University of Maryland-College Park

Advance Scientific Computing I,II

Application of the Spectral Clustering Algorithm

Author:
Danielle Middlebrooks
Dmiddle1@math.umd.edu
Second year AMSC Student

Advisor:
Kasso Okoudjou
Kasso@math.umd.edu
Department of Mathematics

2015-2016

Abstract

Spectral Clustering is a technique used to group together data points of similar behavior in order to analyze the overall data. The goal of this project will be to implement a spectral clustering algorithm on databases in which we will be able to cluster similar images using a similarity matrix derived from the dataset. We will develop code in order to implement each step of the algorithm and optimize to efficiently obtain a reasonable clustering of the dataset.

Contents

1 Introduction

Spectral clustering is clustering technique based on the spectral analysis of a similarity matrix derived from a given data set. The main goal of spectral clustering or any clustering algorithm is to implement a procedure that groups objects in a data set to other objects with similar behavior and have them in different groups from objects that are dissimilar. For this project, we will start with the MNIST Handwritten digits database in order to implement a clustering algorithm that will cluster together same digits and be in a different cluster from different digits. After obtaining various results we will then extend this and apply the spectral clustering algorithm to the Yale face database. Spectral clustering implements a clustering algorithm such as k-means clustering on a reduced dimension which allows the formation of tight clusters. Thus given some data point $X_i \in \mathbb{R}^d$, spectral clustering performs a clustering in $\mathbb{R}^p$ where $p << d$. The advantage of spectral clustering is the simplicity of the algorithm to implement where only the use of standard linear algebra methods are needed in order to solve the problem efficiently. It also has many application areas such as machine learning, exploratory data analysis, computer vision and speech processing.

1.1 Definitions

The motivation behind spectral clustering is given from ideas in graph theory. In this section we define some notation that will be used throughout the motivation section of this report. Define a graph $G = (V, E)$ as a set of vertices together with a set of edges. We assume G is an undirected graph with vertex set $V = \{v_1, ..., v_n\}$. We also assume G is unweighted or in other words each edge has the same weight of 1. Thus the adjacency matrix W is defined to be

$$W = w_{ij} = \begin{cases} 1, & \text{if } v_i, v_j \text{ are connected by an edge} \\ 0, & \text{otherwise} \end{cases}.$$

Since G is undirected we require that $w_{ij} = w_{ji}$ and hence gives a symmetric adjacency matrix. The degree of a vertex $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

This can also be viewed as just the number of edges connected to that vertex. The degree matrix denoted D is a diagonal matrix where each $d_1, ..., d_n$ lies on the diagonal. We denote a subset of vertices $A \subset V$ and its complement as $\bar{A} = V \setminus A$. For simplicity, we define $i \in A$, as the set of indices i of vertices $v_i \in A$. We also define two ways of measuring the size of a subset A of V.

$$|A| = \text{ number of vertices in } A.$$

and

$$vol(A) = \sum_{i \in A} d_i.$$

$|A|$ measures the size of the subset by the number of vertices, while $vol(A)$ measures the size by the number of edges. Finally we define the weight between two subsets $A, B \in V$ as

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}.$$

This counts the number of edges connecting the two subsets. One final definition we would like to introduce in this section is the unnormalized laplacian matrix which is defined as $L = D - W$.

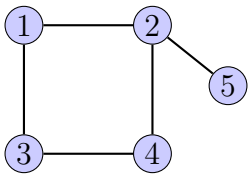
1.2 Motivation

Clustering is a way to separate data points such that similar points are grouped together, and are in a different group from ones that are dissimilar. Another way to think about this is from the viewpoint of graph cuts. Given a graph, we want to partition the vertices such that those connected by edges with high weights are grouped together and separate from the ones connected by low weights. Spectral clustering is motivated by approximating a graph partitioning and in particular approximating the RatioCut or NormalizedCut on a given graph.

One of the most direct ways to partition a graph is to solve the min cut problem. That is, given a similarity graph, we want to partition the graph into k subsets and hence solve the optimization problem of minimizing

$$cut(A_1, ..., A_k) := \frac{1}{2} \sum_{1}^{k} W(A_i, \bar{A}_i) \tag{1}$$

over all partitions where $W(A_i, \bar{A}_i)$ defines the weight between a subset and its complement. In other words, we want to minimize the number of edges cut in order to partition the graph. This is very straightforward and easy to solve, in particular for the case when $k = 2$ by using Karger's algorithm which provides an efficient randomized method for finding this cut. However in some cases it may lead to an unhelpful partition. Consider the example graph below:



To partition this graph into 2 subsets, the min cut problem would cut through the edge connecting 2 to 5 and give one partition to be relatively smaller than the other. This is not helpful in clustering since we want each cluster to be relatively large. To account for this, modifications known as the RatioCut and the normalized cut or NCut can be introduced. For the RatioCut, we want the size of each partition to be measured by the number of vertices in it. As for the NCut, we would like

the size of each partition to be measured by the number of edges. Thus we define the RatioCut and NCut as follows:

$$RatioCut(A_1, ..., A_k) := \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, \bar{A}_i)}{|A_i|} \tag{2}$$

$$NCut(A_1, ..., A_k) := \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, \bar{A}_i)}{vol(A_i)} \tag{3}$$

In both cases, the objective functions try to balance out each partition. This makes solving these versions of the min cut problem NP hard. Spectral clustering allows us to solve relaxed versions of these problems. For this project, we will be focusing on the relaxed version of the NCut problem to solve the clustering problem.

