Clustering with Local and Global Consistency

Markus Breitenbach
Department of Computer Science
University of Colorado, Boulder
Markus.Breitenbach@colorado.edu

Gregory Z. Grudic
Department of Computer Science
University of Colorado, Boulder
grudic@cs.colorado.edu

Abstract

Clustering aims at finding hidden structure in data. In this paper we present a new clustering algorithm that builds upon the local and global consistency method (Zhou, et.*al.*, 2003), a semi-supervised learning technique with the property of learning very smooth functions with respect to the intrinsic structure revealed by the data. Starting from this algorithm, we derive an optimization framework that discovers structure in data without requiring labeled data. This framework is capable of simultaneously optimizing all learning parameters, as well as picking the optimal number of clusters. It also allows easy detection of both global outliers and outliers within clusters. Finally, we show that the learned cluster models can be used to add previously unseen points to the clusters, without re-learning the original cluster model. Encouraging experimental results are obtained on a number of toy and real world problems.

1 Introduction

Clustering aims at finding hidden structure in a dataset and is an important topic in machine learning and pattern recognition. The problem of finding clusters that have a compact shape has been widely studied in the literature. One of the most widely used approaches is the K-Means [1] method for vectorial data. Despite the success these methods have with real life data, they fail to handle data that exposes a manifold structure, i.e. data that is not shaped in the form of point clouds, but winds through a high-dimensional space.

In this paper we present a new clustering algorithm based on the local and global consistency method, a semi-supervised learning technique [2] that has demonstrated impressive performance on relatively complex manifold structures. The idea in semi-supervised learning (or transduction) is to use both labeled and unlabeled data to obtain classification models.

This paper extends this local and global consistency algorithm to unsupervised learning by showing that it naturally lead to an optimization framework that picks clusters on manifolds by minimizing the mean distance between points inside a cluster, while maximizing the mean distance between points in different clusters. We further demonstrate that this optimization framework can simultaneously choose model all parameters, including the number of clusters. To the best of our knowledge, the proposed algorithm is unique in this respect. Other key aspects of the proposed algorithm include automatic global outlier detection (i.e. what points in manifold space are furthest way away from all other points) and cluster outlier detection (what points within a manifold cluster are most on it's extremes). Finally, we demonstrate that we can build a clustering model with one set of points and use

this model to cluster a second, as yet unseen, set of points (without rebuilding the original cluster).

The theoretical formulation for the proposed clustering algorithm is given in Section 2, which also presents a fast heuristic procedure for solving the proposed optimization problem. Section 3 presents detailed experimental results on both synthetic and real data. Section 4 concludes with future work.

The code implementing the proposed clustering algorithm is available at `http://ucsu.colorado.edu/~breitenm/clustering.html`.

2 Algorithm

2.1 Semi-Supervised Learning

In [2] Zhou et.al. introduced the consistency method, a semi-supervised learning technique. We will give a brief summary of the technique here.

Given a set of points $X \in \mathcal{R}^{n\times m}$ and labels $\mathcal{L} = \{1, \cdots, c\}$. Let x_i denote the ith example. Without loss of generality the first l points $(1 \cdots l)$ are labeled and the remaining points $(l+1 \cdots n)$ unlabeled. Define $Y \in \mathcal{N}^{n\times c}$ with $Y_{ij} = 1$ if point x_i has label j and 0 otherwise. Let $\mathcal{F} \subset \mathcal{R}^{n\times c}$ denote all the matrices with nonnegative entries. A matrix $F \in \mathcal{F}$ is a matrix that labels all points x_i with a label $y_i = \arg\max_{j\leq c} F_{ij}$. Define the series $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ with $F(0) = Y, \alpha \in (0,1)$. The entire algorithm is defined as follows:

1. Form the affinity matrix $W_{ij} = exp(-\|x_i - x_j\|^2/(2\sigma^2))$ if $i \neq j$ and 0 otherwise.
2. Compute $S = D^{-1/2}WD^{-1/2}$ with $D_{ii} = \sum_{j=1}^{n} W_{ij}$ and $D_{ij} = 0, i \neq j$.
3. Compute the limit of series $\lim_{t\to\infty} F(t) = F^* = (I - \alpha S)^{-1}Y$. Label each point x_i as $\arg\max_{j\leq c} F^*_{ij}$.

