

Improving neural network based spectral clustering

A Thesis

submitted to

Indian Institute of Science Education and Research Pune

in partial fulfillment of the requirements for the

BS-MS Dual Degree Programme

by

Suraj Yadav



Indian Institute of Science Education and Research Pune

Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

April, 2021

Supervisor: Dr Vaibhav Rajan

© Suraj Yadav 2021

All rights reserved

Certificate

This is to certify that this dissertation entitled Improving neural network based spectral clustering towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Suraj Yadav at Indian Institute of Science Education and Research under the supervision of Dr Vaibhav Rajan, Assistant Professor, Department of Information Systems and Analytics, School of Computing, NUS Singapore, during the academic year 2020-2021.



Dr Vaibhav Rajan

Committee:

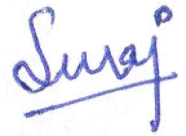
Dr Vaibhav Rajan

Dr Pranay Goel

This thesis is dedicated to my parents Ramrati Yadav and Zile Singh Yadav.

Declaration

I hereby declare that the matter embodied in the report entitled Improving neural network based spectral clustering are the results of the work carried out by me at the Department of Information Systems and Analytics, School of Computing, NUS Singapore, under the supervision of Dr Vaibhav Rajan and the same has not been submitted elsewhere for any other degree.



Suraj Yadav

Acknowledgments

First and foremost, I would like to thank my thesis supervisor, Dr Vaibhav Rajan, for his invaluable guidance and for giving me a great learning experience that I will cherish for years to come. I am truly grateful for the amount of time that he has invested in me, giving me a consistent mentorship. I could not have asked for anything more. I would also like to thank Dr Ragnathan Mariappan for collaborating. His suggestions and views have been crucial in carrying out this project. I am thankful to Debabrata Mahapatra for giving us his key ideas and crucial insights. I will never forget the long discussions that we had, thank you.

I would also like to thank Dr Pranay Goel for being the expert member of the project. He motivated me to pursue clustering analysis during the semester project through an extensive exchange of thoughts. Also, Dr Anindya Goswami, Dr Anisa Chorwadwala and Dr Anupam Kumar Singh have been very supportive and helpful. People in the Mathematics Department have been very kind and considerate. Finally, I am thankful to all my friends who have made my 5 years at the IISER Pune, a memorable experience. Thanks, Rishab and Mrityu for being a part of it.

I will forever be grateful to my parents and sisters for the unconditional love and support.

Abstract

Spectral Clustering is a well known clustering method which overcomes the limitations of traditional clustering algorithms like k-means clustering. The Algorithm involves finding eigenvectors of a graph Laplacian which has two main drawbacks, namely, scalability and out-of-sample predictions. SpectralNet is a deep neural network (NN) based method which overcomes these limitations but uses Cholesky factorization to obtain output orthogonal matrix and is not an end-to-end network. This method only performs well when the Laplacian matrix is highly sparse. The model is also highly sensitive to the hyperparameter setting. We developed an end-to-end neural network architecture called Extended Spectral Clustering (ExSC)¹ which employs β -VAE and cayley map to orthogonalize the input matrices and minimize the spectral loss to update the network weights so as to obtain orthogonal output which better resembles the eigenvector matrix of the Laplacian. The model performs better than the SpectralNet base model. Also, as the model learns an encoder and a decoder, it can also be used as a generative model or a feature extractor to simultaneously perform other tasks.

¹Link to the GitHub repository: <https://github.com/yadav-suraj/ExSC>

Contents

Abstract	xi
List of Figures	3
List of Tables	5
1 Related Work	7
2 Preliminaries	11
2.1 Spectral Clustering	11
2.2 SpectralNet	22
2.3 Trivialization	24
2.4 Variational autoencoders	28
3 Method	35
3.1 Problem Statement:	35
3.2 First method: Geometric Approach	36
3.3 Initial approaches using neural network:	37
3.4 Why Gaussian (Normal) prior	40
3.5 Extended Spectral Clustering	44

4 Experimental Results	47
4.1 Evaluation metric:	47
4.2 Datasets	49
4.3 First Method: Geometric approach	49
4.4 Initial approaches using neural network	50
4.5 Extended Spectral Clustering	51
5 Conclusion and Future work	59
Bibliography	61
Appendix	65

Introduction

Clustering analysis is an essential part of unsupervised learning which has proved to be of great significance in the field of Machine Learning and Deep learning. A few popular application of the field includes feature extraction, disease prediction, exploratory data analysis to figure out subgroups with similar interest in a population, search engines, recommender systems, etc.

Traditional methods such as k-means clustering performs well under the assumption that data is spherical distributed. This is a limitation that spectral clustering (Shi and Malik [2000a], Von Luxburg [2007]) overcomes. There are still two major limitations to the spectral clustering algorithm. Firstly, it is not scalable as finding eigenvectors to the graph laplacian (matrix of size $n \times n$, where n is the number of datapoints; it basically contains information regarding similarity between points) is infeasible when n is large or algebraic multiplicity of an eigenvalue is > 1 . Secondly, it is not possible to make predictions on the unseen datapoints, the problem is referred as inability to do out-of-sample-extension(OOSE). SpectralNet (Shaham et al. [2018]) however overcomes both these limitations but does not have an end-to-end network and also involves finding cholesky decomposition of a matrix.

Spectral clustering is a strong traditional algorithm which finds clusters in small datasets with complicated clusters like as shown in the Figure(1.1). It involves finding an orthogonal matrix over which k-means is performed to obtain the end clusters. As it can be seen that k-means tend to perform very poorly on such datasets and spectral clustering learns these arbitrary shaped clusters very effectively. This advantage of spectral clustering can be leveraged to clusters large datasets with complicated clusters if one can overcome its limitation of not being scalable and inability to do OOSE. The existence of a loss function (a real valued function) with limitations that one can overcome using neural network was the prime motivation of the project. The aim of the project was to build an end-to-end

architecture and improve orthogonality of the matrices over which one performs k-means clustering to obtain end clusters.

Original Contribution

The first two chapters are literature review. Chapter 3 and 4 are the original work which includes the formulation of Algorithm (4) and Section (3.4) which highlights through Proposition (3.4.1) that how selecting gaussian prior in a VAE network could give orthogonal representations of the input. The proposed method of using a β -VAE network for clustering hasn't been proposed in any of the previous work in the field. It was also realized by us that how introducing sparsity in the latent variable through Spike and Slab prior (and approximate posterior) can produce orthogonal representations (latent variables) which are close to the solution of the balanced mincut problem (2.1.5). Extended Spectral Clustering (ExSC)(4) simultaneously leverages variational inference and trivialization to achieve comparable clustering results on benchmark real world datasets. Also, ExSC has a decoder and hence can very well be used for other side jobs such as generative tasks.

Scope of our work

Chapter 1 gives an overview of all the related work which were reviewed and used in the making of the ExSC model. This is followed by Chapter 2 which provides relevant details of these related work upon which the reader can build a strong understanding of what the problem is and the utility of each component of ExSC. The next chapter, Chapter 3 is on methods that were built along the way to tackle the problem of extending Spectral Clustering to neural networks which ends with the details of the ExSC model. Chapter 4 states all the results and involves a discussion over the results as well as the comparison between each of the method used. This is followed by the final chapter, Chapter 5, which summarizes the findings and proposes possible future work. Additionally, Appendix contains all the experimental details and hyperparameter values which will be essential if the reader wants to reproduce the results.

The code for the ExSC model can be found at <https://github.com/yadav-suraj/ExSC>.

List of Figures

1.1	Comparing clustering results of k-means clustering and spectral clustering on toy dataset (Concentric circles). Same colour denotes points belonging to the same cluster.	9
2.1	SpectralNet Model outline where X_m is the mini-batch of size m	22
2.2	Trivialization compared with Dynamic trivialization	26
2.3	Difference between dynamic trivialization(for $K=4$) and RGD on \mathcal{W}	28
2.4	Basic outline of VAE architecture.	28
2.5	Reparameterization trick for gaussian prior and gaussian posterior based VAE model	33
3.1	Basic outline of initial attempts involving orthogonal representations of input and RGD. $Y_{opt} := Y$ obtained upon convergence of spectral loss.	38
3.2	Model architecture as in the Section 3.3.3. d is the number of input features and k are the number of clusters in the dataset.	41
3.3	ExSC architecture as in Section 3.5 (which consists of a β -VAE model with Spike and slab (sns) prior in conjugation with cayley layer). d is the number of input features and k are the number of clusters in the dataset.	41

3.4	Each layers-stack in the Figure(3.3) and Figure(3.2). This assembly of layers forms the basic component of our model	42
4.1	RGD and k-means comparison on toy datasets. Top: entangled CC, Bottom: Concentric circles	50
4.2	Top row shows the original images. Bottom row shows the reconstructed images from the simple β -VAE model.	51
4.3	β -VAE with cayley layer and gaussian prior	52
4.4	Top row shows the original images. Bottom row shows the reconstructed images from the β -VAE model with gaussian prior and cayley layer added.	52
4.5	$\mu_n^T \mu_n$ after training for spike and slab prior. μ_n is normalized μ (Definition 3.4.1).	52
4.6	Top row shows the original images. Bottom row shows the reconstructed images from the β -VAE model with spike and slab prior and cayley layer added.	53
4.7	No cayley layer is used. β -VAE with spectral loss was trained until convergence; Left: Gaussian prior, Right: Spike and slab prior	53
4.8	ExSC model (has Spike and Slab prior)	54
4.9	Overall loss, spectral loss and validation loss for 75nn with sns prior. All loss values are converging.	55
4.10	Confusion matrix for both datasets with predicted labels. Each row and column having single large entry indicates good clustering.	55
4.11	$\mu^T \mu$ for gaussian prior(left) and spike and slab prior(right). Diagonal matrices are better matrices for clustering	56
4.12	Top row shows the original images for FashionMNIST dataset. Bottom row shows the reconstructed images from the β -VAE model with spike and slab prior and cayley layer added.	56

List of Tables

4.1	Results demonstrating impact of the cayley layer addition.	51
4.2	Result comparison of our ExSC model on MNIST dataset with SpectralNet (Shaham et al. [2018])	53
4.3	Result comparison on the MNIST dataset with different values of nearest neighbour for constructing laplacian. Mini-batch size = 100 and number of epochs = 400.	55
4.4	Result comparison on MNIST and FashionMNIST dataset. Note that none of them is an extension of Spectral clustering or uses Laplacian to determine clusters. (Source: Villar-Corrales and Morgenshtern [2020])	57
1	Hyperparameter setting	67

Chapter 1

Related Work

The chapter briefly gives highlight of the traditional method of clustering i.e. spectral clustering and its extension to neural networks through SpectralNet and a brief introduction to the idea of trivialization. These will be discussed in greater details in the following Chapter.

Traditional methods

K-means Clustering

k-means algorithm forms the basis of many algorithm and is one of the widely and well studied clustering algorithm. k-means clustering involves cluster center assignment which changes with each iteration and finally converges to a solution. Algorithm start with initialization of k cluster centers. Distance of each point in the dataset is calculated from each of the cluster centers. Each points is assigned the cluster label to which it is closest to. Hence each of the point is assigned a cluster based on the closest cluster center. Once the clusters are assigned to each point, mean of each cluster is determined by taking mean of all points in each cluster along each feature. Now each of the cluster center is updated to these new corresponding mean of the clusters. The above step of assigning datapoints to one of the cluster centers is repeated until we reach a point where the cluster center do not change upon update.

k-means clustering has many limitations. k-means clustering can only work well for spherical distributed data since it uses euclidean distance to assign each point with a cluster label. k-means is also highly dependent on the starting cluster center assignment and thus in most cases require multiple restarts. k-means algorithm suffer from curse-of-dimensionality if the number of features or points is large. Like any other traditional algorithm it is incapable of overcoming Out-of-sample-extension (we will see this in detail in the coming section).

1.0.1 Spectral Clustering

Spectral clustering is another traditional method of clustering which overcomes the k-means limitation of only finding spherical clusters in the dataset. Spectral clustering is capable of identifying complex clusters in the data set. The algorithm involves construction of graph from the dataset and treat the clustering problem as graph partitioning problem. The graph laplacian is a $n \times n$ matrix which captures similarities between each pair of points. These similarities gives embedding(lower level representations of input) over which k means clustering gives the final clusters. The algorithm is summarized in (1) and is talked in much detail in the subsequent chapter.

Although Spectral clustering finds complex clusters in the data set, it still fails to do Out-of-sample-extension (OOSE). It is also impractical to use spectral clustering for a large dataset as it requires eigenvector calculation of the graph laplacian. Out-of-sample extension is the ability of a model to make predictions on the unseen data. Even in case of addition of a single new point to the dataset, one would need to reapply the complete algorithm to the combined data in order to obtain the clusters. This would also means calculation of eigenvectors of the new laplacian.

SpectralNet

SpectralNet(Shaham et al. (2018)) is an extension of Spectral clustering to the neural network which overcomes both the limitation of Scalability and inability to do OOSE.

The spectral clustering requires calculation of eigenvectors of the graph laplacian. These eigenvectors serves as a solution to the optimization problem (minimizing a real valued function) which is also subjected to the constraint that the solution has to be orthogonal. SpectralNet is a neural network which learns these eigenvectors by minimizing this real valued

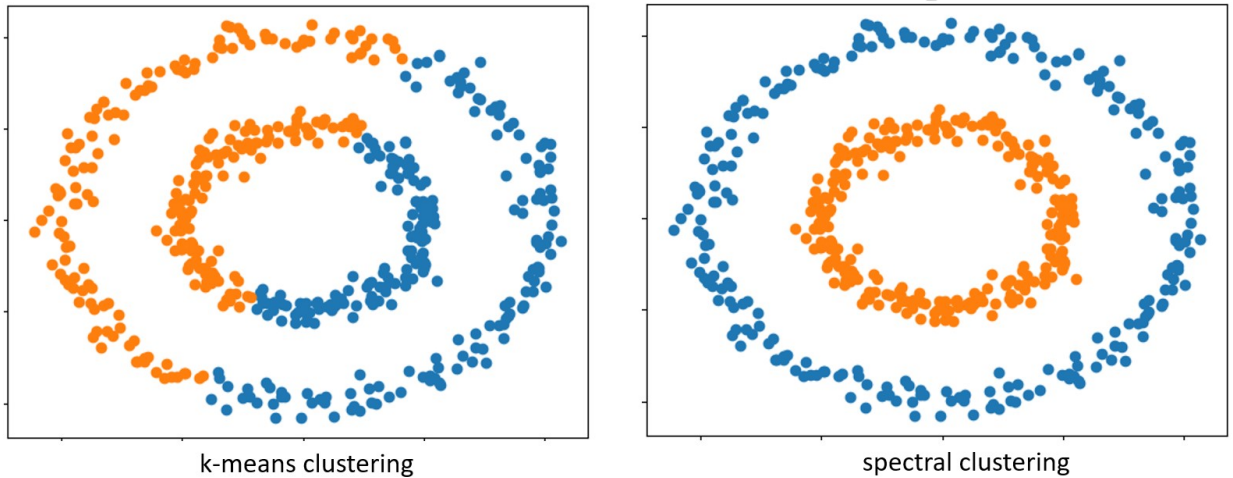


Figure 1.1: Comparing clustering results of k-means clustering and spectral clustering on toy dataset (Concentric circles). Same colour denotes points belonging to the same cluster.

function while satisfying the constraint on the output. SpectralNet implicitly satisfies this orthogonality condition by making use of Cholesky factorization and uses Stochastic gradient descent to update model parameters.

SpectralNet require calculation of cholesky factor of a certain matrix to set weight matrix of the final layer. Cholesky factorization requires a matrix to be non-singular and thus SpectralNet requires multiple restarts with different hyperparameter setting in order to obtain clusters in the data. SpectralNet is highly sensitive to hyperparameter setting and requires hyperparameter tuning in order to decide the best parameters. Hyperparameter tuning requires supervision and with clustering being unsupervised learning, it becomes impractical to use SpectralNet.

Trivialization

Trivialization is generalization of Riemannian gradient descent (RGD). Trivialization translates the optimization problem with manifold constraints into unconstrained optimization problem through parameterization of the manifold in terms of Euclidean space. These parameterization which are called trivializations allows one to use well studied optimization techniques like Stochastic gradient descent or optimizers like Adam, Adagrad or RMSProp to make parameter updates.

Casado [2019] proposes dynamic-trivialization which involves making K (not related to number of clusters in any way) updates on the tangent space around a point and then move on to the tangent space around the new point on the manifold. These tangent vectors are projected on to the manifold through exponential maps after every K updates on the tangent space. For $K=\infty$ dynamic trivialization reduces to RGD and for $k=1$, it reduces to using a single map to parameterize the entire manifold by a euclidean space. Thus dynamic trivialization serves as a generalization of RGD.

Since the problem of spectral clustering involves minimizing a real valued function with orthogonality condition (rectangular orthogonal matrices forms a nice manifold called Stiefel manifold), trivialization is leveraged to solve the spectral clustering's constrained optimization problem.