We will start with approximating the NCut problem for the case where $k = 2$. Relaxing the min NCut problem will derive the motivation behind normalized spectral clustering which we will define later in this section. For the case $k = 2$, we want to solve the optimization problem of minimizing

$$NCut(A, \bar{A}) = \frac{W(A, \bar{A})}{vol(A)} + \frac{W(A, \bar{A})}{vol(\bar{A})} = \frac{W(A, \bar{A})(vol(\bar{A}) + vol(A))}{vol(A)vol(\bar{A})} \tag{4}$$

over both partitions. We define a cluster indicator vector f by

$$f(v_i) = f_i = \begin{cases} \sqrt{\frac{vol(\bar{A})}{vol(A)}}, & \text{if } \ v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(\bar{A})}}, & \text{if } \ v_i \in \bar{A} \end{cases} \tag{5}$$

The cluster indicator vector is giving some value depending on whether the vertex lies in A or $\bar{A}$. Thus by computing $f^T Lf$ and $f^T Df$ we obtain the following:

$$f^T Lf = \sum w_{ij}(f_i - f_j)^2 = W(A, \bar{A})\left(\sqrt{\frac{vol(\bar{A})}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(\bar{A})}}\right)^2 = W(A, \bar{A})\frac{(vol(\bar{A}) + vol(A))^2}{vol(A)vol(\bar{A})} \tag{6}$$

$$f^T Df = \sum d_i f_i^2 = \sum_{i\in A} d_i\left(\sqrt{\frac{vol(\bar{A})}{vol(A)}}\right)^2 + \sum_{j\in\bar{A}} d_j\left(\sqrt{\frac{vol(A)}{vol(\bar{A})}}\right)^2 = vol(\bar{A}) + vol(A) \tag{7}$$

Note that the ratio of the two gives us the NCut problem we want to minimize. Thus minimizing the NCut problem is equivalent to

$$\begin{aligned} &\text{minimize} \quad NCut(A, \bar{A}) = \frac{f^T Lf}{f^T Df} \\ &\text{subject to} \quad f_i = \begin{cases} \sqrt{\frac{vol(\bar{A})}{vol(A)}}, & \text{if } \ v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(\bar{A})}}, & \text{if } \ v_i \in \bar{A} \end{cases} \end{aligned} \tag{8}$$

The relaxation problem is given by

$$\begin{aligned} \underset{f\in\mathbb{R}^n}{\text{minimize}} \quad & \frac{f^T L f}{f^T D f} \\ \text{subject to} \quad & f^T D\mathbf{1} = 0 \end{aligned} \tag{9}$$

where f is allowed to take on real values. It can be shown the relaxation problem is a form of the Rayleigh-Ritz quotient. Since we have the constraint that $f^T D\mathbf{1} = 0$ we want a solution that will not be the constant vector $\mathbf{1}$ of all ones which is the eigenvector of the smallest eigenvalue of 0. Thus we want to find the eigenvector corresponding the the second smallest eigenvalue. Substituting $g = D^{1/2} f$ the problem becomes

$$\begin{aligned} \underset{g\in\mathbb{R}^n}{\text{minimize}} \quad & \frac{g^T (D^{-1/2} L D^{-1/2}) g}{g^T g} \\ \text{subject to} \quad & g \perp D^{1/2}\mathbf{1} \end{aligned} \tag{10}$$

where $D^{1/2}\mathbf{1}$ is the first eigenvector of $L_{sym} = D^{-1/2} L D^{-1/2}$ which corresponds to the lowest eigenvalue of 0. Letting $U = [u_1 u_2 \dots u_n]$ be the matrix whose columns are the orthonormal eigenvectors of L_{sym}. If we only consider vectors g that are orthogonal to u_1 and since L is non negative, then

$$g^T L_{sym} g = \sum_{i=1}^{n} \lambda_i |(U^T x)_i|^2 = \sum_{i=1}^{n} \lambda_i |u_i^T x|^2 = \sum_{i=2}^{n} \lambda_i |u_i^T x|^2$$

This gives a non negative linear combination of $\lambda_2, \lambda_3 \dots, \lambda_n$, thus

$$g^T L_{sym} g = \sum_{i=2}^{n} \lambda_i |u_i^T x|^2 \geq \lambda_2 \sum_{i=2}^{n} |u_i^T x|^2 = \lambda_2 \sum_{i=2}^{n} |(U^T x)_i|^2 = \lambda_2 g^T g$$

provided that g is orthogonal to the first column of U. This inequality becomes equality if we choose $g = u_2$. Therefore

$$\min_{\substack{g\neq 0 \\ g\perp D^{1/2}\mathbf{1}}} \frac{g^T L_{sym} g}{g^T g} = \min_{\substack{g^T g = 1 \\ g\perp D^{1/2}\mathbf{1}}} g^T L_{sym} g = \lambda_2$$

which gives the second smallest eigenvalue and g is the corresponding eigenvector. In general applying the Courant-Fisher theorem, we can find the p smallest eigenvalues and their corresponding eigenvectors.

Theorem 1 (Courant-Fischer Theorem)**.** *Given A a Hermitian matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \lambda_n$, let k be a given integer with $1 \leq k \leq n$, and let $w_i \in \mathbb{C}^n$, then*

$$\max_{w_1, w_2, \dots, w_{k-1}} \min_{\substack{x\neq 0, x\in\mathbb{C}^n \\ x\perp w_1, w_2, \dots, w_{k-1}}} \frac{x^T A x}{x^T x} = \lambda_k$$

We only included part of the theorem which finds the smallest eigenvalue and corresponding eigenvector under some given constraints. For the complete statement and proof of the Courant-Fischer theorem, see appendix A.

This can be extended to the general case for $k > 2$. The outline for this proof comes from the paper by Von Luxberg [1]. In this case, we define a cluster indicator vector f_k by

$$f_j(v_i) = f_j(i) = \begin{cases} \frac{1}{\sqrt{vol(A_j)}}, & \text{if } v_i \in A_j \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

We define the matrix F as the matrix whose columns are the p indicator vectors. Then, $f_i^T f_j = 0$, $f_i^T D f_i = 1$ and $f_i^T L f_i = \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}$. Thus the NCut optimization problem becomes

$$\begin{aligned} \underset{A_1,\ldots,A_k}{\text{minimize}} \quad & Tr(F^T L F) \\ \text{subject to} \quad & F^T D F = I \\ & F \text{ defined in (11)} \end{aligned} \tag{12}$$

Relaxing the second constraint and substituting $T = D^{1/2} F$ gives of the relaxation problem of

$$\begin{aligned} \underset{T \in \mathbb{R}^{n \times p}}{\text{minimize}} \quad & Tr(T^T D^{-1/2} L D^{-1/2} T) \\ \text{subject to} \quad & T^T T = I \end{aligned} \tag{13}$$

This is a standard trace minimization problem in which the solution T is a matrix whose columns are the first p eigenvectors of L_{sym}. The proof of the standard trace minimization problem will be provided in appendix A. L_{sym} is the normalized laplacian matrix defined as

$$L_{sym} = D^{-1/2} L D^{-1/2}. \tag{14}$$

Thus the first p eigenvectors will solve the relaxed version of the min NCut problem.