The regularization framework for this method follows. The cost function associated with the matrix F with regularization parameter $\mu > 0$ is defined as

$$\mathcal{Q}(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij}\left\|\frac{1}{\sqrt{D_{ii}}}F_i - \frac{1}{\sqrt{D_{jj}}}F_j\right\|^2 + \mu\sum_{i=1}^{n}\|F_i - F_j\|^2\right) \tag{1}$$

The first term is the smoothness constraint that associates a cost with change between nearby points. The second term, weighted by μ, is the fitting constraint that associates a cost for change from the initial assignments. The classifying function is defined as $F^* = \arg\min_{F\in\mathcal{F}} \mathcal{Q}(F)$. Differentiating $\mathcal{Q}(F)$ one obtains $F^* - \frac{1}{1+\mu}SF^* - \frac{\mu}{1+\mu}Y$. Define $\alpha = \frac{1}{1+\mu}$ and $\beta = \frac{\mu}{1+\mu}$ (note that $\alpha + \beta = 1$ and the matrix $(I - \alpha S)$ is non-singular) one can obtain

$$F^* = \beta\,(I - \alpha S)^{-1}\,Y \tag{2}$$

For a more in depth discussion about the regularization framework and on how to obtain the closed form expression F^* see [2].

2.2 Clustering with Local and Global Consistency

From equation (2), it is evident that the solution to the semi-supervised learning problem only depends on the labels after the the matrix $(I - \alpha S)$ has been inverted. This matrix only contains the training data inputs, $\{x_1, ..., x_n\}$, and it is this property that we will exploit to derive our clustering algorithm. We define a matrix U as:

$$U = \beta\,(I - \alpha S)^{-1} = \left[u_1^T, ..., u_n^T\right] \tag{3}$$

and note that U defines a graph or diffusion kernel as described in [3, 4]. In addition, the columns of U, denoted by u_i^T, define distances between training points on these graphs,

which can be interpreted as distances along a manifold [5]. The ordering of these distances along each manifold is maintained independent of scaling. From U, we create a new matrix V, by scaling the columns of U to have unit length. We define this V matrix as:

$$V = \left[u_1^T \left\|u_1^T\right\|^{-1}, ..., u_n^T \left\|u_1^T\right\|^{-1}\right] = \left[v_1^T, ..., v_n^T\right] \tag{4}$$

Note that, by definition, $||v_i|| = 1$. Finally, we define a distance (along a manifold specified by U) between points x_i and x_j to be:

$$d_M\left(x_i, x_j\right) = 1 - v_i v_j^T \tag{5}$$

The intuition behind this distance measure is that two points on a manifold are identical, if the order of distances between all other points in the training set is identical and the relative distances are identical. If this is the case for points x_i and x_j, then $d_M(x_i, x_j) = 0$. Conversely, if the point x_i has completely different distances along U to other points in the training data than point x_j, then $d_M(x_i, x_j)$ will approach 1. This leads to our definition of a distance matrix:

$$D_M = 1 - \begin{pmatrix} v_1 v_1^T & \dots & v_1 v_n^T \\ \vdots & \ddots & \vdots \\ v_n v_1^T & \cdots & v_n v_n^T \end{pmatrix} = \begin{pmatrix} d_M\left(x_1, x_1\right) & \dots & d_M\left(x_1, x_n\right) \\ \vdots & \ddots & \vdots \\ d_M\left(x_n, x_1\right) & \cdots & d_M\left(x_n, x_n\right) \end{pmatrix} \tag{6}$$

