Semi-Supervised Graph Clustering: a Kernel Approach

EECS 545 Final Project Report

Hong Moon
University of Michigan
Ann Arbor, MI 48109
`hsmfxw@umich.edu`

Jarrid Rector-Brooks
University of Michigan
Ann Arbor, MI 48109
`jrectorb@umich.edu`

Abstract

Semi-supervised clustering is a technique intended to improve the clustering result of a large amount of unlabeled data set with a small amount of labeled data. However, most existing semi-supervised clustering algorithms can only be applied on data represented in terms of vectors. The authors [1] combined the vector-based and graph-based approaches and empirically demonstrated the effectiveness of their algorithm on both synthetic and real data.

1 Problem Statement

Conceptually, the key idea of semi-supervised clustering is to use the information from a small amount of labeled data to improve the clustering results. From these small amount of labeled data, we can obtain valuable information (or constraints) that some data points must belong to the same clusters, and some which cannot belong to the same clusters. These constraints are called *must-links* and *cannont-links*, respectively. The authors [1] introduce the following called the semi-supervised clustering objective:

$$J(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\boldsymbol{x}_i \in \pi_c} \|\boldsymbol{x}_i - \boldsymbol{m}_c\|^2 - \sum_{\substack{\boldsymbol{x}_i, \boldsymbol{x}_j \in M \\ l_i = l_j}} \frac{w_{ij}}{|\pi_{l_i}|} + \sum_{\substack{\boldsymbol{x}_i, \boldsymbol{x}_j \in C \\ l_i = l_j}} \frac{w_{ij}}{|\pi_{l_i}|} \tag{1}$$

The problem is to find a k disjoint partitioning $\{\pi_c\}_{c=1}^{k}$ of the data $\boldsymbol{x}$ where the semi-supervised clustering objective is minimized. Here, π_c represents the c'th cluster, and $\boldsymbol{m}_c$ represents the c'th cluster center. M denotes a set of must-links constraints, and C denotes a set of cannot-links constraints. l_i refers to the cluster label of $\boldsymbol{x}_i$, and w_{ij} is the penalty cost for violating a constraint between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ [1]. This objective function will be expalained in greater detail later.

2 Motivation

A critical weakness of a k-means clustering is that it is unable to cluster non linearly-separable data. An example of non-kernel k-means can be found in the appendix. Kernel k-means clustering remedies this through the use of a kernel function. Also, with a semi-supervised clustering approach, where we exploit the extra information gained from the small number of labeled data, we could further improve clustering results.

There currently are efficient semi-supervised algorithms for vector-based data. However, most existing semi-supervised clustering algorithms are intended to use data represented in terms of vectors. The authors unified the vector-based and graph-based approaches and empirically demonstrate the effectiveness of their algorithm through empirical results on both synthetic and real data.

3 Literature Review

3.1 Weighted kernel k-means

To make the connection between vector and graph based data, we must look to the weighted kernel k means algorithm. Weighted kernel k means uses the following objective:

$$J(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\boldsymbol{x}_i \in \pi_c} \alpha_i \|\phi(\boldsymbol{x}_i) - \boldsymbol{m}_c\|^2 \quad where \;\; \boldsymbol{m}_c = \frac{\sum_{\boldsymbol{x}_i \in \pi_c} \alpha_i \phi(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_i \in \pi_c} \alpha_i} \tag{2}$$

where α_i is a predefined non-negative weight for the i'th data point, $\phi(\boldsymbol{x}_i)$ is a feature mapping, and $\boldsymbol{m}_c$ is the $\boldsymbol{c}$th cluster center, and π_c represents the c'th cluster. We can then solve the distance calculation in the following way:

$$\begin{aligned} d(\boldsymbol{x}_i, \boldsymbol{m}_c) &= \|\phi(\boldsymbol{x}_i) - \boldsymbol{m}_c\|^2 \\ &= K_{ii} - \frac{2\sum_{\boldsymbol{x}_j \in \pi_c} \alpha_j K_{ij}}{\sum_{\boldsymbol{x}_j \in \pi_c} \alpha_j} - \frac{\sum_{\boldsymbol{x}_j, \boldsymbol{x}_l \in \pi_c} \alpha_j \alpha_l K_{jl}}{(\sum_{\boldsymbol{x}_j \in \pi_c} \alpha_j)^2} \end{aligned} \tag{3}$$

Thus, given a kernel matrix $\boldsymbol{K}$, we can compute the distance from a point to any given cluster center. The expression of k means via the kernel allows us to capture non-linear structures, for instance computing a similarity graph using the Gaussian kernel. Note that opposed to regular k means, weighted kernel k means does not update cluster centers explicitly. Finally, the weighted kernel k means objective function is equivalent to a weighted graph-cut objective function[9], which will allow the method we used to take both graph and vector based data as input.