2 Approach

The following subsections outline the various steps of the project. We will start by developing code to produce a similarity graph from our database. Given the similarity graph we will compute the normalized laplacian matrix. From there we will compute the first p eigenvectors of the laplacian and place in a matrix, perform a dimension reduction on the matrix of eigenvectors and use a clustering algorithm on the reduced dimension in order to cluster the data points.

2.1 Similarity Graph

Given the data set $X_1, ..., X_n$ and a notion of "similar", a similarity graph is a graph where X_i and X_j have an edge between them if they are considered "similar". Consider the Gaussian similarity

function which is defined as $s(X_i, X_j) = e^{\frac{-||X_i - X_j||^2}{2\sigma^2}}$ where σ is a parameter to be determined which varies depending on the dataset used. We used the Gaussian similarity function to define the similarity between any two data points. We then defined a threshold ϵ that determined if two data points are similar enough. If $s(X_i, X_j) > \epsilon$ we considered them similar and connected an edge between X_i and X_j. For the project, choosing the best ϵ was not immediately apparent. In order to choose the best parameters, we randomly select a subset of images from our training set and implemented the algorithm with varied parameters. The parameters that gave the best clustering results are the ones we used for other testing purposes. Note that since each $X_i \in \mathbb{R}^{28\times 28}$, to compute the distance between any two points we use the ℓ^2 norm for matrixes in this case given as

$$||X_i - X_j||_2^2 = \sum_{k=1}^{28}\sum_{l=1}^{28}(X_i(k,l) - X_j(k,l))^2$$

2.2 Laplacian Matrix

Recall that the unnormalized laplacian matrix is defined as $L = D - W$. The normalized laplacian is

$$L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}. \tag{15}$$

The eigenvectors of the normalized laplacian are directly related to the indicator vectors in the NCut problem. Thus finding the p first eigenvectors of the normalized laplacian will allow us to reduce the dimension while still capturing the similarity of the data points and can clustering into k partitions. Recall W is the adjacency matrix. Given the Gaussian similarity function, W and D are defined as

$$W = (w_{ij})_{1\le i,j\le n}, w_{ij} = \begin{cases} 1, & \text{if } \ s(X_i, X_j) > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$D = (d_{ij})_{1\le i,j\le n}, d_{ij} = \sum_{j=1}^{n} w_{ij} \tag{17}$$

which was used in computing the normalized laplacian. Our computation of the normalized laplacian matrix was validated by verification of one of the known eigenvectors. As described above in the motivation section of this report, $L\mathbf{1} = 0$ gives an eigenvalue of 0 corresponding to the eigenvector $\mathbf{1}$. This is equivalent to $L_{sym}D^{1/2}\mathbf{1} = 0$ or the normalized laplacian having an eigenvector of $D^{1/2}\mathbf{1}$. Thus after computing the normalized laplacian in Matlab, it was verified by multiplying L_{sym} by its eigenvector of $D^{1/2}\mathbf{1}$ to obtain a value of zero (its corresponding eigenvalue).

2.3 Computing the first p eigenvectors

We then computed the first p eigenvectors of the normalized laplacian matrix. We used an iterative method called the Power Method to find them. Let $B = D^{-1/2}WD^{-1/2}$.

The largest eigenvalue of B corresponds to the smallest eigenvalue of L_{sym}. Hence this gives us our first eigenvector that we are looking for. To find the next eigenvalues, we implemented the power method with deflation. But when using the power method on any symmetric matrix, the eigenvalues found are the largest ones in magnitude. Since we only want the first p largest we modify our B matrix to make B positive semidefinite which will shift all the eigenvalues to ensure they are positive and we then find the first p of them. To make B a positive semidefinite matrix, we create a new B, denoted B_{mod}, such that

$$B_{mod} = B + \mu I$$

This makes B_{mod} diagonally dominant and hence positive semidefinite. Letting $\mu =$ the largest row sum of B, this gave us B_{mod} which is now a positive semidefinite matrix and we find its largest p eigenvalues and corresponding eigenvectors.

So we actually run the power method on B_{mod} to obtain the first (eigenvalue, eigenvector) pair and use the deflation algorithm in order to find the first nontrivial p of them. Below are outlines of the power method and deflation algorithms.

Power Method Algorithm on Matrix B_{mod}
Start with an initial nonzero vector, v_0.Set tolerance, max iteration and iteration= 1
Repeat
$v_0 = B_{mod} * v_0$;
$v_0 = v_0/norm(v_0, 2)$;
lambda= $v_0' * B_{mod} * v_0$;
converged = (norm($B_{mod} * v_0$− lambda∗v_0,2) < tol);
iter=iter+1;
if iter >= maxiter
warning('Did Not Converge')
Until Converged

Deflation Algorithm for finding the first p eigenvalues
Initialize $d =$ length(B_{mod}); $V =$ zeros(d,p); lambda=zeros(p,1);
for j from $1, \ldots, p$
[lambda(j), V(:,j)] = power-method(B_{mod},v_0);
$B_{mod} = B_{mod}$− lambda(j)∗$V(:, j) * V(:, j)'$;
$v_0 = v_0 - \frac{v_0 \cdot V(:,j)}{v_0 \cdot v_0} * v_0$
end

where λ_j, v_j are the previous eigenvalues and eigenvectors found respectively. Initially v_0 is a randomize vector to approximate the first eigenvector. Then in the deflation algorithm, the j^{th} eigenvector is approximated using the previous initial vector v_0 with the projection of the previous eigenvectors subtracted from it to obtain the new v_0. This removes the orthogonal components of the previous eigenvectors found.

The speed of convergence of this method depends on the size of the eigengap $\gamma_k = |\lambda_k - \lambda_{k+1}|$. [1]. The larger the eigengap, the faster the convergence of the algorithm in computing the first p eigenvectors.

