DISSERTATION

CONVEX AND NON-CONVEX OPTIMIZATION USING CENTROID-ENCODING FOR VISUALIZATION, CLASSIFICATION, AND FEATURE SELECTION

Submitted by

Tomojit Ghosh

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2022

ABSTRACT

CONVEX AND NON-CONVEX OPTIMIZATION USING CENTROID-ENCODING FOR VISUALIZATION, CLASSIFICATION, AND FEATURE SELECTION

Classification, visualization, and feature selection are three essential tasks of machine learning. This Ph.D. dissertation presents convex and non-convex models suitable for these three tasks. We propose Centroid-Encoder (CE), an autoencoder-based supervised tool for visualizing complex and potentially large data sets, such as, SUSY (supersymmetric particles) with 5 million samples and high-dimensional datasets, GSE73072 (Gene Expression Omnibus (GEO) database with identifier GSE73072) clinical challenge data. Unlike an autoencoder, which maps a point to itself, a centroid-encoder has a modified target, i.e., the class centroid in the ambient space. We present a detailed comparative analysis of the method using various data sets and state-of-the-art techniques. We have proposed a variation of the centroid-encoder, Bottleneck Centroid-Encoder (BCE), where additional constraints are imposed at the bottleneck layer to improve generalization performance in the reduced space. We further developed a sparse optimization problem for the non-linear mapping of the centroid-encoder called Sparse Centroid-Encoder (SCE) to determine the set of discriminative features between two or more classes. The sparse model selects variables using the ℓ_1-norm applied to the input feature space. SCE extracts discriminative features from multi-modal data sets, i.e., data whose classes appear to have multiple clusters, by using several centers per class. This approach seems to have advantages over models which use a one-hot-encoding vector. We also provide a feature selection framework that first ranks each feature by its occurrence, and the optimal number of features is chosen using a validation set.

CE and SCE are models based on neural network architectures and require the solution of non-convex optimization problems. Motivated by the CE algorithm, we have developed a convex optimization for the supervised dimensionality reduction technique called Centroid Component

Retrieval (CCR). The CCR model optimizes a multi-objective cost by balancing two complementary terms. The first term pulls the samples of a class towards its centroid by minimizing a sample's distance from its class centroid in low dimensional space. The second term pushes the classes apart in embedded space by maximizing the scattering volume of the ellipsoid formed by the class-centroids. Although the design principle of CCR is similar to LDA (Linear Discriminant Analysis), our experimental results show that CCR exhibits performance advantages over LDA, especially on high-dimensional data sets, e.g., Yale Faces, ORL (Olivetti Research Laboratory), and COIL20 (Columbia Object Image Library). Finally, we present a linear formulation of Centroid-Encoder with orthogonality constraints, called Principal Centroid Component Analysis (PCCA). This formulation is similar to PCA (Principal Component Analysis), except the class labels are used to formulate the objective, resulting in the form of supervised PCA. We show the classification and visualization experiments results with this new linear tool.

ACKNOWLEDGEMENTS

To begin with, I would like to thank Prof. Michael Kirby for his directions, suggestions, and feedback, without which the completion of this thesis would not have been possible. His course Mathematical Modeling of Large Datasets inspired me to examine machine learning from a geometrical perspective. His approach to connecting the subject to geometry has impacted my thinking process profoundly. I would also like to thank Prof. Charles Anderson, Prof. Asa Ben-Hur, and Prof. Henry Adams for serving on my committee and providing valuable feedback. I am also thankful to Manuchehr Aminian (Nuch), Shannon Stiverson, Kartikay Sharma, Eric Kahoe, Karim Karimov, Nathan Mankovich (Nate), Xiaofeng Ma, and Amy Peterson for being valuable colleagues.

I would like to thank my father, late Dilip Kumar Ghosh, mother, Anjali Ghosh, brother, late Nabendu Ghosh, and my sister Paromita Ghosh for their support since my childhood and the encouragement to pursue the higher study. Most of all, I would like to thank my wife, Shatabdi, who was instrumental in supporting my academic endeavors for the last couple of years.

Research is an intellectual exercise driven by some known as well as unknown factors. The known factors are easy to figure out and acknowledge. For example, I know how grateful I'm to my parents, family, colleagues, advisor, and committee members. Perhaps, I can quantify or partially measure their contribution to my dissertation. How about the unknowns? The concept of mapping the samples to their class centroids came to mind in a morning while listening to the classical music of Ludwig van Beethoven. I know that I was working on several clustering techniques and autoencoders during that time. But I was not thinking about it at the very moment when I got the idea of Centroid-Encoder. I've experienced the same many times during my research. These experiences made me think that perhaps an incomprehensible factor influenced my research, and I don't understand it completely. All I can do is acknowledge *it*, and I'm also grateful to *that*!

DEDICATION

To my daughter Trijeeta.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Recent technological advancements have made high-dimensional data readily available, and it has become popular to refer to the *data deluge* being experienced by society in general [13]. This deluge spans basically all areas of the scientific and engineering field, including, e.g., Astronomy, Archeology [14], Signal Processing [15], and Biology [16]. In bioinformatics, researchers seek to understand *omics* data such as the gene expression levels measured by microarrays or next-generation sequencing techniques; in the latter case, samples may consist of over 50,000 measurements [17–20]. Often these high-dimensional features may be noisy, redundant, missing, or irrelevant [21], which has the potential to degrade the performance of Machine Learning tools [22]. Despite these challenges, there is strong evidence that these biological data sets capture information that will enable knowledge discovery from data [23].

The discovery of new facts from data requires the synthesis of three related objectives, including predictive model building, feature selection, and data visualization. The work presented here treats these three topics as inter-related and mutually dependent. Predictive model building allows one to use high-quality features to make accurate statements about samples, such as prognosis or diagnosis of health state. Data (dimension) reduction often improves classification and provides an additional opportunity for visualization. Coming full circle, feature selection helps with the identification of salient features that are useful for discrimination in predictive models. This dissertation addresses the interplay of these ideas in the setting of the *centroid-encoder* that performs classification, visualization, and feature selection.

One of the first stages of data exploration is visualization in two or three-dimensional space. Visualization provides a window into high dimensions, which helps a data explorer in many ways to understand the data, including the characterization of class structure, data separability, and experimental batch effects. The goal of a dimensionality reduction (DR) technique for visualization is to preserve some of the intrinsic properties of interest in visualization space. The intrinsic property can be statistical [24], topological [25], or geometrical [26]; in the presence of labels, the intrinsic property may consist of discriminating features to assign class membership [27]. Principal component analysis [28, 29] continues to be one of the most widely used methods for data reduction and visualization over one hundred years after its initial discovery [30, 31]. Nonetheless, PCA often produces ambiguous results, sometimes collapsing distinct classes into overlapping regions in the setting where class labels are available. It is tempting to infer incorrectly that the data is not separable, even non-linearly, in higher dimensions.

Non-linear extensions to PCA were originally introduced to address the limitations of optimal linear mappings [32–36], also see [37] for additional early references and details. These papers provided the first applications of autoencoder neural networks where data sets are non-linearly mapped to themselves. This is accomplished by first taking the data to higher dimensions before passing through a bottleneck layer of a reduced dimension where data visualization is typically done. In the process of non-linear dimension reduction, novel *latent* or hidden features which are an amalgamation of the observables may be discovered.

Data reduction techniques such as PCA and its non-linear extensions have proven extremely useful for understanding data in high dimensions, particularly as visualization tools. This research addresses how such devices can be adapted to the case where class label information is available, i.e., supervised data reduction. More specifically, one objective of this dissertation is the integration of class label information into the non-linear autoencoder reduction process for data visualization. Here we develop the *centroid-encoder* (CE) algorithm to analyze labeled data. While standard autoencoders are the identity mapping on points, centroid-encoders map points to their class centroid while passing through a low-dimensional representation. This approach provides a low-dimensional encoding for visualization while ensuring that elements with the same label retain their class structure required for classification. The smoothness of mapping functions ensures that similar behavior is captured at the centroid-encoder bottleneck layer.

Inspired by the centroid-encoder, we extend the centroid mapping in a linear setup to propose supervised PCA. The second formulation seeks a linear transformation that reconstructs a data point as its class centroid with the orthogonality constraint.

Typically DR (Dimensionality Reduction) techniques use all the features of the data to produce a low-dimensional embedding for visualization. Such methods don't attempt to distinguish the relative importance of features in the reduction process. However, a data analyst often is interested to learn which features are essential for a given machine learning task. For example, one may be interested to know which biomarkers are crucial to predict neonatal sepsis [38]. Questions like this come naturally while analyzing high-dimensional biological data. The abundance of features demands the development of feature selection algorithms to improve a Machine Learning task, e.g., classification. Which biomarkers are important to characterize a biological process, e.g., the immune response to infection by respiratory viruses such as influenza [39]? Additional benefits of feature selection include improved visualization and understanding of data, reduced storage requirements, and faster algorithm training times.

In this research, we propose a new variable selection approach called Sparse Centroid-Encoder (SCE) to extract features when class labels are available. Our method extends the Centroid-Encoder model [40, 41], with an ℓ_1 penalty [42, 43] sparsity promoting layer between the input and the first hidden layer. We evaluate this Sparse Centroid-Encoder on diverse data sets and show that the selected features produce better generalization than other state-of-the-art techniques. Our results showed that SCE picked fewer features to obtain high classification accuracy. As a feature selection tool, SCE uses a single model for the multi-class problem without the need to create multiple one-against-one binary models typical of linear methods, e.g., Lasso [44], or Sparse SVM [8].

The development of tools for DR (Dimensionality Reduction) is predicated by the nature of the underlying optimization problem, i.e., convex or non-convex. Convex optimization has a unique global extremum, which is algorithmically very attractive. Convex optimization tools, such as the primal-dual interior-point method, provide provable convergence to a global optimum and are generally robust to the selection of the initial condition [45]. In contrast, non-convex problems have local minima making it generally infeasible to determine an optimal solution. In this setting, the initial condition plays a significant role, and if the initialization is bad, then the final solution often gets stuck in a poor local minimum.

Motivated by the intrinsic appeal of convex optimization, we present a new model for the task of supervised DR. Our algorithm, *Centroid Component Retrieval* (CCR), optimizes a multi-objective cost. The resulting model seeks to map any point to the reduced representation of its class centroid, thus increasing class localization. At the same time, the model pushes the classes far apart by maximizing the volume of the scatter of the class-centroids in the low-dimensional space.

1.2 Document Outline

The rest of the dissertation is organized as follows: In chapter 2, we present the related works. Chapter 3 contains the theory and application of the Centroid-Encoder. Bottleneck Centroid-Encoder (BCE), a variant of Centroid-Encoder, is described in chapter 4. Our proposed feature selection tool, Sparse Centroid-Encoder (SCE), is described in chapter 5. Chapter 7 contains the theory and application of the proposed convex model Centroid Component Retrieval (CCR). At last, we present Principal Centroid Component Analysis (PCCA), a formulation of supervised PCA in chapter 6. We present our conclusion and future research direction in chapter 8.

1.3 Research Summary

This section summarizes the research, breaks it down into chapters, and lists each chapter's main contributions. In Table1.1, we list all the five models along with a brief description. Our purpose is to provide a quick overview of the dissertation. The table is helpful in relating all the five models, which will benefit the readers to understand the works presented in this thesis.

1.3.1 Chapter 3: Supervised Visualization using Centroid-Encoder

The main contribution of this chapter is listed below.

Table 1.1: Brief summary of the proposed models.

Model	Brief Description
Centroid-Encoder (CE) $L_C(\theta) = \frac{1}{2N}\sum_{j=1}^{M}\sum_{i\in I_j} \|c_j - f(x_i;\theta))\|_2^2$	A nonlinear model for supervised dimensionality reduction and visualization using bottleneck architecture.
Bottleneck Centroid-Encoder (BCE) $L_{BCE}(\theta) = L_C(\theta) + \frac{\lambda_1}{2N}\sum_{j=1}^{M}\sum_{i\in I_j} \|g(c_j) - g(x_i)\|_2^2$ $+\lambda_2 \sum_{k<l} \frac{1}{1+\|g(c_k)-g(c_l))\|_2^2}$	An extension of CE with two penalty terms applied on the bottleneck layer to increase class localization and class separation.
Sparse Centroid-Encoder (SCE) $L_{SCE}(\theta, \theta_{spl}) = L_C(\theta) + \lambda\|\theta_{spl}\|_1$	A nonlinear feature selection technique using 1-norm coupled with the Centroid-Encoder loss.
Principal Centroid Component Analysis (PCCA) $\underset{\mathbf{a}}{minimize}\ \ \|\tilde{C} - \mathbf{a}\mathbf{a}^T X\|_F^2\ \ subject\ to\ \mathbf{a}^T\mathbf{a} = 1$	A constrained linear formulation of Centroid-Encoder using eigendecomposition
Centroid Component Retrieval (CCR) $\underset{A}{minimize}\ \ \|A^T(\tilde{C} - X)\|_F^2 - \lambda \log\det(A^T\hat{C}\hat{C}^T A + \gamma I)$	A convex formulation that minimize class scatter and maximizes class separation in the embedded space.

1. We proposed a non-linear dimension reduction technique called centroid-encoder. Centroid-Encoder maps data to the corresponding class centroid through a low-dimensional bottleneck analogous to autoencoder. Hence it exploits label information in the non-linear data (dimension) reduction process and may be viewed as a supervised variant of *non-linear principal component analysis* [33–35].

2. We provided convergence proof of Centroid-Encoder that shows how the non-linear mapping moves a sample to its class centroid.

3. As a mapping from the input space to the reduced space, CE generalizes directly to new data without using landmarks or the computation of distance matrices typical of spectral methods [12, 46] or self-organizing mappings [47, 48].

4. The advantage of centroid-encoder over many other data visualization techniques is that it is not necessary to compute pairwise distances, e.g., dt-MCML, dG-MCML, dt-NCA, and dG-NCA [3].

5. We empirically show that CE can preserve the neighborhood relationships of class-labeled data without explicitly computing the neighborhood graph.

6. We show that CE produces similar or better results than other popular supervised methods that explicitly use pair-wise distances or neighborhood graphs.

7. We apply CE to produce the first low-dimensional visualization of the large particle Physics Supersymmetry (SUSY) data set and show that it classifies samples with accuracy comparable to the best high-dimensional models.

8. We also extend the centroid-encoder to capture the multi-modality of data by incorporating multiple centroids per class. We applied the model to the SUSY data set.

1.3.2 Chapter 4: Bottleneck Centroid-Encoder

The highlights of this chapter are mentioned below.

1. We proposed a modification of the Centroid-Encoder, Bottleneck Centroid-Encoder (BCE) model via adding a constraint in the bottleneck layer.

2. The CE calculates the centroids in the ambient space. If the centroids of multiple classes are close to each other, then the corresponding samples will land close in the reduced representation, which in turn causes misclassification. We add two more terms to the bottleneck layer to avoid this situation.

3. BCE employs a centroid constraint at the bottleneck layer. This bottleneck cost has two contrasting terms. The first term pulls all the samples of a class to its centroid by minimizing the sample's distance from its class centroid in reduced space. The second term pushes the classes by maximizing the Euclidean distance between class-pair centroids.

4. We analyzed the importance of the two terms on the model's performance.

5. We empirically showed that the BCE produces better generalization performance than CE and other techniques.

1.3.3 Chapter 5: Sparse Centroid-Encoder

Using the Sparse Centroid-Encoder model, we have achieved the following objectives.

1. We proposed a non-linear feature selection technique, called *Sparse Centroid-Encoder* (SCE) by adding a ℓ_1 penalty to the optimization of Centroid-Encoder.

2. Sparse Centroid-Encoder captures the intra-class variance by using multiple centers per class. This approach detects better discriminatory features and provides valuable information on whether the classes are unimodal (samples belongs to a distribution which has a single peak) or multimodal (samples belongs to a distribution which has multiple peaks). This aspect of SCE distinguishes itself from other feature selection techniques, which don't model intra-class variability.

3. We also provide a feature selection framework that first ranks each feature by its occurrence, and the optimal number of features is chosen using a validation set.

4. We conducted an array of experiments to compare and contrast SCE with a deep features selection technique, DFS [6]. We found that SCE is a more robust approach for weight sparsity promotion.

5. We also analyzed the challenges of minimizing ℓ_1-norm using stochastic optimization e.g. Adam [49].

6. We empirically showed that SCE features produced better classification accuracy on the unseen test data than other state-of-the-art techniques.

1.3.4 Chapter 6: Principal Centroid Component Analysis

The gist of this chapter is presented below.

1. We presented a new formulation of centroid mapping in a linear setup with orthogonality constraints. The resulting model is a linear counterpart of the Centroid-Encoder.

2. We show that the solution comes from the eigendecomposition of symmetric matrices. The formulation is closely related to PCA, except the labels are used, resulting in a new variation of supervised PCA.

3. We established a mathematical relationship between the eigenvalue and the original cost.

4. We present several classification and visualization experiments with our new tool, comparing the vanilla PCA and a supervised version of PCA [50].

1.3.5 Chapter 7: Centroid Component Retrieval

We summarize the main contribution as listed below.

1. We formulated a convex model, Centroid Component Retrieval (CCR), for the task of supervised dimensionality reduction and visualization.

2. The proposed model doesn't require the pair-wise distance calculation, and this aspect makes our model faster than other techniques.

3. The design principle of CCR is similar to LDA, but the optimization problem of CCR is convex, whereas the optimization of LDA is non-convex.

4. The generalized eigenvalue solution of LDA performs poorly when the within-class scatter matrix becomes singular for the high-dimensional data sets [51]. CCR doesn't suffer from this limitation.

5. The experimental results suggest that CCR has performance benefits over LDA, particularly for high-dimensional data sets.

Chapter 2

Background

2.1 Dimensionality Reduction

Data dimensionality reduction for visualization has a long history and remains an active research area. We describe the literature relating to unsupervised and supervised methods where we assume data class labels are unavailable and available, respectively.

2.1.1 Unsupervised Methods

Principal Components Analysis is often the first (and last) tool used for visualization of unlabeled data and is designed to retain as much of the statistical variance as possible [28, 29], see also [24]. Equivalently, PCA minimizes the mean-square approximation error as well as Shannon's entropy [52]. Self-organizing mappings (SOMs) learn nonlinear *topology-preserving* transformations that map data points to centers that are then mapped to the center indices. SOMs have been widely used in data visualization, having been cited over 20,000 times since their introduction [53].

Another class of methods uses interpoint distances as the starting point for data reduction and visualization. For example, Multidimensional Scaling is a spectral method that computes a point configuration based on the computation of the eigenvectors of a doubly centered distance matrix [54]. The goal of the optimization problem behind MDS is to determine a configuration of points whose Euclidean distance matrix is optimally close to the prescribed distance matrix. A related approach, known as Isomap, applies MDS to approximate geodesic distances computed numerically from data on a manifold [55]. Laplacian Eigenmaps is another popular spectral method that uses distance matrices to reduce dimension and conserve neighborhoods [12]. These spectral methods belong to a class of techniques referred to as *manifold learning* algorithms; see also locally linear embedding [46], stochastic neighbor embedding [56], and maximum variance unfolding [57].

The technique t-distributed stochastic neighbor embedding [58], an extension of SNE, was developed to overcome the data crowding or clumping problem often observed with manifold learning methods and is currently a popular method for data visualization. More recently, the uniform manifold approximation and projection (UMAP) algorithm has been proposed, which uses Riemannian geometry and fuzzy simplicial sets to create a low-dimensional and locally uniformly distributed embedding of the data [59]. It has been reported that the algorithm has a better run time than t-SNE and offers compelling visualizations.

Note that these spectral methods including MDS, Laplacian Eigenmaps and UMAP compute embeddings based on solving an eigenvector problem requiring the entire data set. Such methods do not actually create mappings that can be applied to reduce the dimension of new data points without repeating the computation or resorting to the use of landmarks [60]. This, in contrast to methods such as PCA, SOM, parametric t-SNE [2] and autoencoders that serve as mappings for streaming data.

2.1.2 Supervised Methods

Fisher's linear discriminant analysis (LDA) reduces the dimension of labeled data by simultaneously optimizing class separation and within-class scatter [27, 61]. Both LDA and PCA are linear methods in that they construct optimal projection matrices, i.e., linear transformations for reducing the dimension of the data.

It is, in general, possible to add labels to unsupervised methods to create their supervised analogues. A heuristic-based supervised PCA model first selects important features by calculating correlation with the labels and then applies standard PCA of the chosen feature set [62]. Another supervised PCA technique, proposed by [50], uses Hilbert-Schmidt independence criterion to compute the principal components which have maximum dependence on the labels. Colored maximum variance unfolding [63], which is the supervised version of MVU (Maximum Variance Unfolding), is capable of separating different newsgroups better than MVU and PCA.

The projection of neighborhood component analysis (NCA) [64] produces a better coherent structure in two-dimensional space than PCA on several UCI data sets. Parametric embedding [65], which embeds high-dimensional data by preserving the class-posterior probabilities of objects, separates different categories of Japanese web pages better than MDS. Optimizing the NCA objective on a pre-trained deep architecture, Salakhutdinov et al. [66] achieved 1% error rate on MNIST test data with 3-nearest neighbor classifier on 30-dimensional feature space. Min et al. proposed to optimize the NCA and maximally collapsing metric learning (MCML) [67] objective using a Student t-distribution on a pre-trained network [3]. Their approach yielded promising generalization error using a 5-nearest neighbor classifier on MNIST and USPS data on two-dimensional feature space. Neighbor retrieval visualizer (NeRV) [5] optimizes its cost such that the similar objects are mapped close together in embedded space. Its supervised variant uses the class information to produce the low-dimensional embedding. NeRV and its supervised counterpart were reported to outperform some of the state-of-the-art dimension reduction techniques on a variety of datasets.