Chapter 2

Preliminaries

The chapter is a literature review where we build the understanding of what spectral clustering is (Shi and Malik [2000a] and Von Luxburg [2007]) and how it has been extended to neural networks through a model called SpectralNet proposed in Shaham et al. [2018]. The chapter also includes details of trivialization (Casado [2019]) and variational autoencoder (Higgins et al. [2017]) which were later leveraged in our model ExSC to achieve the goal of orthogonalizing the input in a manner so as to obtain solution of the spectral clustering.

2.1 Spectral Clustering

Spectral clustering is a graph partitioning algorithm which partitions nodes of a graph based on weights on the edges (which denotes similarity between the corresponding pair of points). It partitions such that sum of weights going across the partitions is minimized and sum of weights within each partition is maximized. This in fact forms the basis of any clustering algorithm. This is a similarity based clustering where a measure of similarity between each pair of points is established and is used to form clusters. In order to understand the algorithm, we create a graph out of data points details of which will be discussed in the subsequent sections.

2.1.1 Basic definitions and notations

A graph G can be denoted by (V, E) where V is the set of nodes in the graph and E are the edges (pair of nodes). That is, $V = \{x_1, x_2, \dots, x_n\}$ and $E = \{(x_i, x_j) \mid \text{if } i^{\text{th}} \text{ and } j^{\text{th}} \text{ nodes form an edge}\}$. Weights (w_{ij}) on the edges maps relationship between each points. One can think of it as measure of similarity between each pair of points which form an edge. In our setting we assume that $w_{ij} \geq 0$ and $w_{ij} = w_{ji} \quad \forall i, j$.

The weighted adjacency matrix of a graph is the matrix $W = (w_{ij})$. Under the above mentioned assumptions, W is symmetric and each entry is non-negative. $w_{ij} = 0$ for pair of points $(x_i, x_j) \notin E$ (i.e. 0 for pair of points that does not form an edge)

Definition 2.1.1. A degree matrix D is defined as the diagonal matrix with d_1, d_2, \dots, d_n on the diagonal where each d_i is the degree and is defined as

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

d_i is nothing but sum of weights of edges connected to the vertex x_i . For simplicity, $i \in A$ implies vertex $x_i \in A$. For a subset $A \subseteq V$, \bar{A} is the set complement, i.e. $V \setminus A$ and for sets $A, B \subseteq V$ we define

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

$|A|$ denotes the set cardinality and $vol(A) := \sum_{i \in A} d_i$. A subset A is called a connected component if any two points in A can be joined via a path such that all points in the path also belongs to A . A subset A is called a connected component if A is connected and there is no edge going from A to \bar{A} . The non-empty subsets A_1, A_2, \dots, A_k forms a partition of the graph if $A_i \cap A_j = \emptyset$ and $A_i \cup \dots \cup A_k = V$, i.e. each vertex belongs to exactly one of the sets.

2.1.2 Types of similarity graphs

There are different ways to construct the similarity graph and one can choose any of method depending on the type of relation one wants to capture in the data. $G = (V, E)$ denotes the graph that we are constructing. x_1, \dots, x_n are data points and d_{ij} denotes distance between the vertex x_i and x_j .

The ϵ -neighbourhood graph: If the distance between two points is less than ϵ then the two vertices forms an edge in the graph. That is, if $d_{ij} < \epsilon \implies (x_i, x_j) \in E$.

The k-nearest neighbour(knn) graphs: Broadly we want to connect each vertex to its k nearest neighbours. But such a graph would be directed because if x_j is in k nearest neighbours to x_i that doesn't necessarily mean x_i has to be in the k nearest neighbours of x_j . Hence such directed graph can be made undirected in two ways:

- **k-nearest neighbour graph:** join the vertices x_i and x_j if either x_i belongs to the k-nearest neighbours of x_j or if x_j belongs to the k-nearest neighbours of x_i .
- **Mutual k-nearest neighbour graph:** join the vertices x_i and x_j if both x_i belongs to the k nearest neighbours of x_j **and** x_j belongs to the k nearest neighbours of x_i .

After forming the edges one could put weight w_{ij} on the edges using a similarity function. Gaussian kernel is one choice for the similarity function. Gaussian kernel (or gaussian similarity function) is defined as

$$k(i, j) := e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \quad (2.1)$$

where σ parameters maps the local neighbourhood relationship.

The fully connected graph: Each point in the graph is connected to every other point in the graph and weight on the edges is again decided using a similarity function. These graph require careful selection of similarity function which models the local neighbourhood relationship very well. Again, gaussian kernel is a common choice that goes with fully connected graphs. Note that in gaussian kernel points which are far away have small weight on the edge between them and points which are close to each other have large weight on the edge between them.

There is no theoretical evidence which proves that a single type of graph works well across every dataset but knn graphs and fully connected graphs are a common choice.

2.1.3 Graph Laplacians:

Remark 2.1.1. *Throughout the document, by “first k eigenvector” we refer to the eigenvectors corresponding to the smallest k eigenvalues. Also, “eigenvector matrix $M \in \mathbb{R}^{n \times k}$ ”*

refers to the matrix constructed by putting first k eigenvectors along the column. That is, A is called a eigenvector matrix of $B \in \mathbb{R}^{n \times n}$ if $A = [u_1, u_2, \dots, u_k]$ where $u_1 \in \mathbb{R}^n$ is the eigenvector corresponding to the smallest eigenvalue λ_1 of B , $u_2 \in \mathbb{R}^n$ is the eigenvector corresponding to the second smallest eigenvalue λ_2 of B and so on.

A few assumption that we will make are G is undirected, weight matrix (also called affinity matrix) W is symmetric and each of its entry is non-negative.

Definition 2.1.2. A unnormalized graph laplacian L is defined as

$$L = D - W \tag{2.1}$$

where D is the degree matrix and W is the affinity matrix (or the weight matrix).

Lemma 2.1.1. A matrix M is positive semi-definite iff all its eigenvalues are non-negative.

Proposition 2.1.2. L satisfies the following properties:

1. every vector $f \in \mathbb{R}^n$ satisfies

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2$$

2. L is symmetric and positive sem-definite.

3. The smallest eigenvalue of L is 0 and eigenvector corresponding to it is $\mathbf{1}$ (constant one vector).

4. L has n non-negative real valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

Proof. 1.

$$\begin{aligned} f^T L f &= f^T D f - f^T W f = \sum_{i=1}^n d_i f_i^2 - \sum_{i,j=1}^n f_i f_j w_{ij} \\ &= \frac{1}{2} \left(\sum_{i=1}^n d_i f_i^2 - 2 \sum_{i,j=1}^n f_i f_j w_{ij} + \sum_{j=1}^n d_j f_j^2 \right) \end{aligned}$$

now since $d_i = \sum_{j=1}^n w_{ij}$ and $d_j = \sum_{i=1}^n w_{ij}$,

$$\begin{aligned}
&= \frac{1}{2} \left(\sum_{i=1}^n \left(\sum_{j=1}^n w_{ij} \right) f_i^2 - 2 \sum_{i,j=1}^n f_i f_j w_{ij} + \sum_{j=1}^n \left(\sum_{i=1}^n w_{ij} \right) f_j^2 \right) \\
&= \frac{1}{2} \left(\sum_{i,j=1}^n w_{ij} f_i^2 - 2 \sum_{i,j=1}^n f_i f_j w_{ij} + \sum_{i,j=1}^n w_{ij} f_j^2 \right) \\
&= \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2
\end{aligned}$$

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2$$

2. L is symmetric because D and W are symmetric.

That is, $L^T = D^T - W^T = D - W = L$

and a matrix M is positive semi-definite iff for any vector $v \in \mathbb{R}^n$, $v^T M v \geq 0$

Since for any $f \in \mathbb{R}^n$,

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2$$

$(f_i - f_j)^2 \geq 0$ and by assumption $w_{ij} \geq 0 \quad \forall i, j$

$$\implies f^T L f \geq 0 \quad \forall f \in \mathbb{R}^n$$

3. Since,

$$\begin{aligned}
L \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} &= D \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - W \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - \begin{bmatrix} w_{11} & & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \dots & w_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} - \begin{bmatrix} \sum_{i=1}^n w_{1i} \\ \sum_{i=1}^n w_{2i} \\ \vdots \\ \sum_{i=1}^n w_{ni} \end{bmatrix} = 0
\end{aligned}$$

Algorithm 1: Unnormalized spectral clustering

Input: Input $X \in \mathbb{R}^{n \times d}$

// where d are the number of input features

Output: Clusters in the data

- 1 Construct a graph and then its weighted adjacency(or affinity) matrix W with any of the methods described in Section 2.1.2
 - 2 Compute the unnormalized laplacian $L := D - W$
 - 3 Compute the first k eigenvectors (2.1.1) u_1, u_2, \dots, u_k of L .
 - 4 Let $H \in \mathbb{R}^{n \times k}$ the eigenvector matrix of L containing u_1, u_2, \dots, u_k as columns.
 - 5 Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of H .
 - 6 Apply k -means clustering algorithm to cluster $(y_i)_{1 \leq i \leq n}$ into k clusters C_1, \dots, C_k
 - 7 A_1, A_2, \dots, A_k are the final clusters with $A_i = \{x_j \mid y_j \in C_i\}$
-

Hence

$$L \cdot \mathbf{1} = 0 \implies 0 \text{ is a eigenvalue of } L \text{ with eigenvector } \mathbf{1}.$$

Lemma(2.1.1) and the above result implies that the smallest eigenvalue of L is 0 and eigenvector corresponding to it is $\mathbf{1}$

$$(3) \implies (4)$$

□

2.1.4 Unnormalized spectral clustering:

Clustering problem can be thought of as graph partitioning problem where aim is to minimize sum of weight of edges going across the clusters and maximize the sum of weights within the cluster.

A cut in a graph is defined as the partitioning of graph into two disjoint subsets. That is, $cut(A, \bar{A}) = W(A, \bar{A})$. This definition can be extended to partitioning graph into k disjoint subsets. That is,

$$cut(A_1, A_2, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k W(A_i, \bar{A}_i). \quad (2.1)$$

Here factor of $1/2$ comes in to compensate for each edge being added twice in the sum. A way to achieve graph partitioning is by minimizing this cut value. This problem is referred to as **mincut problem**. Minimizing this cut value is minimizing the sum of weight going across the clusters. Exact solution to this problem can be found for $k = 2$ and in most cases,

solution just separates one vertex from rest of the graph. Hence, in most use cases, solving mincut just separates out an outlier from rest of the data. A workaround to this problem is to introduce penalty on small sized clusters. This can be done by defining RationCut (Hagen and Kahng 1992) and NCut (Shi and Malik 2000b).

Definition 2.1.3. *RationCut and NCut for a given partition A_1, \dots, A_k is defined as*

$$\begin{aligned} \text{RatioCut}(A_1, \dots, A_k) &:= \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|} \\ \text{NCut}(A_1, \dots, A_k) &:= \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)} \end{aligned}$$

Now the objective is to minimize the *RatioCut* which is a NP-hard problem.

Definition 2.1.4. *Given a partition $\{A_1, \dots, A_k\}$ of V , an ortho matrix H is defined as $[h_1, \dots, h_k]_{n \times k}$ with $h_j := (h_{1j}, \dots, h_{nj})^T \in \mathbb{R}^n$ where*

$$h_{ij} := \begin{cases} \frac{1}{\sqrt{|A_j|}} & \text{if } i \in A_j \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 2.1.3. *An ortho matrix is orthogonal.*

Proof. Let H be an ortho matrix.

$$\langle h_j, h_j \rangle = \sum_{i=1}^n h_{ij}^2 = \sum_{i \in A_j} h_{ij}^2 = \sum_{i \in A_j} \left(\frac{1}{\sqrt{|A_j|}} \right)^2 \quad (\text{since } h_{ij} = 0 \text{ for } i \notin A_j) \quad (2.2)$$

and

$$\langle h_i, h_j \rangle = \sum_{d=1}^n h_{di} h_{dj} = 0 \quad (2.3)$$

since $\{A_1, \dots, A_k\}$ forms a partition of V , it implies if $d \in A_i \implies d \notin A_j \quad \forall j \neq i$

Hence, (2.2) and (2.3) $\implies H^T H = I$ □

Theorem 2.1.4. For a given partition $\{A_1, \dots, A_k\}$ of V , let H be its ortho matrix. Then,

$$RatioCut(A_1, \dots, A_k) = Tr(H^T L H)$$

Proof. From the properties of laplacian L ,

$$h_p^T L h_p = \frac{1}{2} \sum_{i,j} w_{ij} (h_{ip} - h_{jp})^2$$

Now since $w_{ij} = w_{ji}$

$$\implies \left(\sum_{\substack{i \in A_p \\ j \notin A_p}} w_{ij} (h_{ip} - h_{jp})^2 = \sum_{\substack{i \notin A_p \\ j \in A_p}} w_{ij} (h_{ip} - h_{jp})^2 \right)$$

$$\begin{aligned} h_p^T L h_p &= \frac{1}{2} \sum_{\substack{i \in A_p \\ j \in A_p}} w_{ij} (h_{ip} - h_{jp})^2 + \frac{2}{2} \sum_{\substack{i \in A_p \\ j \notin A_p}} w_{ij} (h_{ip} - h_{jp})^2 \\ &\quad + \frac{1}{2} \sum_{\substack{i \notin A_p \\ j \notin A_p}} w_{ij} (h_{ip} - h_{jp})^2 \\ &= \sum_{\substack{i \in A_p \\ j \notin A_p}} w_{ij} \left(\frac{1}{\sqrt{|A_j|}} \right)^2 = \frac{W(A_p, \bar{A}_p)}{|A_p|} \\ &= \frac{1/2 (W(A_p, \bar{A}_p) + W(\bar{A}_p, A_p))}{|A_p|} = \frac{cut(A_p, \bar{A}_p)}{|A_p|} \end{aligned}$$

Also, $h_p^T L h_p = (H^T L H)_{ii}$

$$\begin{aligned} \implies RatioCut(A_1, \dots, A_k) &= \sum_{p=1}^n \frac{cut(A_p, \bar{A}_p)}{|A_p|} \\ &= \sum_{p=1}^n h_p^T L h_p = \sum_{p=1}^n (H^T L H)_{ii} = Tr(H^T L H) \end{aligned}$$

□

This leads to the following proposition.

Proposition 2.1.5.

$$\min_{\{A_1, \dots, A_k\}} \text{RatioCut}(A_1, \dots, A_k) \quad (2.4)$$



$$\min_{\{A_1, \dots, A_k\}} \text{Tr}(H^T L H) \quad (2.5)$$

where H is as in (2.1.4) and hence further satisfies $H^T H = I$.

These optimization problems are referred as **balanced mincut problem** since *RatioCut* balances cluster sizes. Since the problem is still NP-Hard, the Problem (2.5) can be relaxed to

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times k}} \quad & \text{Tr}(H^T L H) \\ \text{s.t.} \quad & H^T H = I \end{aligned} \quad (2.6)$$

which is referred to as **relaxed RatioCut (or balanced mincut) problem**. The solution to this problem can be given using Rayleigh-Ritz theorem.

Definition 2.1.5. *Rayleigh's Quotient* $R_A(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ is defined as

$$R_A(x) = \frac{x^T A x}{x^T x}$$

Theorem 2.1.6 (Rayleigh-Ritz). *If A is a symmetric $n \times n$ matrix with eigenvalues $\lambda_{\min} = \lambda_1 \leq \dots \leq \lambda_n = \lambda_{\max}$ and let (u_1, \dots, u_n) be the eigenvectors of A , where u_i is the eigenvector associated with λ_i , then*

$$\min_{x \neq 0} R_A(x) = \lambda_1 \quad \text{and} \quad \max_{x \neq 0} R_A(x) = \lambda_n$$

where minimum is attained for $x = u_1$ and maximum is attained for $x = u_n$

Proof. Since $A \in \mathbb{R}^{n \times n}$ is symmetric it can be decomposed into $A = Q \Lambda Q^T$ where Q is orthogonal ($Q^T Q = I$) and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$. (p. 136, Corollary 2.5.11 of [Horn and](#)

$$\begin{aligned}
 x^T Ax &= x^T (Q \Lambda Q^T) x &= (Q^T x)^T \Lambda (Q^T x) &= y^T \Lambda y \\
 &= \sum_{i=1}^n \lambda_i |y_i|^2 &\geq \lambda_{\min} \sum_{i=1}^n |y_i|^2 &= \lambda_{\min} y^T y \\
 &\geq \lambda_{\min} (Q^T x)^T (Q^T x) &= \lambda_{\min} x^T Q Q^T x &= \lambda_{\min} x^T x \\
 &\implies R_A(x) \geq \lambda_{\min} \quad \forall x \in \mathbb{R}^n
 \end{aligned} \tag{2.7}$$

but since

$$\begin{aligned}
 Au_1 &= \lambda_{\min} u_1 \implies \frac{u_1^T Au_1}{u_1^T u_1} = \lambda_{\min} \\
 &\implies \min_{x \neq 0} R_A(x) = \lambda_{\min} = \lambda_1
 \end{aligned}$$

for $x = u_1$. Similarly, $\max_{x \neq 0} R_A(x) = \lambda_n$ for $x = u_n$. □

Theorem 2.1.7 (Rayleigh-Ritz extension). *If A is a symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$ and if (u_1, \dots, u_n) is any orthonormal basis of eigenvectors of A , where u_i is a unit eigenvector associated with λ_i , then for $X \in \mathbb{R}^{n \times k}$*

$$\min_{X^T X = I} \text{Tr}(X^T A X) = \lambda_1 + \dots + \lambda_k \tag{2.8}$$

where minimum is attained for $X = [u_1, \dots, u_n]$

Proof. Proof by induction.