We then put these first p eigenvectors into a matrix and normalize the rows. Let $T \in \mathbb{R}^{n \times p}$ be the eigenvector matrix with rows having norm 1. Set

$$t_{i,j} = \frac{v_{i,j}}{(\sum_s v_{i,s}^2)^{1/2}}$$

Row Normalize on Matrix V
Initialize T =zeros(N, p)
for j from $1, \ldots, N$
rowSum=sum$(V(i,:)^2)$;
$T(i,:) = V(i,:)/(\text{rowSum}^{1/2})$;
end

This will transforms our matrix V consisting of the first p eigenvectors to a new matrix T.

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} & \ldots & v_{1p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{i1} & v_{i2} & v_{i3} & \ldots & v_{ip} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & v_{n3} & \ldots & v_{np} \end{bmatrix} \Rightarrow \begin{bmatrix} t_{11} & t_{12} & t_{13} & \ldots & t_{1p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{i1} & t_{i2} & t_{i3} & \ldots & t_{ip} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & t_{n3} & \ldots & t_{np} \end{bmatrix}$$

We will then project the eigenvectors onto new space. Let $y_i \in \mathbb{R}^p$ be a vector from the i^{th} row of T. This will form the new matrix $Y = T^T$ where each y_i vector is a column of Y.

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} & \ldots & t_{1p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{i1} & t_{i2} & t_{i3} & \ldots & t_{ip} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & t_{n3} & \ldots & t_{np} \end{bmatrix} \Rightarrow y_i = \begin{bmatrix} t_{i1} \\ t_{i2} \\ \vdots \\ t_{ip} \end{bmatrix}$$

We then performed a clustering algorithm on our new matrix Y of a reduced dimension.

2.4 Clustering

We perform a standard k-means clustering algorithm on the new set of vectors of reduced dimension. Since each y_i just corresponds to a row in matrix T, we use k-means clustering in order to cluster the rows of our matrix T.

Clustering Algorithm for clustering into k clusters Randomly select k cluster centroids, z_j Repeat Calculate the distance between each y_i and z_j Assign the data point to the closest centroid Recalculate centroids and distances from data points to new centroids If no data point was reassigned then stop, else reassign data points end

Finally, we assigned the original point X_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

2.5 Clustering Classification

After using k-means on the reduced dimension, this classifies each image to a particular cluster. Since it's not necessarily true that cluster 1 corresponds to digit 0, we developed two methods in order to best classify which cluster belongs to which digit. Both methods compute the error rate by computing $\frac{\text{number of incorrect digits in the cluster}}{\text{total number of digits in the cluster}}$. We then did a comparison to determine which method had the lower error rate as well as the fastest computational time.

2.5.1 Clustering Classification: Method 1

- We compute the distance of 10 fixed images to every image in a given cluster.
- The cluster with the smallest distance to that image is classified as that image.

2.5.2 Clustering Classification: Method 2

- We look at all possible permutations of cluster classifications and choose the one with the smallest classification error rate.
- For example define $\pi = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 2 & 6 & 8 & 4 & 9 & 1 & 0 & 5 & 3 \end{bmatrix}$ as one possible permutation for cluster classification
- Count the number of correct matches and repeat for all permutations

2.5.3 Comparison of Methods

Time in seconds for Method 1 and Method 2 for various number of digits and images per digit. Let d represent the number of digits used in the comparison and N represent the number of images per digit used in the comparison.

	d=5,N=500	d=5,N=1000	d=10,N=500	d=10,N=1000
M1	1.447s	2.712s	10.915s	21.945s
M2	0.008s	0.021s	623.524s	1316.550s

Classification Error from Method 1 and Method 2 for d=5 and N=1000

Index	1	2	3	4	5
Cluster Class	2	3	4	1	0
M1	26%	66%	16%	61%	80%
M2	26%	66%	16%	61%	80%

Classification Error from Method 1 and Method 2 for d=10 and N=1000

Index	1	2	3	4	5	6	7	8	9	10
Class	8	1	7	0	2	9	5	3	4	6
M1	17%	64%	18%	87%	11%	18%	05%	03%	0%	12%
Class	3	1	7	0	5	4	8	6	9	2
M2	43%	64%	18%	87%	21%	31%	04%	27%	03%	14%

Overall method two gave better results than did method one. However, when implementing on the large testing set takes a longer time using method two than method one.

2.6 Addition of Single Datapoint

In implementing the spectral clustering algorithm, the question came up of how to determine the correct cluster of a single digit, say X_{new}, without prior knowledge of what that digit is. There is a standard method known as the Nystrom Method for out of sample extensions. The goal of this method is to use the similarity kernel function, denoted as $K(X, Y)$ in order to embed the new data point X in the reduced dimension.

Proposition 2. *Let $K(x_i, x_j)$ denote a kernel function of L_{sym} such that $L_{sym}(i, j) = K(x_i, x_j)$. Let (v_k, λ_k) be an (eigenvector,eigenvalue) pair that solves $L_{sym}v_k = \lambda_k v_k$. Let (f_k, λ'_k) be an (eigenfunction,eigenvalue) pair that solves $Kf_k = \lambda'_k f_k$. Then $y_k(x)$ is the embedding associated with a*

new datapoint x.

$$\begin{aligned}
\lambda_k' &= \frac{1}{n}\lambda_k \\
f_k(x) &= \frac{\sqrt{n}}{\lambda_k}\sum_{i=1}^{n} v_{ik}K(x, x_j) \\
f_k(x_i) &= \sqrt{(}n)v_{ik} \\
y_k(x) &= \frac{f_k(x)}{\sqrt{n}} = \frac{1}{\lambda_k}\sum_{i=1}^{n} v_{ik}K(x, x_j) \\
y_k(x_i) &= y_{ik}
\end{aligned}$$

where n is the number of points used. [6]

Being that we used the Gaussian similarity function to define our similarity matrix and set a threshold that changed our matrix to 0's and 1's, the kernel function is not so easily defined. So instead in the method described below we did not use a similarity kernel function but rather just projected our new image onto the reduced dimension.