Harnessing the representation power of deep autoencoders, Xie et al. proposed a deeply embedded clustering (DEC) which optimizes the clustering assignment in the representation space of an autoencoder [68]. In DEC, the training samples are first transformed into a low dimensional space by the non-linear mapping of the denoising autoencoder [69]. After that, the decoder is discarded, and the cluster assignment is optimized using an auxiliary target distribution on the output of the encoder. Note, DEC doesn't use the label information of the samples, therefore, it's strictly an unsupervised method. Le et al. proposed a supervised autoencoder (SAE) that includes the unsupervised reconstruction loss on the data and the supervised regression loss [70]. The supervised loss is added to the output layer in a shallow autoencoder (one hidden layer). In a deep autoencoder, the authors suggested adding the supervision in the bottleneck layer. They show that the unsupervised reconstruction loss improves the generalization performance. Adversarial Autoencoders (AAE) [71], fall into the category of generative models, aims to optimize dual objectives - the traditional reconstruction loss of autoencoders and adversarial criteria [72] to match the aggregated posterior distribution of the autoencoder to an arbitrary prior distribution. The adversarial cost forces the encoder to learn the user-defined prior distribution while the decoder imposes the prior distribution while generating the samples. In addition, the supervised counterpart of AAE incorporates labels into the model to guide the data generation.

Supervised Isomap (S-Isomap) [73], which explicitly uses the class information to impose dissimilarity while configuring the neighborhood graph on input data, has a better visualization and classification performance than Isomap. Zhang et al. suggested to use labels to optimize the objective of Local Linear Embedding and their supervised model [74] performed better than LLE. Similarly, the supervised version of Laplacian Eigenmaps [75] yield better quality visualization than its unsupervised counterpart. Stuhlsatz et al. proposed a generalized discriminant analysis based on classical LDA (GerDA) [1], which is built on deep neural network architecture. Min et al. proposed a shallow supervised dimensionality reduction technique where the MCML objective is optimized based on some learned or precomputed exemplars [4].

2.1.3 Limitations of Current Work

Dimensionality reduction is a very active area of research, and the field already has a host of well-performing and widely-used techniques. Here we summarize some of the critical limitations of the current tools. Many techniques e.g., [3], [5], [64], [58], [59], rely on the pair-wise distance calculation to create the embedding. The distance matrix calculation makes these methods costly, i.e., they scale quadratically with the number of data points and can be prohibitively slow as the data set size increases. Spectral methods, e.g., MDS, Laplacian Eigenmaps, require an eigenvector calculation using the entire training set, including testing data. Such methods do not actually create mappings that can be applied to reduce the dimension of new data points without repeating the computation or resorting to the use of landmarks [60]. UMAP involves constructing a second fuzzy simplicial set for test data by rerunning the nearest neighbor search using the training data. Methods like UMAP and its supervised counterpart [59] depend on the initialization. Wang et al. [76] showed that the embedding of UMAP is substantially worse when initialized randomly as opposed to initialized with spectral embedding [77]. Deep neural network-based methods, e.g., deep-NCA (dt/dG-NCA) (see [3]), keep the similar objects close together in the embedding space. These techniques don't consider modeling the dissimilarities among the samples of different classes. Moreover, these techniques use pair-wise distances, which makes them computationally expensive. Supervised autoencoder [70] uses regression loss on the one-hot-encoding target along with the reconstruction loss. Note that one-hot vectors of each class are equidistant from each other; therefore, mapping to one-hot vectors may be helpful for classification but may not reveal the geometric structure of data.

On the other hand, deep-MCML (dt/dG-MCML) (see [3]) collapse all the samples of a class to a point, and at the same time, the model exerts a repulsive force among the different categories to separate them. The repulsive force doesn't consider the global structure of classes in the reduced space. In theory, the repulsive force can push two clusters far apart without considering their distance in the ambient dimension. Therefore, incorporating a repulsive force in the model may improve the k-NN classification accuracy, a standard for comparing DR methods; but it may not reveal the actual global layout of data. Note, like deep-NCA, deep-MCML also relies on pair-wise distance computation.

2.2 Feature Selection

Feature selection can be accomplished in various ways that can be broadly categorized into the filter, wrapper, and embedded methods. In a filter method, each variable is ordered based on a score. After that, a threshold is used to select the relevant features [78]. Variables are usually ranked using correlation [79, 80], and mutual information [81, 82]. In contrast, a wrapper method uses a model and determines the importance of a feature or a group of features by the generalization performance of the predetermined model [83, 84]. Wrapper methods are computationally intensive for larger data sets, in which case search techniques like Genetic Algorithm (GA) [85] or Particle Swarm Optimization (PSO) [86] are used. In embedded methods, feature selection criteria are incorporated within the model, i.e., the variables are picked during the training process [87]. Iterative Feature Removal (IFR) uses the absolute weight of a Sparse SVM model as a criterion to extract features from the high dimensional biological data set [39]. As Sparse Centroid-Encoder is an embedded method, we present the literature review related to embedded techniques.

2.2.1 Feature Selection using Linear Models

Linear models are widely used in Machine Learning for classification and regression. These models approximate the output as a linear combination of input variables (features), i.e. $y \approx f(x) = w^T x + b$ where w and b are the model parameters. From the optimization perspective, a linear models takes the following form: $\underset{\theta}{minimize} \;\; l(y, f(x, \theta))$ where l is a loss function and θ is the parameter set. Adding an ℓ_1 penalty to the parameter set θ gives a feature selector. For example, least absolute shrinkage and selection operator or Lasso [44], has been used extensively for feature selection on various data sets [88–90]. Elastic net, proposed by Zou et al. [91], combined the Lasso penalty with the Ridge Regression penalty [92] to overcome some limitations of Lasso. Elastic net has been widely applied, e.g., [93–95]. Note both Lasso and Elastic net are convex in the parameter space. Support Vector Machines (SVM) [96] are a state-of-the-art model for classification, regression and feature selection. SVM-RFE is a linear feature selection model which iteratively removes the least discriminative features until a parsimonious set of predictive features are selected [97]. IFR [39], on the other hand, selects a group of discriminatory features at each iteration and eliminates them from the data set. The process repeats until the accuracy of the model starts to drop significantly. Note IFR uses Sparse SVM (SSVM), which minimizes the ℓ_1 norm of the model parameters. Lasso, Elastic Net, and SVM-based techniques are mainly applied to binary problems. These models are extended to the multi-class problem by combining multiple binary one-against-one (OAO) or one-against-all (OAA) models. Authors of [8] used 120 Sparse SVM models to select discriminative bands from the Indian Pine data set, which has 16 classes. On the other hand, Random Forest [98], a decision tree-based technique, finds features from multi-class data using a single model. The model doesn't use Lasso or Elastic net penalty for feature selection. Instead, the model weighs the importance of each feature by measuring the out-of-bag error.

2.2.2 Feature Selection using Deep Neural Networks

While the above mentioned linear feature selection models are fast and convex, they don't capture the non-linear relationship among the input features (unless a kernel trick is applied). Because of the shallow architecture, these models don't learn a high-level representation of input features. Moreover, there is no natural way to incorporate multi-class data in a single model. Non-linear models based on deep neural networks overcome these limitations. In this section, we will briefly discuss a handful of such models.

Scardapane et al. [7] used group Lasso [44] to impose the sparsity on a group of variables instead of a single variable. They applied the group sparsity simultaneously on the input and the hidden layers to remove features from the input data and the hidden activation. On MNIST, their algorithm discarded more than 200 features from the input vector with an accuracy of 97% on the test data. Although on the Forest Cover data set, the algorithm used most of the input variables 52.7 out of 54. Li et al. proposed deep feature selection (DFS), which is a multilayer neural network-based feature selection technique [6]. DFS uses a one-to-one linear layer between the input and the first hidden layer. As a sparse regularization, the authors used elastic-net [91] on the variables of the one-to-one layer to induce sparsity. The standard soft-max function is used in the output layer for classification. With this setup, the network is trained in an end-to-end fashion by error backpropagation. Despite the deep architecture, its accuracy is not competitive, and experimental results have shown that the method did not outperform the Random Forest (RF) method. Kim et al. [99] proposed a heuristics based technique to assign importance to each feature. Using the ReLU activation, Roy et al. [100] provided a way to measure the contribution of an input feature towards hidden activation of next layer. Han et al. [101] developed an unsupervised feature selection technique based on the autoencoder architecture. Using a $l_{2,1}$-norm to the weights emanating from each input node, they measure the contribution of each feature while reconstructing the input. The model removes the input features, which have a minimum contribution. Taherkhani et al. [102] proposed an RBM [103,104] based feature selection model which discards a feature if the reconstruction error doesn't increase after setting the corresponding input to zero. Recently, Balin et al. [105] proposed an end-to-end unsupervised feature selection technique, namely Concrete Autoencoders (CAE). The authors utilize the concrete random variable, a continuous approximation of a one-hot vector in the feature selection layer. One of the attractive features of CAE is that its cost function is differentiable, and the model picks a subset of original features by gradually minimizing the temperature of the concrete feature selector layer using an annealing scheme. Yamada et al. [10] proposed Stochastic Gates, which incorporates continuous relaxation of the Bernoulli distribution to approximate ℓ_0-norm. The FsNet, proposed by Singh et al. [11], designed for high-dimensional biological data sets, also uses a Concrete feature selection layer along with Diet Networks [106] to reduce the model size. LassoNet [9], on the other hand, uses proximal gradient descent to incorporate sparsity promoting ℓ_1 norm to update model parameters.

2.2.3 Limitations of Current Techniques

Now we discuss some of the limitations of feature selection techniques mentioned before. Linear methods, such as Sparse SVM [8], LASSO [44], and Elastic Net [91] assume the classes of samples are approximately linearly separable and use an affine mapping of the features to separate the data. Therefore these linear techniques can't model the nonlinear geometry. LASSO, Elastic Net, and Sparse SVM are mainly designed for binary-class problems, and they can't handle multiple classes using a single model. These models are extended to the multi-class problem by combining multiple binary one-against-one (OAO) or one-against-all (OAA) models. Authors in [8] used 120 Sparse Sparse SVM models to select discriminative bands from the Indian Pine data set, which has 16 classes. These limitations are addressed in neural network-based techniques, e.g., DFS [6], Group Sparse ANN [7], LassoNet [9], etc. These models can learn the nonlinear relationship using nonlinear activation functions and handle multi-class data using one-hot-encoding class labels implemented on a single model. Besides, deep neural networks with multiple nonlinear hidden layers learn high-level features, making them useful to represent complex nonlinear structures [107]. In DFS and Group Sparse ANN, the authors used ℓ_1 to promote feature sparsity. Note that the ℓ_1-norm is not differentiable, and the authors implemented symbolic differentiation using the Theano package [108] to calculate the gradient with stochastic optimization. In recent work, Yamada et al. [10] reported that DFS and Group Sparse ANN failed to induce sparsity on several bench-marking feature selection data sets. However, the authors didn't investigate the root cause.

Unlike DFS and Group Sparse ANN, which use non-differentiable ℓ_1-norm, the cost function of CAE [105] is differentiable. Note, unlike the ℓ_1-norm, which selects a subset from the original feature set, the concrete feature selector picks a user-specified number of features. Also, Stochastic Gates [10] approximates ℓ_0-norm by a continuous relaxation of the Bernoulli distribution. Recent articles on Neural Network-based models, CAE, Stochastic Gate, and FsNet, address the non-differentiable aspect of ℓ_1 and provide a new direction to induce feature sparsity. But notice that all of these techniques are optimized on neural network architecture, making them nonconvex models. Training these models will produce non-unique features as a consequence of local minima, and the features may not be optimal for classifying test data. Each different run will potentially produce a different set of features, and the current literature doesn't address how to select a single set of robust features using neural network architecture.

Chapter 3

Supervised Visualization using Centroid-Encoder

Dimensionality reduction techniques such as PCA and its nonlinear extensions, e.g., Kernel PCA and Autoencoders, have proven extremely useful for understanding data in high dimensions, particularly as visualization tools. Here we address the question of how such devices can be adapted to the case where class label information is available, i.e., supervised data-dimensionality reduction. More specifically, this chapter concerns integrating class label information to the non-linear autoencoder reduction process for data visualization. Autoencoder neural networks are the starting point for our approach and are described briefly in the next section.

3.1 Autoencoder

An autoencoder is a dimension reducing mapping that has been optimized to approximate the identity on a set of training data [37, 109, 110]. The mapping is modeled as the composition of a dimension reducing mapping g followed by a dimension increasing reconstruction mapping h, i.e., $f(x) = h(g(x))$ where the *encoder* g is represented $g : U \in \mathbb{R}^n \to V \in \mathbb{R}^m$ and the *decoder* h is represented $h : V \in \mathbb{R}^m \to U \in \mathbb{R}^n$. The composition of g and h, and hence f, is accomplished by minimizing the unconstrained cost

$$L_A(D; \theta) = \frac{1}{2N} \sum_{i=1}^{N} \|x_i - f(x_i; \theta))\|_2^2 \tag{3.1}$$

where $D = \{x_i\}$ is the data and θ represents all the unknown parameters, i.e., weights and biases, in the network. Hence, the autoencoder learns a function f such that $f(x_i; \theta) \approx x_i$ where the model parameters θ are found by solving the minimization problem. The encoder map g takes the input x_i and maps it to a latent representation $y_i = g(x_i) \in V \subset \mathbb{R}^m$. The decoder ensures that the encoder is faithful to the data point, i.e., it serves to reconstruct the reduced point to its original state $x_i \approx h(y_i)$.

The cost L_A is minimized iteratively by backpropagating [111, 112] the gradient w.r.t. θ. Applying chain rule on Equation 3.1 gives us

$$\frac{\partial L_A}{\partial \theta_k} = \sum_k \sum_l \frac{\partial L_A}{\partial f_l} \frac{\partial f_l}{\partial \theta_k} \tag{3.2}$$

where the term $\frac{\partial L_A}{\partial f_l}$ is known as output error/delta (Δ) which is readily calculated to be

$$\Delta_A = \frac{1}{N} \sum_{i=1}^{N} (f(x_i, \theta) - x_i) \tag{3.3}$$

The term $\frac{\partial f_l}{\partial \theta_k}$ is the $(lk)^{th}$ element of the Jacobian matrix of f. The details for using multi-layer perceptrons for training autoencoders can be found in [37].

3.2 Centroid-Encoder

Here we propose a form of supervised autoencoder that exploits label information. Let $D \in \mathbb{R}^{n \times N}$ is the data matrix where N is the total number of samples and n is the dimension of each sample $x_i \in \mathbb{R}^n$. Assume D has M classes $\{C_j\}_{j=1}^{M}$ where the index set of class C_j is denoted by I_j. We define centroid of each class as

$$c_j = \frac{1}{|C_j|} \sum_{i \in I_j} x_i$$

where $|C_j|$ is the cardinality of class C_j. The centroid-encoder is trained by determining a mapping $f = h \circ g$ with input-output pairs

$$\{x_i, c_j\}_{i=1}^{N}$$

where c_j is the target center for data point x_i, i.e., $i \in I_j$. Thus, unlike autoencoder, which maps each point x_i to itself, centroid-encoder will map each point x_i to its class centroid c_j such that $c_j \approx f(x_i)$ while passing the data through a bottleneck layer of dimension m to provide the reduced point $g(x_i)$. The cost function of centroid-encoder is defined as

$$L_C(D;\theta) = \frac{1}{2N}\sum_{j=1}^{M}\sum_{i\in I_j}\|c_j - f(x_i;\theta))\|_2^2 \tag{3.4}$$

where θ represents all the unknown parameters, i.e., weights and biases, in the network. This cost is also referred to as the *distortion error* [113]. Like an autoencoder, the cost is minimized by iteratively updating the parameters (θ) by error backpropagation. We can connect this to the learning procedure for autoencoder by applying chain rule on Equation 3.4

$$\frac{\partial L_C}{\partial \theta_k} = \sum_k\sum_l \frac{\partial L_C}{\partial f_l}\frac{\partial f_l}{\partial \theta_k} \tag{3.5}$$

where the term $\frac{\partial f_l}{\partial \theta_k}$ is identical to an autoencoder and the output delta/error which is being back-propagated is readily calculated to be

$$\Delta_C = \frac{1}{N}\sum_{j=1}^{M}(\sum_{i\in I_j} f(x_i;\theta) - c_j) \tag{3.6}$$

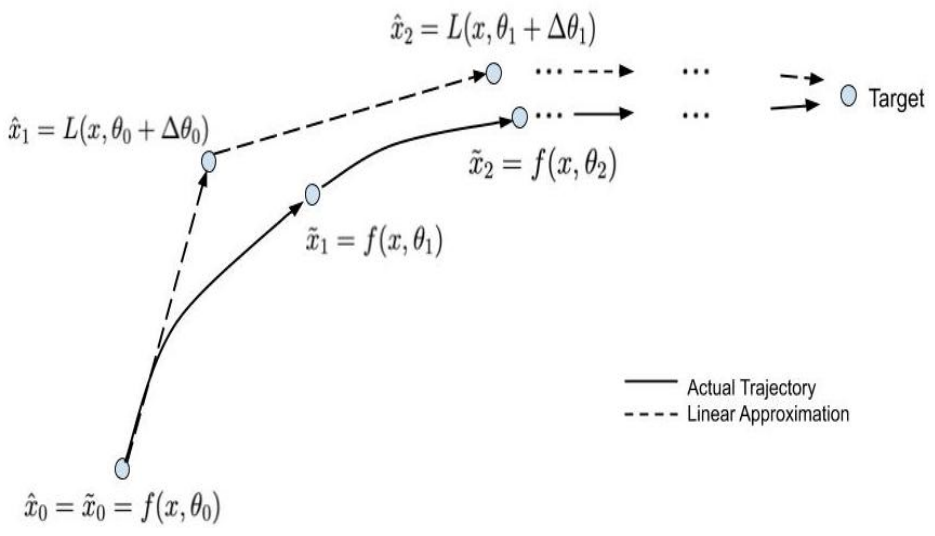
Note, the terms Δ_A and Δ_C capture the difference between auto-encoder and centroid-encoder; the gradient vector g is the same in both cases; see, e.g., [37, 114] for details on the computation of the gradient g.

3.3 Convergence Proof of Centroid-Encoder

We want to minimize the centroid-encoder cost using iterative gradient descent method such that the cost at iteration $k+1$ is less than the cost at iteration k, i.e, $E_{k+1} < E_k$. Let θ_0 is the initial set of parameters which is set randomly. The output of the map $\tilde{x}_0$ is defined as:

$$\tilde{x}_0 := f(x;\theta_0) \tag{3.7}$$

where x is the input to the centroid-encoder. The parameter set θ_0 is updated using back-propagation



$\hat{x}_2 = L(x, \theta_1 + \Delta\theta_1)$
$\hat{x}_1 = L(x, \theta_0 + \Delta\theta_0)$
Target
$\tilde{x}_2 = f(x, \theta_2)$
$\tilde{x}_1 = f(x, \theta_1)$
Actual Trajectory
Linear Approximation
$\hat{x}_0 = \tilde{x}_0 = f(x, \theta_0)$

Figure 3.1: Original trajectory of the of centroid-encoder along with the linear approximation.

which results in the updated parameter set θ_1. This gives a new output $\tilde{x}_1$ as shown in Figure 3.1. The point $\tilde{x}_1$ can be approximated from the initial parameter θ_0 using Taylor series expansion. Let $\hat{x}_0$ is the initial point of the approximation, i.e. $\hat{x}_0 = \tilde{x}_0$.

$$\hat{x}_0 = \tilde{x}_0 = f(x; \theta_0) \tag{3.8}$$

Using Taylor series one can approximate $\tilde{x}_1$ as given below:

$$\hat{x}_1 := f(x; \theta_0 + \Delta\theta_0) \tag{3.9}$$

$$\hat{x}_1 = f(x; \theta_0) + J_{\theta_0}\Delta\theta_0 \tag{3.10}$$

$$\hat{x}_1 = \hat{x}_0 + J_{\theta_0}\Delta\theta_0 \tag{3.11}$$

where J_{θ_0} is known as the *Jacobian matrix* or *derivative matrix* of the function f. Note, that we have ignored the higher order polynomial terms (degree $>= 2$) in the Taylor series expansion.

