Clustering Through Ranking On Manifolds

Markus Breitenbach BREITENM@CS.COLORADO.EDU
Dept. of Computer Science; University of Colorado, Boulder, USA

Gregory Z. Grudic GRUDIC@CS.COLORADO.EDU
Dept. of Computer Science; University of Colorado, Boulder, USA

Abstract

Clustering aims to find useful hidden structures in data. In this paper we present a new clustering algorithm that builds upon the 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. Other methods, e.g. Spectral Clustering, obtain good results on data that reveals such a structure. However, *unlike* Spectral Clustering, our algorithm effectively detects both global and within-class outliers, and the most representative examples in each class. Furthermore, we specify an optimization framework that estimates all learning parameters, including the number of clusters, directly from data. Finally, we show that the learned cluster-models can be used to add previously unseen points to clusters without re-learning the original cluster model. Encouraging experimental results are obtained on a number of real world problems.

1. Introduction

Clustering aims to find 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 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 forms paths through a high-dimensional space. Recently, Spectral Clustering [2, 3] has attracted a lot of attention for its ability to handle this type of data very well.

Although Spectral Clustering algorithms have achieved some success in modeling data that lies on manifolds, there remain a number of open research questions. Perhaps the most important of these is using data to estimate the number of clusters and the distance metrics used in each cluster. In general, these learning parameters are set manually, making spectral clustering difficult to use in practice. Recently, an automated way of choosing these learning parameters was proposed in [4]. However, this algorithm has two main disadvantages: First, there is no framework for assigning points outside of the training set to clusters; second, although the algorithm works well on synthetic data, as demonstrated in Section 3, it has limited success on real world data.

In this paper, we present a new clustering algorithm based on the consistency method, a semi-supervised learning technique [5] that has demonstrated impressive performance on 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 algorithm to unsupervised learning by finding a minimal subset of points that are suitable for the consistency method to use as seeds for clusters. The property we exploit to identify this subset of points is based on a similarity metric we propose. Specifically, our framework considers two points to be identical if they both have identical distance, on the manifold, to all other points. This naturally leads to an optimization framework for estimating learning parameters. Therefore, as with the algorithm proposed in [4], one of the goals of this paper is to directly estimate, from the data, both the number of clusters, and the similarity metrics used to identify them. However, unlike [4], the algorithm proposed here tends to do well on real world data, and can be used to cluster points that are not used during learning.

The similarity metric described above produces a number of properties that differentiate the manifold clustering algorithm proposed in this paper from Spectral Clustering. Specifically:

1. The algorithm directly identifies points that are most representative of each cluster.

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

2. The algorithm directly identifies points that are outliers from all other points.
3. The algorithm directly identifies points that are outliers within each cluster.

These properties can lead to significant improvements in the model quality and give additional insights into the data. As demonstrated in Section 3, Spectral Clustering does not have these characteristics.

The theoretical formulation for the proposed clustering algorithm is given in Section 2: Section 2.2 describes our point ranking similarity metric, and shows how outliers (both global and within clusters) are identified, and how cluster representatives are chosen; Section 2.3 describes the algorithm used to find the points that are used to seed clusters; Section 2.4 defines the optimization function used to estimate the cluster learning parameters from data; and, Section 2.5 defines the algorithm used to cluster data not used during learning. Section 3 presents detailed experimental results on both synthetic and real data. Section 4 concludes with future work. An extension of this work to the robotics domain can be found in [6].

Matlab code implementing the proposed clustering algorithm is available for download from the authors homepages.

2. Algorithm

2.1. Semi-Supervised Learning

In [5] Zhou et.al. introduced the consistency method, a semi-supervised learning technique, which is the basis of the clustering algorithm proposed in this paper. Below is a brief summary of this semi-supervised algorithm.

