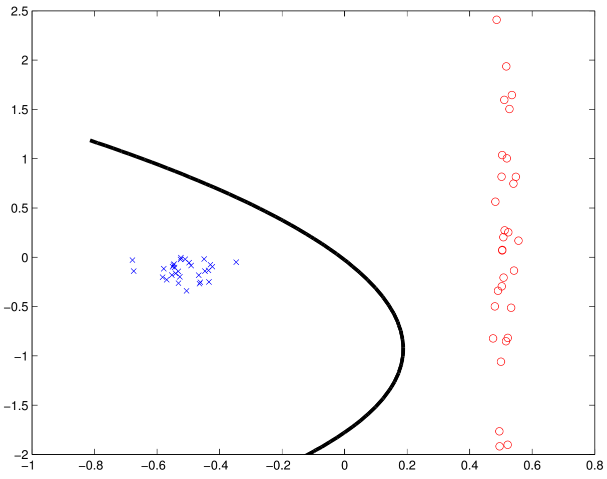

Fig. 6. The data set of Figure 4 is correctly segmented by weighting all points equally.

The presence of the absolute value and the absence of labels in the constraints is the main difference from the SVM constraints. This optimization problem seeks the largest number C to maximize the unsigned distance between data points and the hyperplane.

We can remove the normality requirement on β by changing the constraints:

$$\begin{aligned} &\max_\beta \; C \\ &\;s.t. \;\; |\beta^\top X_i| \geq C\|\beta\| \end{aligned}$$

To see that this optimization problem is difficult, set $\|\beta\| = 1/C$

$$\begin{aligned} &\min_\beta \; \|\beta\|^2 \\ &\;s.t. \;\; |\beta^\top X_i| \geq 1 \end{aligned}$$

This is a quadratic program, except the constraint space is not convex! The SVM constraints $y_i\beta^\top x_i > 1$ are convex because y_i are known. When the y_i are not known, we must consider both negative and postive label assignments, which makes this problem hard.

If we restrict ourselves to solutions that are linear in the data set, we obtain a finite, non-convex problem:

$$\begin{aligned} &\min_\alpha \; \alpha^T\mathbf{K}\alpha \\ &\;s.t. \;\; \alpha^\top K_i^\top K_i\alpha \geq 1, \end{aligned}$$

where K_i is the ith column of $\mathbf{K}$. We have substituted $\beta = \mathbf{X}\alpha$ and squared the constraint $\alpha^\top\mathbf{X}X_i \geq 1$ to obtain a quadratic constraint. We are currently exploring relaxations for solving such a non-convex optimization problem.

8 Conclusion

We have provided a new interpretation for the Normalized Cuts relaxation of Shi and Malik. We showed that this algorithm can be through of as searching for a maximum gap hyperplane in a data set. In fitting this hyperplane, it 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.

We have limited ourselves to experiments with relatively few data points until now. We are currently tuning the algorithms of sections 6 and 7 to handle image-sized data sets to estimate their applicability to image segmentation.

References

1. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
2. D. Verma and M. Meila. A comparison of spectral clustering algorithms. In *http://www.cs.washington.edu/research/spectral*, 2003.

3. S. Schölkopf, B. Mika, S. Burges, C. Knirsch, P. Miiller, K. itsch, and G. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.
4. R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
5. R. Zhang and A. I. Rudnicky. A large scale clustering scheme for kernel k-means. In *ICPR*, 2002.
6. M. Girolani. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, May 2002.
7. B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12 (NIPS)*, 2000.
8. D.M.J. Tax and R.P.W. Duin. Outliers and data descriptions. In *Proceedings of the Seventh Annual Conference of the Advanced School for Computing and Imaging (ASCI)*, June 2001.
9. A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.
10. E.P. Xing and M.I Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report CSD-03-1265, 2003.