B. TECH. PROJECT REPORT

On

Spectral Clustering based on Growing Vector Quantization for Plant Genome Sequence

BY

Aditya Shah

Anant Lal

Aishwary Gagrani

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

September 2016

Spectral Clustering based on Growing Vector Quantization for Plant Genome Sequence

A PROJECT REPORT

Submitted in partial fulfillment of the
requirements for the award of the degrees
of
BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by:
Aditya Shah
Anant Lal
Aishwary Gagrani

Guided by:
Dr. Kapil Ahuja, Assistant Professor, IIT Indore

INDIAN INSTITUTE OF TECHNOLOGY INDORE
September 2016

CANDIDATE'S DECLARATION

We hereby declare that the project entitled "**Spectral Clustering based on Growing Vector Quantization for Plant Genome Sequence"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Dr. Kapil Ahuja, Assistant Professor, Computer Science and Engineering,** IIT Indore is an authentic work.

Further, I/we declare that I/we have not submitted this work for the award of any other degree elsewhere.

Aditya Shah, Anant Lal and Aishwary Gagrani, 28/11/2016

CERTIFICATE by BTP Guide

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

Dr. Kapil Ahuja, Assistant Professor,
Computer Science and Engineering,
IIT Indore

Preface

This report on "**Spectral Clustering based on Growing Vector Quantization for Plant Genome Sequence**" is prepared under the guidance of Dr. Kapil Ahuja.

Through this report we have tried to give a detailed explanation of our effort to develop a system that identifies similar plant species based on whole genome sequencing by Spectral Clustering and Vector Quantization.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added tables and explanatory images wherever necessary.

Aditya Shah

Anant Lal

Aishwary Gagrani

B.Tech IV Year

Discipline of Computer Science and Engineering

IIT Indore

Acknowledgements

It is our privilege to express our gratitude to several persons who helped us directly or indirectly to conduct this research project work. We express our heart full indebtedness to our BTP guide Dr. Kapil Ahuja for his sincere guidance and inspiration in completing this project.

We are extremely thankful to Mr. Aditya Anand Shastri for his coordination and cooperation and for his kind guidance and encouragement.

I also thank my friends who have more or less contributed to the making of this project.

This study has indeed helped us to explore more knowledgeable avenues related to this topic and we are sure it will help us in future.

Aditya Shah

Anant Lal

Aishwary Gagrani

B.Tech IV Year

Discipline of Computer Science and Engineering

IIT Indore

Abstract

In this project report, we present an adaptable solution for clustering and in turn apply this solution to identify similar plant species based on whole genome sequencing. We combine a novel spectral clustering approach with Vector Quantization technique to alleviate the clustering. We upgrade previous clustering algorithms by leveraging spectral clustering and introducing Vector Quantization to improve the time complexity with a minor tradeoff in accuracy.

The Spectral Clustering uses the eigenvalues and eigenvectors to perform clustering based on k-means algorithm. But for reducing time complexity we have initially done sampling via Vector Quantization. For a better understanding we have compared the results obtained by using:-

a)Spectral Clustering

b)Spectral Clustering with Vector Quantization

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1 Motivation

Clustering is one of the most widely used techniques for exploratory data analysis, with applications ranging from statistics, computer science, biology to social sciences or psychology. In virtually every scientific field dealing with **empirical** data, people attempt to get a first impression on their data by trying to identify groups of **"similar behaviour"** in their data. Compared to the "traditional algorithms" such as **k-means** or **single linkage**, **spectral clustering** has many fundamental advantages. Results obtained by spectral clustering often outperform the traditional approaches, spectral clustering is very simple to implement and can be solved efficiently by standard linear algebra methods.

In this article we would like to improve the existing spectral clustering with the aid of **Growing Vector Quantization**. Also the project provides a good learning opportunity to implement many areas together specifically Optimization and Machine Learning. This has motivated us to work on the project.

2. LITERATURE REVIEW

Here we provide a detailed summary of various research papers telling us about the previous work done in this field:

2.1 Spectral Clustering – 2007 [1]

They defined the notion of **similarity** between the data points

They perform clustering by:

(1) Constructing similarity graph and defining the similarity matrix.

(2) Finding the eigenvectors of laplacian defined from similarity matrix.

(3) Clustering based on eigenvectors found above.