Given this definition of similarity between any two points x_i and x_j, our formulation of clustering is as follows. In clustering, we want to pick clusters of points that are most similar to one another, while at the same time most different to points in other clusters. We start by assuming there are c clusters and that each cluster is characterized by a single point. Thus, for c clusters, we have $x_{l_1}, ..., x_{l_c}$ points, where $x_{l_i} \in \{x_1, ..., x_n\}$ is a point in the training data, and $x_{l_i} \neq x_{l_j}$ for $i \neq j$. These points $x_{l_1}, ..., x_{l_c}$ determine clusters as follows. We define a n by c matrix F_V^* by taking the $l_1, ..., l_c$ columns of V (see equation (4)):

$$F_V^* = \left[v_{l_1}^T, ..., v_{l_c}^T\right] \tag{7}$$

Then, as with semi-supervised learning, we assign a point x_i to a class:

$$y_i = \arg\max_{j \leq c} F_{Vij}^* \tag{8}$$

where F_{Vij}^* is the entry of F_V^* given by row i and column j.

2.3 Model Selection For Clustering

Next, let $\mathbf{p}_j$ be the set of points that belong to cluster j. Using matrix D_M we can define the mean distance between points in cluster j as:

$$\overline{D}_M^{jj} = E\left[D_M\left(\mathbf{p}_j, \mathbf{p}_j\right)\right]$$

where $D_M(\mathbf{p}_j, \mathbf{p}_j)$ denotes all entries of D_M corresponding to columns and rows of points $\mathbf{p}_j$ and $E[]$ is the average value of these. Similarly, the mean distance between points in cluster j and points in cluster k is given by:

$$\overline{D}_M^{jk} = E\left[D_M\left(\mathbf{p}_j, \mathbf{p}_k\right)\right]$$

Given that our goal is to find clusters that maximize the distances between points in different clusters, while minimizing the distances between points in the same cluster, we can now state the optimization problem we are solving. Specifically, we want to find σ, $alpha$, c, and $x_{l_1}, ..., x_{l_c}$ to maximize the following:

$$\Omega\left(c\right) = \max_{\alpha, \sigma, c} \left[E\left[\overline{D}_M^{jk}\right]_{\left\{\substack{k=1,...,c \\ j=1,...,c \\ i \neq j}\right\}} - E\left[\overline{D}_M^{jj}\right]_{\{j=1,...,c\}}\right] \tag{9}$$

2.4 Outlier Detection

We define a cluster independent outlier point to be one that is, on average, furthest away to all other points. This can be directly calculated from equation (6) by taking the average of the columns of D_M as follows and defining a outlier cluster independent vector O_d as follows:

$$O_d = \frac{1}{n}\left[\sum D^T_{M1}, ..., \sum D^T_{Mn}\right] = [O_{d1}, ..., O_{dn}] \tag{10}$$

where the element O_{di} is the average distance (in manifold space) between point x_i and all the other points and $D_M = \left[D^T_{M1}, ..., D^T_{Mn}\right]$. Thus by ordering the vales of O_{di} in increasing order, we order the points from furthest to closest, and the points appearing first in the list constitute the outliers.

Similarly, we can find outliers within a cluster j by looking at the $D^{jj}_M = D_M(\mathbf{p}_j, \mathbf{p}_j)$ matrix defined above. Specifically, we obtain an outlier O^j_d vector for cluster j as follows:

$$O^j_d = \frac{1}{n}\left[\sum D^{jjT}_{M1}, ..., \sum D^{jjT}_{Mn}\right] = \left[O^j_{d1}, ..., O^j_{dn}\right] \tag{11}$$

where O^j_{di} is the mean distance of x_j to all other points in its cluster. Thus the point which has maximum O^j_{di} is the one which is *most inside* the cluster, while the point that has minimum O^j_{di} is *most outside* of the cluster.