Assume a set of n training examples $x_1, ..., x_n$, with each training example $x_i \in \Re^m$. Assume also a set of labels $\mathcal{L} = \{1, \cdots, c\}$, where each point belongs to only one label. In a semi-supervised learning framework, 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 $F = [F_1^T, \cdots, F_n^T] \in \mathcal{F}$ is a matrix that labels all points x_i with a label $y_i = \arg\max_{j\leq c} F_{ij}$. Finally, 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 is as follows. The cost function associated with the matrix F with regularization parameter $\mu > 0$ is defined as

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

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

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

For a more in depth discussion about the regularization framework and on how to obtain the closed form expression F^* see [5].

2.2. Clustering

Let us assume for the moment that the parameters σ, α and the number of clusters c are known. We further assume that each cluster exposes a manifold structure without holes, i.e. finding one labeled point per class for the consistency method will allow us to find all the remaining points of that class. 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. To turn the consistency method into a clustering algorithm it suffices to determine which columns of F we need to select, i.e. we need to find the centroid points that are the center of each class, and then assign each point to the class using the classifying function: $y_i = \arg\max_{j\leq c} F_{ij}$. We define a matrix U as:

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

and note that U defines a graph or diffusion kernel as described in [7, 8]. The entries in the columns of U (we symbolize column i of matrix U by the column vector u_i^T) contain the "activation" of all the points in the data set if point i were used for labeling. That means that U_{ii} is the largest number in column i, and the remaining values in U_i get smaller the further the points are away from the centroid, but according to the underlying intrinsic structure. In [9] these values were used to rank with respect to the intrinsic manifold structure, i.e. the activation was used as similarity measure between the points. The ordering of these distances along each manifold is maintained independent of scaling. Therefore, without loss of ranking, we can derive a normalized form of U, called V, as follows:

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

Note that, by definition, $||v_i|| = 1$.

The normalized matrix V allows us to define a rank based similarity metric between any points x_i and x_j as follows:

$$d_M(x_i, x_j) = 1 - v_i v_j^T \quad (5)$$

The intuition behind this distance measure is that two points on a manifold are identical, iff the order of distances between all other points in the training set are identical, and the relative distances are identical - under these conditions, $v_i v_j^T = 1$ and $d_M(x_i, x_j) = 0$. Conversely, if the point x_i has completely different distances along U to other points in the training data as does point x_j, then $d_M(x_i, x_j) = 1$, because $v_i v_j^T = 0$. For notational convenience, we further define the matrix D_M whose entries are defined as $D_{M_{ij}} = d_M(x_i, x_j)$.

Given the definition of similarity in equation (5) 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 from points in other clusters. In order to formalize this, we first define the concept of an outlier within our framework first. 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 by taking the average of the columns of D_M (the elements of which are $D_{M_{ij}} = d_M(x_i, x_j)$ as defined in equation 5) and defining an outlier vector (cluster independent) O_d as follows:

$$O_d = \frac{1}{n}\left[\sum D_{M1}^T, ..., \sum D_{Mn}^T\right] = [O_{d1}, ..., O_{dn}] \quad (6)$$

where the element O_{d_i} is the average distance (on the manifold) between point x_i and all the other points, and, by definition $D_M = \left[D_{M1}^T, ..., D_{Mn}^T\right]$. Thus by ordering the vales of O_d in decreasing order, we order the points from furthest to closest, and *the points appearing first in the list constitute the outliers*.

Similarly, assume that $\mathbf{p}_j = (p_{1j}, ..., p_{Kj})$ is a vector of K indices, where $p_{k_1 j} \neq p_{k_2 j}$, and $p_{kj} \in \{1, ..., n\}$, that defines the K training examples that are part of cluster j. This allows us to define outliers within a cluster j by looking at the sub-matrix $D_M^{jj} = D_M(\mathbf{p}_j, \mathbf{p}_j)$. Specifically, we obtain an outlier vector O_d^j for cluster j as follows:

$$O_d^j = \frac{1}{n}\left[\sum D_{M1}^{jjT}, ..., \sum D_{Mn}^{jjT}\right] = \left[O_{d1}^j, ..., O_{dn}^j\right] \quad (7)$$

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

As outlined below, Equations 6 and 7 constitute the basis for the clustering algorithm proposed in this paper.