Modelling local neighborhood relationships

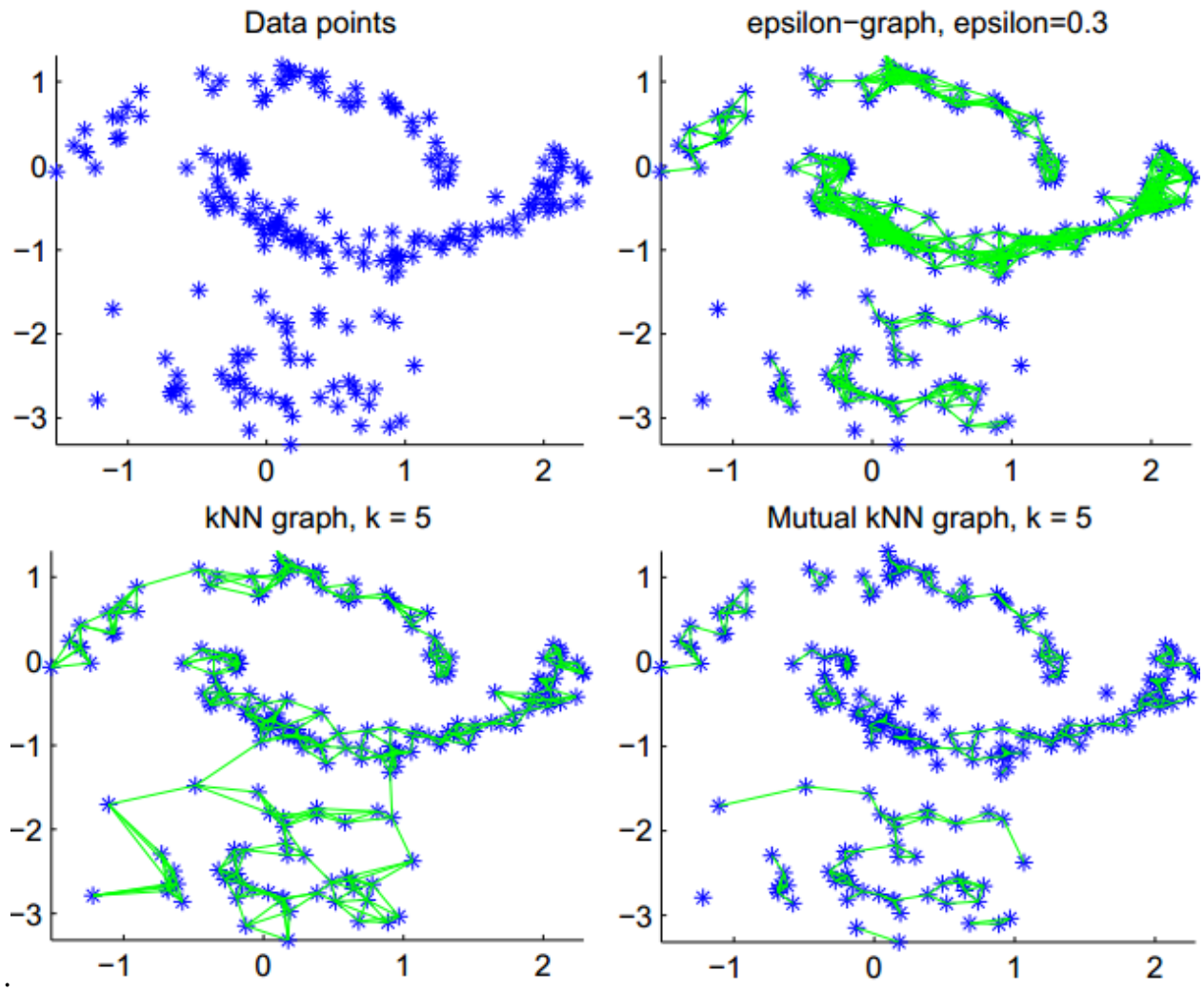
There are several popular constructions to transform a given set $\mathbf{x_1},...,\mathbf{x_n}$ of data points with pairwise similarities $\mathbf{s_{ij}}$ or pairwise distances $\mathbf{d_{ij}}$ into a graph. When constructing similarity graphs the goal is to model the local neighborhood relationships between the data points.