2.5 Finding Points That Define a Cluster

As outlined above, the points $x_{l_1}, ..., x_{l_c}$ are used to specify clusters. These points can be identified by looking at the cluster independent outlier vector O_d defined above. In this paper we use the following greedy heuristic to identify $x_{l_1}, ..., x_{l_c}$. First we assign x_{l_1} to the point that is closest to all other points, which is defined by the point that has the largest value O_{di}. To find x_{l_2}, we multiply each element of O_d by the corresponding element in the column vector $D^T_{Ml_1}$, to obtain an new, re-weighted vector of O^2_d as follows: $O^2_d = \left[O^1_{d1}D^T_{Ml_1}(1), ..., O^1_{dn}D^T_{Ml_1}(n)\right] = \left[O^2_{d1}, ..., O^2_{dn}\right]$ where $O^1_{di} = O_{di}$. The point x_{l_2} then corresponds to the point which has maximum O^2_{di}. The selection procedure for re-weighting a point continues until c points are found. An example of how the mean distances change for each step is given in figure (3.2 a-c).

2.6 Clustering New Points

In order to cluster a new point without adding it to S and re-inverting the matrix $(I - \alpha S)$, we once more use the property that two points are similar if they have similar distances to all other points. However, this time we measure similarity using the S matrix as follows. Given a point x_k, we calculate $W_{kj} = exp(-\|x_k - x_j\|^2/(2\sigma^2))$, for $j = 1, ...n$ and obtain a vector W_k. We then calculate the $D_k = \sum_{j=1}^n W_k(j)$ and compute the vector in the S matrix that is associated with x_k), as $S_k = D_k^{-1/2} W D^{-1/2}$. Finally we normalize S_k to have length 1 and call it S^1_k and similarly normalize the rows of S to also have length 1, denoting this matrix by S^1. We then obtain a set of coefficients $\Theta = (\theta_1, ..., \theta_n)^T = S^1(S^1_k)^T$. This vector has the property that if $x_k = x_i$, then $\theta_i = 1$, but if x_k is very far away from x_i then θ_i will approach zero. Therefore, θ_i measures the closeness of x_k to x_i in S matrix space (with $\theta_i = 1$ being really close and $\theta_i = 0$ really far). We use this property to assign x_k to a cluster by creating an $F_k = [v_1\Theta^T, ..., v_n\Theta^T]$ and assigning $y_c = \arg\max_{j \le c} F_k$.

2.7 Implementation Details

A Matlab implementation of this algorithm can be obtained from `http://ucsu.colorado.edu/~breitenm/clustering.html`. The optimization problem in equation (9) can be solved as follows. For $c = 1, ..., c_{max}$, find σ, α, and $x_{l_1}, ..., x_{l_c}$ that maximize $\Omega(c)$ and then choose the value of choosing the c which has maximum $\Omega(c)$. As this is computationally intensive we choose an approximation of this algorithm for the current paper. Specifically, we maximized (9) for σ by assuming that there is only one cluster and assigning $\alpha = 0.99$ - this amounts to a 1-D optimization. If the experiment required optimizing for σ and $alpha$ together, we did a 2D optimization using the optimal values to find the c that maximized $\Omega(c)$. If the experiment didn't optimize for α, we simply found the c that maximized $\Omega(c)$ without changing the current value of σ or α. This clearly is a suboptimal algorithm and an open research question that we are addressing is to improve it.

3 Experimental Results

We evaluate our method using both toy data problems and real world data. In all the problems the desired assignment to the classes is known and we use this to report an error rate. The parameters σ, α and C are found by using our algorithm unless noted otherwise. We evaluate the assignment to clusters by computing an error rate, i.e. given the correct number of clusters, how many examples are assigned to the wrong cluster. Since the clusters may be discovered in a different order than originally labeled (e.g. clusters are discovered in order $3, 2, 1$ instead of $1, 2, 3$), we use the permutation on the algorithm's label assignments that results in the lowest error rate.