ALGORITHM 1: Basic weighted kernel k-means [1]

$$KERNEL - KMEANS\ (K, k, t_{max}, \boldsymbol{\alpha}, \{\pi_c^{(0)}\}_{c=1}^{k})$$

Input: K: Kernel matrix, k: number of clusters, t_{max}: optional maximum number of iterations, $\boldsymbol{\alpha}$: weight vector, $\{\pi_c^{(0)}\}_{c=1}^{k}$: optional initial clusters
Output: $\{\pi_c^{(n)}\}_{c=1}^{k}$: final partitioning of the points

```
1. Initialize the k clusters from $\{\pi_c^{(0)}\}_{c=1}^{k})$, if provided as input, else randomly.
2. Set t = 0
3. For each point $\boldsymbol{x}_i$ and every cluster $c$, compute $d(\boldsymbol{x}_i, \boldsymbol{m}_c)$ as in Eq (3)
4. Find $c^*(\boldsymbol{x}_i)$ = $argmin_c\ d(\boldsymbol{x}_i, \boldsymbol{m}_c)$, resolving ties arbitrarily.
5. Compute the updated clusters as
       $\pi_c^{(t+1)} = \{\boldsymbol{x}_i : c^*(\boldsymbol{x}_i) = c\}$
6. If not converged or $t_{max} > t$, set $t = t + 1$ and go to Step 3. Otherwise, stop
   and output final clusters $\{\pi_c^{(t+1)}\}_{c=1}^{k}$.
```

3.2 Semi-supervised clustering with Hidden Markov Random Fields (HMRF) k-means

Semi-supervised kernel k means is based upon the HMRF k-means objective function[10]:

$$J(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\boldsymbol{x}_i \in \pi_c} \|\boldsymbol{x}_i - \boldsymbol{m}_c\|^2 + \sum_{\substack{\boldsymbol{x}_i, \boldsymbol{x}_j \in M \\ l_i \neq l_j}} w_{ij} + \sum_{\substack{\boldsymbol{x}_i, \boldsymbol{x}_j \in C \\ l_i = l_j}} w_{ij} \tag{4}$$

HMRF k-means uses the idea of pairwise must-link and cannot-link constraints to incorporate supervision. Two nodes have a must-link constraint if they are known to belong to the same class. Likewise, if two nodes are known to be of different classes, they have a cannot-link constraint. Take w_{ij} to be the penalty, given by the generalized Potts potential[11], for violating a constraint between points x_i and x_j. Then the second and third terms of the objective function are the penalties for breaking must-link and cannot-link constraints respectively, while the first term is the cluster distortion measure.

3.3 Semi-supervised clustering with Spectral Learning

Semi-supervised clustering can be achieved with spectral learning in the followoing way. Form an affinity matrix A, where each entry A_{ij} is represented by the similarity between points i and j. All entries of A are normalized between 0 and 1. Then set $A_{ij} = 1$ (maximum affinity) for all points i and j that have a must-link constraint. And set $A_{ij} = 0$ (minimum affinity) for all points i and j that have a cannot-link constraint. It will later be shown (in section *) that spectral learning can be viewed as a special case of the semi supervised kernel k-means algorithm.

3.4 Spatial Pyramid Kernel

A pyramid kernel is a kernel function that is used in image recognition and clustering. For the Caltech Image Dataset, which will be explained later, the authors used Pyramid Match Kernel [6] for their kernel function. The implementation of Pyramid Match Kernel was separate from the paper we were implementing. Having only 2 group members and lack of time, Spatial Pyramid Match Kernel [7] was adopted for our kernel function instead under the approval of the GSI due its high similarity with the original Pyramid Match Kernel.