2.3. Finding Points That Seed a Cluster

In algorithm 1 we specify our heuristic for identifying the centroid points we use to assign all the data points to a class. Let the points $x_{l_1}, ..., x_{l_c}$ specify the centroid point for each cluster. The algorithm works by first assigning x_{l_1} to the point that is closest to all other points, which is the point that has the largest value O_{d_i}. This is illustrated in figure 1 (a) which shows the mean distance for O_d on the USPS data set (see section 3.2). Note how in each step only one of the clusters has higher values than all the other clusters.

To find x_{l_2}, we multiply each element of O_d by the corresponding element in the column vector $D_{M_{l_1}}^T$, to obtain an new, re-weighted vector O_d. Re-weighting is done using the component-wise vector multiplication ($.*$ in Matlab). Let O_d^n denote the nth re-weighting of the O_d vector. Re-weighting the vector gives all the points that were similar x_{l_i} a small value (with our similarity measure) and all the points that were different a large value. The mean distance after the first re-weighting is illustrated in figure 1 (b). Again we choose the point that is most similar to all the other points, which is the point that has the largest value O_d^n. This procedure of re-weighting and finding the most similar point continues until c points have been found.

Algorithm 1 Find centroid-points. *Input*: Matrix V, number of clusters C *Output*: indices of the clusters $l_1 \cdots l_c$

```
1: $n \leftarrow 1$, compute $O_d^1 = O_d$ and $D_M$ from $V$.
2: $l_1 \leftarrow$ index of the example with maximum of $O_d^1$.
3: $wt \leftarrow (1 - D_{M_{l_1}})$
4: for $2 \leq n \leq C$ do
5:     $l_n \leftarrow$ index of the example with maximum of $O_d^n$.
6:     $wt \leftarrow (1 - D_{M_{l_n}}) .* wt$
7:     $O_d^{n+1} \leftarrow wt .* O_d^n$.
8: end for
```

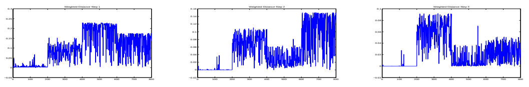

(a) n=1 (b) n=2 (c) n=3

Figure 1. O_d^n **in each step of finding centroid points**: In each step the mean distances change as the O_d^n vector is re-weighted.

2.4. Model Selection For Clustering

Given the above definitions, we now state the optimization framework for estimating the clustering parameters σ, α and c from learning data $x_1, ..., x_n$. let $\mathbf{p}_j$ be the set of points that belong to cluster j (as defined in Section 2.2). Using matrix D_M we can define the mean distance between points in cluster j as:

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

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 $E[\cdot]$ 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, α, c, and $x_{l_1}, ..., x_{l_c}$ to maximize the following:

$$\Omega\left(c\right) = \max_{\alpha,\sigma,c}\left[E\left[\overline{D}_M^{jk}\right]_{\left\{\substack{k=1,\ldots,c\\ j=1,\ldots,c\\ k\neq j}\right\}} - E\left[\overline{D}_M^{jj}\right]_{\{j=1,\ldots,c\}}\right] \tag{8}$$

The optimization problem in equation (8) is non-linear, and currently we use a brute force procedure for solving it. Namely, for each $c = 2, 3, 4, ..., C$, we use the *Matlab* function *fminbnd* to perform a two dimensional optimization in α and σ to maximize $\Omega(c)$. We then choose the number of clusters c by finding the maximum of $\Omega(2), ..., \Omega(C)$, and use the α and σ associated with this number of clusters. Note that this is not the optimal solution, but an approximation that works well in most practical cases.