Let $X = [x_1, \dots, x_k] \in \mathbb{R}^{n \times k}$ where $x_i \in \mathbb{R}^n$. Since $(X^T A X)_{ii} = x_i^T A x_i$

For $k = 1$:

$$\begin{aligned}
 \implies \min_X X^T A X &= \min_X (X^T A X)_{11} = \min_X x_1^T A x_1 = \lambda_1 \quad \text{for } x_1 = u_1. \\
 &\quad \text{(as } x_1^T x_1 = 1 \text{ since } X \text{ is orthogonal)}
 \end{aligned}$$

Now for $k = 2$:

$$\text{Tr}(X^T A X) = (X^T A X)_{11} + (X^T A X)_{22} = x_1^T A x_1 + x_2^T A x_2$$

which are independent terms as $x_1 \cdot x_2 = 0$.

$\min_{x_1} x_1^T A x_1 = \lambda_1$ for $x_1 = u_1$ and since X is orthogonal, $x_2 \cdot x_1 = 0 \implies x_2 \cdot u_1 = 0$ hence $x_2 \neq u_1$, Thus $\min_{x_2} x_2^T A x_2 = \lambda_2$ for $x_2 = u_2$

For $k = d$, assume that the claim is true. That is, $X = [u_1, \dots, u_d]$ minimizes (2.8) with minimum being $\lambda_1 + \dots + \lambda_d$.

Now for $k = d + 1$,

$$\begin{aligned} \min_X \text{Tr}(X^T A X) &= \min_X \sum_{i=1}^{d+1} (X^T A X)_{ii} = \min_{x_1, \dots, x_{d+1}} \left(\sum_{i=1}^d x_i^T A x_i + x_{d+1}^T A x_{d+1} \right) \\ &= (\lambda_1 + \dots + \lambda_d) + \min_{x_{d+1}} x_{d+1}^T A x_{d+1} \end{aligned}$$

since X has to be orthogonal, $x_{d+1} \cdot x_i = 0 \quad \forall 1 \leq i \leq d$

$$\min_{\substack{x_{d+1} \cdot u_i = 0 \\ \forall 1 \leq i \leq d}} x_{d+1}^T A x_{d+1} = \lambda_{d+1}$$

for $x_{d+1} = u_{d+1}$. These statements directly follows from Rayleigh-Ritz theorem (2.1.6)

$$\min_X \text{Tr}(X^T A X) = \lambda_1 + \dots + \lambda_{d+1}$$

□

This directly gives the solution to Relaxed *RatioCut* problem (2.6). Let formalize this.

Proposition 2.1.8. *The solution to the relaxed RatioCut problem is given by*

$$\begin{aligned} \arg \min_{H \in \mathbb{R}^{n \times k}} \text{Tr}(H^T L H) &= [u_1, \dots, u_k] \\ \text{s.t.} \quad H^T H &= I \end{aligned} \tag{2.9}$$

where $\{u_1, \dots, u_k\}$ are the “first k eigenvectors” (2.1.1) of the laplacian L .

But notice before relaxation, the solution was a discrete cluster assignment matrix. In order to obtain the cluster assignment, one way is to do k-means clustering on the row vectors. Since solution to (2.6) is $H = [u_1, \dots, u_k]$ which can be written as $H = [y_1, \dots, y_n]^T$ where y_i 's are the rows of H . Now let C_1, \dots, C_k be the clusters obtained from the k-means

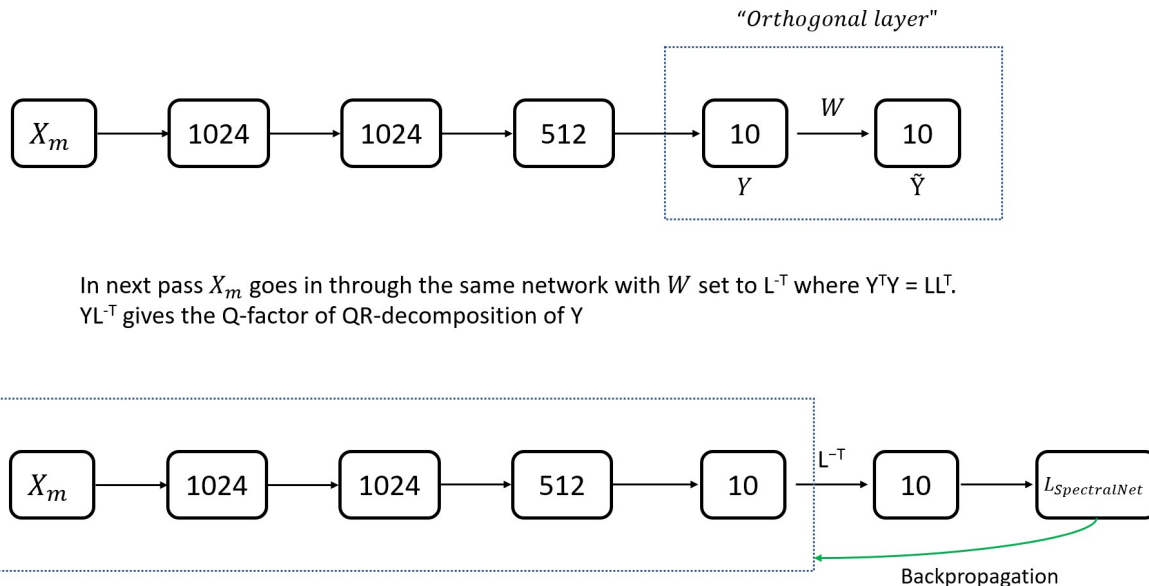


Figure 2.1: SpectralNet Model outline where X_m is the mini-batch of size m

clustering on $\{y_1, \dots, y_k\}$. Then the final clusters can be given by $\{A_1, \dots, A_k\}$ where

$$x_i \in A_j \quad \text{if } y_i \in C_j .$$

Recall, $\{x_1, \dots, x_n\}$ is the input and $\{y_1, \dots, y_n\}$ are the embeddings obtained from the solution to the relaxed *RatioCut* problem. The complete algorithm is summarized in [\(1\)](#).

The solution to the relaxed problem does not guarantee the quality of the clusters and can perform poorly when compared to the exact solution (solution to [2.1.5](#)). One such example is cockroach ladder graph where such comparison can be made. Reader is encouraged to read more on this from [Von Luxburg 2007](#).

2.2 SpectralNet

SpectralNet ([Shaham et al. 2018](#)) is an extension of Spectral clustering to the neural network which overcomes both the limitation of Scalability and inability to do OOSE. SpectralNet implicitly produces orthogonal representations out of the input which can then be used to calculate the spectral loss (objective of the relaxed *RatioCut* problem) [\(2.6\)](#).

It is a feed forward neural network which takes in $\{x_1, \dots, x_n\}$ as input and produces

Algorithm 2: SpectralNet training

Input: $\mathcal{X} \subseteq \mathbb{R}^d$, number of clusters k , batch size m
Output: embeddings $y_1, \dots, y_n, y_i \in \mathbb{R}^k$, cluster assignments $c_1, \dots, c_n, c_i \in \{1, \dots, k\}$

- 1 Construct a training set of positive and negative pairs for the Siamese network;
- 2 Train a Siamese network;
- 3 Randomly initialize the network weights θ ;
- 4 **while** $L_{\text{SpectralNet}}(\theta)$ not converged **do**
- 5 **Orthogonalization step:**
- 6 Sample a random minibatch X of size m ;
- 7 Forward propagate X and compute inputs to orthogonalization layer \tilde{Y} ;
- 8 Compute the Cholesky factorization $LL^T = \tilde{Y}^T\tilde{Y}$;
- 9 Set the weights of the orthogonalization layer to be $\sqrt{m}(L^{-1})^T$;
- 10 **Gradient step:**
- 11 Sample a random minibatch x_1, \dots, x_m ;
- 12 Compute the $m \times m$ affinity matrix W using the Siamese net;
- 13 Forward propagate x_1, \dots, x_m to get y_1, \dots, y_m ;
- 14 Compute the loss (2.1);
- 15 Use the gradient of $L_{\text{SpectralNet}}(\theta)$ to tune all F_θ weights, except those of the output layer;
- 16 Forward propagate x_1, \dots, x_n and obtain F_θ outputs y_1, \dots, y_n ;
- 17 Run k -means on y_1, \dots, y_n to determine cluster centers;

orthogonal representations $\{y_1, \dots, y_n\}$ which serves as the approximation to the eigenvector matrix of the graph laplacian $(D - W)$. Its training requires two forward passes in each iteration. In the first forward pass output of the penultimate layer Y is used to determine L^{-T} factor which is then used to set the weights of the last layer. L is given by $Y^TY = LL^T$ (i.e. Cholesky factorization of Y^TY). This left multiplication of L^{-T} with Y gives an orthogonal output \tilde{Y} (which is actually the Q -factor of the QR -decomposition of Y) (refer [5]). Hence in the second forward pass, \tilde{Y} is calculated and used to minimize the loss (2.1) using mini-batch gradient descent. Finally after training, all the data points are passed to the network to get embeddings $\{y_1, \dots, y_n\}$ over which k -means clustering is performed to obtain clusters. The complete algorithm is summarized in [2].

SpectralNet model has fully connected layers of size 1024, 1024, 512 and 10 with ReLU activation function in between each linear layer. The last layer is termed as "Orthogonal

Layer”. Loss function is defined as

$$L_{SpectralNet}(\theta) = \frac{1}{m^2} Tr(Y^T(D - W)Y) \quad (2.1)$$

where D is the degree matrix and W is the affinity matrix.

SpectralNet uses Siamese network to better learn the relationship between points. Positive pairs and negative pairs are constructed using the nearest neighbours of each point and then the siamese network is trained on the contrastive loss which is defined as

$$L_{siamese}(\theta_{siamese}; x_i, x_j) = \begin{cases} \|z_i - z_j\|^2, & (x_i, x_j) \text{ is a positive pair} \\ \max(c - \|z_i - z_j\|, 0)^2, & (x_i, x_j) \text{ is a negative pair,} \end{cases} \quad (2.2)$$

where c is a margin (hyperparameter).

Then for each pair of points (x_i, x_j) , $\|x_i - x_j\|^2$ in the gaussian kernel (2.1) is replaced by $\|z_i - z_j\|^2$ for calculating laplacian.

2.3 Trivialization

Geodesics are the vectors(or rays) along the shortest path on a manifold.

Definition 2.3.1. *Geodesics $\gamma_{p,v}$ on manifold \mathcal{M} (with complete metric) can be defined as*

$$\begin{aligned} \gamma_{p,v} &: [0, \infty) \longrightarrow \mathcal{M} \\ \gamma_{p,v}(0) &= p, \quad \gamma'_{p,v}(0) = v \quad \text{for } v \in T_p\mathcal{M} \end{aligned}$$

Riemannian exponential map is defined as $\exp_p(tv) := \gamma_{p,v}(t)$ for $t \geq 0$ i.e. mapping from vectors on the tangent space with base at the origin to geodesics on \mathcal{M}

Definition 2.3.2. *For a given real valued function $f : \mathcal{M} \longrightarrow \mathbb{R}$, Riemannian Gradient Descent(RGD) can be defined as*

$$x_{t+1} = \exp_{x_t}(-\eta \nabla f(x_t))$$

$-\nabla f(x_t)$ gives the direction of the steepest descent and is a ray in the tangent space.

x_{t+1} is obtained upon moving along this direction of steepest descent on \mathcal{M} by a step-size of η .

Definition 2.3.3 (Retraction). *A differentiable map $\Gamma : T\mathcal{M} \rightarrow \mathcal{M}$ is called a retraction if for every $p \in \mathcal{M}$, the map $\Gamma_p : T_p\mathcal{M} \rightarrow \mathcal{M}$ satisfies $\Gamma_p(0) = p$ and $(d\Gamma_p)_0 = I$ (identity).*

Since retractions are first order approximations of exponential maps, descent can be made along the retraction, that is, $x_{t+1} = \Gamma_{x_t}(-\eta\nabla f(x_t))$. Retractions are computationally inexpensive as compared to the exponential maps.

Definition 2.3.4. *A map $\phi : \mathbb{R}^n \rightarrow \mathcal{M}$ is a trivialization if it is a diffeomorphism and surjective.*

Such parameterization are required to be diffeomorphism since $\nabla f(x)$ becomes $\nabla(f \circ \phi)(y)$ for $x = \phi(y)$. For a 1-dimensional trivialization, by the chain rule, if $\phi'(y) = 0$ for some $y \in \mathbb{R} \implies \nabla(f \circ \phi)(y) = \nabla f(\phi(y))\phi'(y) = 0$. Hence such parameterizations add saddle points or local minima. Surjectivity is also required. Suppose $\exists x^* \in \mathcal{M}$ such that

$$x^* = \arg \min_{x \in \mathcal{M}} f(x) \tag{2.1}$$

but $\nexists y \in \mathbb{R}^n$ for which $x^* = \phi(y)$. In such case, no matter how one moves in the tangent space, one could never arrive at x^* .

Since a tangent space at a point is isomorphic to \mathbb{R}^n , diffeomorphism and surjectivity ensures that such maps can be used to parameterize a manifold in terms of the euclidean space. Also, it just acts as a change of metric on the manifold.

Theorem 2.3.1. *Let ϕ be a trivialization. Then, solving the problem $\min_{y \in \mathbb{R}^n} f(\phi(y))$ through gradient descent accounts for solving the problem $\min_{x \in \mathcal{M}} f(x)$ using Riemannian gradient descent for a certain metric on \mathcal{M} induced by ϕ .*

Proof. Appendix B of [Casado 2019](#) □

Thus ϕ can be used to translate a constrained optimization problem to an unconstrained optimization problem.

$$\min_{x \in \mathcal{M}} f(x) \xrightarrow{\text{translates to}} \min_{x \in \mathbb{R}^n} f(\phi(x))$$

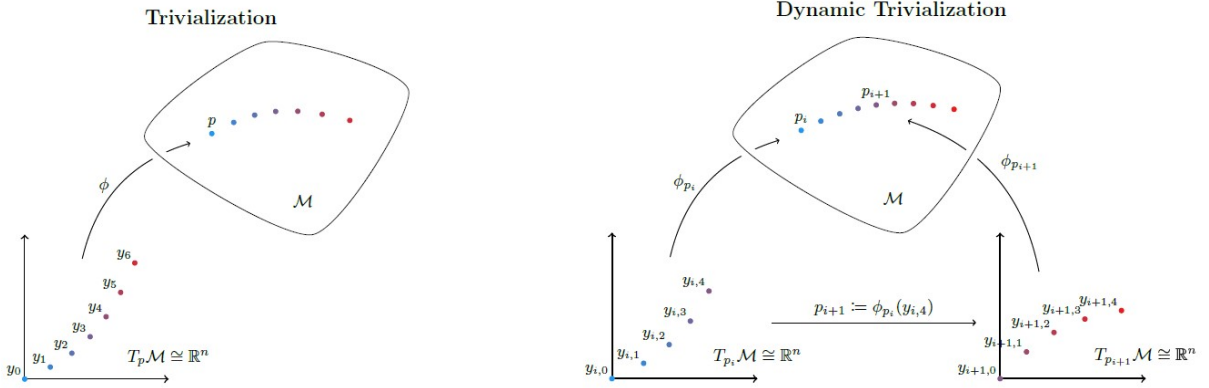


Figure 2.2: Trivialization compared with Dynamic trivialization

Riemannian(or Lie) retractions and exponential maps best serve the purpose since the gradient gets preserved as exponential maps are a diffeomorphism from an open subset of a tangent space of a point on the manifold to the manifold (Theorem 4.4 and Theorem 4.7 [Casado \[2019\]](#)).

[Casado \[2019\]](#) proposes Dynamic trivialization where K number of updates are made using $exp_{\mathcal{M},p}$ at each step and then p is changed to the new point on the manifold.

Thus this does not require the condition of $\phi_p : T_p\mathcal{M} \rightarrow \mathcal{M}$ being surjective. This is because, as long as \mathcal{M} is connected, we can still reach any point in \mathcal{M} in the optimization process by changing the basis of the dynamic trivialization whenever $K < \infty$.