The ε-neighborhood graph: Here we connect all points whose pairwise distances are smaller than **ε**. As the distances between all connected points are roughly of the same scale (at most **ε**), weighting the edges would not incorporate more information about the data to the graph. Hence, the **ε**-neighborhood graph is usually considered as an unweighted graph.

k-nearest neighbor graphs: Here the goal is to connect vertex $\mathbf{v_i}$ with vertex $\mathbf{v_j}$ if $\mathbf{v_j}$ is among the k-nearest neighbors of $\mathbf{v_i}$. However, this definition leads to a directed graph, as the neighborhood relationship is not symmetric. There are two ways of making this graph undirected. The first way is to simply ignore the directions of the edges, that is we connect $\mathbf{v_i}$ and $\mathbf{v_j}$ with an undirected edge if $\mathbf{v_i}$ is among the k-nearest neighbors of $\mathbf{v_j}$ or if $\mathbf{v_j}$ is among the k-nearest neighbors of $\mathbf{v_i}$. The resulting graph is what is usually called the *normal* k-nearest neighbor graph. The second choice is to connect vertices $\mathbf{v_i}$ and $\mathbf{v_j}$ if both $\mathbf{v_i}$ is among the k-nearest neighbors of $\mathbf{v_j}$ and $\mathbf{v_j}$ is among the k-nearest neighbors of $\mathbf{v_i}$. The resulting graph is called the *mutual* k-nearest neighbor graph. In both cases, after connecting the appropriate vertices we weight the edges by the similarity of their endpoints.

The fully connected graph: Here we simply connect all points with positive similarity with each other, and we weight all edges by s_{ij}. As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods. An example for such a similarity function is the **Gaussian** similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$, where the parameter σ controls the width of the neighborhoods. This parameter plays a similar role as the parameter ε in case of the ε-neighborhood graph.

All graphs mentioned above are regularly used in spectral clustering. To our knowledge, theoretical results on the question how the choice of the similarity graph influences the spectral clustering result do not exist.



.

Figure1 : Different Similarity graphs

Limitations

This approach is highly computationally intensive because it needs **O(n^3)** computational operations to perform clustering on **n** data points.

2.2 A Study on Vector Quantization.

Vector quantization is a process that encodes each input vector (a data point) with the closest matching (with minimum distance) vector in current **codebook** (representative data point set), then decodes each vector according to the generated codebook. The crucial part of this method is to design a good codebook (a set of representative data point set).

The well-known algorithm for this design is **Linde-Buzo-Gray** algorithm. The LBG algorithm can provide a encoder that satisfy necessary but not sufficient conditions for optimality. But it needs fixed input number of representative data point and iterated scans of the original data set until the representative data points in the codebook change by a small enough fraction compared with the latest iteration. Given **n** original data points, **k** representative data points in codebook and **t** iterations, LBG needs computational operations **O(nkt)** to construct a codebook. Thus it is time-consuming to applying the traditional vector quantization algorithm directly into the preprocessing of a large set of data point.

Many of the existing vector quantization algorithms (including LBG) neglect the important connection between the decreasing of the original data set by a preprocessing method and the subsequent effect on the clustering which leads to poor accuracy.

2.2.1 Growing Vector Quantization Method (GVQ)

It is hard for a traditional vector quantization algorithm to assign the number of representative data points of a codebook before processing a set of original data points, which is due to the lack of knowledge about the distributional characteristics of the original data set. We also note that a predefined and fixed number

of representative data points is hard for common user to specify, which also troubles the understanding of final clustering results.

When preprocessing a large data set and generating a good set of representative data point to replace the original set, there are two important factors. The first is: the **distortion** of the preprocessing algorithm, which is crucial for the following clusters quality. The second is: the **scan number** of all data points in the original data set, given that many scans of a larger data set is time-consuming. Based on the above considerations, we design a novel growing vector quantization method **GVQ** based on the *minimization of the increment of distortion*, which can produce a good set of representative data point in *one-scan* of the original data set

2.2.2 A Fast Spectral Clustering Method based on Growing Vector Quantization- 2013 [2]

Spectral clustering is a flexible clustering algorithm that can produce high-quality clusters on small scale data sets, but it is limited applicable to large scale data sets because it needs $\mathbf{O(n^3)}$ computational operations to process a data set of **n** data points[1]. Based on the minimization of the increment of distortion, we tackle this problem by developing a novel efficient growing vector quantization method to preprocess a large scale data set, which can compress the original data set into a small set of representative data points in one scan of the original data set. Then we apply spectral clustering algorithm to the small set.

Limitations

This approach has a less accuracy than simple spectral clustering because of the distortion incurred in it.