4 Methodologies Explored and Their Rationale

The HMRF k-means objective may be expressed via kernels, allowing a more generalized algorithm. Further, by viewing graph based objectives in the context of semi supervised kernel k-means, a generalization is found allowing use of graph and vector based data with the studied algorithm. Finally, a connection between spectral learning and the studied algorithm is observed.

4.1 HMRF K-Means to Semi-supervised (SS) Kernel K-Means

The authors propose the following objective for SS Kernel K-Means:

$$J(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\boldsymbol{x}_i \in \pi_c} \|\boldsymbol{x}_i - \boldsymbol{m}_c\|^2 - \sum_{\substack{\boldsymbol{x}_i,\boldsymbol{x}_j \in M \\ l_i = l_j}} \frac{w_{ij}}{|\pi_{l_i}|} + \sum_{\substack{\boldsymbol{x}_i,\boldsymbol{x}_j \in C \\ l_i = l_j}} \frac{w_{ij}}{|\pi_{l_i}|} \tag{5}$$

This objective is highly similar to that of HMRF k-means. Firstly, instead of penalizing based upon unsatisfied must-link constraints, they instead reward satisfied must-link constraints. Further, the penalty and reward values are inversely proportional to the cluster size. That is, a larger reward is given for a must-link satisfied within a smaller cluster and likewise for a cannot-link violated between small clusters. Let E be the matrix of pairwise squared Euclidean distances among the data points. Then, this objective can be expressed as the trace minimization problem[9] with $\boldsymbol{E}_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ and for constraint matrix $\boldsymbol{W}$, $\boldsymbol{W}_{ij} = \boldsymbol{w}_{ij}$ if there is a must-link between nodes $\boldsymbol{i}$ and $\boldsymbol{j}$, $\boldsymbol{W}_{ij} = -w_{ij}$ if there is a cannot-link between nodes $\boldsymbol{i}$ and $\boldsymbol{j}$, and 0 otherwise. Here $w_{ij} = \frac{n}{kC}$ where n is the number of data points, k is the number of clusters, and C is the total number of constraints.

Let S be the similarity matrix $S_{ij} = x_i^T x_j$. And define E = $S_{ii} + S_{jj} - 2S$. Now let us define an indicator vector z_c for cluster c such that $z_c(i)$= 0 if x_i is not in cluster c, and 1 otherwise. Next, define a matrix $\tilde{\boldsymbol{Z}}$ where the column $\boldsymbol{c}$ of $\tilde{\boldsymbol{Z}}$ is equal to $\frac{\boldsymbol{z}_c}{(\boldsymbol{z}_c^\top \boldsymbol{z}_c)^2}$. Then, the authors show that the objective (5) can be re-written as the following [1]:

$$trace(\tilde{\boldsymbol{Z}}^\top (\boldsymbol{E} - 2\boldsymbol{W})\tilde{\boldsymbol{Z}}) \tag{6}$$

As the kernel k-means objective function can be expressed as the trace maximization problem $trace(\tilde{\boldsymbol{Z}}^\top \boldsymbol{K}\tilde{\boldsymbol{Z}})$ with $\boldsymbol{K}$ as the kernel matrix, the authors show that their trace minimization problem can be converted into the maximization problem $trace(\tilde{\boldsymbol{Z}}^\top(\boldsymbol{S}+\boldsymbol{W})\tilde{\boldsymbol{Z}})$. Thus, $\boldsymbol{K}=\boldsymbol{S}+\boldsymbol{W}$. However, $\boldsymbol{K}$ is not guaranteed to be positive semi-definite, a requirement for kernel k-means to converge.

If $\boldsymbol{K}$ is positive semi-definite, the smallest eigenvalue of $\boldsymbol{K}$ is at least 0. So, we set the value of σ as the absolute value of the minimum eigenvalue of $\boldsymbol{K}$ if minimum eigenvalue is less than 0, and 0 otherwise. Therefore, by diagonal shifting by adding a $\sigma\boldsymbol{I}$ to $\boldsymbol{K}$ as in[5], $\boldsymbol{K}$ can be guaranteed to be positive semi-definite and thus converge under kernel-k means. Thus, HMRF k-means can be expressed using the kernel.