Algorithm 2.3.2 (Dynamic trivialization through retractions). *Given a retraction ϕ , an integer $K > 0$ or $K = \infty$, and a starting point p_0 , the dynamic trivialization induced by ϕ is defined as the sequence of problems indexed by $i = 0, 1, \dots$*

$$\min_{y \in T_{p_i}\mathcal{M}} f(\phi_{p_i}(y))$$

where $p_{i+1} := \phi_{p_i}(y_{i,K}) \in \mathcal{M}$, and $y_{i,k} \in T_{p_i}\mathcal{M}$ for $k = 1, \dots, K$, is a sequence of approximations given by a Euclidean optimization algorithm -e.g., SGD, ADAM, ADAGRAD, RMSPROP, ...-applied to the i -th problem with starting point $y_{i,0} = 0$. We say that p_i is the basis at step i .

There are two limit case to the algorithm which proves that the dynamic trivialization is the generalization of the trivialization and the RGD.

When $K = \infty$, it reduces to trivialization. Since a single map is used to make all the updates.

When $K = 1$, it reduces to RGD. The base is changed at every step. Hence ϕ at base p_i is used to obtain p_{i+1} at each step. Since by definition of retractions, $\nabla(f \circ \phi_{p_i})(0) = \nabla f(p_i)$ and $y_{i,1} = -\eta \nabla(f \circ \phi_{p_i})(0) \implies y_{i,1} = -\eta \nabla f(\phi_{p_i}) \implies p_{i+1} = \phi_{p_i}(-\eta \nabla f(\phi_{p_i}))$ which are exactly the updates we make in RGD.

For a Lie group G , lie algebra is the tangent space at the identity, i.e. $\mathfrak{g} = T_e G$ where e is the identity element of G . For a matrix lie group G , any tangent space of G can be written in terms of \mathfrak{g} . That is, if $A \in T_B G$ then $B^{-1}A \in \mathfrak{g}$ and if the metric on the lie group is left-invariant, $B \exp(B^{-1}A)$ lies in the neighbourhood of B . Exponential map thus can be defined as

$$\begin{aligned} \exp_B : T_B G &\longrightarrow G \\ A &\longmapsto B \exp(B^{-1}A) \end{aligned}$$

For connected matrix lie group such as $SO(n)$, $U(n)$, $SL(n)$, or $GL^+(n)$, above map can be used to do dynamic trivialization.

Cayley map is a retraction from tangent space $skew(n)$ of lie group $SO(n)$ to $SO(n)$ and is surjective and is defined as

$$\begin{aligned} cay : skew(n) &\longrightarrow SO(n) \\ A &\longmapsto (I - A)^{-1}(I + A) \end{aligned}$$

for $SO(n) = \{B \in \mathbb{R}^{n \times n} \mid B^T B = I, \det(B) = 1\}$ and $skew(n) \doteq \{A \in \mathbb{R}^{n \times n} \mid A^T + A = 0\}$. Again by similar trick $cay_B(A) = B cay(B^{-1}A)$, for $B \in SO(n)$, $A \in T_B SO(n) = skew(n)$.

Figure [2.3](#) shows difference between dynamic trivialization and RGD where W_t the starting value and a_i are the updates made using SGD. In case of just trivialization, a tangent space around a single point on the manifold is used to parameterize the complete manifold.

Trivialization technique has been used to tackle the vanishing gradient problem of a deep Recurrent Neural Network (RNN) where the weight matrix of the hidden state is constrained to stay on a manifold (particularly on $SO(n)$) through cayley map or matrix exponential.

Our goal had been to leverage trivialization to translate the constrained optimization

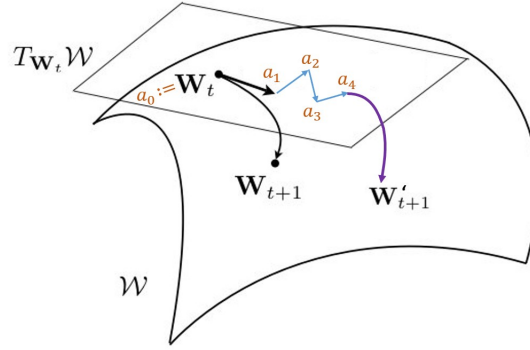


Figure 2.3: Difference between dynamic trivialization(for $K=4$) and RGD on \mathcal{W}

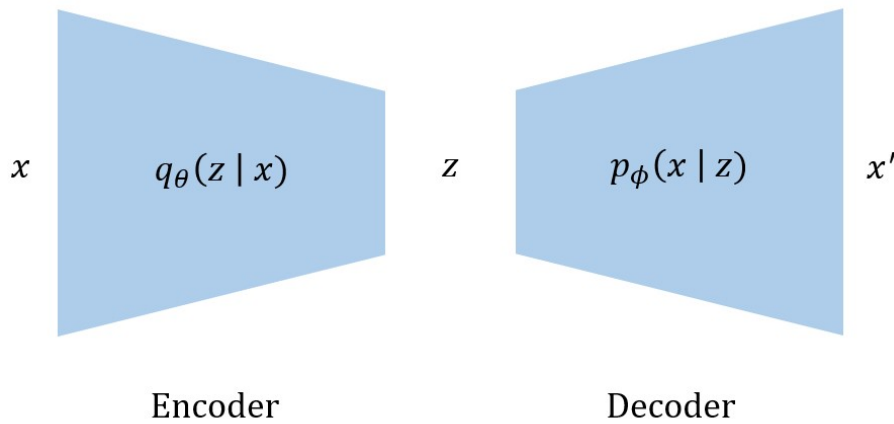


Figure 2.4: Basic outline of VAE architecture.

problem to unconstrained one and use stochastic gradient descent.

2.4 Variational autoencoders

The main goal in Variational Inference is to learn the probability distribution of the given input x . Variational autoencoder (Kingma and Welling [2014]) learns the parameters of the approximate posterior and the likelihood through encoder and decoder system. Basic outline of a VAE can be seen in the Figure [2.4].

2.4.1 Latent variable model

Latent variable models are statistical model which studies the relationship between a observable and a latent variable. Probability distribution of the input(an observable variable)

can be written in terms of likelihood and prior as

$$p(x) = \int p(x | z)p(z)dz$$

where $p(x | z)$ is the likelihood and $p(z)$ is the prior. In some cases, relatively simple expression of $p(x | z)$ and $p(z)$ can give complicated $p(x)$. For example in Mixture of k gaussians, z is a discrete variable and likelihood follows normal distribution, that is

$$\begin{aligned} p(z = i) &= \pi_i \\ p(x | z = i) &= \mathcal{N}(\mu_i, \sigma_i^2) \\ \implies p(x) &= \sum_{i=1}^k p(x, z = i) = \sum_{i=1}^k \pi_i p(x | z = i) \end{aligned}$$

Hence, simple individual gaussian distribution can model complex distributions.

2.4.2 Non-linear Latent variable model

Here, latent variable $z \sim \mathcal{N}_z(0, I)$ and variance of prior is constant. That is,

$$\begin{aligned} p(z) &= \mathcal{N}_z(0, I) \\ p(x | z; \phi) &= \mathcal{N}_x(f(z, \phi) | \sigma^2 * I) \end{aligned}$$

$f(z, \phi)$ is a family of deterministic function parameterized by ϕ . We use this model mainly to compute the posterior, sample data and calculate the likelihood.

Using Bayes rule, posterior can be written as

$$p(z | x) = \frac{p(x | z)p(z)}{p(x)}$$

But since the $p(x)$ is clearly intractable, posterior cannot be calculated. This shows the need to approximate the posterior using simpler probability distributions.

Sample $z \sim \mathcal{N}_z(0, I)$, compute $f(z, \phi)$, then sample $x^* \sim \mathcal{N}(f(z, \phi) | \sigma^2 * I)$

Likelihood can be determined by

$$p(z) = \int p(x, z | \phi) dz = \int \mathcal{N}_x(f(z, \phi) | \sigma^2 * I) \mathcal{N}_z(0, I)$$

which is intractable.

As it is clear that determining true posterior is intractable in most situation, Variational Inference is thus used to approximate the true posterior with $q_z(z | x)$ (called approximate posterior) which is tractable and is close to the true posterior. Variational Inference determines approximate posterior through solving a optimization problem while methods like MCMC performs selection through sampling.

2.4.3 Evidence Lower Bound

Let $q_\theta(z | x_i)$ be the approximate posterior which is to be learnt through an encoder and $p_\phi(x_i | z)$ be the likelihood which is to be learnt through a decoder with parameters ϕ . The KL divergence between the approximate and the real posterior distributions is given by,

$$D_{KL}(q_\theta(z | x_i) || p(z | x_i)) = - \int q_\theta(z | x_i) \log \left(\frac{p(z | x_i)}{q_\theta(z | x_i)} \right) dz \geq 0 \quad (2.1)$$

By Bayes theorem,

$$D_{KL}(q_\theta(z | x_i) || p(z | x_i)) = - \int q_\theta(z | x_i) \log \left(\frac{p_\phi(x_i | z) p(z)}{q_\theta(z | x_i) p(x_i)} \right) dz \geq 0 \quad (2.2)$$

$$D_{KL}(q_\theta(z | x_i) || p(z | x_i)) = - \int q_\theta(z | x_i) \left[\log \left(\frac{p_\phi(x_i | z) p(z)}{q_\theta(z | x_i)} \right) - \log p(x_i) \right] dz \geq 0 \quad (2.3)$$

$$= - \int q_\theta(z | x_i) \log \left(\frac{p_\phi(x_i | z) p(z)}{q_\theta(z | x_i)} \right) dz + \int q_\theta(z | x_i) \log p(x_i) dz \geq 0 \quad (2.4)$$

Since $\log(p(x_i))$ independent of q and is a constant,

$$= - \int q_\theta(z | x_i) \log \left(\frac{p_\phi(x_i | z) p(z)}{q_\theta(z | x_i)} \right) dz + \log p(x_i) \int q_\theta(z | x_i) dz \geq 0 \quad (2.5)$$

$$= - \int q_\theta(z | x_i) \log \left(\frac{p_\phi(x_i | z) p(z)}{q_\theta(z | x_i)} \right) dz + \log p(x_i) \geq 0. \quad (2.6)$$

Finally upon rearranging the terms, we get,

$$\log p(x_i) \geq \int q_\theta(z | x_i) \log \left(\frac{p_\phi(x_i | z) p(z)}{q_\theta(z | x_i)} \right) dz. \quad (2.7)$$

$$\log p(x_i) \geq \int q_\theta(z | x_i) \log \left(\frac{p(z)}{q_\theta(z | x_i)} \right) dz + \int q_\theta(z | x_i) \log p_\phi(x_i | z) dz \quad (2.8)$$

$$\log p(x_i) \geq -D_{KL}(q_\theta(z | x_i) \| p(z)) + E_{\sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] \quad (2.9)$$

where the term on the right is called Evidence Lower Bound (ELBO). From (2.6) and (2.9)

$$D_{KL}(q_\theta(z | x_i) \| p(z | x_i)) = - \overbrace{\left(-D_{KL}(q_\theta(z | x_i) \| p(z)) + E_{\sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] \right)}^{ELBO} + \log p(x_i) \quad (2.10)$$

Since the log probability is constant, minimizing KL divergence between approximate posterior and true posterior is equivalent to maximizing the ELBO.

The first term in ELBO is the KL divergence between the approximate posterior and the prior. The second term is the reconstruction term.

Normal prior

We can make assumption about the distribution of the true posterior. If the approximate posterior $q_\theta(z | x) \sim \mathcal{N}_x(\mu | \sigma^2 * I)$ and the prior $p(z) \sim \mathcal{N}_z(0, I)$ then we have the following closed form expression for the ELBO:

$$-D_{KL}(q_\theta(z | x) \| p(z)) = \frac{1}{L} \sum_{i=1}^L \left(\sum_{j=1}^J \left(\frac{1}{2} [1 + \log(\sigma_{ij}^2) - \sigma_{ij}^2 - \mu_{ij}^2] \right) \right) \quad (2.11)$$

$$E_{\sim q_\theta(z|x)} [\log p_\phi(x | z)] = \frac{1}{L} \sum_{l=1}^L \|x_l - x'_l\|_2^2 \quad (2.12)$$

where $\|\cdot\|$ is the L_2 norm, L is the number of points, J is the number of latent features and $x \in \mathbb{R}^{L \times S}$ (S is the number of input features). For complete derivation of closed form expression of ELBO one can refer (Kingma and Welling [2014]) and (Odaibo [2019]).

Spike and slab prior

The Spike and Slab distribution $\mathcal{S}^2(\alpha, \mu, \sigma)$ is defined over two variables; a binary spike variable s and a continuous slab variable z .

The spike $s \sim \text{Ber}(\alpha)$ is a Bernoulli random variable, the slab $z \sim \mathcal{N}(\mu, \sigma^2)$ is normal random variable when $s = 1$, and $z \sim \delta(z)$ degenerates at $z = 0$ when $s = 0$:

$$\begin{aligned} p(s, z) &= p(s) p(z|s) \\ &= \alpha \delta(s) \mathcal{N}(z|\mu, \sigma) + (1 - \alpha) \delta(s - 1) \delta(z). \end{aligned} \tag{2.13}$$

where $\delta(\cdot)$ is Dirac delta function centered at zero and J is the latent space dimension.

The prior for the multivariate random variable (latent variable in variational setting) can be written as

$$p_s(z) = \prod_{j=1}^J (\alpha \mathcal{N}(z_j; 0, 1) + (1 - \alpha) \delta(z_j))$$

The recognition function $q_\phi(z | x)$ can be chosen to be of form

$$q_\theta(z | x_i) = \prod_{j=1}^J (\gamma_{i,j} \mathcal{N}(z_{i,j}; \mu_{z,i,j}, \sigma_{z,i,j}^2) + (1 - \gamma_{i,j}) \delta(z_{i,j}))$$

where $\mu_{z,i,j}$, $\sigma_{z,i,j}^2$ and $\gamma_{i,j}$ are the outputs of the encoder. The derivation of the ELBO for spike and slab prior and spike and slab approximate posterior can be found in the Appendix B of [Tonolini et al. \[2019\]](#).

2.4.4 Reparameterization trick

Since we learn the parameters of the approximate posterior through neural network, it requires sampling of the latent variable before feeding it to the decoder. This sampling process cannot be backpropagated and hence require a workaround which is achieved by reparameterization trick.

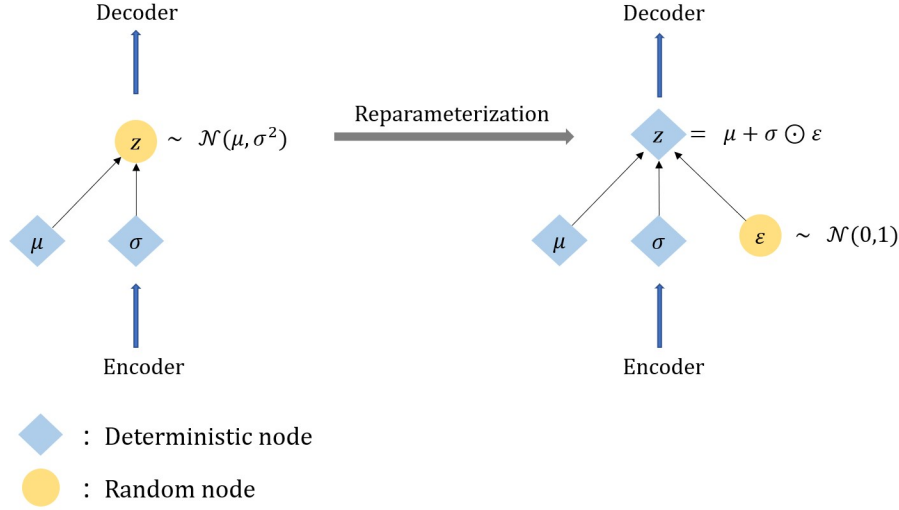


Figure 2.5: Reparameterization trick for gaussian prior and gaussian posterior based VAE model

Gaussian prior

For VAE model with standard normal prior and gaussian posterior is illustrated in the Figure(2.5). Since the encoder gives the parameters (μ, σ) of the approximate posterior, it still requires one to sample z from $\mathcal{N}(\mu, \sigma)$, which makes it a sampling process. Such operation cannot be back propagated. In order to make such sampling deterministic, we add additional variable ϵ which is sampled from $\mathcal{N}(0, 1)$ and the following parameterization is used to obtain the latent variable z

$$z = \mu + \sigma \odot \epsilon$$

which allows one to determine $\frac{\partial L}{\partial z}$ which eventually through chain rule gives $\frac{\partial L}{\partial \theta}$ for encoder parameter update. \odot is the element wise product. Note that it is not necessary that the reparameterization trick can be derived for any kind of probability distribution.