3. DESIGN AND ANALYSIS OF PROPOSED SYSTEM

3.1 Architecture

Figure 2: Our Clustering Algorithm

Data Set: The input data set

Sampling: Performs sampling on input data set

Clustering: Performs clustering on sampled data

Reverse Mapping: Finds the cluster indices of original data points using the cluster indices of sampled data points.

3.2 Vector Quantization Algorithm

We find the closest matching point of each data point in the current codebook. If we succeed to find it, then proceed to process the next point. If not, then we add a new representative data point in the current codebook and then proceed to next point. During this process, we maintain the mapping between original data point and its corresponding representative data point.

3.3 Spectral Clustering Algorithm

Once the representative data set is generated, we move further to clustering. First we construct the similarity graph (choosing one among the three: **ε-neighborhood, k-nearest neighbor, fully connected**) and then define similarity matrix using it. Next we define the *laplacian* depending on whether we are

normalizing the data set or not. Now the first *k*-eigenvectors of laplacian are found out, on which we perform a small processing and then feed to k-means algorithm to find the cluster number of each representative data point. Finally, using the mapping between the original data set and representative data set, we find the cluster number of each original data point.

3.4 Phylogenetic Trees

A phylogenetic tree or evolutionary tree is a *branching* diagram or "*tree*" showing the inferred evolutionary relationships among various **biological** species or other entities—their phylogeny—based upon **similarities** and **differences** in their **physical** or **genetic** characteristics. The taxa joined together in the tree are implied to have descended from a common ancestor.

In a rooted phylogenetic tree, each node with descendants represents the inferred most recent common ancestor of the descendants, and the edge lengths in some trees may be interpreted as time estimates. Each node is called a **taxonomic** unit. Internal nodes are generally called hypothetical taxonomic units, as they cannot be directly observed

Unrooted trees illustrate only the relatedness of the leaf nodes and do not require the ancestral root to be known or inferred.

Construction of Phylogenetic Trees:

Phylogenetic trees composed with a nontrivial number of input sequences are constructed using computational phylogenetics methods. Distance-matrix methods such as **neighbor-joining** or **UPGMA**, which calculate genetic distance from multiple sequence alignments, are simplest to implement, but do not invoke an *evolutionary* model. Many sequence alignment methods such as **ClustalW** also create trees by using the simpler algorithms (i.e. those based on distance) of tree construction. **Maximum parsimony** is another simple method of estimating phylogenetic trees, but implies an implicit model of evolution (i.e. parsimony). More advanced methods use the optimality criterion of **maximum likelihood**, often within a **Bayesian Framework**, and apply an explicit model of evolution to phylogenetic tree estimation. Here we describe two of these methods.

3.4.1 Neighbor Joining

Neighbor Joining is a bottom-up (*agglomerative*) clustering method for the creation of phylogenetic trees.It is usually used for trees based on DNA or protein sequence data, the algorithm requires knowledge of the **distance** between each pair of taxa (e.g., species or sequences) to form the tree.

Algorithm:

Neighbor joining takes as input a distance matrix specifying the distance between each pair of taxa. The algorithm starts with a completely unresolved tree, whose topology corresponds to that of a star network, and iterates over the following steps until the tree is completely resolved and all branch lengths are known:

1) Initialization - Each node is considered as an *active* node

2) Based on the current distance matrix calculate the matrix **Q**.

$$Q(i,j) = (n-2)d(i,j) - \sum_{k=1}^{n} d(i,k) - \sum_{k=1}^{n} d(j,k)$$

3) Find the pair of distinct taxa **i** and **j** (i.e. with **i != j**) for which **Q(i,j)** has its lowest value. These taxa are joined to a newly created node, which is connected to the central node.

4) Calculate the distance from each of the taxa in the pair to this new node.

$$\delta(f,u) = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}\left[\sum_{k=1}^{n} d(f,k) - \sum_{k=1}^{n} d(g,k)\right]$$

$$\delta(g,u) = d(f,g) - \delta(f,u)$$

5) Calculate the distance from each of the taxa outside of this pair to the new node.

$$d(u,k) = \frac{1}{2}[d(f,k) + d(g,k) - d(f,g)]$$

6) Start the algorithm again if number of active nodes currently is greater than 1, replacing the pair of joined neighbors with the new node and using the distances calculated in the previous step; else stop.