4.2 Connection with Graph Based Data

The semi-supervised k-means objective stated earlier has an equivalent for graph clustering objectives. In particular, we focus on ratio association. Let ν_c be the c'th partition of a graph where c = 1,..,n. The objective function for the semi-supervised ratio association is to maximize the following:

$$\sum_{c=1}^{k}\frac{links(\nu_c,\nu_c)}{|\nu_c|}-\sum_{\substack{\boldsymbol{x}_i,\boldsymbol{x}_j\in M\\ l_i=l_j}}\frac{w_{ij}}{|\nu_{l_i}|}-\sum_{\substack{\boldsymbol{x}_i,\boldsymbol{x}_j\in C\\ l_i=l_j}}\frac{w_{ij}}{|\nu_{l_i}|} \tag{7}$$

By a recent theoretical analysis, it has been shown that ratio association is equivalent to kernel k-means with $\boldsymbol{K}$ as the input similarity matrix and $\boldsymbol{A}$ is the graph adjacency matrix, if $\boldsymbol{K}=\sigma\boldsymbol{I}+\boldsymbol{A}+\boldsymbol{W}$ and node weights $\boldsymbol{\alpha}$ is 1 for all nodes[5]. Further, it is consequential that the case of semi-supervised ratio association is the same as the earlier derived semi-supervised kernel k-means objective, now using a graph adjacency matrix as the input similarity matrix.

Further, this method can be connected to spectral clustering [13]. Given $\boldsymbol{A}_{ij}$, the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the authors set $\boldsymbol{W}_{ij}=1-\boldsymbol{A}_{ij}$ when there is a must-link constraint between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For a cannot-link constraint, they set $\boldsymbol{W}_{ij}=-\boldsymbol{A}_{ij}$. With this choice of $\boldsymbol{W}$, the matrix $\boldsymbol{A}+\boldsymbol{W}$ is equivalent to that of the matrix in spectral learning before additive normalization. The normalization is then the same as the normalization used by the authors for ratio cut[1]. Thus, spectral learning can be viewed as a relaxed semi-supervised ratio cut objective, and is a special case of the author's idea.

4.3 Algorithm and Implementation

```
ALGORITHM 2: Semi-supervised kernel k-means for HMRF-KMEANS [1]
                 $KERNEL - HMRF - KMEANS\ (S, k, W, t_{max})$
**Input**: $S$: input similarity matrix, $k$: number of clusters, $W$: constraint penalty matrix, $t_{max}$: optional maximum number of iterations
**Output**: $\{\pi_c\}_{c=1}^k$: final partitioning of the points
1. Form the matrix $K = S + W$.
2. Diagonal-shift $K$ by adding $\sigma I$ to guarantee positive semi-definiteness.
3. Get initial clusters $\{\pi_c^{(0)}\}_{c=1}^k$ using constraints encoded in $W$ and farthest-first initialization (see Algorithm 3).
4, Return $\{\pi_c\}_{c=1}^k$ = KERNEL-KMEANS( $K$, $k$, $t_{max}$, $\mathbf{1}$, $\{\pi_c^{(0)}\}_{c=1}^k$) where $\mathbf{1}$ is the vector of all ones
```

ALGORITHM 3: Farthest-first initialization [1]

```
$FARTHEST - FIRST - INIT\ (A, k, W)$
Input: $A$: input affinity matrix, $k$: number of clusters, $W$: constraint penalty matrix
Output: $\{\pi_c^{(0)}\}_{c=1}^{k}$: initial partitioning of the points
1. $M$ = transitive closure of must-link constraints in W
2. $C_M$ = connected components in $M$.
3. $CC$ = components from $C_M$ disconnected across cannot-link boundaries
4. Set $\pi_1^{(0)}$ = largest connected component from $CC$. set i = 1.
5. If $k > i$, go to Step 6. Else, go to Step 8.
6. Find component $CC_j$ that is farthest from the current set of selected components $\{\pi_c^{(0)}\}_{c=1}^{i}$.
7. Set $\pi_{i+1}^{(0)} = CC_j$. set $i = i + 1$. Go to Step 5.
8. Return $\{\pi_c^{(0)}\}_{c=1}^{k}$
Note: We found the transitive closure using the Floyd Warshall's algorithm, which is typically covered in an undergraduate data structure course.
```