Spike and slab prior

For a model with spike and slab as probability distribution of the prior and the approximate posterior, the latent variable z_i is drawn using the following reparameterization trick

$$z_i = T(\eta_i - 1 + \gamma_i) \odot (\mu_i + \sigma_i \odot \epsilon_i) \quad (2.1)$$

where \odot is the element wise product. Ideally $T(y_i)$ is a step function centered at zero but in order to make the operation differentiable it is defined in terms of scaled Sigmoid as $T(y) = S(cy)$ for some constant c . S is the sigmoid function. As $c \rightarrow 0$, $S(cy)$ tends towards a true binary map. In practice, c has to be a relatively small number for stable gradient ascent. In experiment, best strategy is to gradually increase value of c with training. Also, η_i is sampled from a uniform distribution.

Spike variable is also parameterized in the similar manner. Since spike variable s_i is a binary variable, we can write $s_i = T(\eta_l - 1 + \gamma_i)$ where T is ideally a step function but can be approximated in terms of scaled sigmoid as discussed above and η_l is a noise variable sampled from a uniform distribution which is not dependent of γ_i . For a detailed discussion, refer [Tonolini et al. \[2019\]](#).

Chapter 3

Method

The chapter goes over the problem statement and each of the techniques that were constructed to solve the problem. This also includes initial tests which had their own limitations and were improved with changes in the algorithm. Each of the methods apart from Geometric approach were formulated by us. This majorly includes the use of VAEs to perform clustering.

Our model consists of β -VAE(Variational Autoencoder) (Higgins et al. [2017]) backbone together with a cayley layer which produces orthogonal representations of input. These orthogonal representations then can be used to do spectral clustering. The bottleneck layer of the β -VAE maps input x to latent variable z which has a prior probability distribution $p_\theta(z)$. In our model we chose spike and slab probability distribution (2.13) as the prior and the approximate posterior.

3.1 Problem Statement:

The problem is a constrained optimization problem where the loss function to be minimized is

$$\min_{Y \in \mathbb{R}^{n \times k}} \text{Tr}(Y^T L Y) \text{ subject to } Y^T Y = I_n. \quad (3.1)$$

This is called the spectral loss. As one has seen in the Spectral Clustering section (2.1) that how one arrive at trace minimization problem from minimizing RatioCut, minimization of this real valued function (loss function) ensures that the learned representations are in fact

Algorithm 3: Riemannian gradient descent for spectral clustering

Input: Laplacian L
Output: Clusters in the data

- 1 Initialize Y_0 to be an orthogonal matrix // Any initialization method can be used
- 2 **while** $L_{spectral}$ not converged **do**
- 3 Calculate $G_m = \left[\frac{\partial L}{\partial Y_m[i,j]} \right]$ // G_m is the standard gradient matrix
- 4 Calculate $\Omega(Y_i, G_i) = G_i Y_i^T - Y_i G_i^T$ // Riemannian gradient Ω
- 5 $Y_{i+1} = \Gamma(\eta \Omega(Y_i, G_i)) Y_i$
 // Update step where Γ is the exponential map or a retraction; η is the step size
- 6 $L_{spectral} = tr(Y_{i+1}^T L Y_{i+1})$

/* Let $Y_{optimal}$ be the matrix returned upon loss convergence */

- 7 Do k-means clustering on rows of the matrix $Y_{optimal}$ to obtain clusters

close to the eigenvector matrix (actual solution to the relaxed version of balanced-mincut problem) in equation (2.6). This is because of the fact that the eigenvector matrix forms the solution to this trace minimization problem.

The constraint of matrix being orthogonal is implicitly being satisfied as the bottleneck layer of our model ExSC produces orthogonal output over which the loss function can be minimized. The exact details will be discussed in the section (3.5).

3.2 First method: Geometric Approach

The aim is to minimize the spectral loss while preserving the given orthogonality constraint. Since the collection of rectangular orthogonal matrices forms a well studied manifold called Stiefel manifold, one can use Riemannian Gradient Descent(RGD)(2.3) to move on this manifold while minimizing the given real valued function (i.e. spectral loss). Since this would be an iterative method and hence one would still require a work-around to solve OOSE(as we are not learning any parameters which could make predictions on the unseen data).

Algorithm(3) minimizes spectral loss over Stiefel manifold. Recall, Stiefel manifold is a space of all rectangular real orthogonal matrices. An exponential map or a retraction (Γ)

can be used to map riemannian gradient (Ω) on to the stiefel manifold. Since exponential maps for stiefel manifold does not have a closed form expression, one could use cayley transformation (a retraction) in the Algorithm(3)

3.3 Initial approaches using neural network:

Several methods were tested which employs neural network. Neural networks were mainly used for two reasons:

- **Allows mini-batch gradient descent.** This enables one to work with large datasets and overcomes the scalability limitation of traditional clustering algorithms.
- **Learns parameters.** These parameters can be used to make predictions on the unseen data and hence it allows OOSE.

3.3.1 VAE with iterative schemes

The Variational Autoencoder(VAE)(Kingma and Welling (2014))(2.4) network with the gaussian prior produces near orthogonal representation in the bottle-neck layer. We will discuss in a while that why the choice of a gaussian prior produces these near orthogonal matrices (3.4). These near orthogonal representations can be fine tuned so as to resemble the eigenvector matrix. This fine-tuning can be done using the spectral loss and minimizing the spectral loss over these orthogonal matrices(each mini-batch produces one) can be done in several ways. As shown in the Figure(3.1), trivialization (2.3) and Riemannian Gradient Descent (RGD) can be used to solve the trace minimization problem (minimizing the spectral loss). Since there is no clear way to back-propagate gradients to the parameters of the VAE, such optimization techniques are still an iterative scheme which does not learn any parameters during training. This does not allow one to do OOSE. Hence, RGD cannot directly be used. This led to the next way of solving the problem by incorporating the spectral loss in the loss function of the VAE itself.

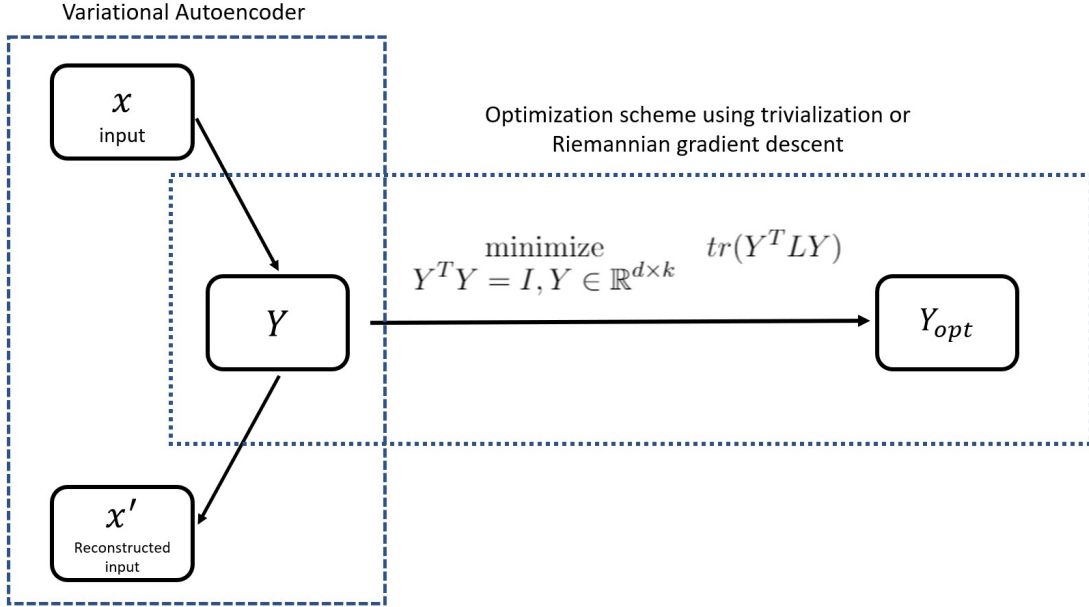


Figure 3.1: Basic outline of initial attempts involving orthogonal representations of input and RGD. $Y_{opt} := Y$ obtained upon convergence of spectral loss.

Loss function of a VAE is the Evidence lower bound (ELBO) (2.4.3) which is defined as

$$\begin{aligned}
 L_{VAE} &= D_{KL}(q_{\phi}(z|x)|p_{\theta}(z|x)) \\
 &= \frac{1}{m} \sum_{l=1}^m \|x_l - x'_l\|_2^2 + D_{KL}(q_{\phi}(z|x)|p_{\theta}(z))
 \end{aligned}$$

where $D_{KL}(q_{\phi}(z|x)|p_{\theta}(z))$ varies with the prior $p_{\theta}(z)$ and m is the mini-batch size. In general, prior is selected to be a standard normal distribution and such VAE is called VAE with gaussian prior. Generally the prior is selected to be a simple and tractable probability distribution.

3.3.2 VAE with spectral loss

This method involved using simple VAE with gaussian prior. The bottleneck layer of this model produces orthogonal representations which can directly be used to minimize the spectral loss. The gradient of this loss function with respect to each of the parameters in the model can then be used to update the parameters. Training can be done until convergence

in overall loss where overall loss L_{model} is

$$L_{model} = D_{KL}(q_{\phi}(z|x)|p_{\theta}(z|x)) + tr(\mu^T L \mu)$$

where first term is the VAE loss (ELBO) and second term is the spectral loss. L is the laplacian of the input.

This type of training can produce orthogonal representations which forms solution to the problem statement (3.1) and also minimizes the VAE loss. This method still had limitation. The limitation being restrictive search space. Since the same matrix was used to minimize the VAE loss and spectral loss, it was producing orthogonal matrices from a subspace of stiefel manifold. In order to further allow orthogonal matrices to come from a much larger space, cayley transformation (3.1) was used to parametrize the weight matrix of the cayleyNN layer. A switch from the VAE to the β -VAE is also made due to the additional parameter β in β -VAE which allows one to put more emphasis on learning the probability distribution of the approximate posterior, $q_{\theta}(z|x)$. In β -VAE loss function changes to

$$L_{VAE} = \frac{1}{m} \sum_{l=1}^m \|x_i - x'_i\|_2^2 + \beta D_{KL}(q_{\phi}(z|x)|p_{\theta}(z)) \quad (3.1)$$

where x, x' are input and reconstructed input respectively and D_{KL} is KL Divergence between two probability distributions.

The inclusion of β parameter allows one to better learn the parameters of the latent space probability distribution as the loss function puts more weight on minimizing the KL-Divergence (D_{KL}). Hence, β -VAE produces better orthogonal matrices at the bottleneck.

3.3.3 VAE with parametrization (Gaussian prior)

The model is as shown in the Figure (3.2). The model consists of the β -VAE (true posterior is assumed to be following gaussian distribution and hence prior is taken to be standard normal distribution) in conjugation with a cayley layer (3.1).

This method of fine tuning through cayley transformation can be generalized to:

$$y = Mx$$

where M is parameterized to stay orthogonal ($M^T M = I$). Now since x is already orthogonal, y is also orthogonal. This is because $(Mx)^T Mx = x^T (M^T M)x = x^T Ix = x^T x = I$. Various methods (like use of exponential maps) from trivialization (Casado 2019) can be used to restrict M to stay on $O(n)$ manifold (orthogonal group).

Loss function of the model is:

$$L_{model} = D_{KL}(q_\phi(z|x)|p_\theta(z|x)) + tr(\mu_{new}^T L \mu_{new}) \quad (3.1)$$

$$= \frac{1}{m} \sum_{l=1}^m \|x_i - x'_i\|_2^2 + D_{KL}(\mathcal{N}(\mu, \sigma^2) || \mathcal{N}(\mathbf{0}, I)) + tr(\mu_{new}^T L \mu_{new}) \quad (3.2)$$

$$= \frac{1}{m} \sum_{l=1}^m \|x_i - x'_i\|_2^2 - \frac{1}{m} \sum_{i=1}^m \left[\sum_{j=1}^k \frac{1}{2} [1 + \log(\sigma_{ij}^2) - \sigma_{ij}^2 - \mu_{ij}^2] \right] + tr(\mu_{new}^T L \mu_{new}) \quad (3.3)$$

where μ and σ^2 are the mean and variance of the approximate posterior q and k is the number of features of the latent variable z . Recall that the true posterior is assumed to following gaussian distribution and thus $q \sim \mathcal{N}(\mu, \sigma^2)$. The second terms changes for different prior $p_\theta(z)$ and has a closed form expression for tractable probability distributions (such as gaussian distribution itself). μ_{new} is as defined in Equation (3.4) The final model ExSC uses spike and slab distribution (Tonolini et al. 2019) as the prior and the approximate posterior in the β -VAE and normalized output of cayley layer is then used to do k-means clustering upon convergence of overall loss.

3.4 Why Gaussian (Normal) prior

When the prior is chosen to be standard normal distribution, the minimization of the VAE loss function leads to the minimization of the difference between the true posterior $p_\theta(z|x)$ and the approximate posterior $q_\phi(z|x)$. This leads to $q_\phi(z|x)$ follow a gaussian distribution. Now if all the entries of a matrix are sampled from gaussian distribution, it can be made near orthogonal matrix using normalization operation which is differentiable operation.

Let X be a random variable being sampled from a standard normal distribution. i.e. $X \sim \mathcal{N}(0, 1)$, then $\mathbb{E}[X] = 0$ and $\mathbb{E}[X^2] = 1$

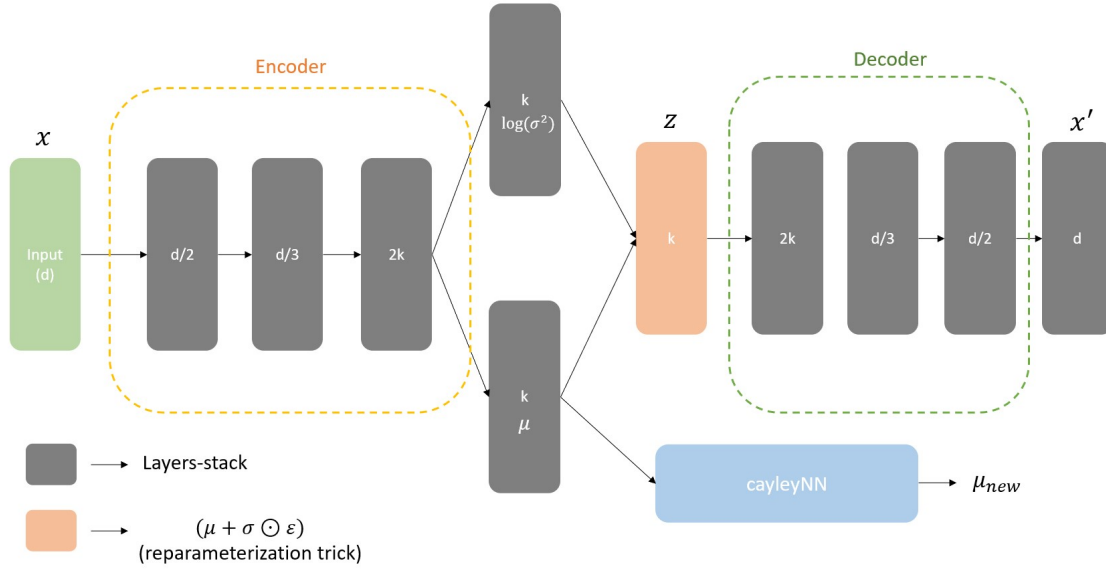


Figure 3.2: Model architecture as in the Section [3.3.3](#). d is the number of input features and k are the number of clusters in the dataset.

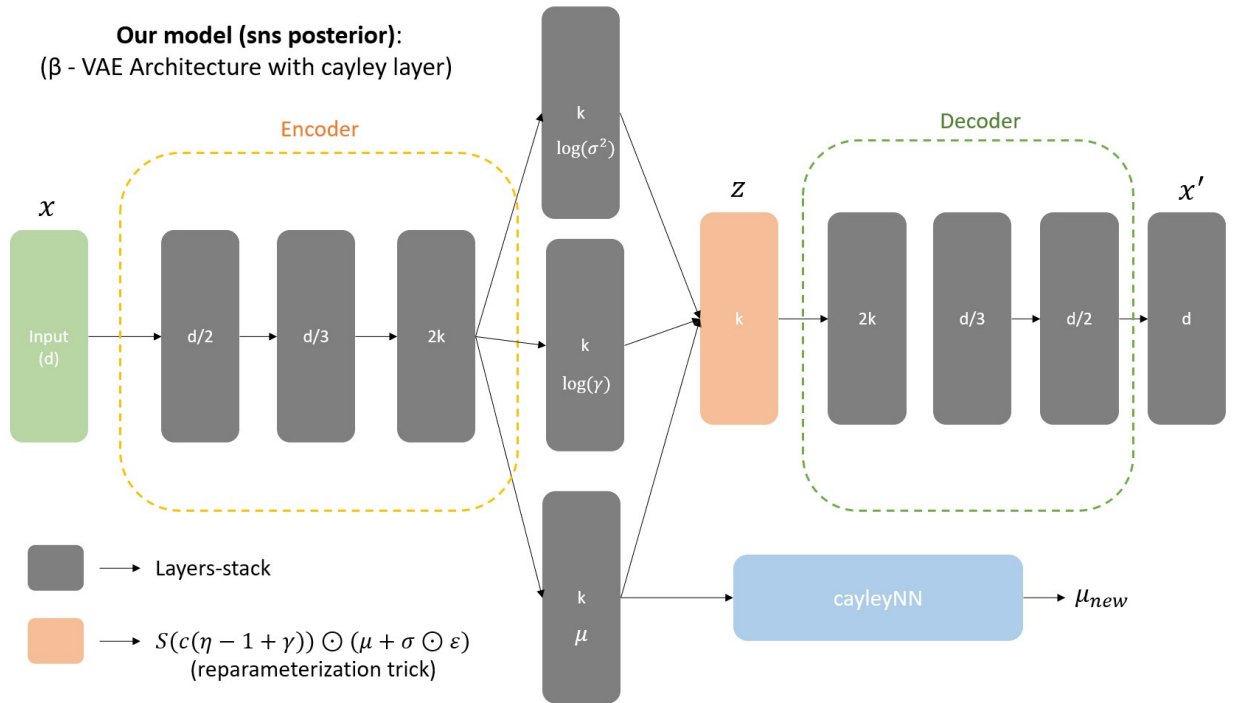


Figure 3.3: ExSC architecture as in Section [3.5](#) (which consists of a β -VAE model with Spike and slab (sns) prior in conjugation with cayley layer). d is the number of input features and k are the number of clusters in the dataset.