Hence $\hat{x}_1$ is a linear approximation of $\tilde{x}_1$. Let,

$$\Delta\hat{x}_0 := J_{\theta_0}\Delta\theta_0. \tag{3.12}$$

Therefore Equation 3.11 becomes,

$$\hat{x}_1 = \hat{x}_0 + \Delta\hat{x}_0 \tag{3.13}$$

We can use Equation 3.12 in the context of centroid-encoder to calculate the $\Delta\hat{x}_0$. Now use Equation 3.5 where $\frac{\partial L_C}{\partial\theta_i}$ corresponds to $\Delta\theta_0$ in Equation 3.12. Writing the above equation in terms of Jacobian matrix gives us:

$$\frac{\partial L_C}{\partial\theta_i} = J_\theta^T \sum_j \frac{\partial L_C}{\partial f_j} \tag{3.14}$$

Here J_θ is a $K \times M$ Jacobian matrix, where K is the number of nodes in the output layer and M is the total number of parameters (weights and biases). Substituting the value of $\Delta\theta_0$ in Equation 3.12 gives us:

$$\Delta\hat{x}_0 = J_{\theta_0}J_{\theta_0}^T \sum_j \frac{\partial L_C}{\partial f_j} \tag{3.15}$$

Replacing $\sum_j \frac{\partial L_C}{\partial f_j}$ by $(\hat{x}_0 - c)$ (see Equation 3.6) gives us:

$$\Delta\hat{x}_0 = J_{\theta_0}J_{\theta_0}^T(\hat{x}_0 - c) \tag{3.16}$$

The Jacobian contains the partial derivatives of a function and these derivatives point in the direction of the maximum increase of the function. As we are minimizing the cost of centroid-encoder, we should consider the direction of maximum decrease. For a sufficiently small $\epsilon_0 > 0$, the change in $\Delta\hat{x}_0$ can be written as:

$$\begin{aligned}\Delta\hat{x}_0 &= -\epsilon_0 J_{\theta_0}J_{\theta_0}^T(\hat{x}_0 - c)\\ \Delta\hat{x}_0 &= -\epsilon_0 Q_{\theta_0}(\hat{x}_0 - c)\end{aligned} \tag{3.17}$$

where $Q_{\theta_0} = J_{\theta_0} J_{\theta_0}^T$ is a PSD (positive semi-definite) matrix. Let's consider the square distance of $\hat{x}_1$ from its class center c

$$\|c - \hat{x}_1\|_2^2 = (c - \hat{x}_1)^T (c - \tilde{x}_1) \tag{3.18}$$

$$\|c - \hat{x}_1\|_2^2 = c^T c - 2c^T \hat{x}_1 + \hat{x}_1^T \hat{x}_1 \tag{3.19}$$

Substituting the value of $\hat{x}_1$ from Equation 3.13,

$$\begin{aligned} \|c - \hat{x}_1\|_2^2 = c^T c - 2c^T(\hat{x}_0 + \Delta\hat{x}_0) + \\ (\hat{x}_0 + \Delta\hat{x}_0)^T(\hat{x}_0 + \Delta\hat{x}_0) \end{aligned} \tag{3.20}$$

from which it follows

$$\|c - \hat{x}_1\|_2^2 = \|c - \hat{x}_0\|_2^2 + 2(\hat{x}_0 - c)^T \Delta\hat{x}_0 + \Delta\hat{x}_0^T \Delta\hat{x}_0 \tag{3.21}$$

Putting the value of $\Delta\hat{x}_0$ from Equation 3.17, gives us:

$$\begin{aligned} \|c - \hat{x}_1\|_2^2 = \|c - \hat{x}_0\|_2^2 - 2\epsilon_0(\hat{x}_0 - c)^T Q_{\theta_0}(\hat{x}_0 - c) + \\ {\epsilon_0}^2(\hat{x}_0 - c)^T Q_{\theta_0}^2(\hat{x}_0 - c) \end{aligned} \tag{3.22}$$

If we choose ϵ_0 as given below:

$$\frac{2(\hat{x}_0 - c)^T Q_{\theta_0}(\hat{x}_0 - c)}{(\hat{x}_0 - c)^T Q_{\theta_0}^2(\hat{x}_0 - c)} > \epsilon_0 \tag{3.23}$$

then Equation 3.22 becomes:

$$\|c - \hat{x}_1\|_2^2 = \|c - \hat{x}_0\|_2^2 - \kappa^2 \tag{3.24}$$

where $\kappa^2 > 0$. Equation 3.24 implies that:

$$\|c - \hat{x}_1\|_2^2 < \|c - \hat{x}_0\|_2^2 \; for \; \frac{2(\hat{x}_0 - c)^T Q_{\theta_0}(\hat{x}_0 - c)}{(\hat{x}_0 - c)^T Q_{\theta_0}^2(\hat{x}_0 - c)} > \epsilon_0 \tag{3.25}$$

Similarly we can write:

$$\|c - \hat{x}_2\|_2^2 < \|c - \hat{x}_1\|_2^2 \; for \; \frac{2(\hat{x}_1 - c)^T Q_{\theta_1}(\hat{x}_1 - c)}{(\hat{x}_1 - c)^T Q_{\theta_1}^2(\hat{x}_1 - c)} > \epsilon_1 \tag{3.26}$$

and

$$\|c - \hat{x}_{k+1}\|_2^2 < \|c - \hat{x}_k\|_2^2 \; for \; \frac{2(\hat{x}_k - c)^T Q_{\theta_k}(\hat{x}_k - c)}{(\hat{x}_k - c)^T Q_{\theta_k}^2(\hat{x}_k - c)} > \epsilon_k \tag{3.27}$$

Therefore

$$\|c - \hat{x}_0\|_2^2 > \|c - \hat{x}_1\|_2^2 > \|c - \hat{x}_2\|_2^2 ... > \|c - \hat{x}\|_2^2 > \\ \|c - \hat{x}_{k+1}\|_2^2$$

Therefore the sequence $\{\hat{x}_k\}$ converges to c completing the proof.

3.4 The Centroid-Encoder Training Algorithm

The training is very similar to that of an autoencoder and the steps are described in Algorithm 1. As with autoencoders, the centroid-encoder is a composition of two maps $f(x) = (h \circ g)(x)$. For visualization, we train a centroid-encoder network using a bottleneck architecture, meaning that the dimension of the image of the map g is 2 or 3.

3.5 Pre-training Centroid-Encoder

Here we describe the pre-training strategy of centroid-encoder. We employ a technique we refer to as *pre-training with layer-freeze*, i.e., pre-training is done by adding new hidden layers while mapping samples of a class to its centroid; see Figure 3.2. In this approach, weights of

Algorithm 1: Supervised Non-linear Centroid-Encoder (without pre-training).

```
Input: Labeled data (D) = $\{x^i\}_{i=1}^N$ with M classes, $I_j$ index set of class $C_j$. User defined
        parameters: error tolerance $\tau$, learning rate $\mu$, bottleneck dimension $m$.
Output: Bottleneck output $y^i = g(x^i)$; network parameters $\theta$.
Result: Non-linear embedding of data in $m$ dimensions.
Initialization: Class centroids $c_j = \frac{1}{|C_j|}\sum_{i \in I_j} x^i, j = 1, \ldots, M$. Iteration $t \leftarrow 0$.
        Partition D into training (Tr) and validation set (V).
1 while $|L_C^{t+1}(V)$ - $L_C^t(V)| > \tau$ do
2     Compute the loss/error $L_C(V)$ and $L_C(Tr)$ using Equation 3.4
3     Compute backpropogation error $\Delta_C$ using Equation 3.6 on training set Tr.
4     Update model parameters $\theta$ by $\theta^{t+1} = \theta^t - \mu\Delta_C.\mathbf{g}^t$
5 end
```

hidden layers are learned sequentially. Initially $\theta = (W_1, \tilde{W}_1)$ and by the end of training $\theta = (W_1, \ldots, W_n, \tilde{W}_n, \ldots, \tilde{W}_1)$ using our θ notation for the set of unknown parameters. The algorithm starts by learning the parameters of the first hidden layer using standard error backpropagation [111]. Then the second hidden layer is added in the network with weights initialized randomly. At this point, the associated parameters of the first hidden layer are kept frozen. Now the parameters of the second hidden layer are updated using backpropagation. We repeat this step until pre-training is done for each hidden layer. Once pre-training is complete, we do an end-to-end fine-tuning by updating the parameters of all the layers at the same time. We comment that our pre-training approach is different than the greedy layer-wise pre-training proposed by [103] and [110,115]. The unsupervised pre-training of Hinton et al. and Bengio et al. uses the activation of the l^{th} hidden layer as the input for the $(l+1)^{th}$ layer. In our approach, we use the original input-output pair (x^i, c_j) to pre-train each hidden layer. As we use the labels to calculate the centroids c_j, so our pre-training is supervised.

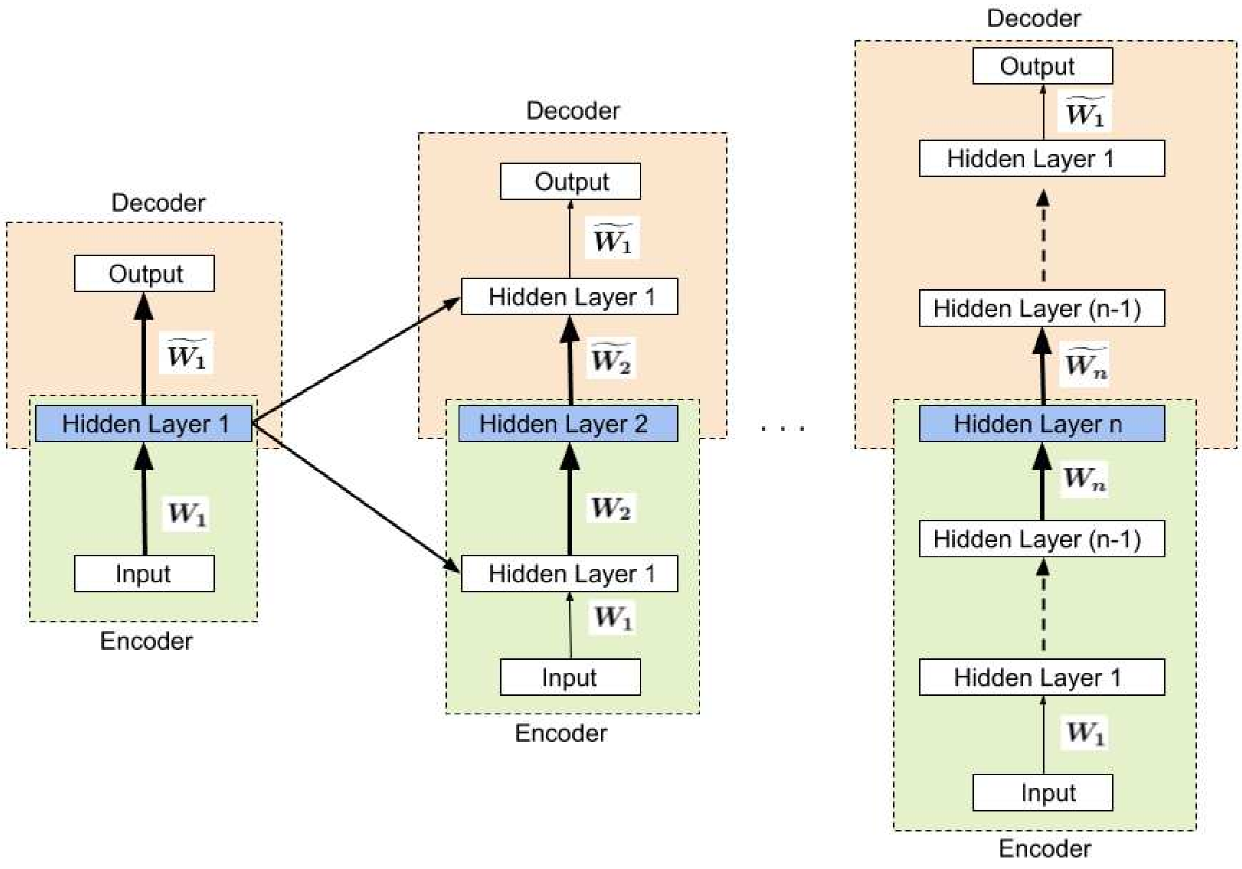


Figure 3.2: Pre-training a deep centroid-encoder by layer-freeze approach. In the first step, a centroid-encoder with the first hidden layer is pre-trained (left diagram). The pre-trained weights are $(W_1, \widetilde{W_1})$. In the next step, a new hidden layer is added by extending the network architecture. After that, the weights $(W_2, \widetilde{W_2})$ associated with the new hidden layer are updated while keeping the other weights $(W_1, \widetilde{W_1})$ fixed. This process is repeated to add more hidden layers.

3.6 Visualization Experiments

To evaluate centroid-encoder we select three suites of data sets from the literature and run three bench-marking experiments comparing centroid-encoder with a range of other supervised dimension reduction techniques. To objectively compare the performance of these supervised models we employ a standard class prediction error on the two-dimensional visualization domain.[1] The classification error is defined as

$$Error\ (\%) = \frac{100}{N} \sum_{i=1}^{N} I[l_i \neq f(\tilde{x}_i)] \tag{3.28}$$

[1]This corresponds to the bottleneck layer for CE.

Here N is total number of test samples, l_i is the true label of the i^{th} test sample, and f is a classification function which returns the predicted label of the embedded test sample $\tilde{x_i}$. Here I denotes the indicator function. Following the supervised visualization literature [2, 3, 5, 64], we chose the k-nearest neighbor (k-NN) as the classification function (f). All results share the following workflow:

- Select a small data set from the training set and run 10-fold cross validation to determine the network architecture and hyper-parameters
- Using this architecture train K models on different data partitions as described below
- Using the sequestered test set compute average k-NN ($k = 5$) classification errors on the 2D representation with standard deviations

We follow this common evaluation strategy in the three bench-marking experiments so that our CE results are comparable to the published results. We describe details of each experiment including how the training and test partitions were selected below. In addition to prediction error, we also visualize the two-dimensional embedding of test samples of each model to do a subjective comparison. We fix the reduction dimension as 2 in all networks for our visualization application.

3.6.1 Data Sets

Here we provide a brief description of the eight data sets used in visualization and classification experiments.

MNIST Digits: This is a widely used collection of digital images of handwritten digits (0..9)[2] with separate training (60,000 samples) and test set (10,000 samples). Each sample is a grey level image consisting of 1-byte pixels normalized to fit into a 28 x 28 bounding box resulting in *vecced* points in $\mathbb{R}^{784}$.

[2]The data set is available at http://yann.lecun.com/exdb/mnist/index.html.

USPS Data: A data set of handwritten digits $(0 \ldots 9)$[3], where each element is a 16x16 image in gray-scale resulting in *vecced* points in $\mathbb{R}^{256}$. Each of the ten classes has 1100 digits for a total of 11,000 digits.

Phoneme Data: This data set is created from Finish speech recorded continuously from the same speaker. A 20-dimensional vector represents each phoneme. The data set consists of 3924 number of samples distributed over 13 classes. The data set is taken from the LVQ-PAK[4] software package.

Letter Recognition Data: Available in the UCI Machine Learning Repository [116], this data set is comprised of 20,000 samples where each of them represents an upper case letter of the English alphabet A to Z selected from twenty different fonts. Each sample is randomly distorted to produce a unique representation that is converted to 16 numerical values and 26 classes.

Landsat Satellite Data: This data set consists of satellite images with four different spectral bands. Given the pixel values of a 3x3 neighborhood, the task is to predict the central pixel of each 3x3 region. Each sample consists of 9-pixel values from 4 different bands, which makes the dimension 36. There are six different classes and 6435 samples. The data set is also available in UCI Machine Learning Repository [116].

Iris Data: The UCI Iris data set [116] is comprised of three classes and is widely used in the Machine Learning literature. Each class has 50 samples and, each sample has four features.

Sonar Data: This UCI data set [116] has two classes: mine and rock. Each sample is represented by a 60-dimensional vector. The mine class has 111 patterns, whereas the rock category has 97 samples.

SUSY Data: It is a high-energy particle physics data set with 4.5 million samples in the training set and 0.5 million samples in the test set. Each data point has 18 features, where the first 8 represent kinematic properties measured by the particle detectors, and the last ten features are derived by the Physicist from the kinematic properties. The goal is to determine when supersym-

[3]The data set is available at https://cs.nyu.edu/~roweis/data.html.

[4]The software is avilable at http://www.cis.hut.fi/research/software.

metric particles are produced from the background process where no particles are detected. The data set is available in UCI repository [116].

3.6.2 Experimental Details

Here we present the details of the bench marking experiments. Our goal is to compare centroid-encoder with a wide range of other methods to establish its relative performance attributes. To this end, we picked the published results from three papers [3, 5, 50] for benchmarking. For an apples-to-apples comparison, we followed the same experimental methodology as described in those papers. This approach permitted a direct comparison to deep neural network-based models (deep-MCML and deep-NCA), kernel methods, and non-linear neighborhood preserving models using the authors' best results. It is an implicit assumption in this approach that the authors of these papers have carefully optimized their work, and hence these results are most useful for comparison.

Experiment 1: The first bench-marking experiment is conducted on the widely studied MNIST and USPS data sets. We compared CE with the following methods: autoencoder, non-linear NCA [66], supervised UMAP [59], GerDA [1], HOPE [4], parametric t-SNE [2], t-distributed NCA [3], t-distributed MCML [3], and supervised UMAP. All MNIST models were trained using the entire training set and evaluated on the standard test set. For USPS, we followed the strategy in [3], where we randomly split the entire data set into a training set of 8000 samples and a test set consisting of 3000 samples. We repeat the experiments $K = 10$ times and report the average error rate with standard deviation. To determine the model architecture in each case we took a subset from the training data (3000 and 30,000 for USPS and MNIST respectively), picked randomly, and ran 10-fold cross-validation. We implemented non-linear-NCA and Autoencoders in Python, and we used the scikit-learn [117] package to run supervised UMAP. For the rest of the methods, we took the published results for comparison.

Experiment 2: We conducted the second experiment on Letter, Landsat, and Phoneme data sets. For evaluation, we compared CE with the following techniques: supervised neighbor retrieval visualizer (SNeRV) by [5], multiple relational embedding (MRE) by [118], colored maximum vari-

Table 3.1: Network topology used for CE on various data sets. The number d is the input dimension of the network and is data set dependent.

Dataset	Network topology	Activation
MNIST	$d \to 1000 \to 500 \to 125 \to 2$	tanh
USPS	$d \to 2000 \to 1000 \to 500 \to 2$	relu
Phoneme	$d \to 250 \to 150 \to 2$	relu
Letter	$d \to 250 \to 150 \to 2$	relu
Landsat	$d \to 250 \to 150 \to 2$	relu
Iris	$d \to 100 \to 2$	relu
Sonar	$d \to 500 \to 250 \to 2$	relu
SUSY	$d \to 500 \to 250 \to 125 \to 2$	relu

ance unfolding (MUHSIC) by [63], supervised isomap (S-Isomap) by [73], parametric embedding (PE) by [65], and neighborhood component analysis (NCA) by [64] and supervised UMAP. Following the experimental setup in [5], we randomly selected 1500 samples from each data set and ran 10-fold cross-validation ($K = 10$). Except for UMAP and supervised UMAP, we used the published results in [5]. We picked the architecture (for CE and AE) and other model specific hyper-parameters by running 10-fold internal cross-validation on the training set.

Experiment 3: For the third experiment, we compared the performance of centroid-encoder with supervised PCA (SPCA), and kernel supervised PCA (KSPCA) on the Iris, Sonar, and a subset of USPS data set. Following [50], test error rates were obtained by averaging over $K = 25$ runs on randomly generated 70/30 splits of each data set. For USPS, we randomly picked 1000 cases and used that subset for our experiment, as done by [50]. Model specific hyper-parameters are tuned using 10-fold internal cross-validation on the training set. We implemented SPCA and KSPCA in Python.

3.6.3 Implementation

We have implemented centroid-encoder in PyTorch to run on GPUs. To train CE with an optimal number of epochs, we used 10% of training samples as a validation set in all of our visualization experiments. We measure the generalization error on the validation set after every training epochs. Once the validation error doesn't improve, we stop the training. Finally, we merge the validation set with the training samples and train the model for additional epochs. The model parameters are updated using Adam optimizer [49]. Table 3.1 provides the network architecture used in training of CE on various data sets. The implementation is available at: https://github.com/Tomojit1/Centroid-encoder/tree/master/GPU. Apart from centroid-encoder, we have implemented autoencoder (AE), non-linear NCA (NNCA), supervised PCA (SPCA) and kernel supervised PCA (KSPCA). These models require the tuning of hyper-parameters as listed in Table 3.2.

Table 3.2: Hyper-parameters for different models.

Model	Hyper parameter	Range of Values
CE, AE	$learning\_rate$	0.1, 0.01, 0.001, 0.0001, 0.0002, 0.0004, 0.0008
	$mini\_batch\_size$	16, 25, 32, 50, 64, 75,128, 256, 512, 1024
	$weight\_decay$	0.001, 0.0001, 0.00001, 0.00002, 0.00004, 0.00008
SUMAP	$n\_neighbors$	5, 10, 20, 40, 80
	min_dist	0.0125, 0.05, 0.2, 0.8
KSPCA	$kernel\_parameter(\gamma)$	0.001, 0.01, 0.025, 0.035, 0.05, 0.1, 1.0, 2.5, 3.0, 5.0, 7.5, 10.0

3.7 Results

Now we present the results from a comprehensive quantitative and qualitative analysis across these diverse data sets. Note that the primary objective of our assessment is to determine the quality of information obtained from visualization, including class separability, data scatter, and neighborhood structure. Since this assessment is by its very nature subjective, we also employ label information to determine classification rates that provide additional quantitative insight into the data reduction for visualization. Computational expense is also an essential factor, and we will see that this is a primary advantage of CE over other methods when the visualizations and error rates are comparable.

3.7.1 MNIST

Table 3.3: Error rates (%) of k-NN (k=5) on the 2D embedded data by various dimensionality reduction techniques trained with pre-training. Results of GerDA and pt-SNE are taken from [1] and [2] correspondingly. Error rates of the variants of NCA and MCML are reported from [3]. NA indicates that the result was not reported in the original source.