We have listed the final algorithms above. K is formed as described in the previous section. It is then diagonally shifted to ensure positive semi-definiteness. While unsupervised k-means usually initializes cluster centers randomly, as we know some information about the structure of the data through constraints, we can initialize clusters more intelligently using farthest first initialization as in Algorithm 3. First, we attempt to infer additional constraints from those given. To do this, we take the transitive closure of the constraint matrix W and find its connected components. Then, for any two nodes x_i and x_j in the same connected component we add a must-link constraint between them if one does not already exist. Next, if two nodes $x_i \in C_m$, $x_j \in C_n$ where C is the set of all connected components share a cannot-link constraint, we then add a cannot-link between every node in C_m and C_n. To calculate distances between component centers, we derived the following:

$$\begin{aligned}\|\boldsymbol{m}_d - \boldsymbol{m}_c\|^2 &= m_d^\top m_d - 2m_d^\top m_c + m_c^\top m_c \\ &= \frac{\sum_{\boldsymbol{x}_i,\boldsymbol{x}_j \in \pi_d} K_{ij}}{|\pi_d|^2} - \frac{2\sum_{\boldsymbol{x}_i \in \pi_d, \boldsymbol{x}_j \in \pi_c} K_{ij}}{|\pi_d||\pi_c|} + \frac{\sum_{\boldsymbol{x}_i,\boldsymbol{x}_j \in \pi_c} K_{ij}}{|\pi_c|^2} \end{aligned} \tag{8}$$

Then, using these initial clusters, we run and return the results of standard weighted kernel k-means.

5 Evaluation

5.1 Data Sets

We tested our implementation with three data sets; the same as the authors used to attempt to ensure correctness of our code. Here, we randomly generated must-link constraints and cannont-link constraints. The data sets we used are as follows.

1. Two Circles. Although it is a synthetic data set, the reason we use this data is to be able to plot the clustering result and show our method's effectiveness. Since each data point is in two-dimension, we can plot the data. But the other real data sets we have are well over two-dimensions (one of which is 16 dimension) so we cannot visualize the clustering result. Another reason is to show that weighted kernel k-means and HMRF kernel k-means can separate non-linear boundaries, which we cannot with k-means since it is only able separate data with linear decision boundaries. This data was generated by selecting two radii and creating one inner circle, class 1, and one outer circle, class 2, each consisting of 100 points.

2. Pendigits. This is a real data set for handwritten digit recognition from the UCI Machine Learning Repository[2]. Figure [1] shows an example of a handwritten digit 8. We selected 10% of the classes [3,8,9] since, as mentioned by the authors, distinguishing between these classes is a difficult task. Our data set contained 101 points, each in 16 dimensional space.

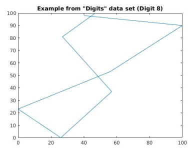


Figure 1: Example of Digit 8 from ”Pendigits” data set.

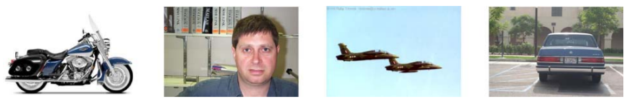


Figure 2: Example images from Caltech Image data set.

3. Caltech Image Data. We used this data as a graph-based data set as there is no explicit vector representation of the data. Here we used a set of 300 images of motorcycles, faces, airplanes, and cars as in Figure 3 obtained from the Caltech Categories data set[4]. To use this data, we first used the spatial pyramid match kernel[7], which itself used SIFT (Scale Invariant Feature Transform) [8] descriptors, to create a kernel matrix. Then, we viewed this kernel matrix as a dense graph. Examples of images from this data set can be seen in Figure 3.

5.2 Methodology

For measuring the performance of our models we used 2 fold cross validation with our metric being Normalized Mutual Information (NMI). NMI measures the amount of information obtainable from one random variable from another random variable. Let C be the random variable denoting the cluster assignments of data points, and K be the random variable denoting the underlying class labels on data points. Then,

$$NMI = \frac{I(C;K)}{(H(C)+H(K))/2} \tag{9}$$

where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the random variables X and Y. $H(X)$ is the Shannon entropy of X, and $H(X|Y)$ is the conditional entropy of X given by Y. The term $(H(C) + H(K))/2$ is the average entropy of C and K. The mutual information is normalized by the average entropy of C and K to make the value of NMI fit between 0 and 1[12]. NMI is the standard measure for clustering performance.