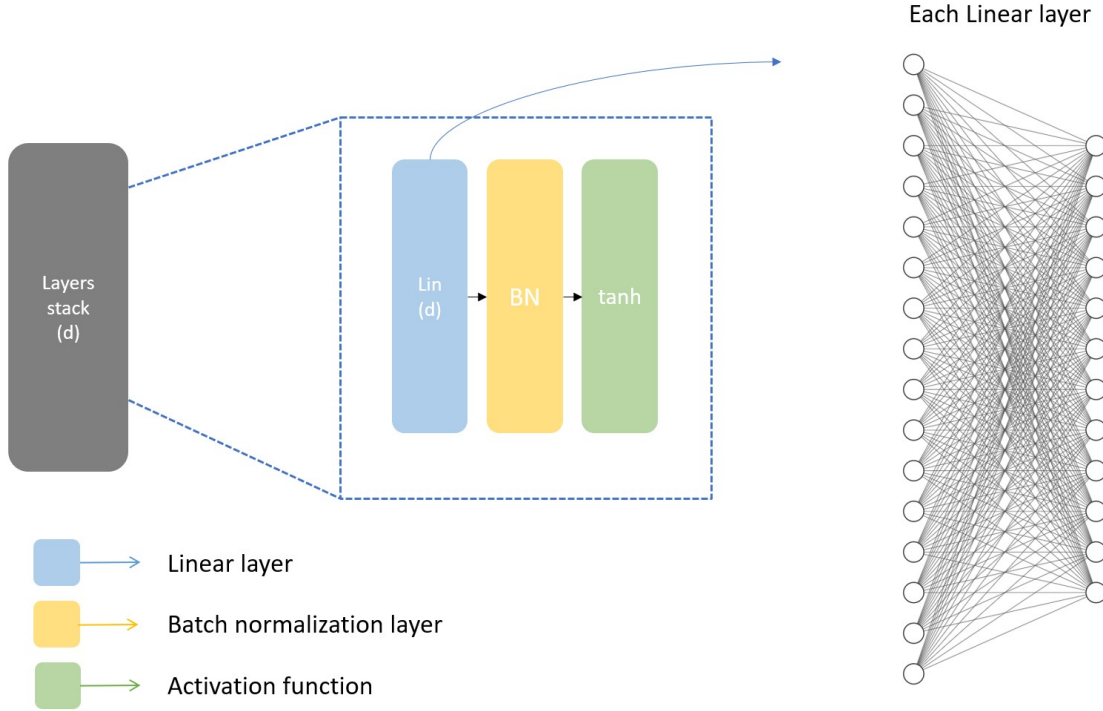


Figure 3.4: Each layers-stack in the Figure(3.3) and Figure(3.2). This assembly of layers forms the basic component of our model

For $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$\mathbb{E}[X] = \mu \text{ and } \mathbb{E}[X^2] = \mu^2 + \sigma^2 . \quad (3.1)$$

Either use the relation $Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ or the pdf of a normal distribution to derive the second moment above.

Now for any matrix $B = \{b_{ij}\} \in \mathbb{R}^{n \times k}$, let $M = B^T B \in \mathbb{R}^{k \times k}$. Since,

$$m_{ii} = \sum_{j=1}^n b_{ij}^2$$

and if each $b_{ij} \sim \mathcal{N}(\mu, \sigma^2)$, from equation (3.1)

$$\implies \mathbb{E} \left[\sum_{i=1}^n b_{ij}^2 \right] = n(\mu^2 + \sigma^2) \quad (3.2)$$

$$\implies m_{ii} \approx n(\mu^2 + \sigma^2) \implies m'_{ii} \approx 1 \quad (3.3)$$

for $M' = B'^T B'$ where $B' = \text{normalize}(B)$

Definition 3.4.1. *In the above equation, the normalize transformation can be defined as*

- $\phi(B) := \text{normalize}(B) := \left\{ \frac{b_{ij}}{\sqrt{\sum_{i=1}^n b_{ij}^2}} \right\}_{\substack{1 \leq i \leq n \\ 1 \leq j \leq k}}$
- $\phi_{\mu, \sigma}(B) := \text{normalize}_{\mu, \sigma}(B) := B / (\sqrt{n(\mu^2 + \sigma^2)})$

Second normalize transformation can be regarded as an approximation to the first one because of the equation (3.2).

Each entry of M is of form,

$$m_{ij} = \sum_{k=1}^n b_{ki} b_{kj} . \quad (3.4)$$

Since for i.i.ds $X, Y \sim \mathcal{N}(0, 1)$,

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y] = 0$$

and for i.i.d $X, Y \sim \mathcal{N}(\mu, \sigma^2)$,

$$n\mathbb{E}[XY] = n\mathbb{E}[X]\mathbb{E}[Y] = n\mu^2$$

where n is a constant.

Hence, if each b_{pq} in equation (3.4) is sampled independently from a normal distribution, then

$$\sum_{k=1}^n b_{ki} b_{kj} \approx n\mu^2 \quad \forall i \neq j$$

From equation (3.1), equation (3.3) and the definition of the normalization transformation above,

$$\implies m'_{ij} \approx \frac{n\mu^2}{n(\mu^2 + \sigma^2)} \approx 0 \quad \text{for either } \mu = 0 \text{ or } \mu < \sigma \text{ (sufficiently smaller)} \quad (3.5)$$

Hence the normalize transformation over matrix that is being sampled from a normal distribution with mean μ and variance σ^2 is implicitly satisfying the orthogonality constraint. In addition, the (3.5) proves the improvement in orthogonal representations with choosing β -VAE over VAE. Since the β hyperparameter in (3.1) leads to further minimization of KL divergence between the approximate posterior $q_\phi(z|x)$ (which gives the bottleneck output)

and the prior $p_\theta(z)$ (which is assumed to standard normal), this leads to $\mu_q \rightarrow 0$ and $\sigma_q \rightarrow 1$ for each sample where (μ_q, σ_q) are the output of the encoder(parameters of the approximate posterior) q . This sets $m'_{ij} \rightarrow 0$ for $i \neq j$ with training.

In summary, this gives a matrix B for which $Y(= \phi_{\mu, \sigma}(B))$ is near orthogonal as $(Y^T Y)_{ii} \rightarrow 1$ and $(Y^T Y)_{ij} \rightarrow 0$ for $i \neq j$. This leads to the following proposition.

Proposition 3.4.1. *The relaxed RatioCut problem (2.6) is equivalent to*

$$\begin{aligned} & \text{minimize} && \text{Tr}(Y^T LY) \\ & Y = \phi_{\mu, \sigma}(B), B = \{b_{ij}\} \in \mathbb{R}^{n \times k} && \\ & \text{s.t } b_{ij} \sim \mathcal{N}(\mu, \sigma^2) && \end{aligned} \tag{3.6}$$

Hence minimizing the trace value (spectral loss) over a large matrix which is obtained after applying the normalization transformation to a matrix that is being sampled from a gaussian posterior is same as finding solutions to the relaxed *RatioCut* (or balanced-mincut) problem.

3.5 Extended Spectral Clustering

Extended Spectral Clustering (ExSC) model broadly consists of an **encoder**, **bottleneck layers**, a **decoder** and a **cayley layer** as can be seen in the Figure(3.3) . The encoder consists of 3 layers-stack. Each layers-stack consist of linear (also called dense) layer followed by a batch-normalization layer and a tanh activation function as in the Figure(3.4). Each linear layer is a feed forward fully connected layer(as on the right in the Figure(3.4)). A linear layer of size (d) gives an output of shape (m,d) where m is the batch size. The components in the bottle-neck layers changes with changes in the prior. The model with spike-and-slab prior has additional parameter spike variable which is obtained by adding additional layers in the bottleneck. These parameters which are sampled using a neural network are used to sample points from the approximate posterior distribution using the reparametrization trick as stated in the section(2.4.4)

Cayley layer(CayleyNN) is defined as:

$$y = \text{cay}(A)x = (I - A)^{-1}(I + A)x \tag{3.1}$$

where A is initialized to be a skew-symmetric matrix using [5]. x is the input and y is the output. A is the parameter of this layer which allows fine tuning of input orthogonal matrix to new orthogonal matrix which minimizes the spectral loss. $(I - A)^{-1}(I + A)$ is called cayley transformation.

The loss function of the model is defined as:

$$L_{model} = \overbrace{D_{KL}(q_\phi(z|x)|p_\theta(z|x))}^{ELBO} + \overbrace{tr(\mu_{new}^T L \mu_{new})}^{\text{Spectral loss}} \quad (3.2)$$

$$= \frac{1}{m} \sum_{l=1}^m \|x_i - x'_i\|_2^2 + \beta D_{KL}(q_\phi(z|x)|p_\theta(z)) + tr(\mu_{new}^T L \mu_{new}) \quad (3.3)$$

where

$$D_{KL}(q_\phi(z|x)|p_\theta(z)) = \sum_{i=1}^m \left[\sum_{j=1}^k \left[\frac{\gamma_{ij}}{2} (1 + \log(\sigma_{ij}^2) - (\mu_{new})_{ij}^2 - \sigma_{ij}^2) - (1 - \gamma_{ij}) \left(\frac{1 - \alpha}{1 - \gamma_{ij}} \right) - \gamma_{ij} \left(\frac{\alpha}{\gamma_{ij}} \right) \right] \right]$$

since the prior is the spike and slab distribution (refer to the Appendix B of Tonolini et al. [2019] for derivation of the *ELBO*).

μ_{new} in Equation [3.2] is:

$$\mu_{new} = \text{normalize}(\text{cay}(A) \mu) = \text{normalize}((I - A)^{-1}(I + A) \mu) \quad (3.4)$$

where *normalize* transformation is as in Definition [3.4.1].

Hence,

$$L_{model} = \beta \sum_{i=1}^m \left[\sum_{j=1}^k \left[\frac{\gamma_{ij}}{2} (1 + \log(\sigma_{ij}^2) - (\mu_{new})_{ij}^2 - \sigma_{ij}^2) - (1 - \gamma_{ij}) \left(\frac{1 - \alpha}{1 - \gamma_{ij}} \right) - \gamma_{ij} \left(\frac{\alpha}{\gamma_{ij}} \right) \right] \right] + \frac{1}{m} \sum_{l=1}^m \|x_i - x'_i\|_2^2 + \frac{1}{m} tr(\mu_{new}^T L \mu_{new}) \quad (3.5)$$

where k is the number of features of the latent variable which is equal to the number of clusters in the dataset. (γ, μ, σ) are the output of the encoder. m is the mini-batch size. This correction in spectral loss comes in to normalize the spectral loss irrespective of the

mini-batch size for mini-batch training. μ_{new} is as defined in the Equation (3.4).

The spike and slab probability distribution introduces sparsity to the orthogonal representations which closely resembles the matrices in the solution space of the *RatioCut* problem (or the balanced mincut problem) in (2.1.5). This sparsity in the latent variable sampling allows it to have a structure close to the cluster assignment structure (exact one non-zero entry in each row). We lose this cluster assignment structure when we relax the *RatioCut* problem. Hence sampling from spike and slab distribution produces matrices which are close to the exact solution.

Algorithm 4: ExSC training

Input: $X \in \mathbb{R}^{n \times d}$, number of clusters k

Output: Embeddings y_1, y_2, \dots, y_n and cluster assignments c_1, c_2, \dots, c_n where $c_i \in 1, 2, \dots, k$

- 1 Initialize the β -VAE network weights
 - 2 Initialize the cayley layer weight $A \in skew(p)$ using (5)
 - 3 **while** *loss is not converged* **do**
 - 4 Sample a random mini-batch X of size m ;
 - 5 Calculate the affinity matrix W using the gaussian kernel;
 - 6 Calculate the unnormalized laplacian L ;
 - 7 Forward propagate X to get μ and then through CayleyNN to get μ_{new} ;
 - 8 Compute the loss in (3.5);
 - 9 Use the gradient of each term of the loss function w.r.t each parameter to update all model parameters;
 - 10 Forward propagate all data points x_1, x_2, \dots, x_n to obtain $\mu_1, \mu_2, \dots, \mu_n$
 - 11 Normalize them using (3.4.1) to obtain y_1, y_2, \dots, y_n
 - 12 Do k-means clustering on y_1, y_2, \dots, y_n to obtain clusters
-

Chapter 4

Experimental Results

We compare the performance of our model with previous spectral clustering algorithm on benchmark real datasets MNIST and FashionMNIST. Experiments were performed mostly over MNIST dataset with models proposed in the previous chapter. The proposed models include geometric approach (through RGD), VAE trained on spectral loss and β -VAE with parameterization and different choices of prior (gaussian and spike & slab). The ExSC model which consist of β -VAE architecture (with spike and slab prior) together with parameterization is tested against various other clustering model on MNIST and FashionMNIST dataset.

4.1 Evaluation metric:

The clustering accuracy can be determined using two well known metrics(Cai et al. 2011): Normalized Mutual Information(NMI) and unsupervised clustering accuracy (ACC). Both the NMI score and the ACC score lie between 0 and 1.

NMI(or ACC) score of 1 denotes perfect clustering, that is, the two sets of clusters are identical (perfect overlap). Whereas, 0 score denotes that the two set of clusters are independent. Higher the score, better is the clustering.

Normalized Mutual Information (NMI)

Let T be set of clusters obtained from the ground truth and T' be the set of clusters obtained from a model's prediction.

The Mutual Information(MI) metric is defined as:

$$\text{MI}(T, T') = \sum_{t_i \in T, t'_j \in T'} p(t_i, t'_j) \log_2 \left(\frac{p(t_i, t'_j)}{p(t_i) \cdot p(t'_j)} \right) \quad (4.1)$$

where $p(t_i)$ and $p(t'_j)$ are the probability that an arbitrary point picked from the data space belongs to the clusters t_i and t'_j respectively. $p(t_i, t'_j)$ is the joint probability that a point picked at random belongs to both the clusters t_i and t'_j .

The Entropy $H(C)$ of a cluster C is defined as:

$$H(C) = \sum_{i=1}^k p(c_i) \cdot \log_2(p(c_i)) \quad (4.2)$$

where $C = \{c_1, c_2, \dots, c_k\}$ are the k cluster labels.

Finally, the Normalized Mutual Information (NMI) is defined as:

$$\text{NMI}(T, T') = \frac{\text{MI}(T, T')}{\max(H(T), H(T'))} \quad (4.3)$$

ACC

Let $S = \{s_1, s_2, \dots, s_n\}$ and $C = \{c_1, c_2, \dots, c_n\}$ denote the true labels and predicted labels for all n data points. That is, for datapoint x_0 , let s_0 and c_0 denotes true label and predicted label respectively.

ACC is defined as:

$$\text{ACC} = \frac{1}{n} \max_{\phi \in \Pi} \sum_{i=0}^{k-1} \delta(t_i, \phi(t'_i)) \quad (4.4)$$

where δ is the Delta function and Π denotes the space of all permutations of $\{1, 2, \dots, k\}$.

$$\delta(x, y) = \begin{cases} 1, & x = y \\ 0, & \text{otherwise} \end{cases}$$

The best mapping ϕ can be obtained using the Kuhn-Munkres Algorithm (Munkres 1957).

4.2 Datasets

MNIST Dataset

MNIST dataset is a collection of 70,000 gray-scale hand-written digits(0-9) of shape (28×28) where each value denotes the pixel intensity. Each values lies between 0-255 where 0 denotes completely black pixel and 255 denotes the completely white pixel. This accounts to 784 input features for each image. Furthermore, dataset is divided into 60,000 training images and 10,000 test images.