Method	Dataset	
	MNIST	USPS
Centroid-Encoder	2.61 ± 0.09	2.91 ± 0.31
Supervised UMAP	3.45 ± 0.03	6.17 ± 0.23
NNCA	4.71 ± 0.57	6.58 ± 0.80
Autoencoder	22.04 ± 0.78	$16.49 \pm .91$
dt-MCML	**2.03**	**2.46 ± 0.35**
dG-MCML	2.13	3.37 ± 0.18
GerDA	3.2	NA
dt-NCA	3.48	5.11 ± 0.28
dG-NCA	7.95	10.22 ± 0.76
pt-SNE	9.90	NA

As shown in Tables 3.3 (results with pre-training) and 3.4 (results without pre-training), the models with relatively low error rates for the MNIST data amongst our suite of visualization methods are dt-MCML, dG-MCML, centroid-encoder, and HOPE. With pre-training, the error rate of CE is comparable to the top-performing model dt-MCML and superior to NNCA and supervised

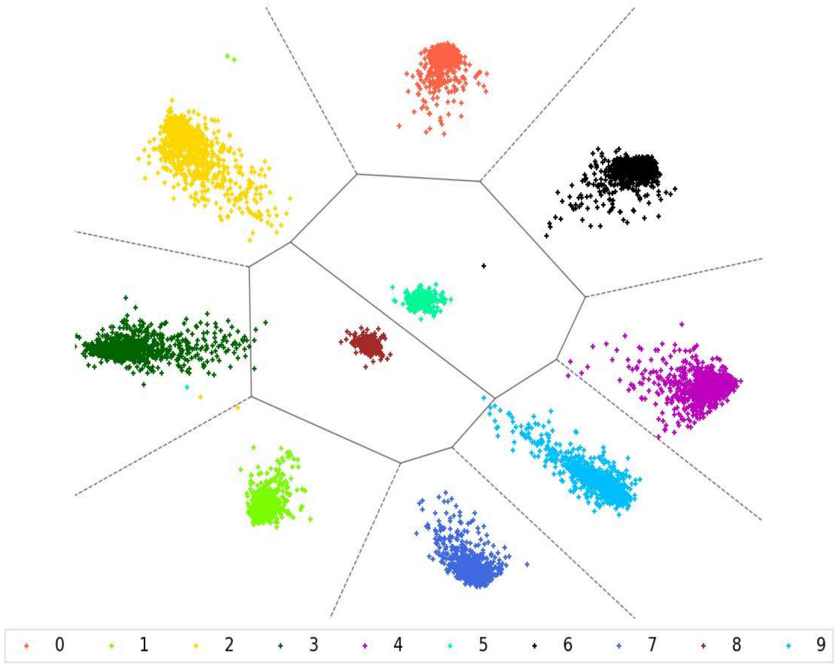
UMAP by a margin of 2.1% and 0.84%, respectively. Note that methods in the MCML and NCA families require the computation of distance matrix over the data set, making them significantly more expensive than CE, which only requires distance computations between the data of a class and its center. It's noteworthy that the error of CE is lower than the other non-linear variants of NCA: dt-NCA and dG-NCA. Among all the models, parametric t-SNE (pt-SNE) and autoencoder exhibit the worst performance with error rates on MNIST data are 9.90% and 22.04%, respectively. These high error rates are not surprising given these methods do not use label information.

The prediction error of CE with pre-training is relatively low at 2.6%, as shown in Table 3.3 behind the more computationally expensive dt-MCML and dG-MCML algorithms. With no pre-training, the numeric performance of centroid-encoder is statistically equivalent to dt-MCML and HOPE.

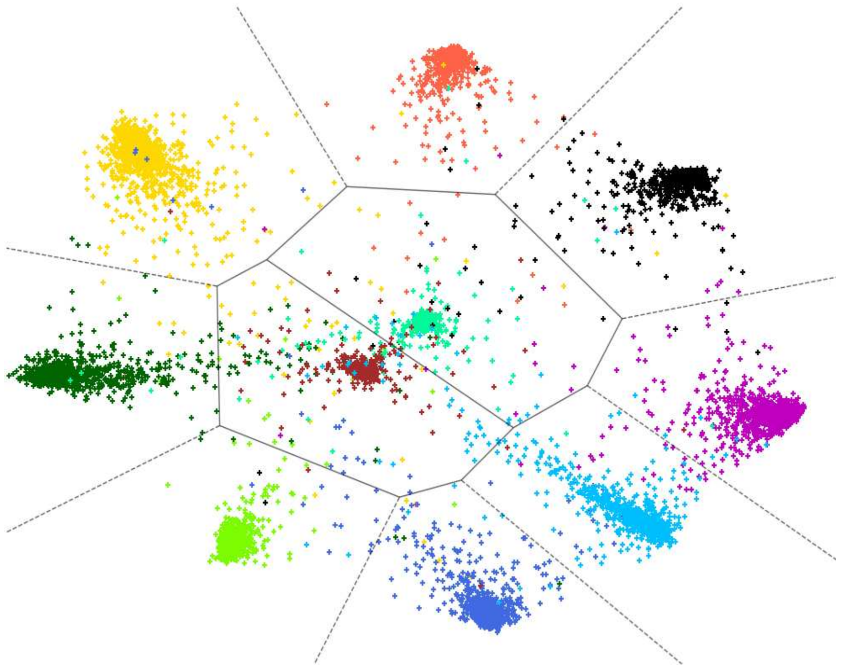
Table 3.4: Error rates ($\%$) of k-NN (k=5) on the 2-dimensional data by various techniques trained without pre-training. Error rate of HOPE, dt-NCA and dt-MCML are reported from [4].

Method	Dataset	
	MNIST	USPS
Centroid-Encoder	**3.17 ± 0.24**	**2.98 ± 0.67**
Autoencoder	21.55 ± 0.47	15.17 ± 0.85
HOPE	3.20	3.03
dt-MCML	3.35	4.07
dt-NCA	3.48	5.11

While the prediction error is essential as a quantitative measure, the visualizations reveal information not encapsulated in this number. The neighborhood relationships established by the embedding provide potentially valuable insight into the structure of the data set. The two-dimensional centroid-encoder visualization of the MNIST training and test sets are shown in Figure 3.3. The entire 10,000 test samples are shown in b) while only a subset of the training data set, 1000 digits picked randomly from each class, is shown in a). The separation among the ten classes is easily visible in both training and test data, although there are few overlaps among the categories in test



(a) Visualuzation of 1000 digits per class from MNIST training set.



(b) Visualization of the 10,000 MNIST test samples.

Figure 3.3: Voronoi cells in 2D of the MNIST data using centroid-encoder. The network architecture of $784 \rightarrow [1000, 500, 125, 2, 125, 500, 1000] \rightarrow 784$ is employed to map the data onto the 2D space. The centroid of the training samples mapped to 2D are used to form the Voronoi regions for each digit class.

data. Consistent with the low error rate, the majority of the test digits are assigned to the correct Voronoi cells.

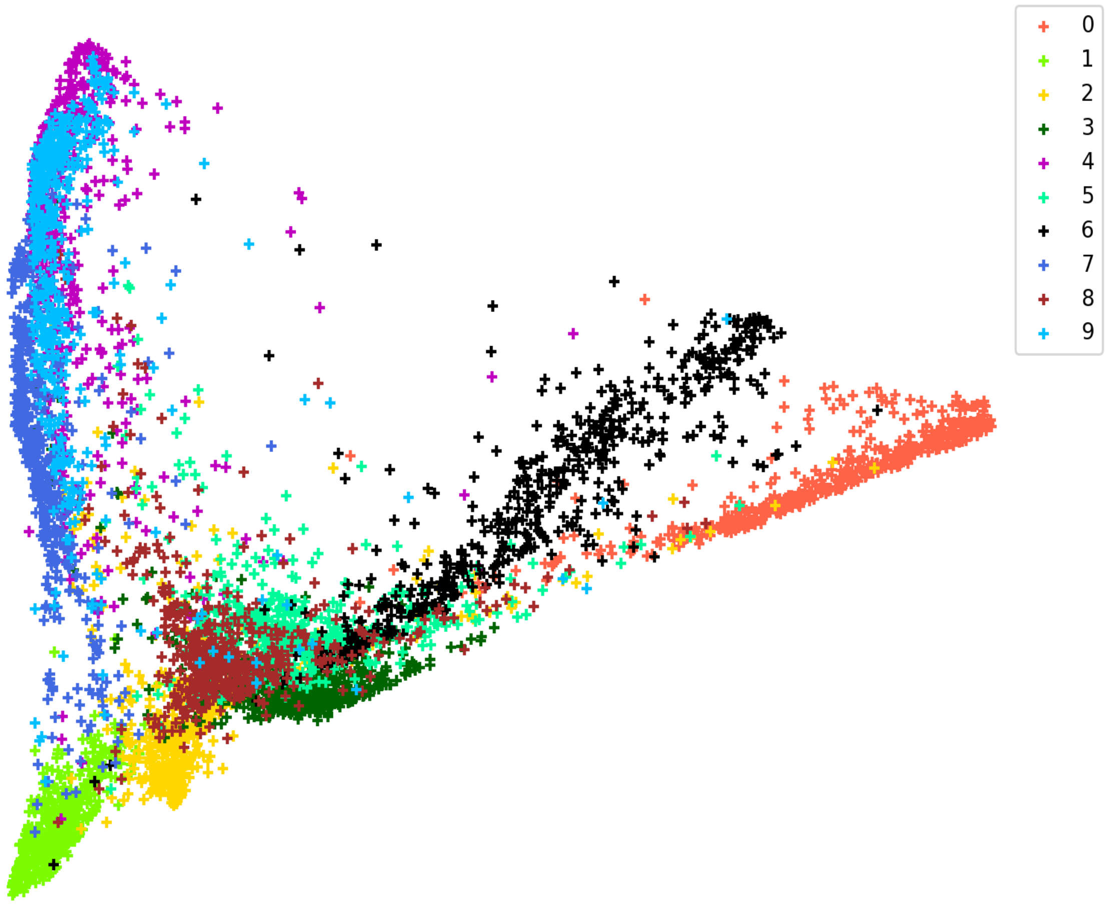


Figure 3.4: Two dimensional visualization of 10,000 MNIST test digits by Laplacian Eigenmap.

Now let's consider the visualization from a neighborhood relationship perspective. This comparison aims to verify whether CE can preserve class neighborhood structure without calculating the neighborhood graph. We compare the results with those of Laplacian Eigenmaps (LE), as shown in Figure 3.4, an unsupervised manifold learning spectral method that solves an optimization problem preserving neighborhood relations [12]. Both CE and LE place digits 5 and 8 as neighbors at the center, allowing us to view all digits as being perturbations of these numbers. Digits 7, 9, and 4 are collapsed to one region; in contrast, CE is built to exploit label information and separates these neighboring digits. LE clumps 7, 9, 4 next to 1 as well, consistent with the CE visualization. The digit 0 neighbors 6 for both CE and LE, but LE significantly overlaps the 6 with the digit 5.

Figure 3.5 shows the two-dimensional arrangement of 10,000 MNIST test set digits using non-linear NCA, supervised UMAP, UMAP, autoencoder, and t-SNE. It is apparent that the embedding of CE is more similar to LE in terms of the relative positioning of the digits in 2D than these other visualization methods. There are some similarities across all the models, e.g., 0 and 6 are neighbors for all methods (except for SUMAP), as are the digits 4, 9, and 7. The separation among the different digit groups more prominent in CE than all the methods save for SUMAP, but as seen above, quantitatively, SUMAP has a higher error rate than CE. We note that unsupervised UMAP, as well as t-SNE, both agree with CE that the digits 5 and 8 should be in the center. However, in contrast, CE provides a mapping that can be applied to streaming data.

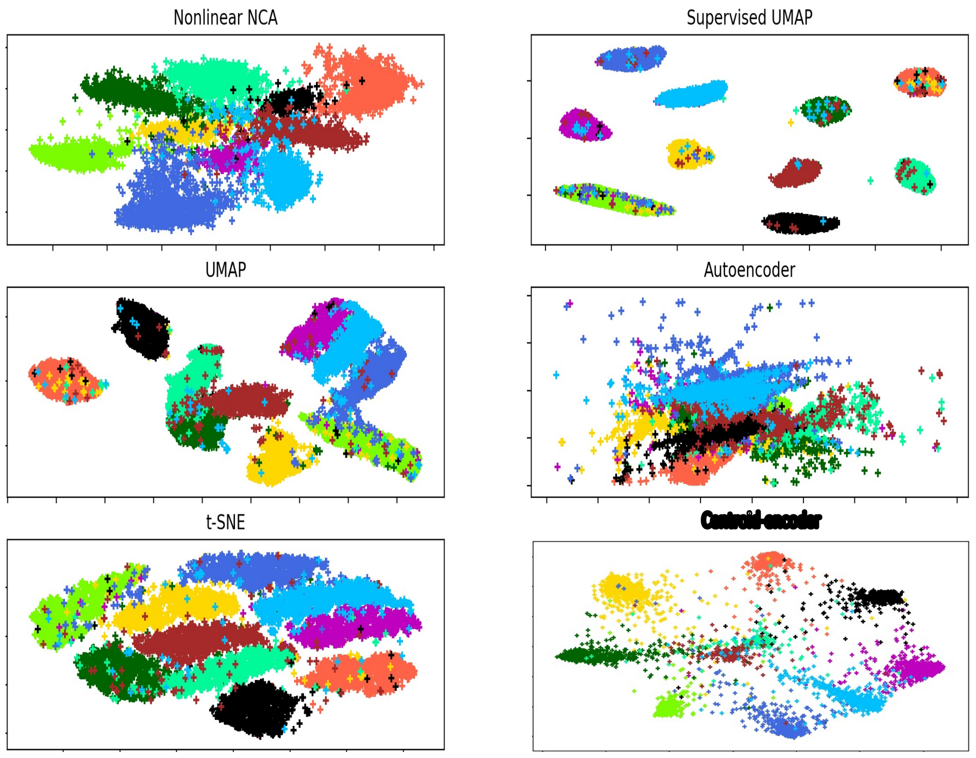


Figure 3.5: A comparison of the visualizations of 10,000 MNIST test digits by different methods.

3.7.2 USPS

On USPS, centroid-encoder without pre-training has a prediction error slightly lower than HOPE but outperformed dt-MCML and dt-NCA by the margins of 1.09% and 2.12%, respectively, see Table 3.4. Without pre-training, the centroid-encoder performed better than NNCA with pre-training and supervised-UMAP on both MNIST and USPS datasets.

On the USPS data, the top three models with pre-training are dt-MCML, centroid-encoder, and dG-MCML with the error rates of 2.46%, 2.91%, and 3.37%, respectively. The error rate of centroid-encoder is better than NNCA and supervised-UMAP by a margin of 3.67% and 3.26%, respectively. Like in MNIST, dt-NCA and dG-NCA performed relatively poorly compared to centroid-encoder. Again, autoencoder performed the worst among all the models.

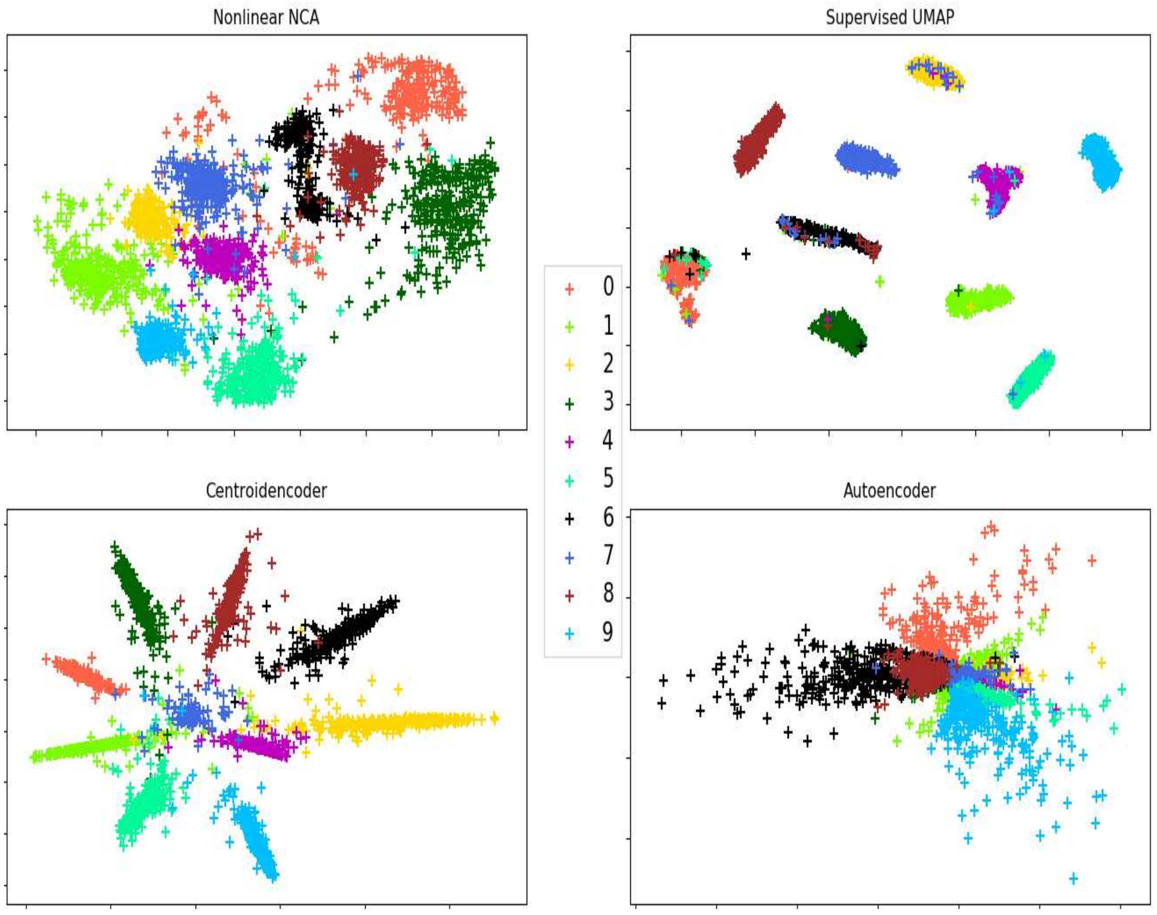


Figure 3.6: Two dimensional plot of 3000 USPS test digits by different dimensionality reduction methods.

In Figure 3.6, we show the two-dimensional visualization of 3,000 test samples using non-linear NCA, supervised UMAP, centroid-encoder, and autoencoder. Like MNIST, the neighborhoods of CE and the other methods share many similarities. Digits 4, 9, and 7 are still neighbors in all the techniques but now sit more centrally. Digits 8 and 6 are now consistently neighbors across all methods. The within-class scatter of each digit is again the highest for autoencoder since it is unsupervised. Non-linear NCA also has considerable variance across all the digit classes. Supervised UMAP has tighter clumping of the classes – apparently an integral feature of SUMAP as well as UMAP. However, supervised UMAP has some misplaced digits compared to CE, which is evident from the error rate in Table 3.3. These observations are consistent with MNIST visualizations. SUMAP emphasizes the importance of local structure over the global structure of the data. In contrast to CE, Autoencoder doesn't provide any meaningful information about the neighborhood structure.

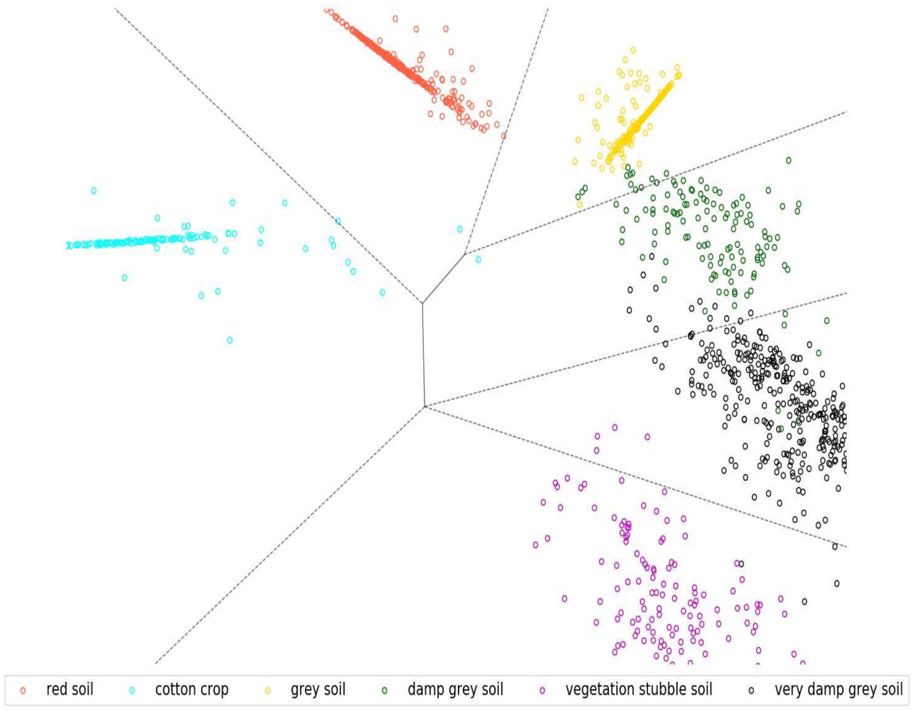
3.7.3 Letter, Landsat and Phoneme Data Sets

Here we compare centroid-encoder with another suite of supervised methods including, SNeRV, PE, S-Isomap, MUHSIC, MRE, and NCA. Table 3.5 shows the results on three bench-marking data sets, including UCI Letter, Landsat, and Phoneme data set. We included UMAP for additional comparison. CE produces the smallest prediction error on Landsat and Phoneme data sets and is ranked second on the Letter data set. On Landsat, the top three models are CE, SUMAP and UMAP. On the Phoneme data, the centroid-encoder outperforms the second-best model, which is SUMAP, by a small margin of 0.09%. Notably, the variance of errors of centroid-encoder (1.33) is better than the SUMAP (2.84). On the Letter dataset, the centroid-encoder achieves the error rate of 23.82%, which is the second-best model after SNeRV ($\lambda = 0.1$), although the variance of the result of centroid-encoder is better than SNeRV by a margin of 1.47. Surprisingly the performance of SUMAP and UMAP degrade significantly on this particular data set. It's also noteworthy that the variance of errors for centroid-encoder is the lowest *in every case*.