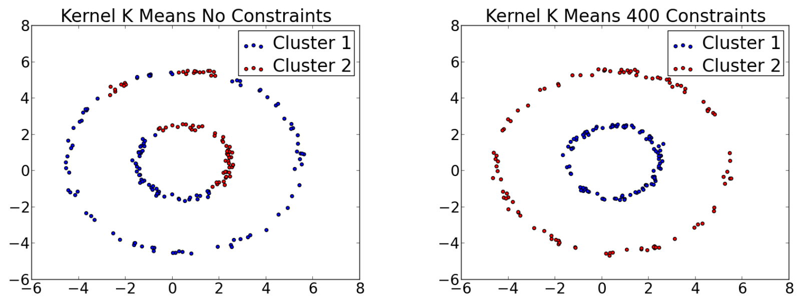
To tune parameters, we tested our implementation with a range of values (specifically for the Gaussian kernel) with the previously mentioned framework. As we only used parameters for the Two Circles and Pendigits data sets, we only tuned parameters for these data sets. We chose the parameters which yielded the best NMI scores.

We compared our results against two competitors: for vector based data we compared against Agglomerative Hierarchical Clustering using Ward Linkage, a tree based clustering objective, while for graph based data we compared against Spectral Clustering.

5.3 Results

1. Two Circles. Figure [4a] shows the result for kernel K-means with no constraints on Two Circles data. As noted by the authors, this data set was used to demonstrate the capability to cluster non-linear data. Using the RBF (Gaussian) kernel, we observed perfect clustering with 300 or more constraints, with the results rising for the most part, while for the linear kernel and ward clustering the NMI was nearly zero. These results matched those found by the authors. Figure [3b] shows the result for semi-supervised kernel K-means with 400 constraints, while figure [3a] shows semi-supervised kernel k-means with no constraints. With only 400 constraints given, we could perfectly separate the data with a circular boundary, which we would never be able to with a standard k-means clustering.

We have identified that the trend of our graph matches correctly with that of the graph produced by the original authors. With the Gaussian kernel, our graph has reached the NMI value of 1 with fewer



(a) Kernel k-means Clustering with No Constraints on TwoCircles data

(b) Semi-supervised kernel k-means with 400 constraints

Figure 3: Comparison between kernel k-means and Semi-supervised kernel k-means

number of constraints. A possible explanation is that we have used 2-fold cross validation to select the parameter of the Gaussian kernel, while the original authors mention that an arbitrary value was chosen for the parameter, which they did not reveal. The semi-supervised clustering with a linear kernel and the agglomerative hierarchical clustering with Ward linkage [13] yieled NMI values of almost 0 regardless of the increasing number of constraints.

2. Pendigits. Figure [4b] shows the result for semi-supervised kernel K-means on Pendigits data set. As noted earlier, the digits data set was used due to the difficulty of clustering the data set. Here, our results show a dip at the beginning which was also seen by the authors. They conjecture that this happens as the number of kernel parameters learned using too few constraints are unreliable. Finally, we saw a spike from at 200 and 250 constraints differing from the results of the authors. We believe this may be due to the authors randomly selecting their data from the three classes. As the data was randomly selected, we could not guarantee we used the same data set. Further, it is possible that the authors used different parameters than us, resulting in differing results. Ward clustering and SS Kernel K Means with the linear kernel both performed well and were quite stable, though neither had a max NMI value as high as the Gaussian kernel.

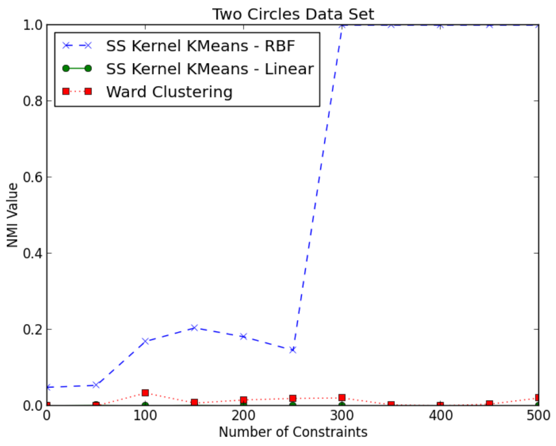
3. Caltech Image Data. Figure [4c] shows the result for semi-supervised kernel K-means on Caltech Image Data. Our results were extraordinarily spiky, and we would classify this as a negative result. We hypothesize this difference is due to differing methods of obtaining the data sets between ourselves and the authors. As the authors again randomly selected data, we were unable to ensure we had exactly the same images. More importantly, the data was transformed using the pyramid match kernel and SIFT. We were unable to find libraries implementing exactly the methods the authors did, and so we were unable to guarantee our kernel matrix was formed as theirs was. Still, although it is more spiky, our results do still resemble those of the authors. Their results showed ratio association outperforming spectral learning. While this was true for some numbers of constraints in our tests, we believe spectral clustering to here have been superior due to its steadiness throughout the test.