- Create a similarity vector, denoted as X_{sim} of 0's and 1's to measure the similarity between the new digit and the previous digits tested on.
- Normalize the similarity vector by multiplying it by $D^{-1/2}$ similar to how we computed the normalized laplacian matrix
- Compute the projection of the similarity vector onto the eigenvectors of the normalized laplacian matrix and normalize to 1. This projection is done by computing the inner product of X_{sim} with each of the eigenvectors. This will give a new vector, denoted as C_{sim} that lives in $\mathbb{R}^p$ where again p is the number of eigenvectors found.
- Find the centroid that is closest to C_{sim} by calculating the distance between C_{sim} and each centroid and finding the minimum.
- Finally, using the cluster classification, assign the new digit X_{new} to its correct digit.

Below is a table of error percentage and run time that was obtained by using this method to add a single new datapoint and averaged over 100 times of implementing.

	Error	Runtime
Averaged over 100 digits	61%	12.6sec

To create X_{sim} we measured the similarity between X_{new} and 2000 previously tested images. If we implemented this measuring the similarity between X_{new} and more images already tested on, we would obtain a lower error.

3 Implementation

The spectral clustering algorithm outlined above was implemented in the programming language Matlab R2016b. This is the programming language that we are most comfortable with using with the most prior knowledge of this language. This was ran on a personal laptop, a Macbook Pro which is a 2.5 GHz Intel Core processor and has 4 GB of Memory. For large subsets (ie greater than 20,000 images) the code was also ran on a more powerful desktop computer located in the Norbert Wiener Center. The desktop computer has 128GB of memory and was implemented on Matlab 2015b.

4 Database

The main database used was the MNIST Handwritten digits database. The database includes a training set of 60,000 images and a testing set of 10,000 images. We used various subsets of the training set of 60,000 images in order to find the best parameters that gave the smallest clustering error. Each image is of size 28×28 pixels. We denote an image $X_i \in \mathbb{R}^{28\times 28}$. A variety of methods have been tested using this database. For this project, we used these images for analyzing the spectral clustering algorithm to see if we can cluster the images such that same digits are clustered together despite the different handwritings. In our code, each image is read into a 3 array called $image(n1, n2, N)$ where n1,n2 represents the size of the image, and N represents its number in the entire list of training or testing sets. Thus N ranges from $1 - 60,000$. The link below can be used to view this database.

http://yann.lecun.com/exdb/mnist/

The following results are based on using subsets of size $N = 1000$, $N = 2000$, $N = 10000$, and $N = 20000$. We used $\sigma = 2000$ and varied ϵ in order to produce the smallest error. We also could have kept ϵ a constant and varied σ to obtain the smallest error but we believe it would have resulted in similar results. We used ϵ varying between 0.30 and 0.48 depending on the dataset size, $p = 3$ (first 3 nontrivial eigenvectors), and k=10 (number of clusters).Error=$\frac{\text{Number of incorrect digits in cluster}}{\text{Total number of digits in cluster}}$

For N=1000: $\epsilon = 0.47$, cluster classification, classification error and total error

Digit	0	1	2	3	4	5	6	7	8	9
Cluster Class	9	4	5	6	1	10	3	7	8	2

Cluster class	1	2	3	4	5	6	7	8	9	10
Error	35%	80%	22%	43%	51%	50%	61%	73%	13%	91%

Overall Error=$\frac{\text{Total number of incorrect digits}}{\text{Total number of digits}} = 53\%$

For N=2000: $\epsilon = 0.343$, cluster classification, classification error and total error

Digit	0	1	2	3	4	5	6	7	8	9
Cluster Class	6	3	9	5	4	10	2	1	8	7

Cluster class	1	2	3	4	5	6	7	8	9	10
Error	50%	44%	31%	60%	51%	19%	68%	64%	54%	71%

Overall Error=$\frac{\text{Total number of incorrect digits}}{\text{Total number of digits}} = 50\%$

For N=10000: $\epsilon = .3405$, cluster classification, classification error and total error

Digit	0	1	2	3	4	5	6	7	8	9
Cluster Class	3	7	6	9	2	10	4	1	8	5

Cluster class	1	2	3	4	5	6	7	8	9	10
Error	41%	65%	12%	47%	80%	44%	28%	68%	49%	89%

Overall Error=$\frac{\text{Total number of incorrect digits}}{\text{Total number of digits}} = 49\%$

For N=20000:$\epsilon = .34$ Cluster classification, classification error and total error

Digit	0	1	2	3	4	5	6	7	8	9
Cluster Class	8	1	3	10	6	7	2	5	9	4

Cluster class	1	2	3	4	5	6	7	8	9	10
Error	41%	27%	49%	47%	40%	60%	87%	12%	49%	48%

Overall Error=$\frac{\text{Total number of incorrect digits}}{\text{Total number of digits}} = 48\%$

5 Validation

There are various phases in which we can validate various steps of the algorithm.

5.1 Validation of Computation of Eigenvectors

We compared our results from the power method algorithm with the eigenvectors computed by using the Matlab command eigs(L_{sym}). Figures 1 and 2 (located at the end of the report) show the first 5 eigenvalues found and a comparison to the ones found using the built in Matlab function eigs on 2000 and 10000 images respectively.

The residual between the first 5 eigenvectors found from the power method and the eigs function were also computed. Both the graph and tables are shown for first a set of 2,000 images and then a set of 10,000 images.

	λ_1	λ_2	λ_3	λ_4	λ_5
r	4.24E-15	2.40E-15	1.82E-13	2.65E-12	1.38E-08

	λ_1	λ_2	λ_3	λ_4	λ_5
r	9.17E-15	2.64E-15	7.94E-14	8.44E-12	1.08E-08

Here, $r = ||B_{mod}/\lambda_i * v_i - B_{mod}/\hat{\lambda}_i * \hat{v}_i||_2$ where v_i, λ_i comes from using the power method and $\hat{v}_i, \hat{\lambda}_i$ comes from the eigs function and $1 \leq i \leq 5$ corresponding to the first 5 eigenvalues. In both cases the first few eigenvalues found from the power method give a reasonable approximation to the ones found from the Matlab command. After the fifth eigenvector and so on, the residuals found start to get larger mostly because of rounding errors. The deflation algorithm relies on available eigenvalues and eigenvectors already found. Since the previous eigenvalues and eigenvectors were estimates that came from the power method, the new approximated matrix is subject to round off errors and thus the error can increase as more eigenvectors are found. However, since we will only be using the first 3 or 4 non-trivial eigenvectors more specifically v_2, v_3, v_4 and possibly v_5, with pretty good confidence the power method can still be applied to larger subsets in the training set of 60,000 images and a good approximation to those eigenvalues and eigenvectors can be used.