2.5. Clustering New Points

In order to cluster a new point without adding it to S and re-inverting the matrix $(I - \alpha S)$ (as defined in Section 2.1), 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 $j = 1, ...n$, obtaining a vector W_k. We then calculate $D_k = \sum_{j=1}^{n} W_k(j)$ and compute the vector in the S matrix that is associated with x_k, as $S_k = D_k^{-1/2} W D^{-1/2}$. We further normalize S_k to have length 1, and call it S_k^1. Similarly the rows of S are normalized to length 1, denoting this matrix by S^1. We then obtain a set of coefficients $\Theta = (\theta_1, ..., \theta_n)^T = S^1(S_k^1)^T$. 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 a vector $F_k = [v_{l_1}\Theta^T, ..., v_{l_c}\Theta^T]$, where $v_{l_1}, ..., v_{l_c}$ are the columns of V (defined in Equation 4) which correspond to the cluster seed points in defined in Section 2.3. Point x_k is then assigned to a cluster y_c using the following function: $y_c = \arg\max_{j \leq c} F_k$, where j refers to an element in the vector F_k.

3. Experimental Results

We evaluate our method using both synthetic data problems and real world data. The results will be compared with Ng's Spectral Clustering algorithm presented in [3] and Self-Tuning Spectral Clustering presented in [4], a variant of Ng's algorithm that can determine the Kernel matrix and the number of clusters automatically which demonstrated good results on image segmentation problems. To evaluate the performance of out-of-sample techniques we will report error rates on the data sets for our method and the techniques for Spectral Clustering presented in [10]. In all the problems, the desired assignment to the classes is known, and we use this to report an error rate. 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. The parameters σ, α and C, the number of clusters, are found by the algorithms unless noted otherwise. To have means of comparison of how well our algorithm finds outliers, we compare this to Spectral Clustering as well. We determine outliers with Spectral Clustering by using the distance to the K-Means cluster centers.

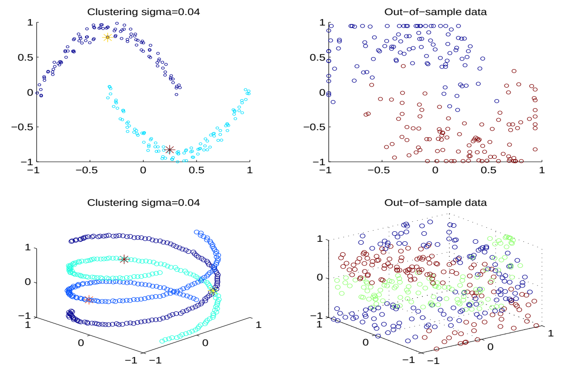
Note that clustering algorithms with parameters (except the number of clusters) do require a fine-tuning until the user is satisfied with the solution. Since we know the true labels, but do not want to be "training on the test-set", we will give results for a range of the parameters and the error-rate obtained if the algorithm was unable to find the "right" solution automatically.

3.1. Synthetic Data

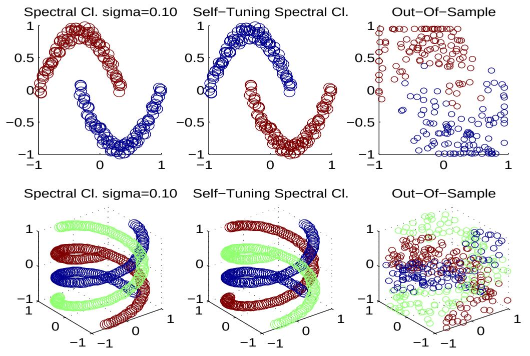
In this experiment, we consider the two-moon and spiral synthetic data sets. The spiral data that was used in [11] and two-moon has been used as an example in numerous other manifold related experiments [5, 4]. Note that these synthetic data sets can not be clustered in a meaningful way by methods that assume a compact shape for the data like K-means [5, 11].