6 Conclusion

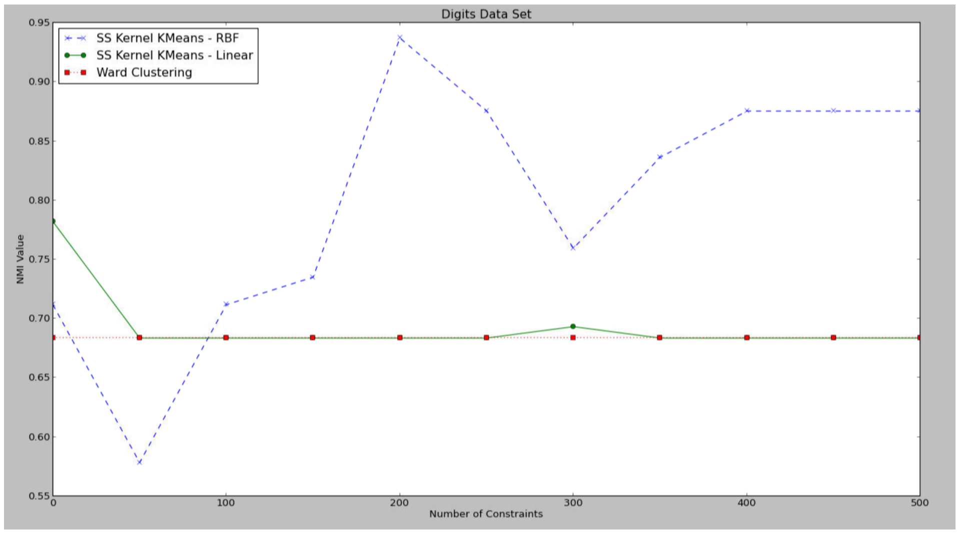
We learned much about semi-supervised clustering. We learned about constraint augmentation, different initialization schemes for kernel k-means, and a number of items about graph based machine learning. Further, we were able to learn more of graph cut objectives, as well as about Hidden Markov Random Fields.

As for strengths of semi-supervised kernel k-means, unlike the standard k-means clustering, our method can separate data with non-linear decision boundaries. Also, unlike the existing algorithms which are intended for vector-based data, our method can be applied on graph-based data as well. Further, for the vector based tests we performed, semi-supervised kernel k-means outperformed its competition, agglomerative clustering.

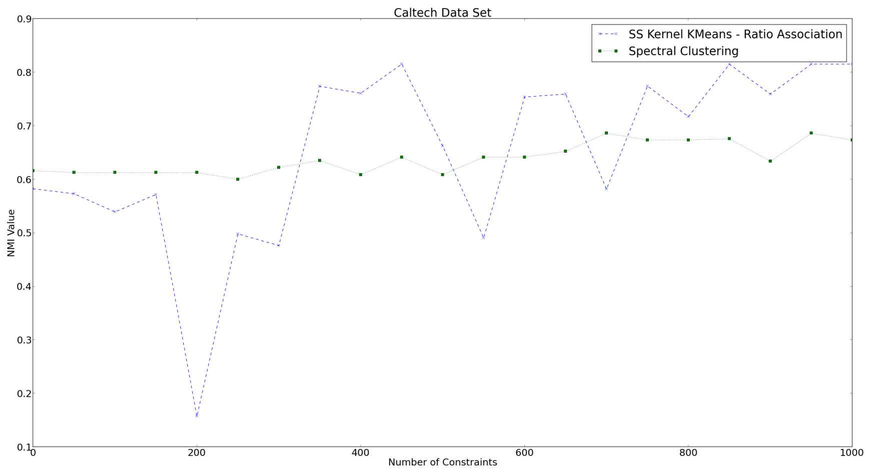
For weaknesses, firstly the NMI values often drop at when only a few constraints are added. The reason for this is touched upon in the results section. Further, our results were quite spiky for our



(a) Result of Two Circles



(b) Result of Pendigits



(c) Result of Caltech

Figure 4: Experimental Results

graph based tests. This could be due to anomalies within the data, but with the results we obtained we believe spectral learning to be a superior solution to semi-supervised graph data. Still, the ability to generalize to both vector and graph data is highly valuable.