3.1 Toy Data

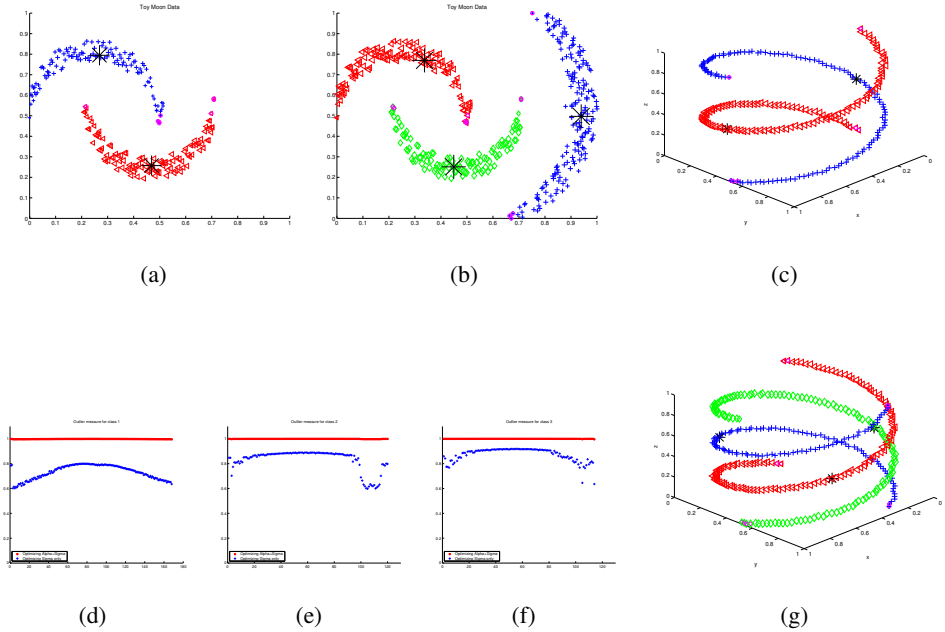
In this experiment we consider the moons toy-problem as depicted in figure 1 (b). We allowed σ and α to be optimized as described in section 2.7. Using all three moons ($\sigma = 0.0354, \alpha = 0.9999$) we see that our algorithm perfectly determined the number of clusters as described in section 2.7. The three classes have been separated out perfectly with no errors. The centroid points determined by our algorithm have been marked with a star. The size of the dots are proportional to their largest value in F^*. We can see that these three points have the maximum distance from each other. The outliers of each class is denoted as circles and are located furthest away from the class centroid at the ends of each of the moons. If we allow only the optimization of σ and set $\alpha = 0.99$, we see how the outlier measurements in figure 1 (d)-(f) change to a lower value, i.e. the separation of the data becomes more difficult. In this case σ was determined to be 0.0553.

We also use the spiral data that was used in [6]. The algorithm determined $\sigma = 0.0724$ and $\alpha = 0.9989$ for the two spirals. The number of clusters was correctly determined as $C = 2$. In the case of three spirals the heuristic splits each spiral into two clusters so we had to provide the number of clusters in this case. Outliers are located at the end of the spirals since the centroid of each spiral is in the middle.

Note that these toy data problems can not be clustered in a meaningful way by methods that assume a compact shape for the data like K-means [2, 6].

3.2 USPS Digits Data

In this experiment we address a classification task using the USPS dataset. The set consists of 16x16 images of handwritten digits. We use digits 1, 2, 3, and 4 in our experiments with the first 200 examples from the training set and the following 200 examples as unseen examples that will be added to the clusters.