For the two-moon problem (see figure 2 (a)) we used σ and α as determined by our approximation of the optimization problem in equation (8). For two-moons our algorithm correctly determined the number of clusters and automatically determined parameters ($\sigma = 0.0415, \alpha = 0.9999999$) that separate the two clusters. 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 the intensity gets smaller the further away the point is from the centroid.

For the spiral data our algorithm determined automatically that there are 3 clusters and determined the parameters $\sigma = 0.039$ and $\alpha = 0.999999$ for the three spirals. The points with the smallest intensity are located at the end of each spirals since the centroid of each spiral is in the middle.



(a)



(b)

Figure 2. **Synthetic Data:** Both algorithms were able to automatically determine the parameters and number of clusters. The size of each dot shows the "inlierness". The plot in the right-most column shows out-of-sample points. (a) Results on the two moon and spiral synthetic data with our algorithm. The centroid-point for our method is marked with a star. (b) Results using Spectral Clustering. Left and middle: manually determined $\sigma = 0.1$, automatically determined σ.

In figure 2 (b) we show the the cluster assignments of Spectral Clustering for the two data sets. In the figure, from left to right, we show the result with a manually determined kernel σ, automatically determined kernel matrix (Self-Tuning Spectral Clustering) and the assignment of previously unseen points to the existing clustering solution. The unseen points were generated by distorting the data set by adding a uniformly distributed random number within ± 0.2 to each coordinate.

Out-of-sample: The Spectral Clustering out-of-sample extension obtained an 0.0429 error rate on the two moon data and a 0.1587 error rate on the three spirals. Our method obtained an 0.0429 error rate on the two moon data and a 0.1455 error rate on the three spirals.

Outliers: Also note the difference in the size of points near the centroid of the two moons and the ends of each moon, i.e. points at the ends of each moon are more likely to be outliers than points closer to the center. Also note that with Spectral Clustering (Figure 2 (b), top left) the intensity for all the points is mostly the same. In figure 5 we show a 3D-plot showing the distance to the center for each data point.

α-parameter: To use Spectral Clustering one needs a suitable Kernel-σ and the number of clusters. Since our algorithm has one more parameter than Spectral Clustering, we will now demonstrate how this parameter influences the outcome of the label assignments. In figure 3 we can see that a change in α can result in a drastic changes in the separability of the three spiral data as some points are assigned to the wrong spiral. We see that the clustering can change significantly, if α is not optimized.

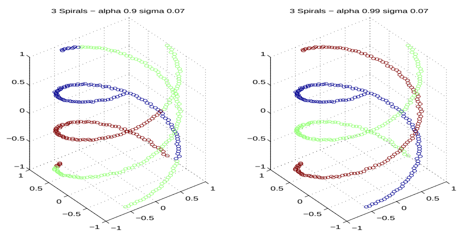


Figure 3. **Influence of parameter α:** Clustering of three spirals with $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)

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.

We first use our algorithm to discover the different classes and label one example of each class. Our algorithm determined $\sigma = 0.811396$ and $\alpha = 0.999981$. The number of clusters was automatically determined to be $C = 4$. This results in an error rate of 0.0238 which is a slightly better error rate than the error rate obtained with the semi-supervised consistency method with 1 random labeled point per class.

Using Ng's algorithm for spectral clustering, we manually determine the optimal value for sigma ($\sigma = 5$). We obtained a 0.07 error rate on the digits data, and the rotation alignment method correctly predicts 4 clusters with the manually determined sigma. In table 1, we list the different error rates and number-of-cluster predictions for different sigmas. Note how the performance varies based on the sigma. Using the self-tuning techniques – either with Ng's

σ	0.7	0.9	2	5	7
Error	0.477	0.292	0.748	0.07	0.076
Clusters	7	6	5	4	4

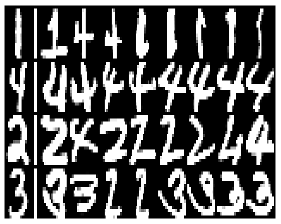
Table 1. Spectral Clustering on USPS Digits Data

algorithm or using the rotation method – results in a 0.75 error rate, because the the algorithm assigns all the digits to one cluster, despite the fact that we specified the number of clusters.