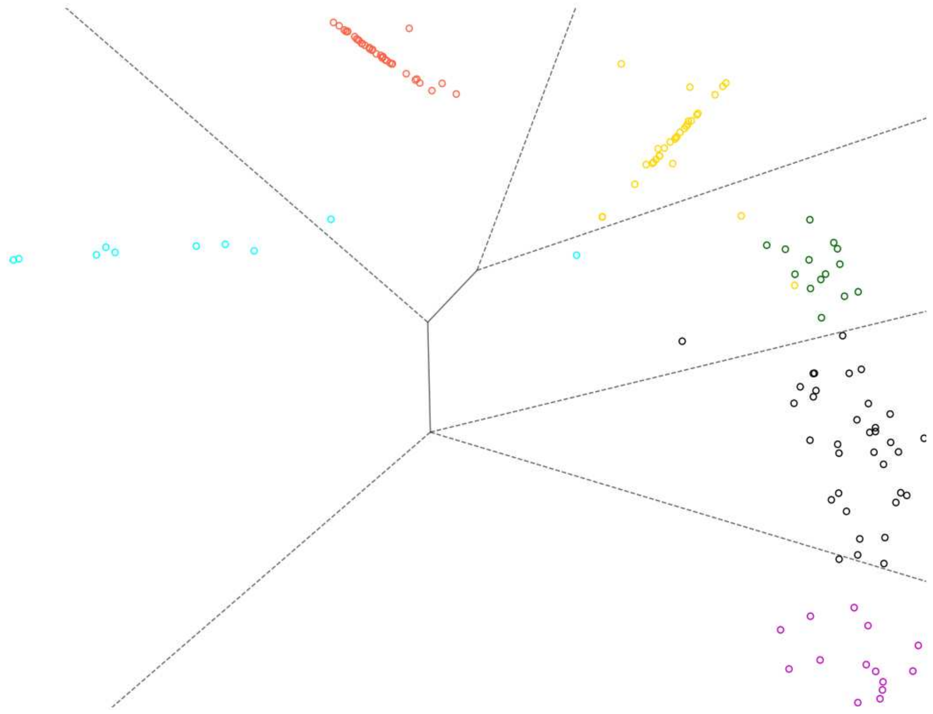
Table 3.5: Classification error ($\%$) of k-NN (k=5) on the 2D embedded data by different supervised embedding methods on Letter, Landsat and Phoneme data. Average misclassifications over ten-fold cross-validation are shown along with standard deviation. Results of methods other than centroid-encoder, UMAP and SUMAP are reported from [5].

Method	Dataset		
	Letter	Landsat	Phoneme
CE	23.82 ± 3.13	**8.97** $\pm$ **2.55**	**7.52** $\pm$ **1.33**
Supervised UMAP	33.57 ± 5.29	9.68 ± 3.12	7.61 ± 2.84
UMAP	39.15 ± 5.51	11.89 ± 2.83	8.47 ± 3.08
SNeRV $\lambda = 0.1$	**22.96** $\pm$ **4.6**	14.34 ± 7.38	9.43 ± 7.79
SNeRV $\lambda = 0.3$	24.59 ± 4.6	13.93 ± 6.97	9.02 ± 7.38
PE	31.15 ± 4.92	14.75 ± 8.20	9.84 ± 6.15
S-Isomap	31.97 ± 7.38	15.16 ± 9.02	8.61 ± 5.74
NCA	62.30 ± 5.74	15.57 ± 7.38	20.49 ± 5.73
MUHSIC	79.51 ± 4.92	15.37 ± 4.10	14.75 ± 4.1
MRE	90.98 ± 7.38	53.28 ± 34.2	45.08 ± 18.03

The visualization of the two-dimensional embedding using centroid-encoder on Landsat is shown in Figure 3.7. We might conclude that the water content in the samples is causing scatter

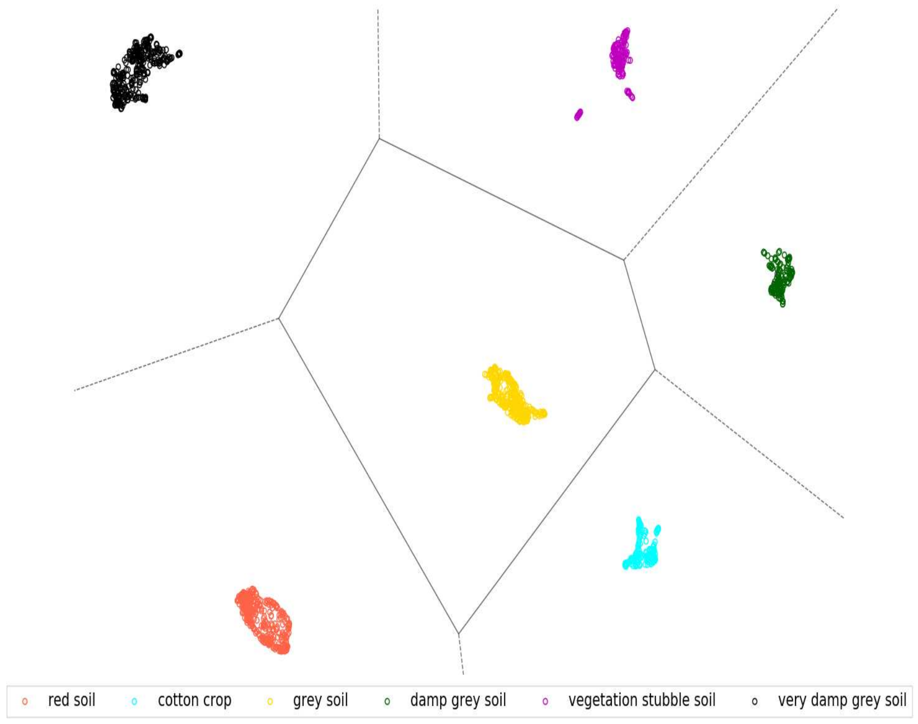


(a) Two-dimensional plot of Landsat training samples.

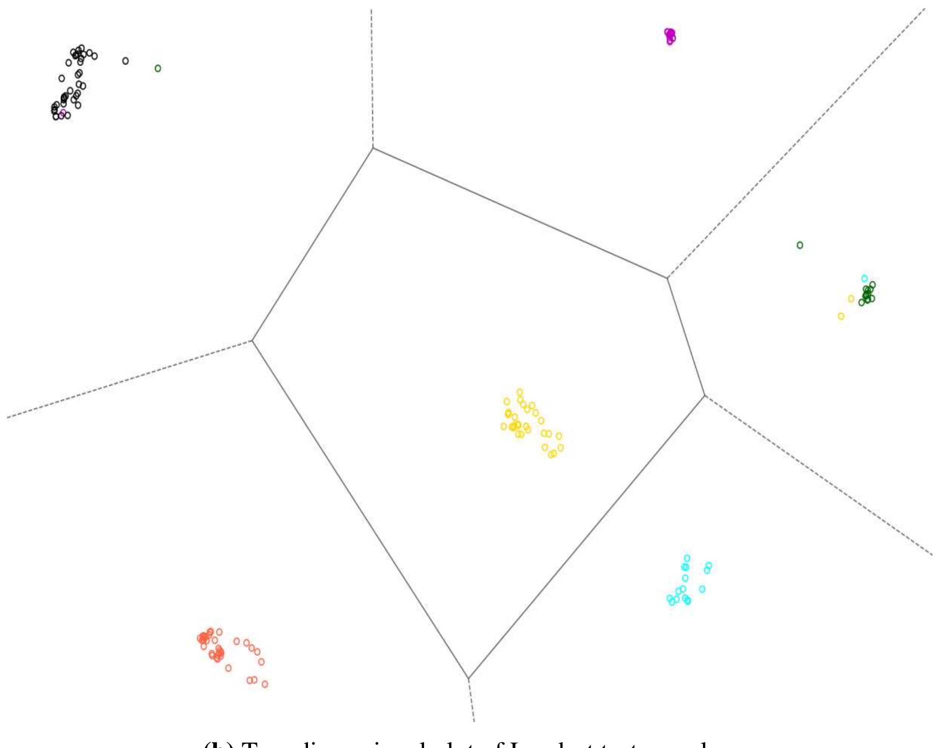


(b) Two-dimensional plot of Landsat test samples.

Figure 3.7: Voronoi cells of two-dimensional Landsat data using a $16 \rightarrow [250, 150, 2, 150, 250] \rightarrow 16$ centroid-encoder. Voronoi regions for each soil type are formed from the training set.



(a) Two-dimensional plot of Landsat training samples.



(b) Two-dimensional plot of Landsat test samples.

Figure 3.8: Voronoi cells of two-dimensional Landsat data using Supervised UMAP. Voronoi regions for each soil type are formed from the training set.

in the 2D plots and decreases as we proceed counter-clockwise. The test samples encode essentially the same structure. In Figure 3.8, we show the visualization using SUMAP. We see tighter clusters, but there is no difference between the damp and non-damp soil types. The neighborhood relationships, demonstrated by Laplacian Eigemaps, also show this transition by dampness consistent with CE. Damp and very damp soils are, however, not neighbors using SUMAP, making the visualization less informative.

3.7.4 Iris, Sonar and USPS (revisited) Data Sets

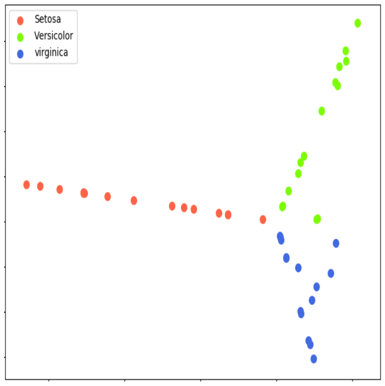
In this experiment, we compare supervised PCA and kernel supervised PCA to centroid-encoder. Table 3.6 contains the classification error using a 5-NN classifier, showing that centroid-encoder outperforms SPCA and KSPCA by considerable margins by this objective measure. In the top row of Figure 3.9, we present the two-dimensional visualization of Iris test data. Among the three methods, centroid-encoder and KSPCA produce better separation compared to SPCA. SPCA can't separate the Iris plants Versicolor and Virginica in 2D space. The dispersion of the three Iris categories is relatively higher in KSPCA and SPCA compared to centroid-encoder. Surprisingly in KSPCA, three classes are mapped on three separate lines.

Table 3.6: k-NN ($k = 5$) error (%) on the 2D embedded data by different supervised embedding methods on Iris, Sonar and USPS data. Note that given the limitations of KSPCA we have restricted the USPS data set to have 1000 total samples.

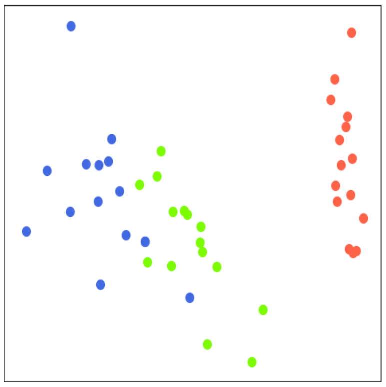
Method	Dataset		
	IRIS	Sonar	USPS
Centroid-Encoder	**3.29** $\pm$ **2.10**	**14.24** $\pm$ **2.99**	**12.16** $\pm$ **2.01**
KSPCA	5.16 ± 4.56	21.06 ± 5.38	51.63 ± 2.11
SPCA	5.24 ± 2.23	32.75 ± 6.78	54.18 ± 0.66

The two-dimensional visualization of Sonar test samples is shown in the middle row of Figure 3.9. None of the methods can completely separate the two classes. The embedding of KSPCA is slightly better than SPCA, which doesn't separate the data at all. In centroid-encoder, the two classes are grouped relatively far away from each other. Most of the rock samples are mapped at the bottom-left of the plot, whereas the mine samples are projected at the top-right corner.

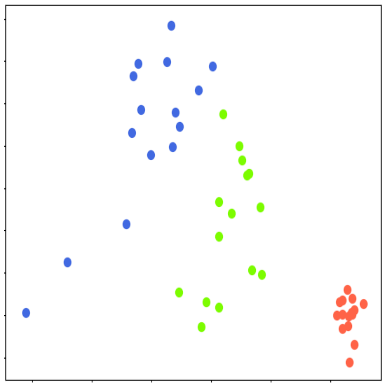
The visualization of USPS test data is shown at the bottom of Figure 3.9. Both SPCA and KSPCA are unable to separate the ten classes, although KSPCA separates digit 0s from the rest of the samples. In contrast, the centroid-encoder puts the ten digit-classes in 2-D space without much overlap. Qualitatively, the embedding of centroid-encoder is better than the other two.



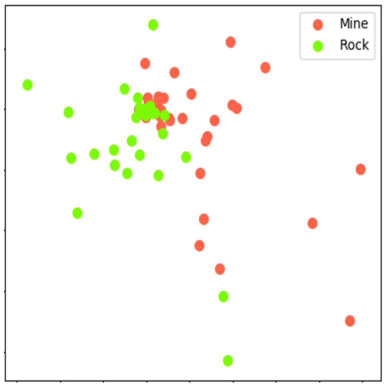
(a) KSPCA



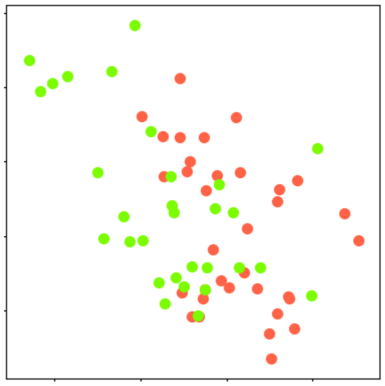
(b) SPCA



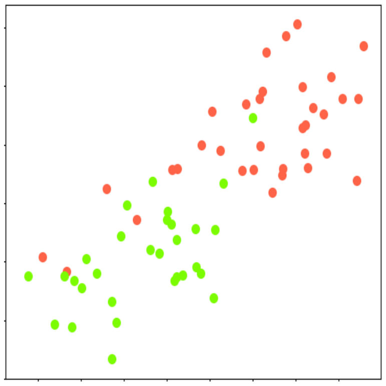
(c) Centroid-Encoder



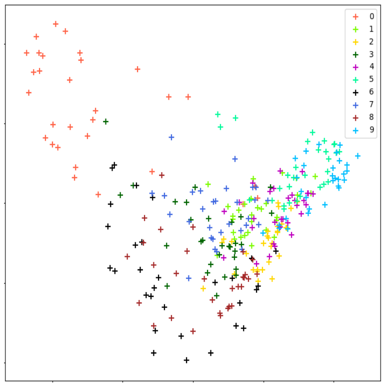
(d) KSPCA



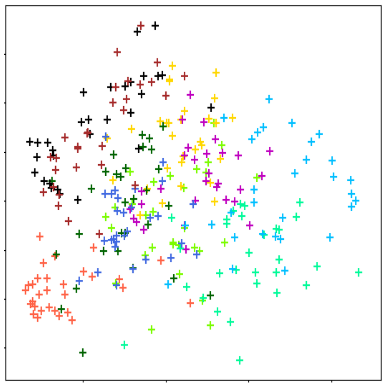
(e) SPCA



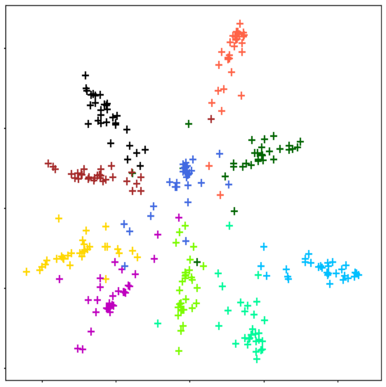
(f) Centroid-Encoder



(g) KSPCA



(h) SPCA



(i) Centroid-Encoder

Figure 3.9: Visualization of Iris (top row), Sonar (middle row) and USPS (bottom row) data using different dimensionality reduction techniques.

3.7.5 SUSY data set

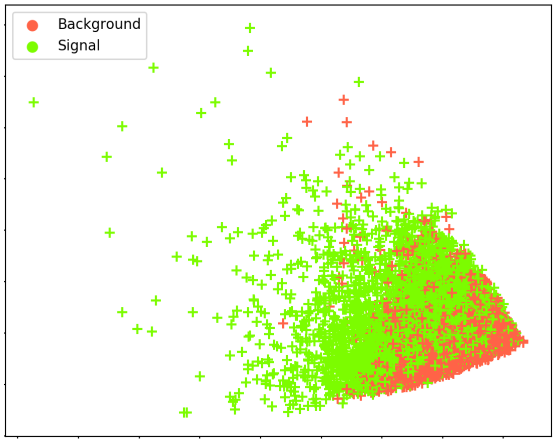
The high-energy supersymmetry particle data (SUSY), is a large data set with 4,500,000 training points and 0.5 million test points [119]. Here we demonstrate that CE is effective at visualizing larger data sets; taken as a whole this data set is too large for many of the other methods compared here such Laplacian Eigenmaps, t-SNE, KSPCA, SPCA. The sklearn implementation of UMAP required 9 hours 12 minutes to complete on this data set.

Table 3.7: Balanced success rate (BSR) and area under the curve (AUC) measure of different methods on SUSY data sets. NA indicates result not reported.

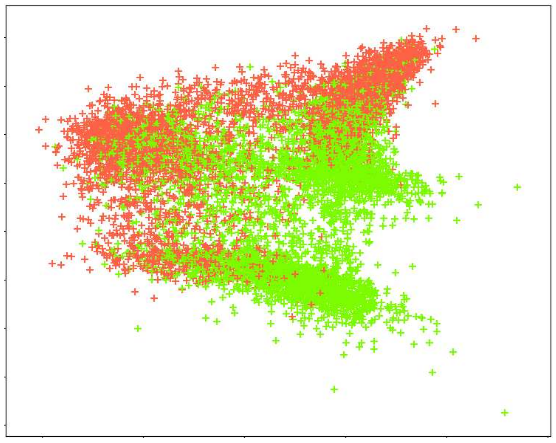
Method	Dim. of classification space	BSR	AUC
CE Classifier	2	**80.41 ± 0.02**	0.875 ± 0.002
Baldi et al. [119]	300	NA	**0.879**
Le et al. [70]	not reported	77.79 ± 0.02	NA

In Figure 3.10, we show the 2-dimensional embedding of SUSY test samples using centroid-encoder, supervised UMAP and PCA. Unlike previous exmaples in this paper, here we used a multi-center per class approach for centroid-encoder. For both the signal and background classes two centers were determined using standard k-Means algorithm [120,121]. Now the CE algorithm is applied as before. We observe that the data is not completely separable in two-dimensional space, as evident from the visualization of each model. PCA and centroid-encoder put the two classes close together, although centroid-encoder split the two classes better than PCA. On the other hand, SUMAP produces three distinct clusters: one for the signals for two for the backgrounds. Notice the overlap of the samples from signal to the background.

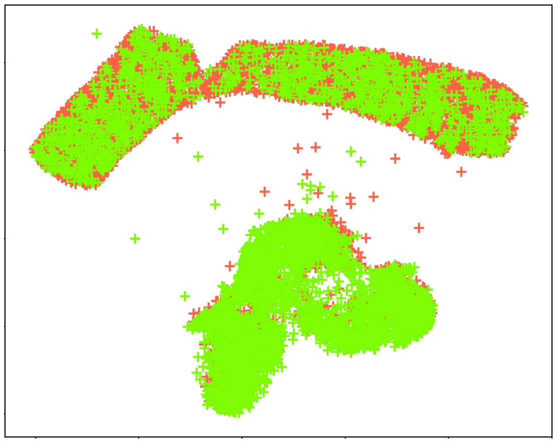
To quantify the quality of the embedding of these models, we again ran k-NN classification on the two-dimensional space. We vary the number of nearest neighbors (hyper-parameter k) to investigate the robustness of the embedding as we transition from local to more global structure. We see that the accuracies remain almost the same for supervised UMAP as we increase the value of k. On the other hand, the classification improves for centroid-encoder as k increases. The accuracy starts to saturate after $k = 50$. The low classification accuracy of PCA is expected as it's an unsupervised model. It's noteworthy that the classification performance using the embedding of centroid-encoder is better than supervised UMAP for each k. The higher classification accuracies for the small and large values of k indicate that the embedding of centroid-encoder is robust than supervised UMAP. From the neighborhood perspective, we can say that the centroid-encoder better preserves the neighborhood of the data both from local to more global sense.



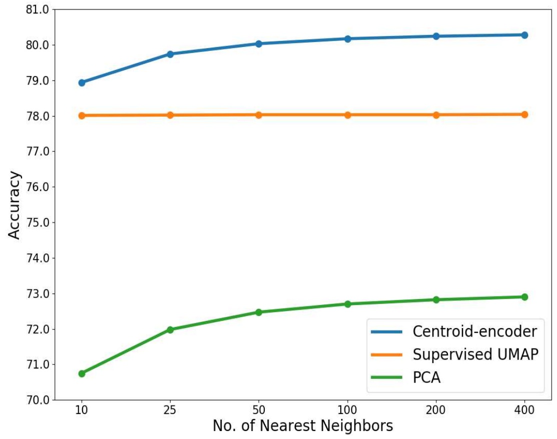
(a) PCA projection of the test data set.



(b) CE reduction of the test data set.



(c) SUMAP representation of the test data set.



(d) k-NN classification accuracy.

Figure 3.10: Visualization of SUSY data set using PCA, centroid-encoder and supervised UMAP. We built the models on the training set, and then project the test samples using the trained models. We display a subset of the test set (5,000 samples selected randomly from each class) using the three models. The k-NN classification accuracy is shown in panel (d).

To illustrate how CE can be used as a classifier, we train a one-hot encoded neural network on the visualized data. In this architecture the decoder is discarded and a softmax layer with one-hot-encoding is attached with the encoder. The network is further trained to learn the class label. In this experiment a bottleneck architecture with two nodes in the bottleneck layer is used; see Table 3.1. The resulting balanced success rate (BSR) of CE is better than the supervised autoencoder model of [70] by a margin of 2.65. The difference of AUC score between our model to [119] is not significant. We should note that we performed the classification on the 2-dimensional space, whereas [119] did the classification on 300-dimensional space; see Table 3.7.

3.8 Comparison between CE and Bottleneck-ANN Classifier

In this section, we compare the visualization of CE with an artificial neural networks classifier with bottleneck architecture. For simplicity, we will call the later model bottleneck-ANN, which maps a sample to its class label, unlike the centroid-encoder, which maps a sample to its corresponding class-centroid. The comparison aims to explore relative properties and features of the bottleneck-ANN and CE embeddings. We note that mapping to class-centroids is not the same as mapping to class labels. The bottleneck-ANN uses one-hot-encoding vector to map a sample to its class label. Let's consider a C-class problem where each data point lives in $\mathbb{R}^d$. The one-hot-encoding class labels are $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \dots \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$ where each target vector has C number of elements. Therefore the bottleneck-ANN will be a mapping $f_{ANN} : \mathbb{R}^d \to \mathbb{R}^C$. In contrast, the mapping of centroid-encoder is $f_{CE} : \mathbb{R}^d \to \mathbb{R}^d$ as the centroids live in $\mathbb{R}^d$-dimensional space.