(a) (b) (c)

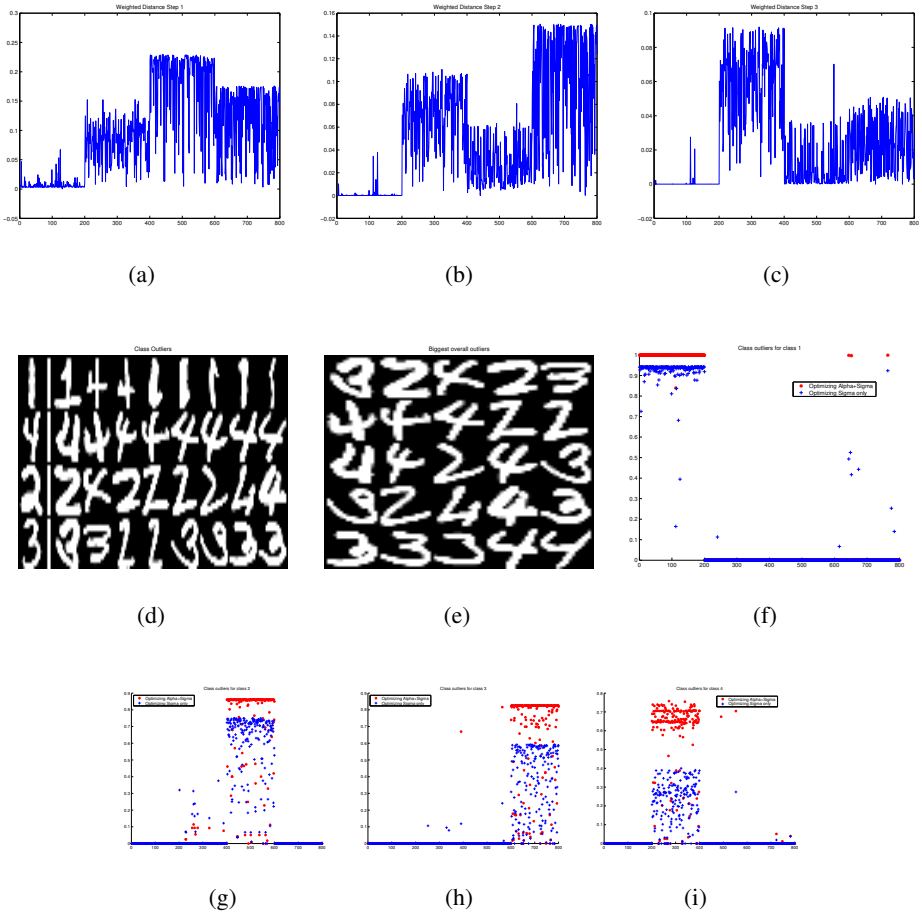
(d) (e) (f) (g)

Figure 1: Toy Data Experiments: (a) two clusters with (σ, α, C) determined by the algorithm; (b) three clusters were found with (σ, α, C) determined by the algorithm. The moons were labeled in the order they were discovered; (c) spiral data with (σ, α, C) determined by the algorithm; (d)-(f) the outlier measurements for optimization of α and σ vs. optimization of σ only for the three moons; (g) three spirals with (σ, α) determined by the algorithm

In figure 2 (a)-(c) we can see how the mean difference develops in each step. In each step the point with the largest distance is chosen, the points are re-weighted and the process is repeated as described in section 2.5.

We first use our algorithm to discover the different classes and label one example of each class. Using the consistency method the remaining points were labeled. Our algorithm determined $\sigma = 0.811396$ and $\alpha = 0.999981$. The number of clusters was not fixed and determined to be $C = 4$. This results in an error rate of 0.0238 which is a slightly better error rate than the consistency method had with 4 marked points. The outliers found for digit 1 to 4 are shown in figure 2 (f)-(i) marked with stars. We can see how some points in the right side of the plot are obviously misclassified, other outlier points have very small values. Many of the outliers are not just misclassified examples, but digits written in an unusual way as we can see in figure 2 (d) and (e). We assign the unseen examples to the existing clusters (without recomputing F^*) using the method in section 2.6 and obtain an error rate of 0.0425.