5.2 Validation of k-means clustering

We validated the k-means clustering algorithm on a well known clustered set. Since we are able to repeat the initial randomize starting centroids, we can repeat the algorithm on say the Swiss Roll dataset to obtain a "good" clustering. Figures 3 and 4 show the Swiss Roll dataset in 2 and 3 dimensions. Figures 5 and 6 show the clustering of the Swiss Roll dataset into four clusters using matlab's built in k-means and our implementation of the k-means algorithm, respectively.

5.3 Validation of Final solution

We visually validated the solution of the algorithm by displaying the clusters and seeing if similar images are grouped together as predicted. Figures 7,8 and 9 are the visual outputs of a subset of each of the clusters.

6 Testing

For more testing purposes we implemented the spectral clustering algorithm on another database. We used the public database supplied by Yale University called "The Yale Face Database". This database includes 165 grayscale images of 15 individuals. There are 11 images per individual, each having a different facial expression or configuration which are center-light, with glasses, happy, left-light, without glasses, normal, right-light, sad, sleepy, surprised, and one eye winked. Each of these images are of size 32×32 pixels. The objective in analyzing spectral clustering using this database was to see if the same individual can be clustered with all of his images despite the different configurations. The following results are obtained by using a random subset of the database using 10 individuals and 5 images per individual. The link below can be used to view this database.

http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

When testing on this database, we expected the algorithm to give similar results. Ultimately the algorithm should cluster similar faces together and put in different clusters of faces that were dissimilar and hopefully have around the same error rate (or less) than the results from the MNIST digits database. Below are the results using $\sigma = 2000$, $\epsilon = .465$, $p = 4$ and $k = 10$, the best cluster classification, a table of error for each cluster classification. Figures 10,11 and 12 show visual validation of results. We see that we get a little over half of the images getting put into the incorrect cluster. I believe this may be due to the size of the dataset as well as the various lighting. As we saw from the MNIST digits database, the more images we used for clustering, the better our clustering results became. Being that we only used a total of 50 images, I believe extending this to more images can result in a lower overall error. Also being that we are measuring the distance between two grayscale images it may seem that images that have poor lighting or are darker may seem more closely related to images that are brighter. With more time I could try implementing again so that each image has about the same lighting intensity and hence would probably give much better results.

Image	1	2	3	4	5	6	7	8	9	10
Cluster Class	5	6	8	4	2	7	9	10	3	1

Error=$\frac{\text{Number of incorrect faces in cluster}}{\text{Total number of faces in cluster}}$

Cluster class	1	2	3	4	5	6	7	8	9	10
Error	71%	33%	60%	83%	0%	66%	44%	40%	60%	66%

Overall Error=$\frac{\text{Total number of incorrect faces}}{\text{Total number of faces}} = 54\%$

7 Project Schedule/ Milestones

We have split the project into different phases and allocated time to complete each phase.

- End of October/ Early November
  Develop code to generate a Similarity Graph and Normalized Laplacian matrix from the MNIST database. This will include testing for the correct parameter σ in the Gaussian Similarity function as described previously.

- End of November/ Early December
  Compute first p eigenvectors of the Normalized Laplacian matrix as well as validate this. Also prepare for the mid-year presentation and report.

- February
  Normalize the rows of matrix of eigenvectors and perform dimension reduction.

- March/April
  Cluster the points using k-means clustering algorithm and validate this step.

- End of Spring semester: Implement entire algorithm, optimize and obtain final results as well as prepare for the final presentation and final report.

8 Deliverables

The deliverables for this project are the MNIST digits database and Yale face database with the code that delivers this. We will deliver code that implements the spectral clustering algorithm and code that was use for testing and validations at various steps. If time allows this code will be optimized for effective performance. We will also deliver reports at the various periods throughout the course as requested which covers the approach, implementation, validation, testing and milestones of the project. Finally we will give the various presentations throughout the course that introduce the project, give a mid-year update and a final presentation and this report of results found.

9 Conclusion

Spectral clustering is a well known clustering technique and gives relatively good clusters depending on the size of the dataset. The more images used, the better the results tend to be as seen on the two databases I have implemented the algorithm on. The benefit of using this clustering algorithm as opposed to another is the ease of implementation and the reduced dimension that the actual clustering takes place. However it is not as advanced as some of the other clustering algorithms such as Hierarchical Clustering.[7] Overall my implementation of the spectral clustering algorithm could possibly be improved based on the normalized laplacian and/or the similarity graph used and well as increading the number of eigenvectors used for the clustering. [1]

10 Appendix

Theorem (Courant-Fischer Theorem). *Given A a Hermitian matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \lambda_n$, let k be a given integer with $1 \leq k \leq n$, and let $w_i \in \mathbb{C}^n$, then*

$$\max_{w_1,w_2,\ldots,w_{k-1}} \min_{\substack{x\neq 0, x\in\mathbb{C}^n \\ x\perp w_1,w_2,\ldots,w_{k-1}}} \frac{x^TAx}{x^Tx} = \lambda_k$$

and

$$\min_{w_1,w_2,\ldots,w_{n-k}} \max_{\substack{x\neq 0, x\in\mathbb{C}^n \\ x\perp w_1,w_2,\ldots,w_{n-k}}} \frac{x^TAx}{x^Tx} = \lambda_k$$

[5.]