7 Description of individual effort

We were originally in a group of 3. But one of the members made zero contribution to the project from the start and dropped the class on 12/13. We have written about 1200 lines of Python code of our own. We have done almost every task together.

- Member contribution: Hong - Weighted kernel k-means, semi-supervised kernel k-means, graphs and data analysis, writing report.
- Member contribution: Jarrid - Weighted kernel k-means, semi-supervised kernel k-means, graphs and data analysis, writing report.

8 Appendix

ALGORITHM 4: Semi-supervised graph clustering algorithm [1]

$$SS - KERNEL - KMEANS\ (A, obj, k, W, t_{max})$$

Input: A: input affinity matrix, obj: clustering objective, k: number of clusters, W: constraint penalty matrix, t_{max}: optional maximum number of iterations
Output: $\{\pi_c\}_{c=1}^{k}$: final partitioning of the points

```
1. Generate a vector $\boldsymbol{\alpha}$ of node weights based on obj using Table 3.
2. If obj is SS-ratio-cut, reset A as A-D.
3. Form $K = \sigma * diag(\boldsymbol{\alpha})^{-1} + diag(\boldsymbol{\alpha})^{-1}\ (A + W)\ diag(\boldsymbol{\alpha})^{-1}$
4. Get initial clusters $\{\pi_c^{(0)}\}_{c=1}^{k}$ using constraints encoded in $W$ and farthest-first initialization (see Algorithm 3).
5, Return $\{\pi_c\}_{c=1}^{k}$ = KERNEL-KMEANS( $K$, $k$, $t_{max}$, $\boldsymbol{\alpha}$, $\{\pi_c^{(0)}\}_{c=1}^{k}$)
```

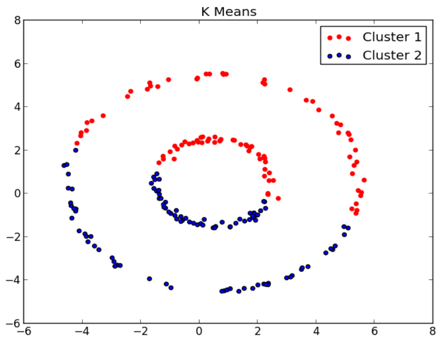


Figure 5: K-means clustering on TwoCircles data.

References

[1] Kulis, B., Basu, S., Dhillon, I. et al. *Mach Learn* (2009) 74: 1.doi:10.1007/s10994-008-5084-4

[2] E. Alpaydin, Fevzi. Alimoglu. (1998). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[3] David J. Slate. (1991). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[4] ”Visual Geometry Group Home Page”. Robots.ox.ac.uk. N.p., 2016. Web. 16 Nov. 2016.

[5] Dhillon, I., Guan, Y., & Kulis, B. (2007). Weighted graph cuts without eigenvectors: a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(11), 19441957.

[6] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In Proc. ICCV, 2005.

[7] Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 21692178).

[8] Lowe, D.G. International Journal of Computer Vision (2004) 60: 91. doi:10.1023/B:VISI.0000029664.99615.94

[9] Dhillon, I., Guan, Y., & Kulis, B. (2004). Kernel k-means, spectral clustering and normalized cuts. In Pro ceedings of the 10th international conference on knowledge discovery and data mining(pp. 551556).

[10] Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering In Proceedings of 10th ACM SIGKDD international conference on knowledge discovery and data mining(KDD-2004)(pp. 5968).

[11] Kleinberg, J., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise rela- tionships: metric labeling and Markov random fields. In: Proceedings of the 40th IEEE symposium on foundations of computer science (FOCS-99), (pp. 1423).

[12] Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on web-page clustering. In Workshop on artificial intelligence for web search (AAAI).

[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.