Example:-

Table 1.1: Neighbor Joining Example(1)

Distance Matrix:

	a	b	c	d	e
a	0	5	9	9	8
b	5	0	10	10	9
c	9	10	0	8	7
d	9	10	8	0	3
e	8	9	7	3	0

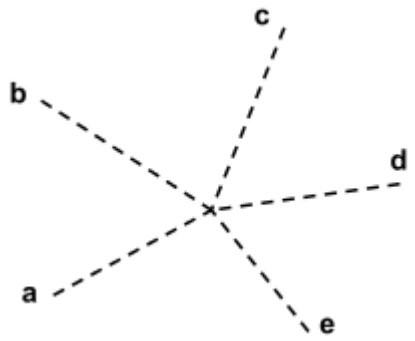


Figure 3.1 : Neighbor Joining Tree(1)

Table 1.2: Neighbor Joining Example(2)

Corresponding Q matrix:

	a	b	c	d	e
a		-50	-38	-34	-34
b	-50		-38	-34	-34
c	-38	-38		-40	-40
d	-34	-34	-40		-48
e	-34	-34	-40	-48	

Table 1.3: Neighbor Joining Example(4)

Distance Matrix:

	u(a,b)	c	d	e
u(a,b)	0	7	7	6
c	7	0	8	7
d	7	8	0	3
e	6	7	0	3

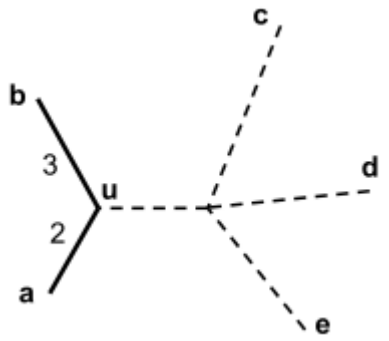


Figure 3.2 : Neighbor Joining Tree(2)

Table 1.4: Neighbor Joining Example(4)

Corresponding Q matrix:

	u	c	d	e
u		-28	-24	-24
c	-28		-24	-24
d	-24	-24		-28

e	-24	-24	-28	

Table 1.5: Neighbor Joining Example(5)

Distance:

	v((a,b),c)	**d**	**e**
v((a,b),c)	0	4	3
d	4	0	3
e	3	3	0

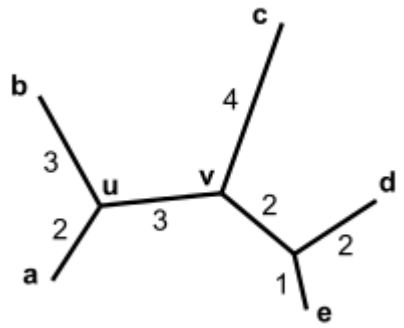


Figure 3.3 : Neighbor Joining Tree(3)

3.4.2 Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

It is a simple agglomerative (bottom-up) hierarchical clustering method for the creation of phylogenetic trees. The algorithm requires knowledge of the **distance** between each pair of taxa (e.g. species or sequences) to form the tree.

Algorithm:

The UPGMA algorithm constructs a rooted tree that reflects the structure present in a pairwise similarity matrix (or a dissimilarity matrix)

1) Initialization - Each point is considered as an individual cluster

2) At each step, the nearest two clusters are combined into a higher-level cluster. The distance between any two clusters **A** and **B**, each of size |**A**| and |**B**|, is taken to be the average of all distances ***d(x, y)*** between pairs of objects ***x*** in **A** and ***y*** in **B**, i.e. the mean distance between elements of each cluster:

$$\frac{1}{|\mathcal{A}| \cdot |\mathcal{B}|} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} d(x, y)$$

3) At each clustering step, the updated distance between the joined clusters **A∪B** and a new cluster **X** is given by the proportional averaging of the $d_{A,X}$ and $d_{B,X}$ distances:

$$d_{(\mathcal{A} \cup \mathcal{B}),X} = \frac{|\mathcal{A}| \cdot d_{\mathcal{A},X} + |\mathcal{B}| \cdot d_{\mathcal{B},X}}{|\mathcal{A}| + |\mathcal{B}|}$$

4) Go to step 2 if number of clusters currently are greater than 1 else stop

Example:

Table 2.1: UPGMA Example(1)

First Clustering:

	a	b	c	d	e
a	0	17	21	31	23
b	17	0	30	34	21
c	21	30	0	28	39
d	31	34	28	0	43
e	23	21	39	43	0