To compare these two models, we used a biological dataset with three classes: one healthy (control) and two sick classes. The disease class 2 is more severe than class 1, and the samples of class 1 lie between controls and class 2. Please note that the relationship mentioned above is shared with us by the data provider, and a two-dimensional PCA plot (see Figure 3.11) also confirms the fact. For a fair comparison we used the same bottleneck architecture $250 \to 2 \to 250$. We used 70% samples from each class in training and the rest for the test. Figure 3.12 displays the two-dimensional visualization of the three classes using CE and Bottleneck-ANN. Notice that CE placed the sick class 1 (red samples) between the healthy and the sick class 2, maintaining the class relationship in the low-dimensional space. On the other hand, Bottleneck-ANN put the three classes equidistantly, failing to preserve the geometry of the class structure. The embedding of Bottleneck-ANN is not surprising; the reason lies in the target of the mapping (f_{ANN}). The one-hot-encoding targets of the three classes are $\begin{bmatrix}1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\end{bmatrix}$, and $\begin{bmatrix}0\\0\\1\end{bmatrix}$. Notice the distance between any two classes in $\sqrt{2}$. Hence the mapping (f_{ANN}) is enforcing the classes to be equidistant in the bottleneck space. This demonstration suggests that the centroid-encoder is a better visualization tool than Bottleneck-ANN on this dataset.

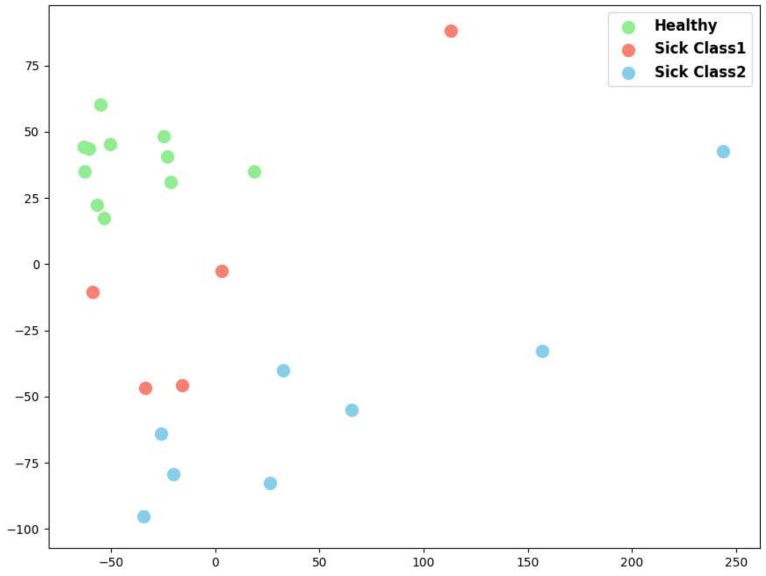


Figure 3.11: Two-dimensional PCA plot of healthy and sick samples.

Now we compare the two models on the widely used MNIST digits using a hidden layer architecture of $1000, 500, 125, 2, 125, 500, 1000$. Note, CE maps the output of the last hidden layer to a class-centroid, whereas the Bottleneck-ANN maps to a one-hot-encoding class label. In Figure 3.13, we show the two-dimensional plot of MNIST digits. Observe that the model puts digits '7' (dark blue) away from classes '4' (purple) and '9' (sky blue). These three classes are neighbors, as shown by several models, e.g., Laplacian Eigenmap (Figure 3.4), UMAP (Figure 3.5), t-SNE (Figure 3.5) and Centroid-Encoder (Figure 3.3). Bottleneck-ANN thinks digits '1' (light green) and '6' (black) are neighbors, although these two digits are not similar looking. None of the other models

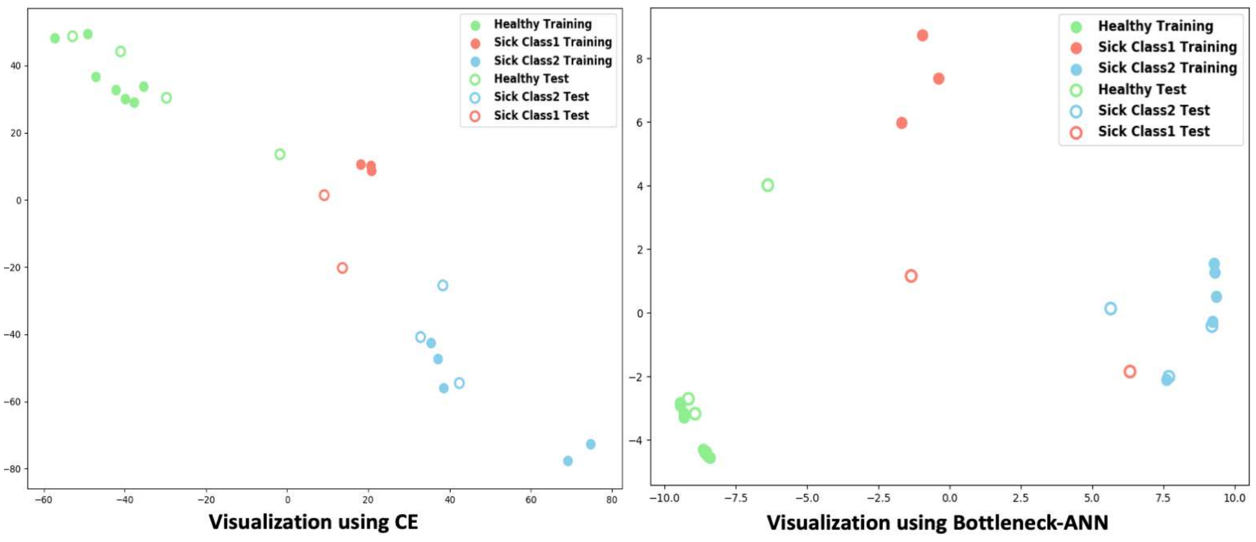


Figure 3.12: Two-dimensional visualization of healthy and sick samples using CE and Bottleneck-ANN using . Unlike CE, which maps a sample to its class-centroid, bottleneck-ANN maps a sample to its class label. Both the models use the same hidden layer layout $250 \rightarrow 2 \rightarrow 250$.

(LE, UMAP, t-SNA, and CE) put these two classes in neighboring space. Clearly, bottleneck-ANN doesn't preserve the neighborhood relationship. To objectively compare the embedding of CE and bottleneck-ANN, we did a 5-NN classification on the embedding space. The error rate of bottleneck ANN is 4.46 ± 0.32, compare to CE 2.61 ± 0.09.

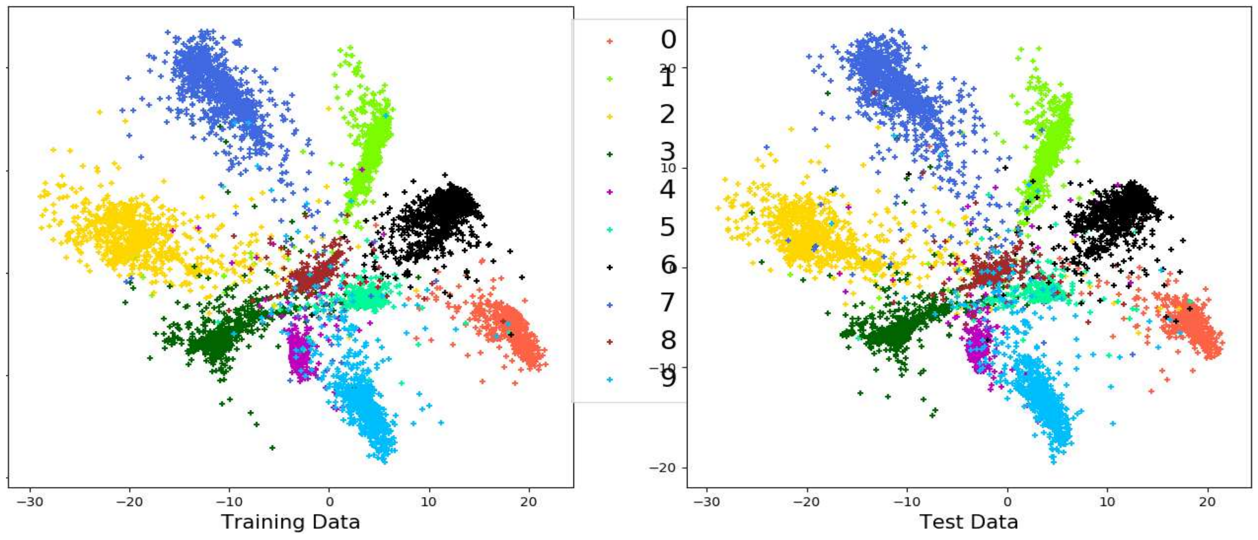


Figure 3.13: Two-dimensional visualization of MNIST digits using Bottleneck-ANN with a network architecture $784 \rightarrow [1000, 500, 125, 2, 125, 500, 1000] \rightarrow 10$. Each sample is mapped to the one-hot-encoding class label.

3.9 Discussion and Analysis of Results

The experimental results in Section 3.7 establish that centroid-encoder performs competitively, i.e., it is the best, or near the best, on a diverse set of data sets. The algorithms that produce similar classification rates require the computation of distance matrices and are hence more expensive than centroid-encoders. Here we analyze the CE algorithm in more detail to discover why this improved performance is perhaps not surprising. This has to do, at least in part, with the fact that centroid-encoder, implementing a non-linear mapping, is actually able to capture more variance than PCA, and this is particularly important in lower dimensions.

3.9.1 Variance Plot to Explain the Complexity of Data

PCA captures the maximum variance over the class of orthogonal transformations [37]. The total variance captured as a function of the number of dimensions is a measure of the complexity of a data set. We can use variance to establish, e.g., that the digits 0 and 1 are less complex than the digits 4 and 9. To this end, we use two subsets from the MNIST training set: the first one contains all the digits 0 and 1, and the second one has all the digits 4 and 9. Next, we compute the total variance captured as a function of dimension; see Figure 3.14. Clearly, the variance is more spread out across the dimensions for the digits 4/9. In Figure 3.15, we show the visualization of these two subsets using PCA. We see that the 0/1 samples in subset1 are well separated in two-dimensional space, whereas the data in subset2 are clumped together. This lies in the fact that in low dimension (2D) PCA captures more variance from subset1 (around 41%) as compared to subset2 (about 22%). We can say that subset 2 is more complex than subset1 .

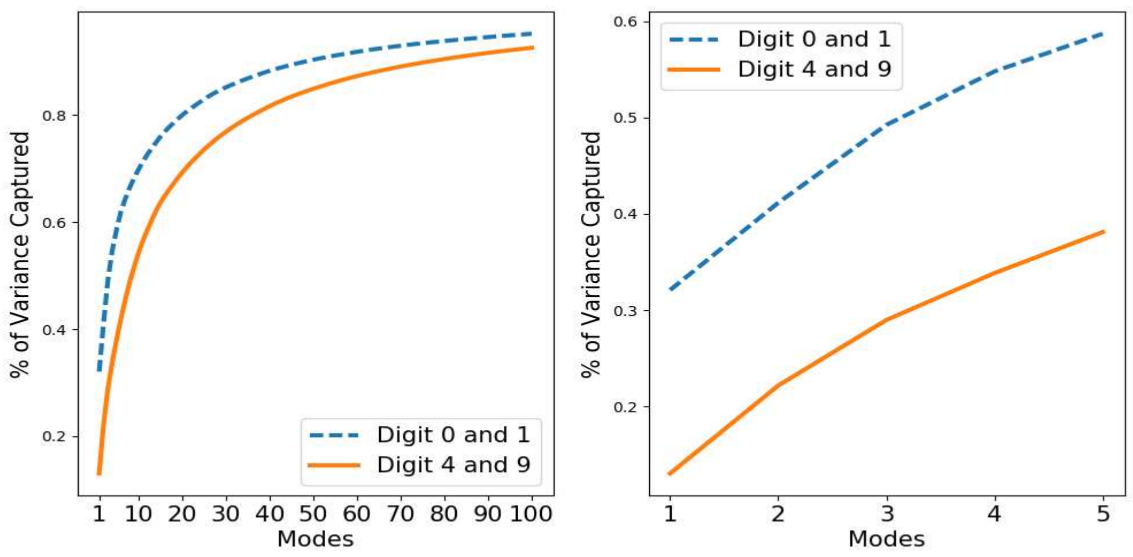
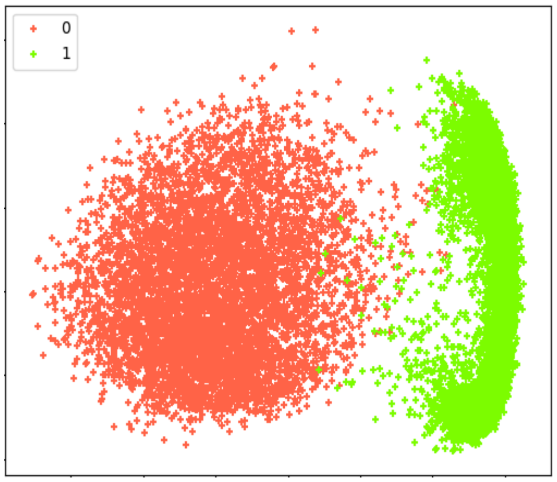


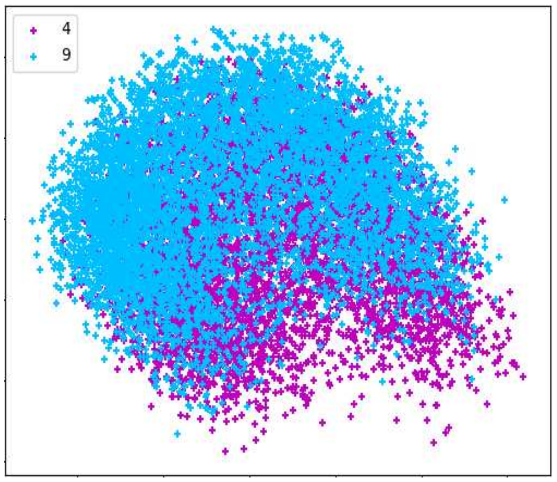
Figure 3.14: Variance plot using two different subsets (subset1: all the digits 0 and 1, subset2: all the digits 4 and 9) of MNIST data. Left: comparison of variance plot of the two subsets using the first 100 dimensions. Right: a blowup of the plot on the left showing the % of variance captured in low dimensions.

3.9.2 Centroid-Encoder and Variance

In this section, we demonstrate how centroid-encoder maps a higher fraction of the variance of the data to lower dimensions. Again, consider the subset of digits 4 and 9 from the experiment above. We use all the digits 4 and 9 from the MNIST test set as a validation set. We trained a centroid-encoder with the architecture $784 \rightarrow [784] \rightarrow 784$ where we used hyperbolic tangent ('tanh') as the activation function. We keep the number of nodes the same in all the three layers. After the training, we pass the training and test samples through the network and capture the hidden activation which we call CE-transformed data. We show the variance plot and the two-dimensional embedding by PCA on the CE-transformed data. Figure 3.16 compares the variance plot between the original data and the CE-transformed data. In the original data, only 22% of the total variance is captured, whereas about 89% of the total variance is captured in CE-transformed data. It appears that the non-linear transformation of centroid-encoder puts the original high dimensional data in a relatively low-dimensional space. Visualization using PCA of CE-transformed data (both training and test) is shown in Figure 3.17. This time, PCA separates the classes in two-dimensional space.



(a) Subset1: digit 0 and 1



(b) Subset 2: digit 4 and 9

Figure 3.15: Two dimensional embedding using PCA for subset1 and subset 2.

Comparing Figure 3.17 with Figure 3.15b, it appears that the transformation by centroid-encoder is helpful for low-dimensional visualizations.

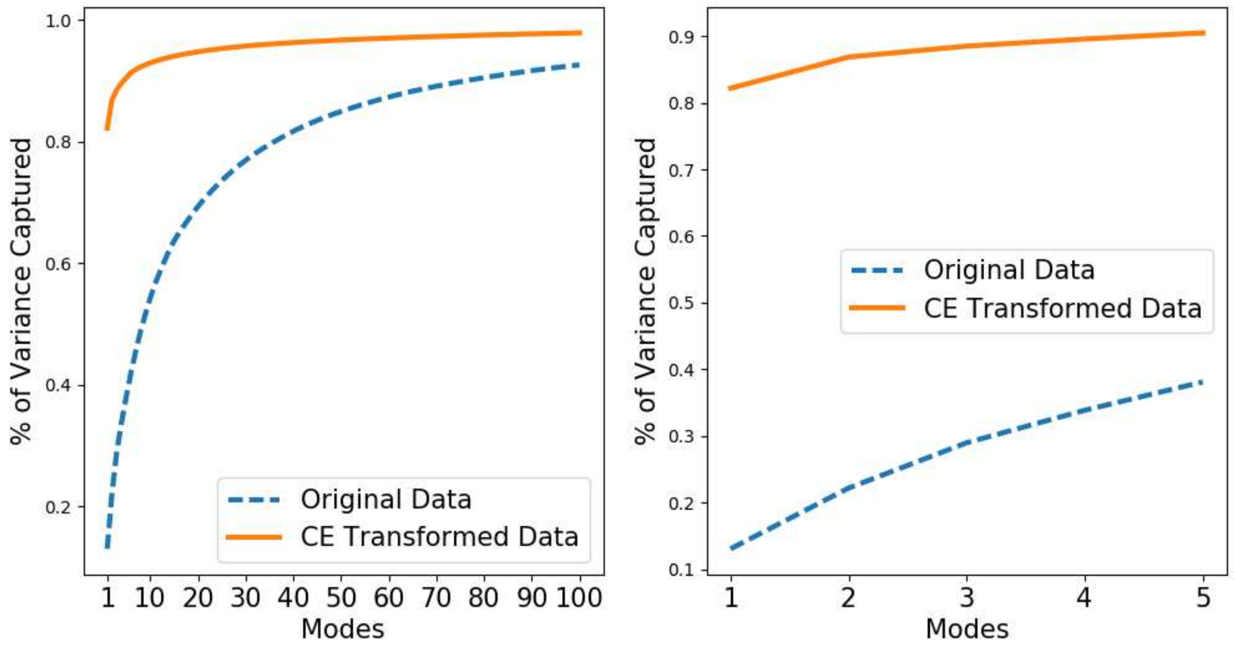
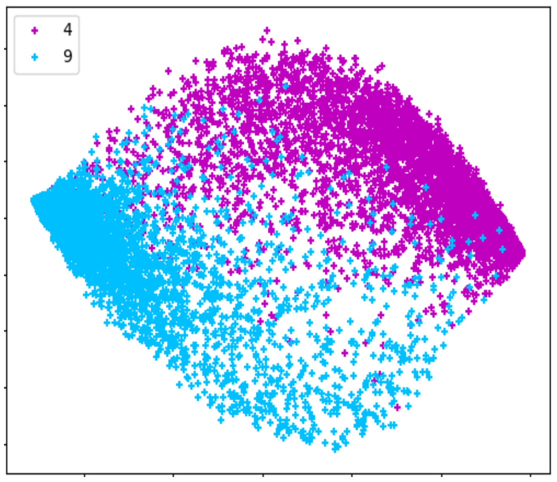
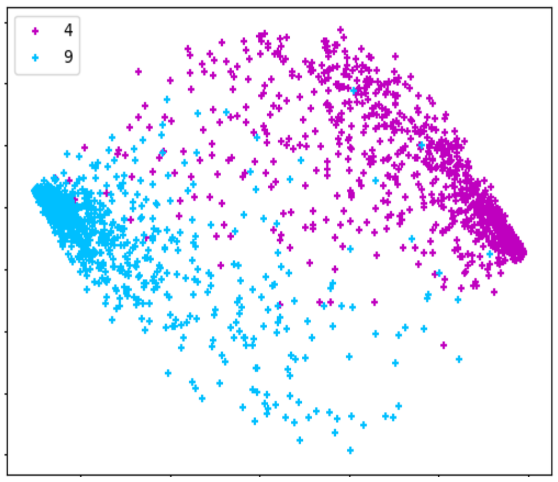


Figure 3.16: Variance plots for the 4/9 digit data. Left: comparison of variance plot of the original data and CE transformed data using the first 100 dimensions. Right: a blowup of the plot on the left showing the % of variance captured in low dimensions.



(a) Digits 4 and 9 from training set.



(b) Digits 4 and 9 from test set.

Figure 3.17: Two dimensional embedding using PCA on the CE-transformed data.

3.9.3 Computational Complexity and Scalability

The computational complexity of the centroid-encoder is effectively the same as the autoencoder. Note, both methods, as is typical of neural network approaches, require the determination of a suitably tuned network architecture. This is a hidden cost for neural networks in general. The advantage of centroid-encoder over many other data visualization techniques is that it is not necessary to compute pairwise distances. If everything else is fixed, we can view centroid-encoder as scaling linearly with the number of data points. Techniques that require distance matrices scale quadratically with the number of data points and can be prohibitively slow as the data set size increases. Methods based on distance matrices include NNCA, dt-NCA, dG-NCA, dt-MCML, and dG-MCML, as well as the spectral methods including MDS, UMAP, Laplacian Eigenmaps, Isomap and their supervised counterparts. The only overhead of centroid-encoder is the calculation of centroids of each class, but this is also linear with the number of classes.

3.10 Conclusions

We presented the centroid-encoder as a new tool for data visualization when class labels are available. CE is well suited to multi-class data visualization tasks for very large data sets, e.g., over a million data points, such as the SUSY analysis presented here. In our experience, it is also beneficial for smaller biological data sets residing in high-dimensions, e.g., 1000 samples with 10,000-50,000 features; this domain-specific application will appear elsewhere. The objective function can also be easily adapted via the addition of penalty terms, such as the addition of a sparsity promoting ℓ_1-norm term. In future work, we will explore the application of this penalty term with CE as a sparse non-linear approach for supervised data reduction that can be applied to the optimal feature extraction problem. The wide variety of examples presented here suggest that CE shares many qualitative performance similarities with techniques such as Laplacian eigenmaps and Fisher Discriminant analysis, but has the advantages associated with autoencoders and deep neural networks including the ability to model very large data sets with highly parameterized deep architectures.