Out-of-sample: We assign previously unseen examples to the existing clusters (without recomputing F^*) using the method in section 2.5 and obtain an error rate of 0.0425. Using the Bengio's Out-of-Sample framework for Spectral Clustering ($\sigma = 5$) results in an 0.1175 error rate.

α-parameter: To test how unstable the model is for changes in α we run our 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 results as above 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.

Outliers: In figure 4 (a) we evaluate how well the methods can find outliers. Our method finds centroid points that are clearly "average" examples of their respective class. The outliers for all the digits have some twist to them: a one with an underline, a misclassified 4, rotated digits, or otherwise illegibly written digits. In figure 5 (b) we can see that Spectral Clustering with a manually determined sigma does not produce a usable outlier measure. The points closest to their respective center are sometime outliers. Self-Tuning Spectral Clustering failed on this data set, and we found the outliers to be unusable. In figure 5 (c) we show a plot of the distances as determined by Spectral Clustering for each digit to their respective center.


(a)

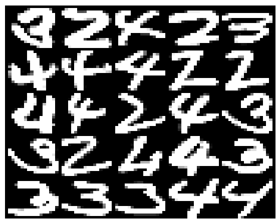
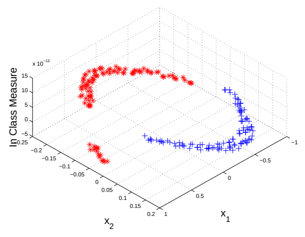
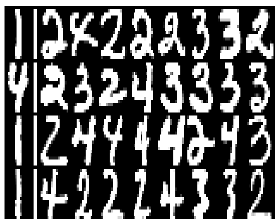

(b)

Figure 4. **USPS handwritten digits data**: **(a)** the left-most digit is the centroid for each class followed by the worst outliers for the class; **(b)** overall worst outliers;



(a)


(b)

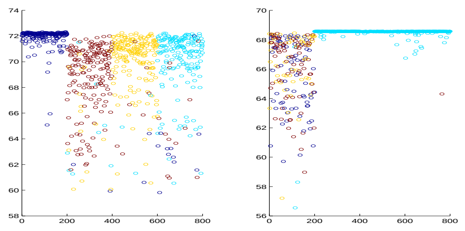

(c)

Figure 5. **Outliers determined by Spectral Clustering**: **(a)** Class-Inlierness plotted for the two-moon data **(b)** Digit outliers; the left-most digit is the digit closest to the respective center **(c)** Distance to Cluster Center for each digit (Spectral Clustering vs. Self-Tuning Spectral Clustering)

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 with 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. By 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 [5], using $W_{ij} = exp(-(1 - < x_i, x_j > / \parallel x_i \parallel \parallel x_j \parallel)/(2\sigma^2))$.

Our algorithm discovered only 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 $\alpha = 0.99$, then obtained the same error rate of 0.5659.We attribute this to the fact that, in the original experiment in [5], the consistency method required 50 labeled examples in the semi-supervised learning scenario instead of the 4 we have provided with the clustering. We believe that the manifolds of each cluster have holes, which violates the assumptions we made for our method.

For Spectral Clustering, we used the same Kernel that was used with our method. Using Spectral Clustering with a

σ	0.2	0.7	0.9	2	3	4
Error	0.739	0.545	0.147	0.111	0.109	0.11
Clusters	6	5	3	3	3	3

Table 2. Spectral Clustering on Newsgroup data: Error-rate and predicted number of clusters with varying sigma.