Table 2.2: UPGMA Example(2)

Second Clustering:

	(a,b)	c	d	e
(a,b)	0	25.5	32.5	22
c	25.5	0	28	39
d	32.5	28	0	43
e	22	39	43	0

Table 2.3: UPGMA Example(3)

Third Clustering:

	((a,b),e)	c	d
((a,b),e)	0	30	36
c	30	0	28
d	36	28	0

Table 2.4: UPGMA Example(4)

Final Clustering:

	((a,b),e)	(c,d)
((a,b),e)	0	33
(c,d)	33	0

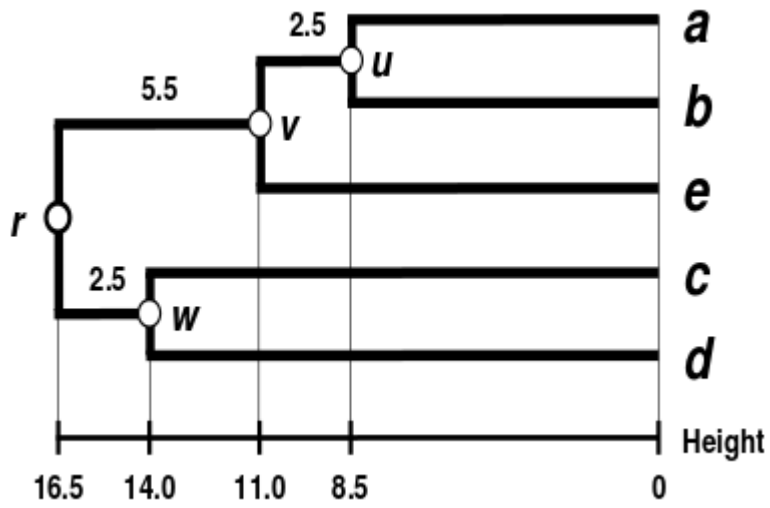


Figure 4 : UPGMA Tree

4. PRACTICAL ASPECT : IDENTIFYING SIMILAR SOYBEAN SPECIES

4.1 Basic Algorithm

Input: A set of input soybean accessions/genotypes $\mathbf{X}=\{X_1,X_2,.....,X_n\}$ and a reference genotype **R**

Step 1) Filtering the input genotypes using the reference genotype.

Step 2) Finding the dissimilarity **S(i, j)** between $\mathbf{X_i}$ and $\mathbf{X_j}$. The dissimilarity between any two genotype $\mathbf{X_i}$ and $\mathbf{X_j}$ is defined as

S(i, j) = SNP(i, j)

where SNP(i, j) = Number of positions at which genotypes $\mathbf{X_i}$ and $\mathbf{X_j}$ differ.

Step 3) Applying the spectral clustering algorithm as described previously

Figure 5 : Identifying Similar Soybean Species

4.2 k-medoids

k-medoids clustering is a partitioning method commonly used in domains that require robustness to outlier data, arbitrary distance metrics, or ones for which the mean or median does not have a clear definition.

It is similar to *k*-means, and the goal of both methods is to divide a set of measurements or observations into *k* subsets or clusters so that the subsets minimize the sum of distances between a measurement and a center of the measurements cluster. In the *k*-means algorithm, the center of the subset is the mean of

measurements in the subset, often called a centroid. In the *k*-medoids algorithm, the center of the subset is a member of the subset, called a medoid.

The *k*-medoids algorithm returns medoids which are the actual data points in the data set. This allows you to use the algorithm in situations where the mean of the data does not exist within the data set. This is the main difference between *k*-medoids and *k*-means where the centroids returned by *k*-means may not be within the data set. Hence *k*-medoids is useful for clustering categorical data where a mean is impossible to define or interpret.

4.3 Applying Vector Quantization on genotypes

We use the k-medoids algorithm to generate a representative set of genotypes

Algorithm:

Input: n data points $\{x_i\}$, i = 1..n and number of representative points k

Output: m-way partition of the input data

1. Perform k-means with k clusters on $x_1, \ldots, x_n$ to:

    a) Compute the cluster centroids $y_1, \ldots, y_k$ as the k representative points.

    b) Build a correspondence table to associate each x_i with the nearest cluster centroid y_j .

2. Run a spectral clustering algorithm on $y_1, \ldots, y_k$ to obtain an m-way cluster membership for each of y_i .

3. Recover the cluster membership for each x_i by looking up the cluster membership of the corresponding centroid y_j in the correspondence table.