FashionMNIST Dataset

FashionMNIST dataset is similar to MNIST dataset but relatively challenging in both the classification problem and clustering. The dataset has 70,000 (60,000 train and 10,000 test images) gray-scale images belonging to one of these 10 classes: {T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag and Ankle boot}.

4.3 First Method: Geometric approach

Mainly two toy datasets were created for testing RGD for clustering, viz. concentric circles and entangled CC. As it can be seen in the Figure(4.1) the spectral loss converges for both data and $Y_{optimal}$ from Algorithm(3) gives out desired clusters. k-means as expected formed spherical clusters.

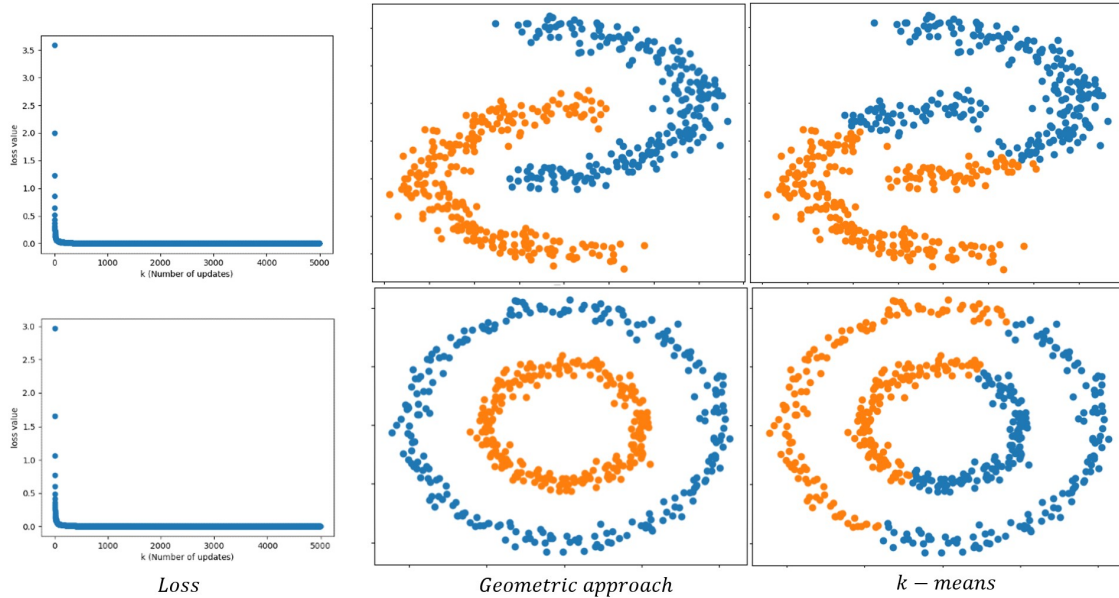


Figure 4.1: RGD and k-means comparison on toy datasets. Top: entangled CC, Bottom: Concentric circles

4.4 Initial approaches using neural network

4.4.1 β -VAE without any parametrization

Since the VAE bottleneck layer produces near orthogonal matrices (matrices with orthogonality index [5] close to 0), these representation were directly used for clustering in the initial tests. These orthogonal representation were directly treated as the embeddings of the input and k-means clustering was done on it. As expected, those representation were far from the eigenvector matrix (2.1.1) of L and produced results which were same as doing k-means clustering on the input itself. On over 60,000 training images, NMI came out to be 0.41 and a ACC score of 0.5. The reconstructed images were also not good enough (when compared to the 4.6) as can be seen in the Figure (4.2).

The next variation to this model was using β -VAE (gaussian prior) without any parameterization trained upon the sum of VAE loss and spectral loss as in the section (3.3.2). This model performs well when compared to the previous approaches but the Table (4.1) shows how crucial it is to add the cayley layer (or some other parameterization). This in fact shows how such parameterizations can fine tune the orthogonal matrix at the bottleneck to produce better orthogonal matrix (i.e. with lower orthogonality index).

Prior	Cayley layer	Train		Test	
		NMI	ACC	NMI	ACC
gaussian	✗	0.52	0.66	0.51	0.62
	✓	0.60	0.70	0.58	0.62
spike and slab	✗	0.46	0.52	0.49	0.54
	✓	0.61	0.64	0.66	0.69

Table 4.1: Results demonstrating impact of the cayley layer addition.

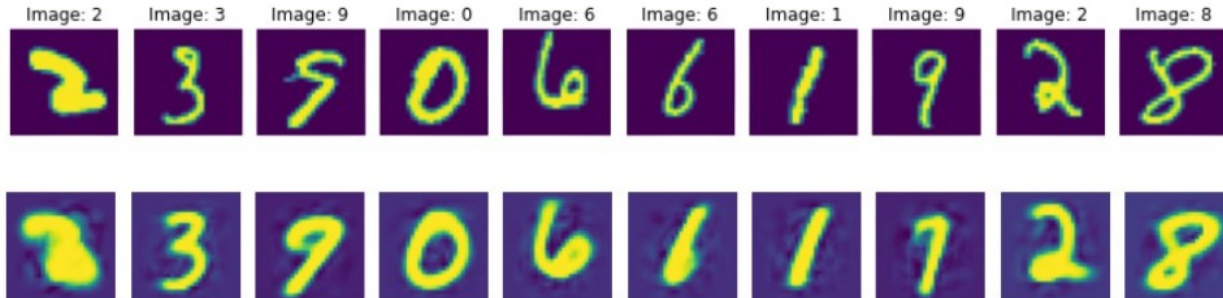


Figure 4.2: Top row shows the original images. Bottom row shows the reconstructed images from the simple β -VAE model.

4.4.2 β -VAE with parameterization and gaussian prior

The model is as in the section(3.3.3). Reconstructed images for the VAE model with cayley for gaussian prior is as shown in Figure(4.4). Normalized μ is a near orthogonal matrix at the end of the training but it further improves in terms of orthogonality upon changing the prior from gaussian to spike and slab.

4.5 Extended Spectral Clustering

The ExSC model with architecture as shown in Figure(3.3) was tested on MNIST and FashionMNIST datasets. The graph is constructed by joining each nodes with its 75 nearest neighbours and assigning weight to each edge using gaussian kernel (2.1). The scale variable σ in gaussian kernel was calculated by taking mean of distance of each point to its 3rd nearest neighbour (same way as SpectralNet decided σ). If the graph has components, spectral clustering tend to give each component as a separate cluster. Since the batch size (bs) is taken to be 100, choosing a high value of $k(\geq 0.5 \times bs)$ ensures that each graph constructed (for each mini-batch) is connected. Mini-batch size was taken to be 100 in all experiments.

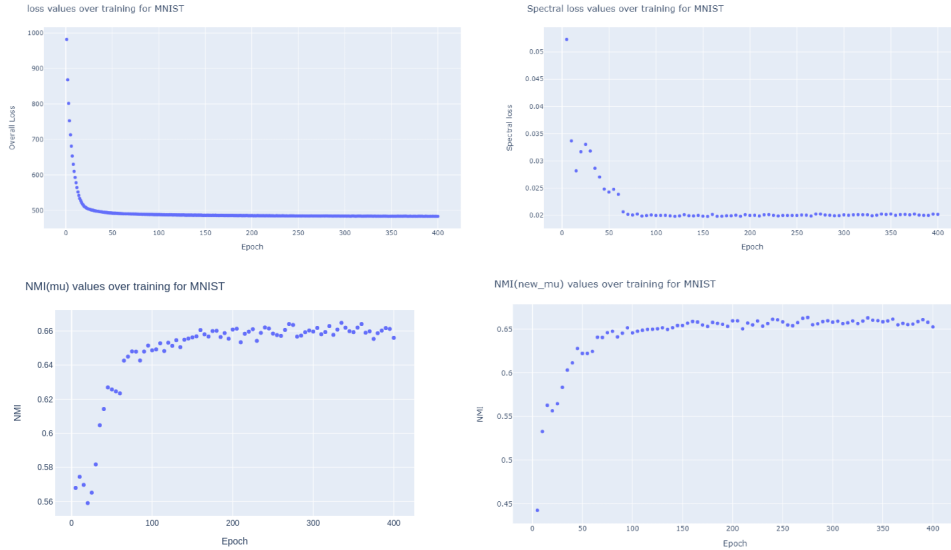


Figure 4.3: β -VAE with cayley layer and gaussian prior

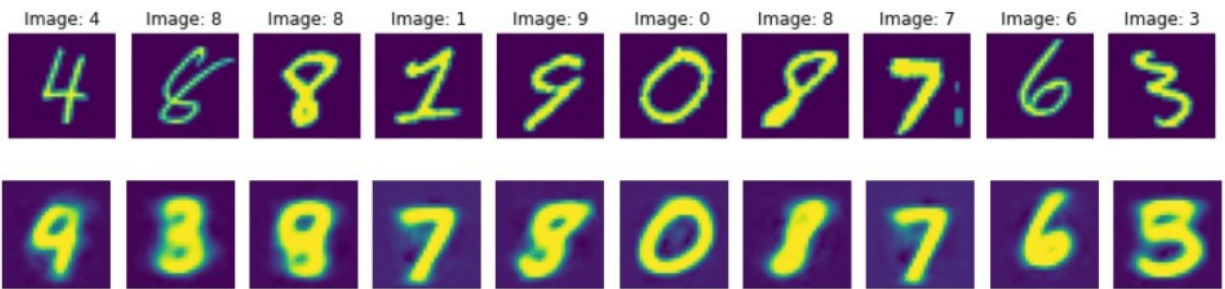


Figure 4.4: Top row shows the original images. Bottom row shows the reconstructed images from the β -VAE model with gaussian prior and cayley layer added.

The model also learn decoder during training, which can be used for multiple side tasks like generative model, data augmentation, feature extraction, recommender systems and search engines. Images reconstructed from the model are as shown in Figure(4.6).

Clustering results over MNIST with the ExSC model can be seen in Table(4.2). The model outperforms SpectralNet base model which uses euclidean metric to determine

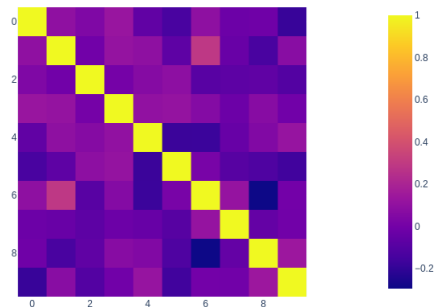


Figure 4.5: $\mu_n^T \mu_n$ after training for spike and slab prior. μ_n is normalized μ (Definition 3.4.1).

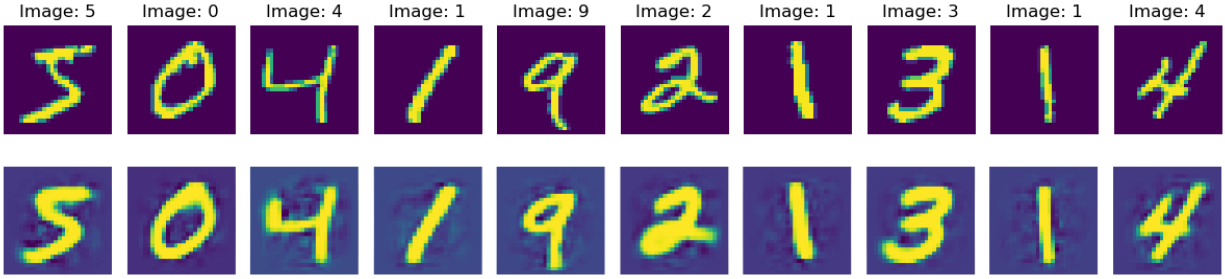


Figure 4.6: Top row shows the original images. Bottom row shows the reconstructed images from the β -VAE model with spike and slab prior and Cayley layer added.

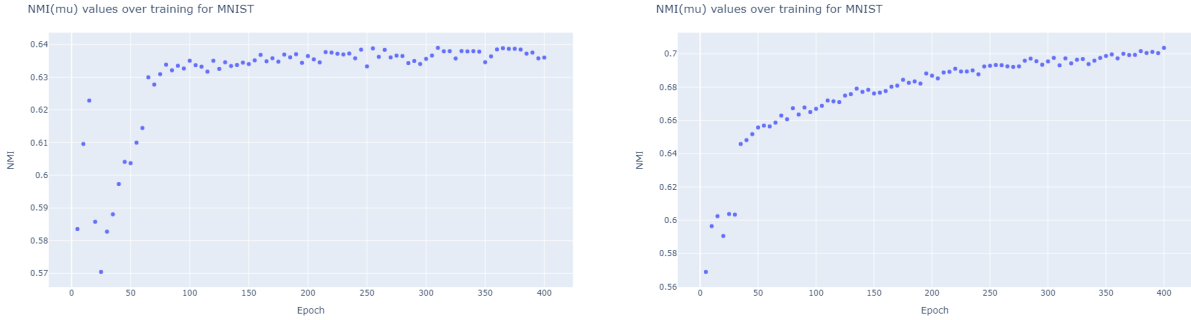


Figure 4.7: No Cayley layer is used. β -VAE with spectral loss was trained until convergence; Left: Gaussian prior, Right: Spike and slab prior

	Epochs	Train		Test	
		NMI	ACC	NMI	ACC
Our model	800	0.665	0.710	0.693	0.734
	980	0.675	0.766	0.687	0.765
	970	0.699	0.775	0.711	0.782
SpectralNet	-	0.671	0.62	-	-

Table 4.2: Result comparison of our ExSC model on MNIST dataset with SpectralNet (Shaham et al. [2018])

similarity (affinity matrix) between points

and uses pixel values directly as an input instead of using a code space (a pre-trained model which gives embeddings of an input. These embeddings can then be used as an input to SpectralNet instead of pixel values). There is significant improvement over NMI value and nearly 25% improvement in ACC score.

There was also a significant improvement in the accuracy of the model with addition of batch normalization. Further improvement in results (even on individual mini-batch) can be

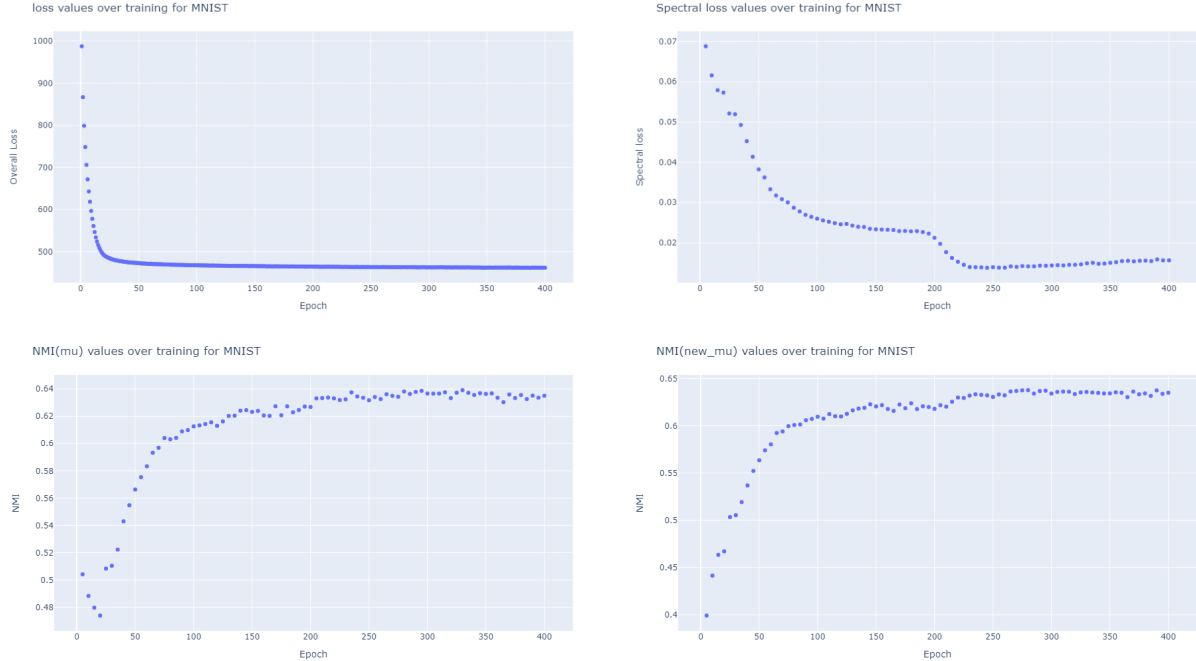


Figure 4.8: ExSC model (has Spike and Slab prior)

observed in Figure(4.7) by using Spike and slab prior instead of Gaussian prior in the β -VAE architecture. Also, the overall loss and the spectral loss converges in all runs (Figure 4.3, 4.8).