We compared CE to state-of-the-art techniques and showed advantages empirically over other methods. Our examples illustrate that CE captures the topological structure, i.e., neighborhoods of the data, comparable to methods, but with lower cost and generally better prediction error. These experiments include comparisons to the unsupervised Laplacian eigenmaps, UMAP, pt-SNE, autoencoders; the supervised UMAP, SPCA, KSPCA as well as the MCML and NCA class of algorithms. The algorithms that have the most similar prediction errors when compared to CE require the computation of distances between all pairs of points. We demonstrated that the 2D embedding of centroid-encoder produces competitive, if not optimal, prediction errors. We also showed empirically that the model globally captures high statistical variance relative to optimal linear transformations, i.e., more than PCA.

In addition to capturing the global topological structure of the data in low-dimensions, CE exploits data labels to minimize the within-class variance. This improves the localization of the mapping such that points are mapped more faithfully to their Voronoi regions in low-dimensions. CE also provides a mapping model that can be applied to new data without any retraining being necessary. This is in contrast to the spectral methods described here that require the entire training and testing set for computing the embedding eigenvectors. Also, while the experiments run in this chapter focus on visualization of labeled data sets, it would be interesting to explore the classification performance of bottleneck dimensions larger than three.

Chapter 4

Bottleneck Centroid-Encoder

In this chapter, we present a nonlinear dimensionality reduction technique called "bottleneck centroid-encoder" (BCE) that extends the framework of centroid-encoders (CE). In centroid-encoder, the centroids are computed explicitly from the data and are not optimized for classification in the reduced space. Bottleneck centroid-encoders employ a centroid constraint at the bottleneck layer. This bottleneck cost has two contrasting terms. The first term pulls all the class samples to their centroid by minimizing their distance from their class centroid in reduced space. The second term pushes the classes by maximizing the Euclidean distance between class-pair centroids. We present a detailed analysis of the method, including various numerical experiments using several data sets. The results are compared to other supervised dimensionality reduction techniques, including supervised UMAP (SUMAP), nonlinear NCA (NNCA), t-distributed NCA, t-distributed MCML.

4.1 Formulation of BCE

The centroid-encoder algorithm calculates its cost on the output layer and then backpropagates the gradient through the hidden layers. Note that the gradient of a sample depends on the distance from itself to its class-centroid, which is calculated in the ambient space. If the centroids of multiple classes are close in the ambient space, then the corresponding samples will land close in the reduced space. This will cause misclassification. To avoid this situation, we add two more terms to the bottleneck layer, i.e., at the output of the encoder g. Our modified objective function is now

$$\mathcal{L}_{BCE}(\theta) = \frac{1}{2N}\sum_{j=1}^{M}\sum_{i\in I_j}(\|c_j - f(x_i;\theta)\|_2^2 + \lambda_1\|g(c_j) - g(x_i)\|_2^2) + \lambda_2\sum_{k<l}\frac{1}{1+\|g(c_k) - g(c_l))\|_2^2} \tag{4.1}$$

The term $\|g(c_j) - g(x_i)\|_2^2$ will further pull a sample x_i towards it centroid which will improve the class localization in reduced space. As we want to avoid mapping different classes close in the bottleneck, it's desirable to have a force that will push the classes far apart. We can achieve this by maximizing the distances (equivalently the square of the ℓ_2-norm) between all class-pairs centroids. We introduced the third term to fulfill the purpose. Note, as the original optimization is a minimization problem, we choose to minimize $\sum_{k<l} \frac{1}{1+\|g(c_k)-g(c_l)\|_2^2}$ which will ultimately increase the distance between the centroids of class k and l. We added 1 in the denominator for numerical stability. The hyper parameters λ_1 and λ_2 will control the class localization and separation in the embedded space. We used validation set to determine their values. This modification of the centroid-encoder at the bottleneck will be referred to as bottleneck centroid-encoder (BCE).

4.2 Analysis of Hyper-parameters

The BCE model incorporates two hyper-parameters λ_1 and λ_2 to control the class scatter and separation in embedded space. These parameters need to be tuned using a validation set for optimal result on the test set. This section analyzes the importance of the two hyper-parameters on the model's performance. The analysis is done on the MNIST training digits over a range of values. We put aside 20% of samples from each class as a validation set. The rest of the data set is used to train a BCE model for each combination of λ_1 and λ_2. After the training, the validation set is used to compute the error rate using a 5-NN classifier in the two-dimensional space. Figure 4.1 shows the errors for different combinations of λ_1 and λ_2 in a heat map. Observe that the error rate increases with λ_2 when λ_1 is zero. The behavior is not surprising as setting λ_1 to zero nullifies the effect of the second term (see Equation 4.1), which would hold the samples tight around their centroid in reduced space. The gradient from the first term will exert a pulling force to bind the samples around their centroids, but the gradient coming from the third term will dominate the gradient of the first term as λ_2 increases. As an effect, the class-scatter increases in low dimensional space resulting in misclassifications. As soon as λ_1 increases to 0.1, the error rate decreases significantly. After that, a higher value of λ_2 doesn't change the results too much. It's also noteworthy that an increase in λ_1 doesn't decrease the error rate too much. The minimum validation error occurs for $\lambda_1 = 0.6$ and $\lambda_2 = 0.1$. The analysis reveals that λ_1 is relatively more important than λ_2.

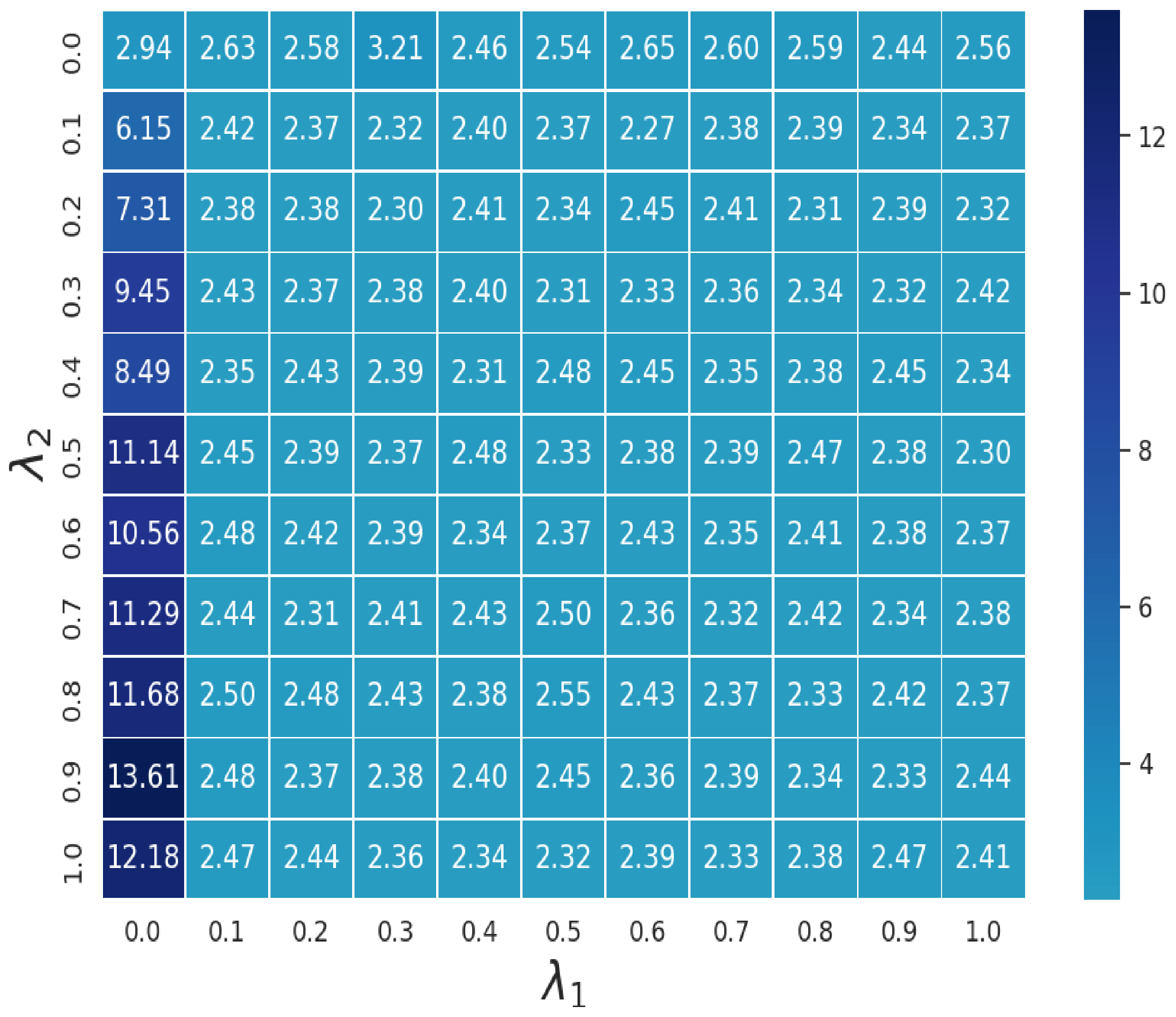


Figure 4.1: Analysis of error rate with changes to λ_1 and λ_2.

4.3 Experimental Results

In this section, we objectively compare the bottleneck centroid-encoder with other supervised models on four datasets. We begin by describing the datasets next.

4.3.1 Data Sets

Here we provide a brief description of the data sets used in the bench-marking experiments.

USPS: A data set of handwritten digits $(0 \ldots 9)$[5], where each digit is a 16x16 gray-scale image. Each of the ten classes has 1100 digits for a total of 11,000 digits.

MNIST: This is widely used collection of digital images of handwritten digits (0..9)[6] with separate training (60,000 samples) and test set (10,000 samples). Each sample is a 28 x 28 grey level image.

Fashion MNIST/FMNIST: It's a data set of 10 different fashion items (T-shirts/tops, shoes, etc) with separate training (60,000 samples) and test set (10,000 samples) [122]. Each sample is a 28 x 28 grey scale image.

4.3.2 Methodology

To objectively compare models, we employ a standard class prediction error using a test data set on the two-dimensional visualization domain.

The class prediction error is defined as

$$Error\ (\%) = \frac{100}{N} \sum_{i=1}^{N} I[l_i \neq f(\tilde{x_i})] \tag{4.2}$$

Here N is total number of test samples, l_i is the true label of the i^{th} test sample, and f is a classification function which returns the predicted label of the embedded test sample $\tilde{x_i}$. Here I denotes the indicator function, which has the value 1 if the argument is true (label incorrect), and 0 otherwise (label correct). A common choice of f for bench-marking algorithms in the supervised visualization literature is the k-nearest neighbor (k-NN) classifier [2, 3, 5, 64].

As MNIST and FMNIST have a separate test set, we train the models on the training set and then used the test set to calculate the generalization error. On the USPS following [3]; we randomly picked $8,000$ cases from the original set to train the models and then used the rest of the $3,000$ samples to test. We took 80% of the samples from every 21 classes to train models on the Mouse

[5]The data set is available at https://cs.nyu.edu/ roweis/data.html.

[6]The data is available at http://yann.lecun.com/exdb/mnist/index.html.

data, and we use the rest of the data for testing. In all the cases we repeat the process 10 times and report the mean error rate with standard deviation.

Like autoencoder, BCE is a model based on deep architecture whose performance varies based on network topology along with other hyper parameters: learning rate[7], mini-batch[8] size, and weight decay[9] constant. As with all neural network learning algorithms, these parameters need to be tuned to yield optimal performance. We randomly picked some samples from each dataset ($10,000$ from MNIST and FMNIST and 2000 from USPS) to use as a validation set to finetune the hyperparameters. For the Mouse data, we chose 10% random samples from each class as a validation set. We implemented our models (along with Autoencoder) in PyTorch to run on GPUs. We used the Adam optimizer ([49]) to update the model parameters.

Beside our own models, we have implemented nonlinear NCA (NNCA) in Python following [66]. We used the scikit-learn [117] package to run supervised UMAP (SUMAP). For the rest of the methods, we took the published results for comparison.

4.3.3 Quantitative and Visual Analysis

Now we present the results from a comprehensive quantitative and qualitative analysis across various data sets.

MNIST: First we discussed the result on MNIST set.

As shown in Tables 4.1, the models with relatively low error rates for the MNIST data amongst our suite of visualization methods are dt-MCML, BCE, dG-MCML and centroid-encoder. The error rate of our models is comparable to the top-performing model dt-MCML and superior to NNCA and supervised UMAP. Note that methods in the MCML and NCA families require the computation of distance matrix over the data set, making them significantly more expensive than BCE and CE. It's noteworthy that the error of BCE and CE is lower than the other non-linear variants of

[7]Learning rate was selected from the following list of values: 0.1, 0.01, 0.001, 0.0001, 0.0002, 0.0004, 0.0008.

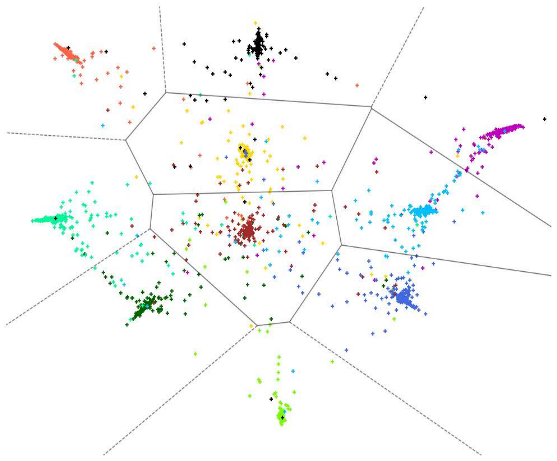
[8]Mini-batch size was selected from the following values: 16, 32, 50, 64, 128, 256, 512, 1024.

[9]Weight decay was chosen from the following values: 0.001, 0.0001, 0.00001, 0.00002, 0.00004, 0.00008.

Table 4.1: Error rates (%) of k-NN (k=5) on the 2D embedded data by various dimensionality reduction techniques. Results of GerDA and pt-SNE are taken from [1] and [2] correspondingly. Error rates of the variants of NCA and MCML are reported from [3].

Method	Dataset	
	MNIST	USPS
CE	2.61 ± 0.09	2.91 ± 0.31
BCE	2.12 ± 0.08	2.53 ± 0.19
SUMAP	3.45 ± 0.03	6.17 ± 0.23
NNCA	4.71 ± 0.57	6.58 ± 0.80
Autoencoder	22.04 ± 0.78	$16.49 \pm .91$
dt-MCML	**2.03**	**2.46 ± 0.35**
dG-MCML	2.13	3.37 ± 0.18
GerDA	3.2	NA
dt-NCA	3.48	5.11 ± 0.28
dG-NCA	7.95	10.22 ± 0.76
pt-SNE	9.90	NA

NCA: dt-NCA and dG-NCA. Among all the models, parametric t-SNE (pt-SNE) and autoencoder exhibit the worst performance with error rates on MNIST data are 9.90% and 22.35%, respectively. These high error rates are not surprising given these methods do not use label information.


(a) Embedding of BCE.

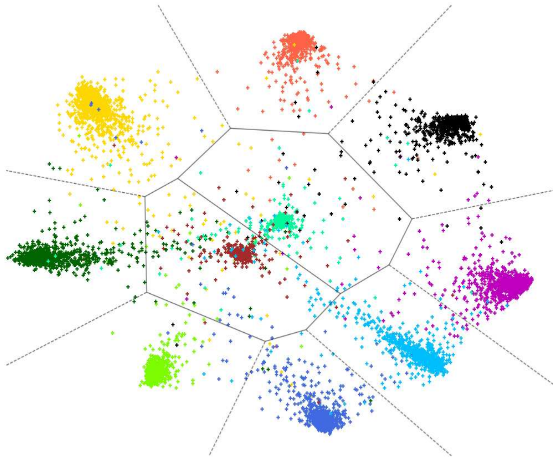

(b) Embedding of CE.

Figure 4.2: Two dimensional embedding of MNIST test samples using BCE and CE.

While the prediction error is essential as a quantitative measure, the visualizations reveal information not encapsulated in this number. The neighborhood relationships established by the embedding provide potentially valuable insight into the structure of the data set. The two-dimensional visualization of BCE and CE on the MNIST test set is shown in Figure 4.2. Both the models separated the ten classes. Although the scatter of each digit class is more prominent in CE than BCE. The additional terms in the objective function in the bottleneck layer of BCE makes the clumping tighter for each class and separated from each other. This observation supports the improved generalization of BCE over centroid-encoder in Table 4.1.

Now let's consider the visualization from a neighborhood relationship perspective. We compare the results with those of Laplacian Eigenmaps (LE), as shown in Figure 3.4, an unsupervised manifold learning spectral method which is formulated to preserve neighborhood relations [12].

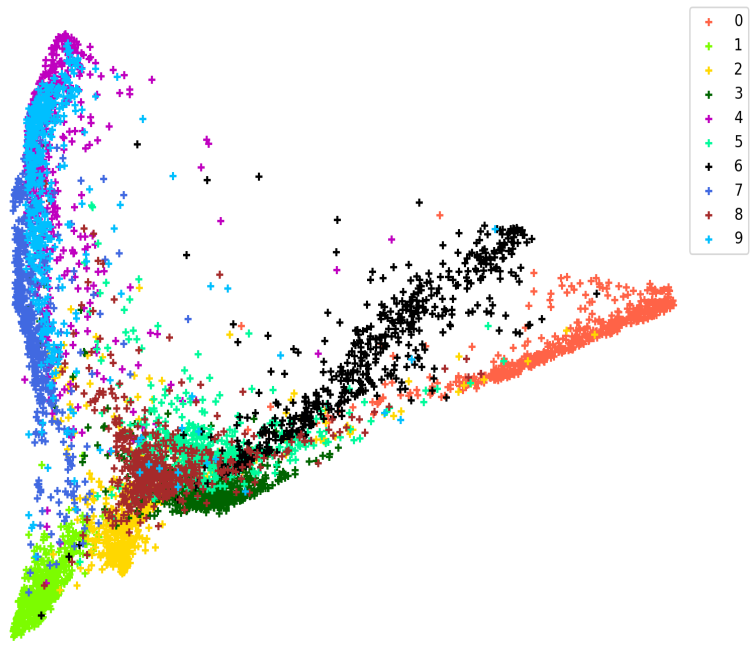


Figure 4.3: Two dimensional visualization of 10,000 MNIST test digits by Laplacian Eigenmaps [12].

LE placed the digit 0 and 6 as neighbors, and the same neighbor-relationship is visible in both BCE and CE. Digits 7, 9, and 4 are collapsed to one region in the embedding of LE; in contrast, BCE and CE are built to exploit label information and separates these neighboring digits. Digits 3, 5, and 8 are neighbors in LE with a significant overlap; this neighborhood relationship is also maintained in BCE and CE, although we see clear separation among these digit classes. Like

LE, BCE and CE also mapped the digits 0 and 1 far apart as these two classes have significant dissimilarities.

USPS: On USPS, BCE and CE are among the top three performing models. The difference of error rate between the BCE and the top performing model dt-MCML is very slim. Both of our models have lower error rate than supervised UMAP and nonlinear NCA. Figure 4.4 shows the two-dimensional representation of the USPS test digits by all the models. When comparing CE to BCE, we see that the latter produces more compact clusters of the ten classes with better separation. In the embedding of NNCA, each digit class has dispersed more compare to other methods. SUMAP emphasizes the importance of local structure over the global structure of the data.

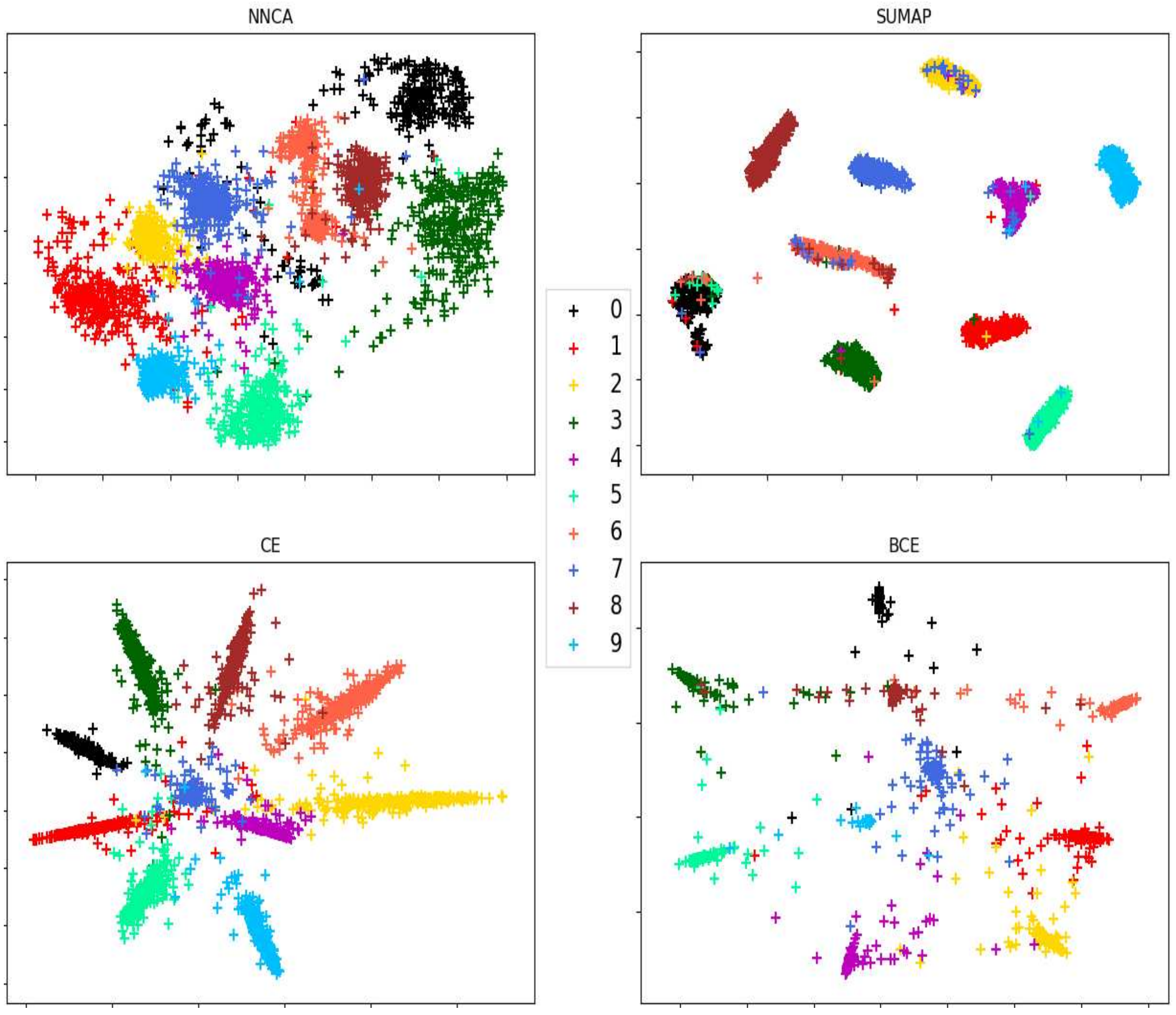


Figure 4.4: Two dimensional embedding of 3,000 USPS test digits by different methods.