5. IMPLEMENTATION AND RESULTS

5.1 Data Sets

Our dataset comprises of two parts:

1) Testing Data Set

2) Real Data Set

Testing Data Set

It contains 5 data sets each containing 30-50 genotypes and the length of each genotype varies between 3000-5000

Real Data Set

It contains 31 genotypes each of length 4847

5.2 Performance parameters

Silhouette : It is a measure of how *similar* an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from **-1** to **1**, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters

For each datum **i**, let **a(i)** be the average dissimilarity of **i** with all other data within the same cluster. We can interpret **a(i)** as how well **i** is assigned to its cluster (the smaller the value, the better the assignment). We then define the average dissimilarity of point **i** to a cluster **c** as the average of the distance from **i** to all points in **c**.

Let **b(i)** be the lowest average dissimilarity of **i** to any other cluster, of which **i** is not a member. The cluster with this lowest average dissimilarity is said to be the "*neighbouring cluster*" of **i** because it is the next best fit cluster for point **i**. We now define a silhouette:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

The average **s(i)** over all data of a cluster is a measure of how tightly grouped all the data in the cluster are. Thus the average s(i) over all data of the entire dataset is a measure of how appropriately the data have been clustered.

Sigma(σ): It decides the shape of gaussian distribution curve.

No. of Clusters(k): It decides into how many clusters the input data should be grouped. Choosing **k** for a given data set also affects the clustering performance, and hence finding optimal **k** for a data set is also a good study.

5.3 Result Comparison: SC+VQ vs PT

Table 3: Comparison of Silhouette values on Test Data 1

No. of clusters (k)	Spectral Clustering	Spectral Clustering + VQ	Neighbor Joining	UPGMA
4	0.6992	0.6992	0.7057	0.6992
5	0.6448	0.6448	0.6688	0.5691
6	0.5959	0.5959	0.5595	0.5131
7	0.5412	0.5412	0.5711	0.4951
8	0.5142	0.5142	0.5267	0.4531
9	0.4888	0.4888	0.5303	0.4475
10	0.4624	0.4624	0.4954	0.4586
11	0.4340	0.4340	0.4716	0.4217
12	0.4162	0.4162	0.1766	0.4164
13	0.3851	0.3851	0.1818	0.3898
14	0.3827	0.3827	0.2134	0.3047

Table 4: Comparison of Silhouette values on Test Data 2

No. of Clusters (k)	Spectral Clustering	Spectral Clustering + VQ	Neighbor Joining	UPGMA
4	0.7007	0.7007	0.7068	0.7007
5	0.6388	0.6388	0.6613	0.5548
6	0.5993	0.5993	0.5569	0.5159
7	0.5409	0.5086	0.5687	0.4898
8	0.5108	0.5064	0.5383	0.4536
9	0.4683	0.4482	0.4538	0.4403
10	0.4456	0.4159	0.3767	0.4500
11	0.4343	0.4376	0.4090	0.4228
12	0.4166	0.4166	0.4200	0.4201
13	0.3845	0.3945	0.2027	0.3945
14	0.3875	0.2909	0.2103	0.3144

Table 5: Comparison of Silhouette values on Test Data 3

No. of Clusters (k)	Spectral Clustering	Spectral Clustering + VQ	Neighbor Joining	UPGMA
4	0.6192	0.6192	0.6417	0.5257
5	0.5458	0.5458	0.5597	0.5284
6	0.4891	0.4891	0.4754	0.4820
7	0.4506	0.4448	0.4841	0.4110
8	0.4186	0.4148	0.4520	0.4186
9	0.3994	0.3994	0.2710	0.3105
10	0.3911	0.3252	0.2886	0.3302
11	0.3616	0.3579	0.3337	0.3634
12	0.3787	0.3686	0.3066	0.3787
13	0.4200	0.4199	0.3483	0.4147
14	0.4315	0.4326	0.3966	0.4429

Table 6: Comparison of Silhouette values on Test Data 4

No. of Clusters (k)	Spectral Clustering	Spectral Clustering + VQ	Neighbor Joining	UPGMA
4	0.6162	0.6162	0.5420	0.5306
5	0.5576	0.5576	0.5963	0.4832
6	0.5141	0.5141	0.5006	0.4728
7	0.4655	0.4236	0.5129	0.4138
8	0.4260	0.4030	0.2518	0.4220
9	0.4033	0.2722	0.2071	0.4031
10	0.3501	0.2851	0.2315	0.3169
11	0.3526	0.2950	0.2306	0.3322
12	0.3882	0.3625	0.2698	0.3560
13	0.4063	0.3160	0.3172	0.3946
14	0.4226	0.4122	0.3519	0.4244