We rerun the algorithm again without letting it optimize for α and fix α to 0.99. The algorithm determines the optimal σ as 0.0553. In this case we get similar results with the same number of clusters. The error rate on the training set increased to 0.0388, but the error rate for the previously unseen points changed to 0.035. This demonstrates that optimizing for both σ and α gives better results. Again we can see in figure 2 (f)-(i) that the separability of the data gets worse if only σ is optimized.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 2: USPS handwritten digits data: (a)-(c) Mean distance for all digits, changing in each step; (d) the left-most digit is the centroid for each class followed by the worst outliers for the class; (e) overall worst outliers; (f)-(i) values for outliers with different optimization parameters (α, σ vs. fixed α)

3.3 20 Newsgroups Dataset

In the first experiment we will try to cluster natural language text from the 20 newsgroups dataset (version 20-news-18828). Analogous to the experiments with the consistency method, we choose the topics in $rec.*$ which contains autos, baseball, hockey and motorcycles. The articles were preprocessed using the Rainbow software package using the following options: (1) skipping any header as they contain the correct newsgroup; (2) stemming all words using the Porter stemmer; (3) removing words that are on the SMART system's stop list; (4) ignoring words that occur in 5 or fewer documents. Removing documents that have less than 5 words, we obtained 3970 document vectors in 8014 dimensional space. The documents were normalized into TFIDF representation and the distance matrix was computed, as in [2], using $W_{ij} = exp(-(1- < x_i, x_j > / \parallel x_i \parallel \parallel x_j \parallel)/(2\sigma^2))$.

Our algorithm discovers two clusters on this dataset that do not make sense intuitively so we fixed the number of clusters to $C = 4$. We let the algorithm optimize for α and σ and obtain an error rate of 0.5659. We rerun the same experiment and set $\alpha = 0.99$ and obtain the same error rate of 0.5659.We attribute this to the fact consistency method required more

labeled examples in the semi-supervised learning scenario than we have provided. We did not want to set a higher number of clusters as it would generate a model that is very difficult to interpret.

3.4 Control Data

We use the Synthetic Control Chart Time Series dataset from the UCI database as it was suggested for clustering. The dataset contains 600 examples of control charts that have been synthetically generated. The clusters that are supposed to be found have 100 examples each.

We first find that our method does not determine the number of clusters correctly so we fix it to $C = 6$. Working with a fixed number of clusters, we determine σ to be 0.145351 and $\alpha = 0.999963$. However, the assignment to the clusters does not work satisfactorily with a 0.4117 error rate. We therefore used equation (10) to remove the 50 worst outliers and rerun the same experiment. The values for σ and α change to $\sigma = 0.22723$ and $\alpha = 0.992217$. The error rate sinks to 0.2764. This clearly demonstrates that our method can successfully identify outliers that interfere with the clustering process.

4 Conclusion

We have proposed a new clustering algorithm that 1) directly optimizes for all model parameters; 2) can detect both global outliers and outliers within each cluster; and 3) builds a cluster model that can cluster previously unseen points without relearning or modifying the original model. The proposed framework is based on a recently proposed semi-supervised learning technique [2] which frames learning as a search for local and global consistency. In this paper we show that this framework naturally leads to an optimization framework for clustering unlabelled data. Experimental evidence on both real and synthetic data supports the proposed algorithm.

This paper opens a number of interesting theoretical questions. The first of these concerns obtaining efficient algorithms for solving the proposed optimization problem (this paper introduced a fast heuristic solution which leaves much room for improvement). Second, our optimization framework can optimize a different set of model parameters for each cluster, a concept that may have wide ranging consequences. Finally, we measure closeness between points on a manifold not by a standard measure on a manifold, but by how each point orders distances to *other* points in the manifold. This is a unique distance metric that needs further theoretical study.

References

[1] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[2] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schlkopf. Learning with local and global consistency. Cambridge, Mass., 2004. MIT Press.

[3] J. R Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1983.

[4] J. Shrager, T. Hogg, and B. A. Huberman. Observation of phase transitions in spreading activation networks. *Science*, 236:1092–1094, 1987.

[5] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schlkopf. Ranking on data manifolds. Cambridge, Mass., 2004. MIT Press.

[6] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. Proceedings of the ICML, 2003.