σ	0.2	0.7	0.9	2	3	4
Error	0.625	0.5	0.5	**0.245**	0.276	0.505
Clusters	10	**6**	4	3	5	8

Table 3. UCI Control data: Error-rate and number of clusters predicted for Spectral Clustering

manually determined $\sigma = 3$, we obtained an error rate of 0.11. Self-Tuning Spectral Clustering with the number of clusters specified results in a 0.7499 error rate as almost all documents are assigned to the same cluster. The number of clusters, using a manually determined sigma, was falsely predicted as 3.

3.4. Control Data

We use the Synthetic Control Chart Time Series dataset, a time-series problem, 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.

Our method determines $C = 6$ clusters with $\sigma = 0.2517$ and $\alpha = 0.99$. However, the assignment to the clusters produces a 0.4117 error rate. Since we are unhappy with the model we obtained, we use equation (6) to remove the 50 worst outliers and rerun the experiment for $C = 6$ clusters. The values for σ and α change to $\sigma = 0.2067$ and $\alpha = 0.99$. The error rate sinks to 0.28. This clearly demonstrates that our method can successfully identify outliers that interfere with the clustering process. For this dataset our algorithm did not automatically give the best results. Since the result for this data set is not satisfying, we give error rates for different σ and α values as one might have found them manually in table 4. Note that the error rates differ significantly with varying α, which shows that this parameter can have a significant influence on the cluster assignment of the data.

Using Spectral Clustering we find the best error rate with manually determined $\sigma = 2$ of 0.245. A table with error-rates for different sigmas is in table 3. Specifying the number of clusters, Self-Tuning Spectral Clustering results in an error rate of 0.5.

3.5. Yale Face-Database B

We now consider the setup used the in [12] and use the Yale Face Database B [13]. We use images of individuals 2, 5 and 10 and down-sample each image to 30x40 pixels. This gives us 1755 images with 1200 dimensions to work with. In the original setup PCA was applied to reduce the images to 3 dimensions, but we will apply both our algorithm and Spectral Clustering to the 1200 dimensional problem. Our algorithm determines that there are only 2 clusters, putting the faces of individuals 2 and 10 into one cluster. We set the number of clusters to 3 and rerun the experiment. Given the number of clusters the algorithm automatically determines $\sigma = 1.0803$ and $\alpha = 0.99999999$ and separates the three individuals faces with no errors. Note that no preprocessing, like PCA, was used in this case.

σ	0.2	0.3	0.4	0.6	0.8
Error, $\alpha = 0.99$	0.59	0.476	0.656	0.83	0.83
Error, $\alpha = 0.8$	0.49	**0.201**	0.666	0.52	0.52
Error, $\alpha = 0.7$	0.3	0.246	0.476	0.34	0.41

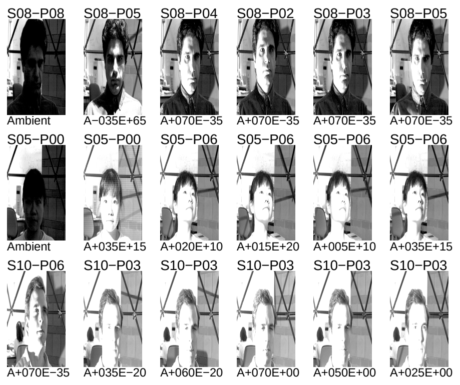
Table 4. UCI Control data: Error-rate for varying σ and α

In comparison, Self-Tuning Spectral Clustering solved the problem without errors as well and correctly predicted 3 clusters. Spectral Clustering with a manually determined $\sigma = 0.9$ solved the problem without errors as well. Due to space constraints we can not make a comparison with the outliers determined by Spectral Clustering.