Validation loss is a good measure of convergence to a solution in unsupervised setting. 10% of the total training data was used as validation set. Convergence in validation loss can be seen with convergence in overall loss in Figure(4.9). All loss values converges with training. Validation loss can be used as a stopping criteria. Figure(4.10) is the confusion matrix plot for MNIST and FashionMNIST datasets. It shows good clustering as each row and column has single large entry. Since the predicted label will not be same as the true label, larger entries are not along the diagonal as one would expect for a confusion matrix(for example, label = 4 in prediction might refer to clusters of all handwritten digits “7”).

In the Figure(4.11), it can be clearly seen that changing the prior distribution has improved the quality of orthogonal representation at the bottleneck. More the $\mu^T \mu$ is close to being diagonal matrix, the better will be the normalized(μ) in terms of orthogonality. Figure(4.5) shows that the normalized μ in case of the spike and slab prior is nearly an orthogonal matrix and thus minimization of spectral loss over collection of such matrices is leading to good clustering results.

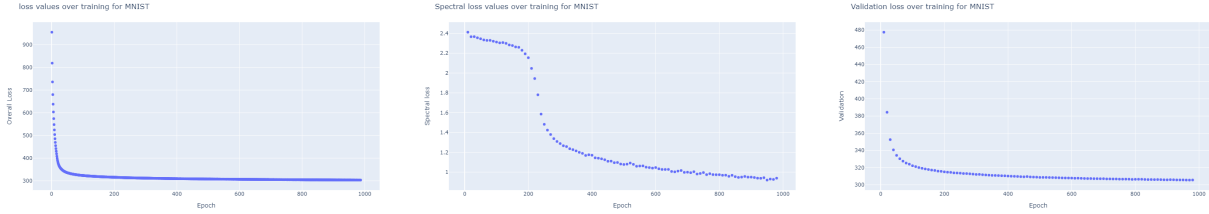


Figure 4.9: Overall loss, spectral loss and validation loss for 75nn with sns prior. All loss values are converging.

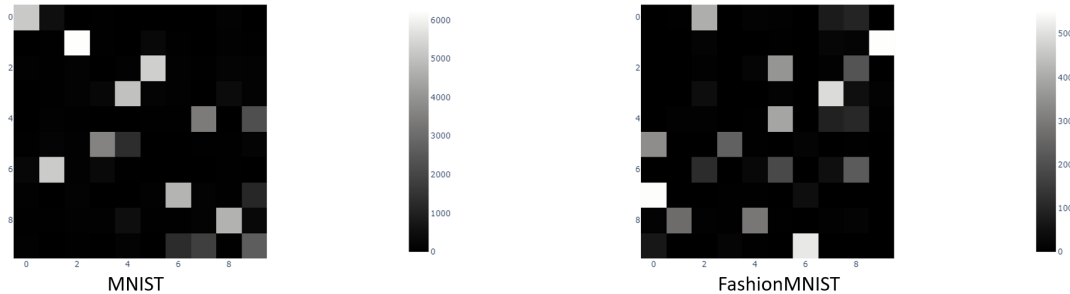


Figure 4.10: Confusion matrix for both datasets with predicted labels. Each row and column having single large entry indicates good clustering.

Nearest neighbours	Train		Test	
	NMI	ACC	NMI	ACC
25	0.60	0.63	0.64	0.65
50	0.62	0.70	0.63	0.69
75	0.63	0.66	0.67	0.69
100	0.63	0.67	0.68	0.70

Table 4.3: Result comparison on the MNIST dataset with different values of nearest neighbour for constructing laplacian. Mini-batch size = 100 and number of epochs = 400.

Our model is robust to selection of nearest neighbours while constructing the graph (which in turn gives laplacian). SpectralNet result drastically change with change in this parameter and can go as low as $NMI \sim 0.25$. This robustness can be observed in the Table(4.3)

Reconstructed images for FashionMNIST dataset are as shown in Figure(4.12) and confusion matrix for clustering results is as shown in Figure(4.10) (right). Clustering on the FashionMNIST dataset achieve $NMI \sim 0.63$ and $ACC \sim 0.61$.

We compared our model results with k-means (Macqueen [1967]), PSSC (Villar-Corrales and Morgenshtern [2020]), DEC (Xie et al. [2016]), IDEC (Guo et al. [2017]), JULE (Yang et al. [2016]), DCN (Yang et al. [2017]), DEPICT (Dizaji et al. [2017]), DDC (Ren et al. [2018]), N2D (McConville et al. [2020]) and ADEC (Mrabah et al. [2019]). Our model

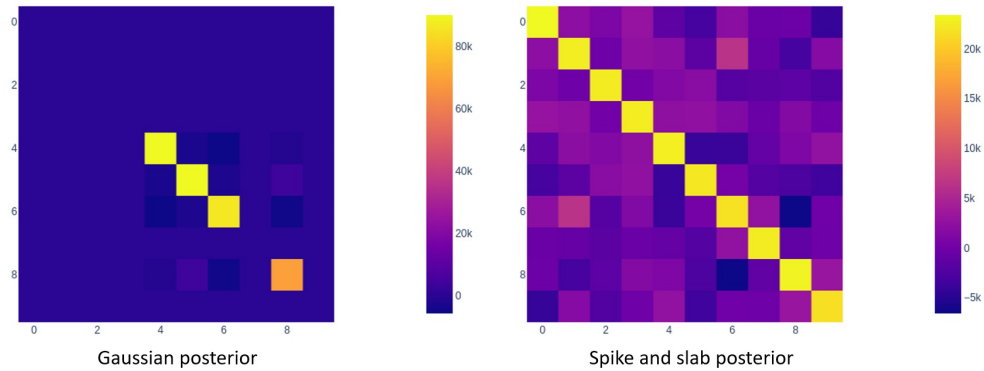


Figure 4.11: $\mu^T \mu$ for gaussian prior(left) and spike and slab prior(right). Diagonal matrices are better matrices for clustering

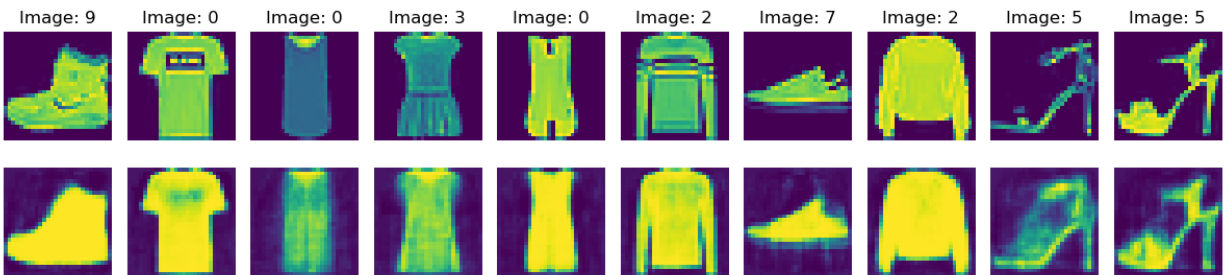


Figure 4.12: Top row shows the original images for FashionMNIST dataset. Bottom row shows the reconstructed images from the β -VAE model with spike and slab prior and Cayley layer added.

performed well against other clustering algorithms and outperformed most of them (Table 4.4) on FashionMNIST dataset.

Note: All the figures and plots are produced from training on MNIST dataset. For FashionMNIST dataset results and plots, it is explicitly mentioned in the captions.

Models	MNIST		FashionMNIST	
	ACC	NMI	ACC	NMI
k-means	0.584	0.497	0.474	0.512
PSSC	0.965	0.913	0.628	0.644
DEC	0.863	0.834	0.518	0.546
IDEC	0.881	0.867	0.529	0.557
JULE	0.964	0.913	0.563	0.608
DCN	0.830	0.810	0.501	0.558
DEPICT	0.965	0.917	0.392	0.392
DDC	0.965	0.932	0.619	0.682
N2D	0.979	0.942	0.672	0.684
ADEC	0.986	0.961	0.586	0.662
ExSC (Our)	0.699	0.775	0.611	0.627

Table 4.4: Result comparison on MNIST and FashionMNIST dataset. Note that none of them is an extension of Spectral clustering or uses Laplacian to determine clusters. (Source: Villar-Corrales and Morgenshtern [2020])

Chapter 5

Conclusion and Future work

Around 25% improvement in the ACC score and 4% improvement in NMI score can be observed over the SpectralNet’s result on MNIST dataset. The model also produces comparable results to the other clustering models on FashionMNIST dataset.

The ExSC model(our) consists of a trained decoder which can very well be used for multiple side tasks. ExSC model performs well even on the unseen data as it can be seen in the Table(4.2) that similar clustering accuracy is observed over test set (of 10,000 unseen data points) after training. There were two major addition to the SpectralNet base model, that were, the Siamese network (to better learn similarity between points) and the use of code space (to only keep important input features for learning). Both of these are independent of SpectralNet base model and can very well be implemented on top of our model as well. Addition of these two components could possibly lead to similar improvement in the results as the SpectralNet observed.

SpectralNet had a few major drawbacks. It doesn’t work across different datasets and is highly dependent on the hyperparameter setting. The robustness to hyperparameter setting is an important property for a clustering model since hyperparameter tuning requires supervision. Even the slightest of changes to the hyperparameters of the SpectralNet leads to a singular matrix before the “orthogonal layer” of SpectralNet and since Cholesky decomposition in the last layer requires matrix to be non-singular, the SpectralNet model does not work. Other limitation being loss of information. In the hyperparameter setting under which SpectralNet’s best results were obtained, only 3 nearest neighbours were chosen to

construct affinity matrix. This lead to laplacian being highly sparse. With mini-batch of size 1024 they only had 4 non-zero entries in each row of the laplacian. Our model works well under all these conditions and even full affinity can be used to do the clustering. Our model is also not much sensitive to the hyperparameter setting as one can see in Table(4.3) that for various values of nearest neighbour the model produces similar clustering results. Notice there is a difference in these results and the best results because of less number of epochs (fixed to 400 in these tests).

Clustering over the FashionMNIST using our model is promising as the model is giving comparable results to the other clustering algorithm (Table 4.4). Addition of a Siamese network with a code space can likely produce state-of-the-art result on the dataset. More theoretical work can be pursued on the topic. One can also review ways to improve CayleyNN layer architecture which can be made batch size independent through operation like ($y = x \text{cay}(A)^T$) and multiple such layer can be used before minimizing the spectral loss.

Bibliography

- Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000a.
- Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.
- Uri Shiham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. *arXiv preprint arXiv:1801.01587*, 2018.
- Angel Villar-Corrales and Veniamin I. Morgenshtern. Scattering transform based image clustering using projection onto orthogonal complement. *CoRR*, abs/2011.11586, 2020. URL <https://arxiv.org/abs/2011.11586>.
- Mario Lezcano Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems*, pages 9157–9168, 2019.
- I. Higgins, Loïc Matthey, A. Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.
- L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9): 1074–1085, 1992. doi: 10.1109/43.159993.
- Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000b. doi: 10.1109/34.868688.

- Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, USA, 2nd edition, 2012. ISBN 0521548233.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.
- Stephen Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function, 2019.
- Francesco Tonolini, Bjorn Sand Jensen, and Roderick Murray-Smith. Variational sparse coding, 2019. URL <https://openreview.net/forum?id=SkeJ6iR9Km>.
- D. Cai, X. He, and J. Han. Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge & Data Engineering*, 23(06):902–913, jun 2011. ISSN 1558-2191. doi: 10.1109/TKDE.2010.165.
- James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957. ISSN 03684245. URL <http://www.jstor.org/stable/2098689>.
- J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
- Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2016.
- Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1753–1759, 2017. doi: 10.24963/ijcai.2017/243. URL <https://doi.org/10.24963/ijcai.2017/243>.
- Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters, 2016.
- Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering, 2017.
- Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization, 2017.

Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. Deep density-based image clustering, 2018.

Ryan McConville, Raul Santos-Rodriguez, Robert J Piechocki, and Ian Craddock. N2d: (not too) deep clustering via clustering the local manifold of an autoencoded embedding, 2020.

Nairouz Mrabah, Mohamed Bouguessa, and Riadh Ksantini. Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift, 2019.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform, 2018.

Appendix

Experiment details:

The model architecture for the MNIST and FashionMNIST dataset: [784 – 392 – 261 – 20 – 10] for the encoder and decoder being [10 – 20 – 261 – 392 – 784] with cayley layer as [10 – 10]. These number denotes the number of in-features and out-features of each of the layers in the model. These number are specific to the MNIST and FashionMNIST dataset and vary as according to (3.3).

The recognition function (encoder) takes in input $x_i \in \mathbb{R}^{1 \times d}$ and returns the mean $\mu_i \in \mathbb{R}^{1 \times d}$, log variance $\log(\sigma_i^2) \in \mathbb{R}^{1 \times d}$ and log spike $\log(\gamma_i) \in \mathbb{R}^{1 \times d}$ where (μ, σ, γ) are the parameters of the approximate posterior. Each entry of γ_i has to lie in $[0, 1]$, thus we output $\log(\gamma_i)$ which ensures $\gamma_i \geq 0$ and further we can ensure $\gamma_i \leq 1$ by

$$\log(\gamma_i) = -\text{ReLU}(-v_i)$$

where v_i is the output of the encoder which goes into the bottle-neck layers.

z_i is sampled using the reparameterization trick as in the section (2.4.4). c is initialized with 50 and is gradually increased using $\delta_c = 0.001$ in each iteration.

The dataloader is set to produce random mini-batch in each iteration. These mini-batches also changes across epoch since keeping mini-batches same across epoch would mean only a sequence of diagonal blocks of size $m \times m$ in W_{full} ($n \times n$ affinity matrix) will be used for training and hence most of the entries of W_{full} are ignored.

Adam optimizer was used to make updates to the model parameters as it convergences faster, is computationally efficient and robust to the hyperparameter selection. The hyper-

parameters being the learning rate and the weight decay (L_2 penalty).

In initial testing, dropout layers were also used to further regularize the model but the model performed poorly (in producing orthogonal representations) across all experimental settings and hence dropout layers were dropped from the final model ExSC.

Stopping criteria

Stopping criteria was based on the validation loss. The last five validation loss values v_1, \dots, v_5 were taken and if $|v_i - v_{i+1}| \leq t \quad \forall 1 \leq i < 5$, the training stops. Here, t is the threshold and hence a hyperparameter. Update strategy was used on t where t is doubled in every 50 epochs after 800 epochs of training.

Orthogonality index

Orthogonality index of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

$$o(A) = \|A^T A - I\|_F$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix and the Frobenius norm $\|\cdot\|_F$ for a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}$$

Hyperparameter selection in our ExSC model was based on the quality of output orthogonal representations. The output was assessed based on the orthogonality index. This doesn't make use of data labels and hence can be used in the unsupervised setting of clustering problem.

Nearest neighbour hyperparameter is used in constructing the laplacian and scale neighbour parameter determines the σ in the gaussian kernel (2.1). Let scale neighbour = l then σ is the median of distances of each point from its l^{th} nearest neighbour. This strategy was proposed by Shaham et al. [2018] and was followed by us for a fair comparison of the results.

General	
Hyperparameter	Values
batch size	100
Validation set fraction	10%
number of clusters	10
Optimizer	Adam
activation function	Tanh
learning rate	1e-4
weight decay	1e-4
nearest neighbours	75
scale neighbours	2
threshold t	0.1

Spike and slab prior	
Hyperparameter	Values
α	0.5
c	50
δ_c	+0.001
β	0.1
δ_β	0

gaussian prior	
Hyperparameter	Values
β	20

Table 1: Hyperparameter setting

Skew-symmetric matrix initialization

We use the method proposed in [Helfrich et al. \[2018\]](#) to initialize the weight matrix of the cayley layer. $A \in skew(n)$ is initialized to be block diagonal matrix.

$$A = \begin{bmatrix} B_1 & & \\ & \ddots & \\ & & B_{\lfloor n/2 \rfloor} \end{bmatrix} \quad \text{where} \quad B_j = \begin{bmatrix} 0 & s_j \\ -s_j & 0 \end{bmatrix}$$

$$s_j = \sqrt{\frac{1-\cos(t_j)}{1+\cos(t_j)}} \quad \text{and} \quad t_j \text{ is sampled from } \mathcal{U}\left[0, \frac{\pi}{2}\right].$$

QR and Cholesky decomposition relation

Since we know these two relations:

$$Y = QR \tag{1}$$

$$Y^T Y = LL^T \tag{2}$$

Where first is the QR-decomposition of Y and second is the Cholesky factorization of $Y^T Y$.

$$\implies Q = YR^{-1}$$

Now since $Q^T Q = I$

$$\implies (YR^{-1})^T (YR^{-1}) = I$$

$$\implies (R^{-T} Y^T) (YR^{-1}) = I$$

$$\implies R^T R = Y^T Y = LL^T$$

$$\implies R^T R = LL^T$$

$$\implies L = R^T$$

$$\implies YL^{-T} = YR^{-1} = Q$$

This implies that right matrix multiplying Y with L^{-T} is giving the Q component of QR-decomposition of Y

So as we were taking $Q' = YL^{-T}$ in SpectralNet is same as taking Q from QR-decomposition as in the equation (1).