Proof. Since A is Hermitian, there exist a unitary matrix $U \in M_n$ such that $A = U\Lambda U^T$ with $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$. Let $1 \leq k \leq n$. If $x \neq 0$ then

$$\frac{x^TAx}{x^Tx} = \frac{(U^Tx)^T\Lambda(U^Tx)}{x^Tx} = \frac{(U^Tx)^T\Lambda(U^Tx)}{(U^Tx)^T(U^Tx)}$$

and $\{U^Tx | x \in \mathbb{C}^n \text{ and } x \neq 0\} = \{y \in \mathbb{C}^n | y \neq 0\}$. Thus if $w_1, w_2, ..., w_{k-1} \in \mathbb{C}^n$ are given, then

$$\begin{aligned}
\inf_{\substack{x\neq 0 \\ x\perp w_1,w_2,\ldots,w_{k-1}}} \frac{x^TAx}{x^Tx} &= \inf_{\substack{y\neq 0 \\ y\perp U^Tw_1,U^Tw_2,\ldots,U^Tw_{k-1}}} \frac{y^T\Lambda y}{y^Ty} \\
&= \inf_{\substack{y^Ty=1 \\ y\perp U^Tw_1,U^Tw_2,\ldots,U^Tw_{k-1}}} \sum_{i=1}^{n} \lambda_i|y_i|^2 \\
&\geq \inf_{\substack{y^y=1 \\ y\perp U^Tw_1,U^Tw_2,\ldots,U^Tw_{k-1} \\ y_k=y_{k+1}=\ldots=y_n=0}} \sum_{i=1}^{n} \lambda_i|y_i|^2 \\
&= \inf_{\substack{|y_1|^2+|y_2|^2+\ldots+|y_{k-1}|^2=1 \\ y\perp U^Tw_1,U^Tw_2,\ldots,U^Tw_{k-1}}} \sum_{i=1}^{k} \lambda_i|y_i|^2 \geq \lambda_k
\end{aligned}$$

This shows that

$$\inf_{\substack{x\neq 0 \\ x\perp w_1,w_2,\ldots,w_{k-1}}} \frac{x^TAx}{x^Tx} \geq \lambda_k$$

for any $k-1$ vectors. But equality will hold for one choice of the vectors which is $w_i = u_{n-i+k}$, where $U = [u_1 \ldots u_n]$. Thus,

$$\sup_{w_1,\ldots,w_{k-1}} \inf_{\substack{x\neq 0 \\ x\perp w_1,w_2,\ldots,w_{k-1}}} \frac{x^TAx}{x^Tx} = \lambda_k$$

and we can replace inf and sup with min and max, respectfully, since the extremum is achieved. The proof for the second case is similar. □

Theorem (Min Trace Problem)**.** *Let A be Hermitian matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \lambda_n$, then*

$$\begin{aligned} &\underset{X \in \mathbb{R}^{n \times k}}{minimize} \quad Tr(X^T A X) = \sum_1^k \lambda_i \\ &subject\ to \quad X^T X = I \end{aligned} \tag{18}$$

and the columns of X contain the corresponding eigenvectors of the k smallest eigenvalues of A.[1.]

Proof. Let $h(X) = tr(X^T A X)$. Then

$$\begin{aligned} h(X+Y) - h(X) &= tr((X^T + Y^T)A(X+Y)) - tr(X^T A X) \\ &= tr(X^T A X) + tr(X^T A Y) + tr(Y^T A X) + tr(Y^T A Y) - tr(X^T A X) \\ &= 2tr(X^T A Y) + tr(Y^T A Y) \end{aligned}$$

Since

$$\lim_{||Y|| \to 0} \frac{tr(Y^T A Y)}{||Y||} = 0$$

and

$$\lim_{||Y|| \to 0} \frac{h(X+Y) - h(X) - tr(Y^T A Y)}{||Y||} = 0$$

then

$$D_X h(Y) = 2tr(X^T A Y)$$

So the lagrange problem to be solved is

$$D_X h(Y) = X^T \Lambda$$

hence

$$\begin{aligned} 2X^T A &= 2X^T \Lambda \\ \Rightarrow AX &= \Lambda X \end{aligned}$$

which gives

$$\begin{aligned} Ax_1 &= \lambda_1 x_1 \\ Ax_2 &= \lambda_2 x_2 \\ &\vdots \\ Ax_k &= \lambda_k x_k \end{aligned}$$

Thus the solution X of the eigenvalue problem is the matrix whose columns are the eigenvectors of the corresponding eigenvalues of A. □

Theorem (Diagonally dominant matrix)**.** *A Hermitian diagonally dominant matrix A with real non-negative diagonal entries is positive semidefinite. [5.]*

Proof. Let A be a Hermitian diagonally dominant matrix with real nonnegative diagonal entries; then its eigenvalues are real and, by Gershgorin's circle theorem, for each eigenvalue an index i exists such that:

$$\lambda \in \left[a_{ii} - \sum_{j \neq i} |a_{ij}|, a_{ii} + \sum_{j \neq i} |a_{ij}|\right]$$

which implies, by definition of diagonally dominance, $\lambda \geq 0$ and thus A is positive semidefinite. □

11 References

[1] Von Luxburg, U. *A Tutorial on Spectral Clustering.* Statistics and Computing, 7 (2007) 4.

[2] Shi, J. and Malik J. *Normalized cuts and image segmentation.* IEEE Transations on Pattern Analysis and Machine Intelligence, 22 (2000) 8.

[3] Chung, Fan. "Spectral Graph Theory". American Mathematical Society. Regional Conference Series in Mathematics. 1997. Ser. 92.

[4] Vishnoi, Nisheeth K. *$Lx = b$ Laplacian Solvers and their Algorithmic Applications.* Foundations and Trends in Theoretical Computer Science, 2012.

[5] Horn, R. and Johnson, C. "Matrix Analysis". Cambridge University Press, 1985.