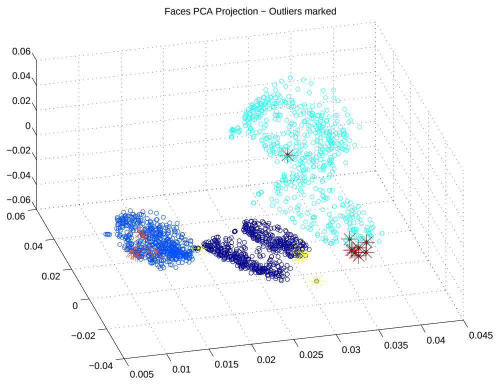
Outliers: In figure 6 (a) we show the cluster representants in the left most column. Note that in two out of three clusters it is the ambient shot of the person. In one case it is the center that one would have wished for: the ambient shot from the front, i.e. the clustering found the “real” center of the data as all the other data points are changes of pose and angle. In figure 6 (b) we show a PCA projection of the faces down to 3 dimensions and marked the outliers that are shown in (a). Note that the outliers are indeed the most outside points of the point clouds. Also note that the worst outlier for subject 5 (middle row, second to left) is an image that is slightly corrupted and was therefor correctly marked as the most outlieing face.

4. Conclusion

We propose a new clustering algorithm that has the following characteristics: 1) all clustering parameters (including the number of clusters) can be estimated directly from the data; 2) points that are most representative of each cluster are implicitly identified; 3) points that are outliers from *all* other points are implicitly identified; 4) points that are outliers *within* each cluster are implicitly identified; and, 5) the learned model can cluster previously unseen points without relearning or modifying the original model. Within the proposed framework, similarity between points is measured by how each point orders distances to *other* points on the manifold. This, similarity measure naturally gives rise to all of the above characteristics, and differentiates the algorithm from other manifold clustering algorithms such as spectral clustering [2, 3, 4].



(a)



(b)

Figure 6. **Yale Face Database B**: **(a)** left-most column are the cluster centers, middle and right are the two worst outliers per class. The upper title of each image designated subject and pose of the Yale Face Database B. The lower title of the image designates the light source direction with respect to the camera axis in azimuth (e.g. ’A+035’) and the elevation in degrees (e.g. ’E+40’); **(b)** a PCA-projection of the face data. The outliers plotted in (a) that the algorithm identified have been marked and are at the most outside points of the blobs.

Experimental evidence is presented showing that the proposed algorithm significantly outperforms spectral clustering in detecting global and within cluster outlier points, as well as identifying points that are most representative of the class. Furthermore, standard spectral clustering algorithms have two learning parameters: the number of clusters and the affinity parameter σ used to define the clusters. The algorithm proposed in this paper introduces a third parameter α which controls the dispersion of labels across points. This third parameter can lead to significant improvement in clustering performance over standard Spectral Clustering. Furthermore, experimental results show that, on real world data, the proposed algorithm can outperform other algorithms which estimate clustering learning parameters from data (see [4]).

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 uses standard Matlab functions for nonlinear 2D optimization, leaving much room for improvement). Second, our optimization framework can optimize a different set of model parameters for each cluster, which may significantly improve clustering performance on data that has different scales for different clusters. Finally, we measure closeness between points not by standard manifold distance metrics, but by how each point orders distances to *other* points in the manifold. This similarity metric warrants further theoretical study.

References

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

[2] Deepak Verma and Marina Meila. A comparison of spectral clustering algorithms, technical report uw-cse-03-05-01, university of washington.

[3] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[4] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

[5] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In L. Saul S. Thrun and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, Mass., 2004. MIT Press.

[6] Gregory Z. Grudic and Jane Mulligan. Topological mapping with multiple visual manifolds. *Proceedings of Robotics: Science and Systems*, 2005.

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

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

[9] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In L. Saul S. Thrun and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, Mass., 2004. MIT Press.

[10] Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

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

[12] R. Vidal, Y. Ma, and J. Piazzi. A new gpca algorithm for clustering subspaces by fitting, differentiating and dividing polynomials. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:510–517, 2004.

[13] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.