Table 7: Comparison of Silhouette values on Test Data 5

No. of Clusters (k)	Spectral Clustering	Spectral Clustering + VQ	Neighbor Joining	UPGMA
4	0.6620	0.6620	0.6698	0.5757
5	0.5968	0.5968	0.6274	0.5838
6	0.5528	0.5528	0.5303	0.5493
7	0.5050	0.5050	0.5433	0.4690
8	0.4726	0.4726	0.5112	0.4726
9	0.4414	0.1914	0.1813	0.4169
10	0.4197	0.4197	0.2030	0.3210
11	0.3458	0.3364	0.2298	0.3320
12	0.3493	0.2670	0.2600	0.3315
13	0.3479	0.2951	0.2119	0.3362
14	0.3627	0.2993	0.2373	0.3469

Table 8: Comparison of Silhouette values on Real Data

No. of Clusters (k)	Spectral Clustering	Spectral Clustering + VQ	Neighbor Joining	UPGMA
4	0.2053	0.2159	0.2546	0.2192
5	0.2421	0.1700	0.2791	0.2488
6	0.2771	0.2639	0.2389	0.2771
7	0.2990	0.2446	0.2612	0.2736
8	0.3451	0.2727	0.2902	0.2874
9	0.3490	0.2861	0.3110	0.3031
10	0.3522	0.3361	0.3430	0.2966
11	0.3687	0.3035	0.3831	0.3476
12	0.3799	0.3299	0.4089	0.3569
13	0.4329	0.4268	0.4153	0.3829
14	0.4470	0.4128	0.4610	0.4403

6. CONCLUSION

This project was a confluence of Genomic study, Optimization and Clustering in matlab.

The objective of the project was to explore the latest research and technique in the field of Computer Science and develop a sophisticated system to identify similar soybean plant species in a given data set. The implementation of the project was possible only after the detailed study of identifying similar genomes, Vector Quantization techniques, Clustering Concepts and Matlab Programming.

While working on the project, we were successfully able to implement a eigenvalues and eigenvectors inspired algorithm which is able to identify similar soybean species based on their genomic sequences. We compared the results with the standard phylogeny inference methods like phylogenetic trees on our dataset and found that Spectral Clustering is better than the phylogenetic tree method

Once identifying the similarity between various accessions/genotypes we are able to identify similar accessions using Spectral Clustering. The complete project has been implemented on matlab.

7. Future Work

To extend this project in future, a lot of work can be done further on this application. We highlight some of the possibilities here as:

1) Heuristics exist which performs Neighbor Joining in ~ $O(n^2)$

   Complexity of Spectral Clustering + Vector Quantization ~ $O(n^3/x^3) = dn^3/x^3$

   Analyzing VQ method for $n / x > c / d$

2) Applying the idea of dimensionality reduction - compressing genotypes if their lengths are huge.

3) Refining the similarity matrix

4) Use k-means for VQ using consensus sequence and compare results.

8.Bibliography and References

[1] Ulrike von Luxburg: **A Tutorial on Spectral Clustering**. Max Planck Institute for Biological Cybernetics. The article appears in Statistics and Computing, 17(4), 2007.

[2] Xiujun Wang, Xiao Zheng, Feng Qin, and Baohua Zhao: **A Fast Spectral Clustering Method Based on Growing Vector Quantization for Large Data Sets.** School of Computer Science and Technology, Anhui University of Technology, China and School of Computer Science and Technology, University of Science and Technology of China, 2013.

[3] Andrew Y. Ng, Michael I. Jordan and Yair Weiss: **On Spectral Clustering: Analysis and an algorithm**. U.C. Berkeley and School of CS & Engineering, The Hebrew University

[4] Tae-Ho Lee, Hui Guo, Xiyin Wang, Changsoo Kim and Andrew H Paterson: **SNPhylo: a pipeline to construct a phylogenetic tree from huge SNP data**. National Academy of Agricultural Science, University of Georgia, Hebei Union University, Chungnam National University. The article appears in BMC Genomics, February 2014