[6] Benjio, Y, Paiement, J, Vincent, P. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. 2003

[7] Kowsari, K. Comparison three methods of clustering: K-means, Spectral Clustering and Hierarchical Clustering. 2013

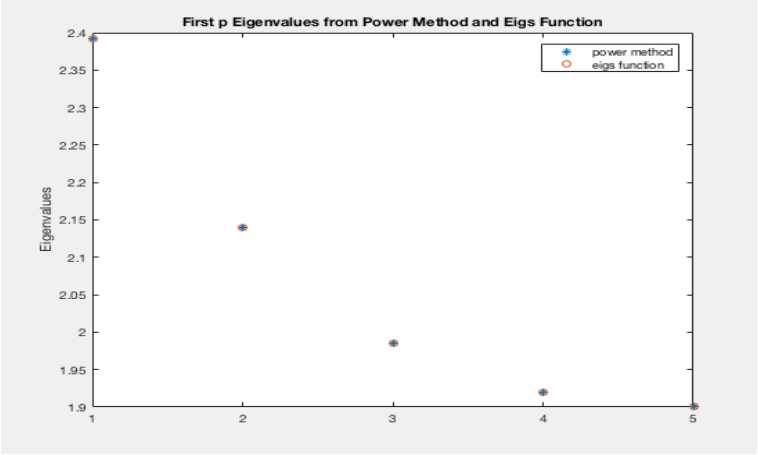


Figure 1: Comparison of eigenvalues found on sample of 2,000 images

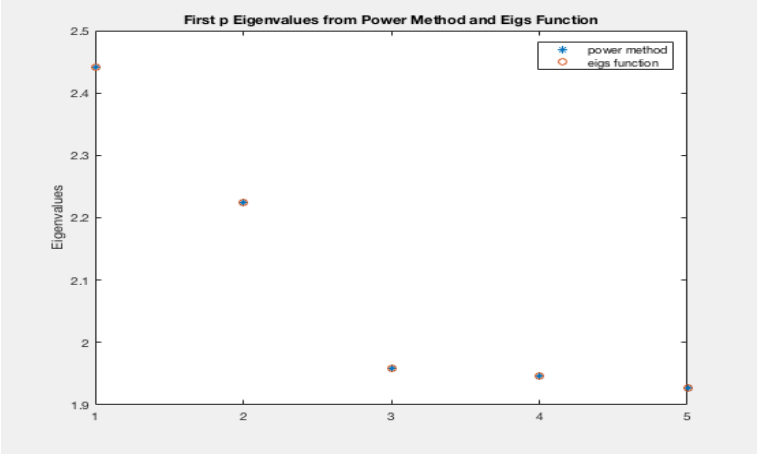


Figure 2: Comparison of eigenvalues found on sample of 10,000 images

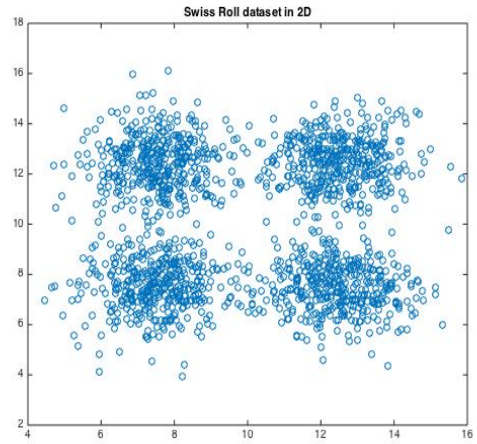


Figure 3: Swiss roll Dataset in 2-dimensions

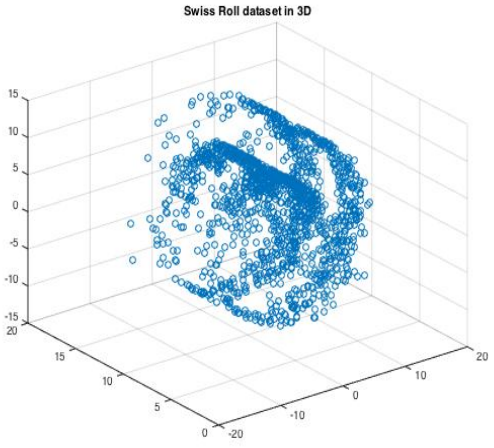


Figure 4: Swiss roll Dataset in 3-dimension

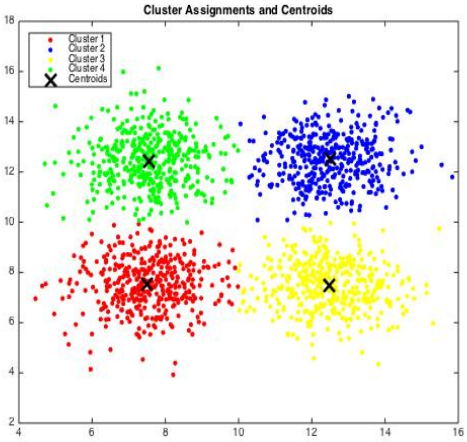


Figure 5: K means clustering via Matlab

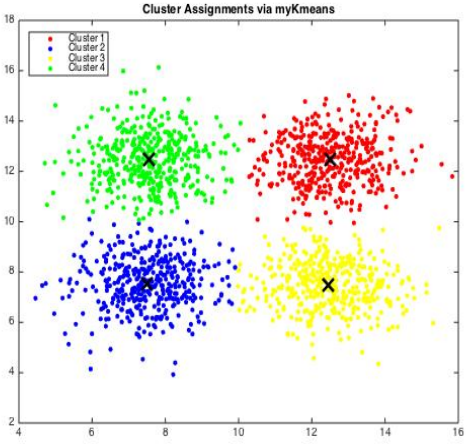


Figure 6: K means clustering via myKMeans

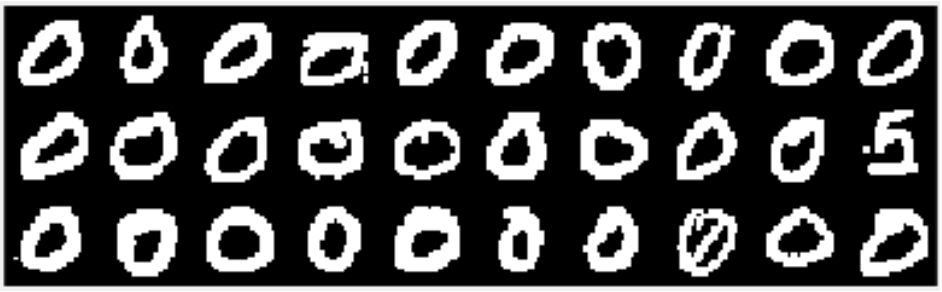


Figure 7: Cluster of digit '0'

Figure 8: Cluster of digit '7'

Figure 9: Cluster of digit '3'

Figure 10: Cluster 5 of face database

Figure 11: Cluster 4 of face database

Figure 12: Cluster 2 of face database