FMNIST: Table 4.2, shows the error rates on FMNIST using four supervised models. BCE is the best among all the models leaving centroid-encoder the second-best model. The classification error of BCE is better by a margin of at least 3% from SUMAP and NNCA. Figure 4.5 shows visualization of FMNIST training and test samples using BCE. It can be argued that the BCE embedding is better compared to NNCA and SUMAP as shown in Figure 4.6. NNCA didn't produce tighter clusters compared to BCE. For example, the shirts (silver), coats (magenta) and bags (brown) are dispersed too much. Unlike BCE, the dresses (dark green) are mapped too close to the other clothing items. The SUMAP (4.6b) managed to produce ten clusters, but the model failed to separate the classes. The clothing groups are significantly overlapped with samples from other categories. In particular, two cloth categories, the t-shirts (red) and the pullover (golden), are not visible at all. The shirts (silver) overlapped significantly with other groups. This justifies the better error rate of BCE (9.79%) compared to NNCA (13.02%) and SUMAP (14.79%).

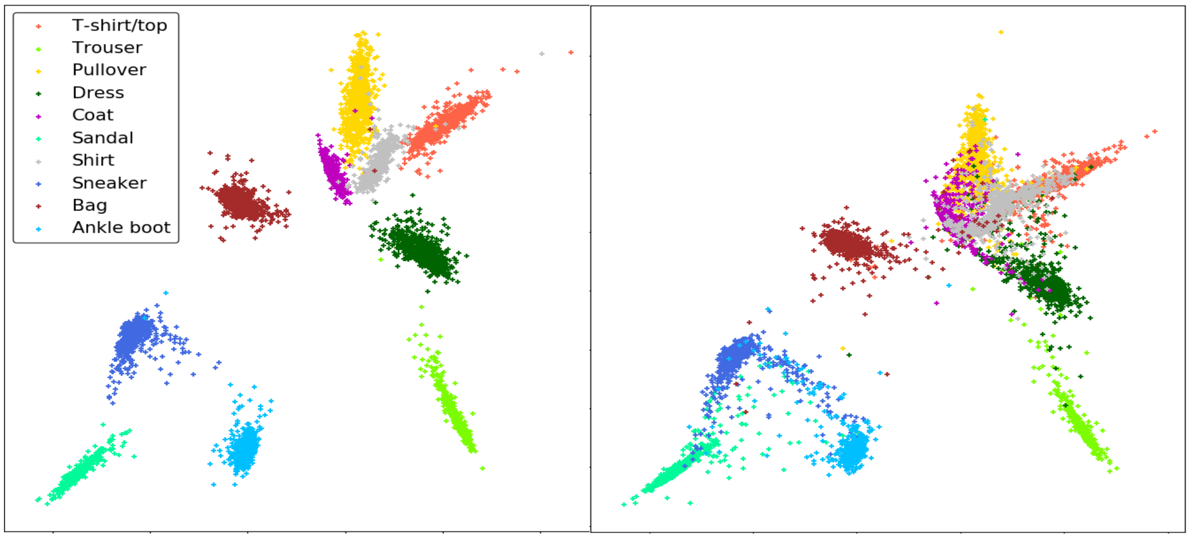
Table 4.2: Error rates ($\%$) of k-NN (k=5) on the 2D embedded data by various dimensionality reduction techniques on the FMNIST data.

Method	Error Rate
CE	10.34 ± 0.54
BCE	$\mathbf{9.79 \pm 0.24}$
SUMAP	14.79 ± 0.00
NNCA	13.02 ± 0.54

Bottleneck Centroid-Encoder with Reconstruction Loss in the Output Layer The BCE incorporates a CE loss at the output layer. Le et al. have shown that the unsupervised reconstruction loss of autoencoders improves the generalization performance [70].

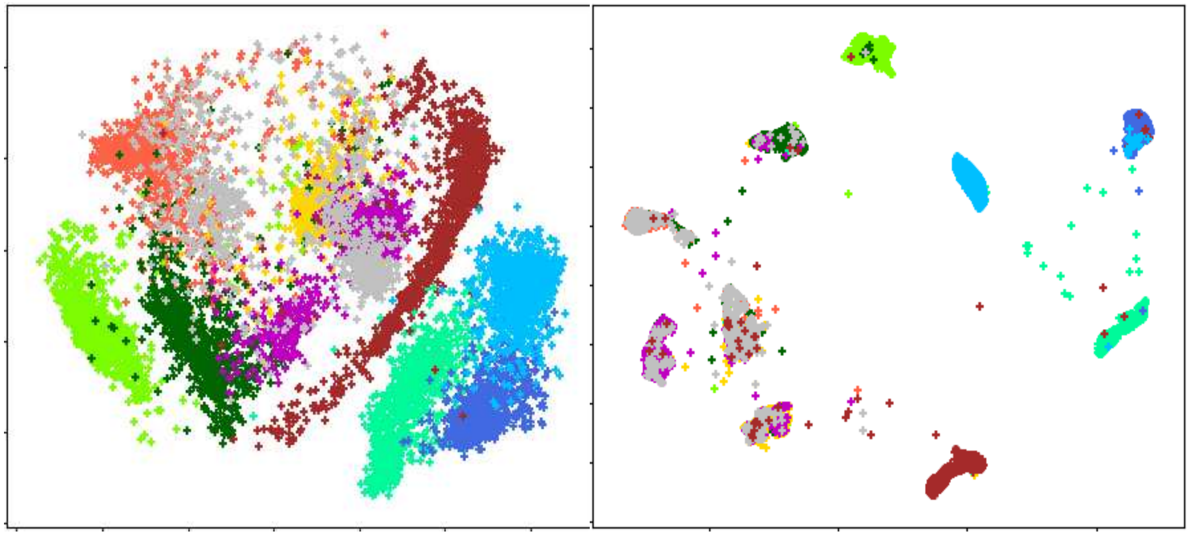
$$\mathcal{L}(\theta) = \frac{1}{2N}\sum \|x_i - f(x_i;\theta)\|_2^2 + \frac{1}{2N}\sum_{j=1}^{M}\sum_{i\in I_j} \lambda_1 \|g(c_j) - g(x_i)\|_2^2 + \lambda_2 \sum_{k<l} \frac{1}{1 + \|g(c_k) - g(c_l))\|_2^2} \tag{4.3}$$

Motivated by their work, we incorporate the reconstruction loss in the BCE framework as shown in Equation 4.3. Notice the difference between the original BCE cost (Equation 4.1) and the



(a) Training data. **(b)** Test data.

Figure 4.5: Two dimensional embedding of FMNIST data using bottleneck centroid-encoder.



(a) NNCA. **(b)** SUMAP.

Figure 4.6: Two dimensional embedding of FMNIST test cases using NNCA and SUMAP.

proposed one (Equation 4.3); the CE loss at the output layer is replaced by the reconstruction loss of autoencoders. With this modification to the BCE cost, we run a 5-NN classification on the two-dimensional embedding space using MNIST data. The modified BCE cost produced a 5-NN classification of 2.24 ± 0.06, slightly higher than the original BCE error of 2.12 ± 0.08. This experiment suggests that the reconstruction loss at the output layer doesn't help improve the generalization performance of the BCE model.

4.4 Bottleneck Centroid-Encoder and Variance

The experimental results in Section 4.3.3 establish that the proposed algorithm BCE produces more compact data reduction with visually tighter clusters and numerically improved classification. Here we analyze the BCE algorithm in more detail to discover why this improved performance is perhaps not surprising. This has to do, at least in part, with the fact that BCE, implementing a nonlinear mapping optimized at the bottleneck, captures more variance than PCA, and this is particularly important in lower dimensions.

PCA captures the maximum variance over the class of orthogonal transformations [37]. The total variance captured as a function of the number of dimensions is a measure of the complexity of a data set. We can use variance to establish, e.g., that the digits 0 and 1 are less complex than the digits 4 and 9. To this end, we use two subsets from the MNIST training set: the first one contains all the digits 0 and 1, and the second one has all the digits 4 and 9. Next, we compute the total variance captured as a function of dimension; see Figure 4.7. Clearly, the variance is more

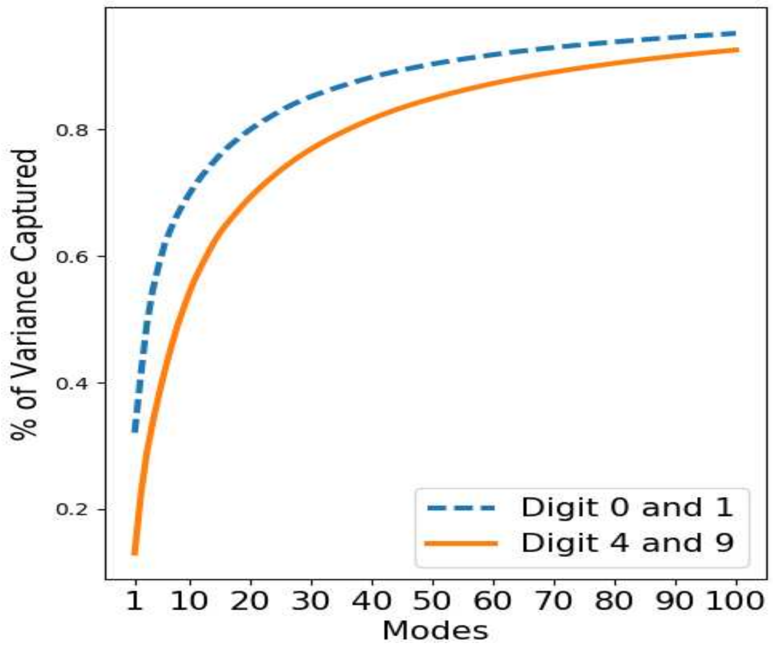
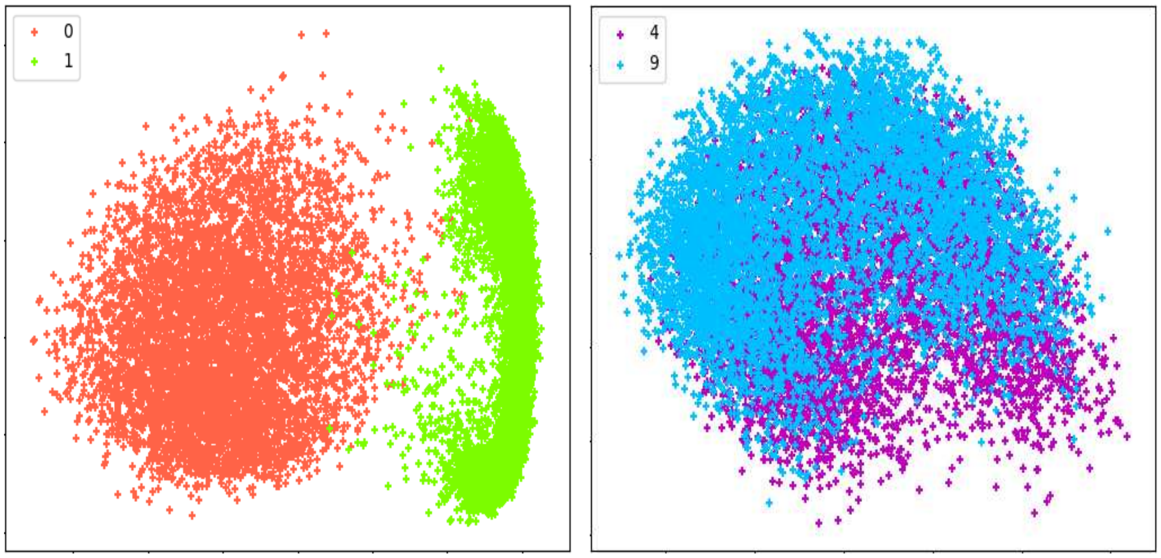


Figure 4.7: Comparison of variance plot of the two subsets using the first 100 dimensions; subset1: all the digits 0 and 1, subset2: all the digits 4 and 9 of MNIST data.



(a) Subset1: digit 0 and 1 **(b)** Subset 2: digit 4 and 9

Figure 4.8: Two dimensional embedding using PCA for subset1 and subset 2.

spread out across the dimensions for the digits 4/9. Figure 4.8, shows the visualization of these two subsets using PCA. We see that the samples in subset1 are well separated in two-dimensional space, whereas the data in subset2 are clumped together. This lies in the fact that in low dimension (2D) PCA captures more variance from subset1 (around 41%) as compared to subset2 (about 22%). We can say that subset2 is more complex than subset1 .

Now, we demonstrate how BCE maps a higher fraction of the variance of the data to lower dimensions. Again, consider the subset of digits 4 and 9 from the experiment above. We use all the digits 4 and 9 from the MNIST test set as a validation set. We trained a bottleneck centroid-encoder with the architecture $784 \to [784] \to 784$. We keep the number of nodes the same in all the three layers. After the training, we pass the training and test samples through the network and capture the hidden activation which we call BCE-transformed data. We show the variance plot and the two-dimensional embedding by PCA on the BCE-transformed data.

Figure 4.9 compares the variance plot between the original data and the BCE-transformed data. In the original data, only 22% of the total variance is captured, whereas about 92% of the total variance is captured in BCE-transformed data.

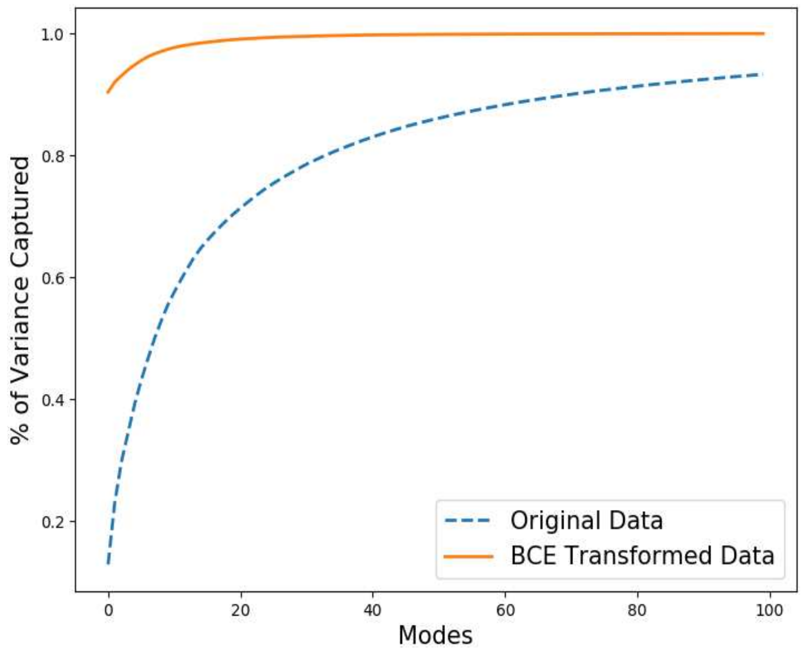
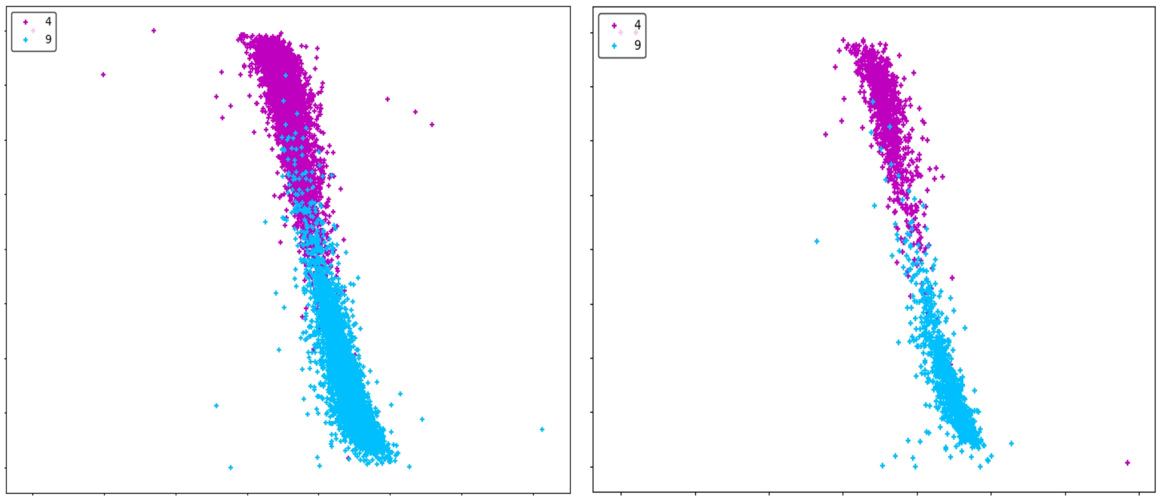


Figure 4.9: Comparison of variance plot of the original data and BCE transformed data using the first 100 dimensions.

Visualization using PCA of BCE-transformed data (both training and test) is shown in Figure 4.10. This time, PCA separates the classes in two-dimensional space. Comparing Figure 4.10 with Figure 4.8b, it appears that the transformation by bottleneck centroid-encoder is helpful for low-dimensional visualizations.



(a) Digits 4 and 9 from training set. **(b)** Digits 4 and 9 from test set.

Figure 4.10: Two dimensional embedding using PCA on the BCE-transformed data.

4.5 Conclusion

In this chapter we propose the bottleneck centroid-encoder architecture to optimize the location of centroids in the reduced space. The objective function includes terms that serve to enhance the separation between classes while making classes compact. These characteristics make BCE particularly well suited to the task of data visualization when class labels are available. Our numerical experiments showed advantages over a range of other methods and established that methods with approximately equivalent performance are considerably more computationally expensive. Our examples illustrate that BCE captures the topological structure, i.e., neighborhoods of the data, comparable to other methods such as Laplacian Eigenmaps but with generally better prediction error. These experiments also include comparisons to the unsupervised pt-SNE, autoencoders as well as the supervised MCML and NCA class of algorithms. The algorithms that have the most similar prediction errors when compared to BCE require the computation of distances between all pairs of points. We demonstrated that the 2D embedding using BCE produces classification rates that reflect an improvement over CE. We also showed empirically that the model globally captures high statistical variance relative to optimal linear transformations, i.e., more than PCA. In addition to capturing the global topological structure of the data in low-dimensions, BCE exploits data labels to minimize the within-class variance. This improves the localization of the mapping such that points are mapped more faithfully to their Voronoi regions in low-dimensions. Both CE and BCE algorithms provide a mapping model that can be applied to new data without any retraining being necessary. This is in contrast to spectral methods, such as Laplacian Eigenmaps, that require the entire training and testing set for computing the embedding eigenvectors.

Chapter 5

Sparse Centroid-Encoder

This chapter presents a non-linear feature selection technique called Sparse Centroid-Encoder (SCE). We develop the sparse optimization problem by adding a sparsity promoting ℓ_1-norm term to the objective of the centroid-encoder. The goal of the algorithm is to extract discriminatory features in groups while mapping a point to its class centroid. This approach has the advantage that it can be applied to determine all features which separate multiple classes simultaneously. The algorithm is applied to a wide variety of data sets including, single-cell biological data, high-dimensional biological data, hyperspectral data, image data, and GIS data. We compared our method to various state-of-the-art feature selection techniques, including six neural network-based models (DFS, SG-L1-NN, G-L1-NN, STG, CAE, and FsNet), Sparse SVM, and Random Forest. We empirically showed that SCE features produced better classification accuracy on the unseen test data, often with fewer or the same number of features.

5.1 Sparse Centroid-Encoder

The Sparse Centroid-Encoder (SCE) is a modification to the Centroid-Encoder architecture as shown in Figure 5.1. The input layer is connected to the first hidden layer via the sparsity promoting layer (SPL). Each node of the input layer has a weighted one-to-one connection to each node of SPL. The number of nodes in these two layer are the same. The nodes in SPL don't have any bias and non-linearity. The SPL is fully connected to the first hidden layer, therefore the weighted input from the SPL will be passed to the hidden layer in the same way that of a standard feed forward network. During training, a ℓ_1 penalty will be applied to the weights connecting the input layer and SPL layer. The ℓ_1 penalty will drive most of the weights to near zero and the corresponding input nodes/features can be discarded. Therefore the purpose of the SPL is to select important features from the original input. Note we only apply the ℓ_1 penalty to the parameters of the SPL.

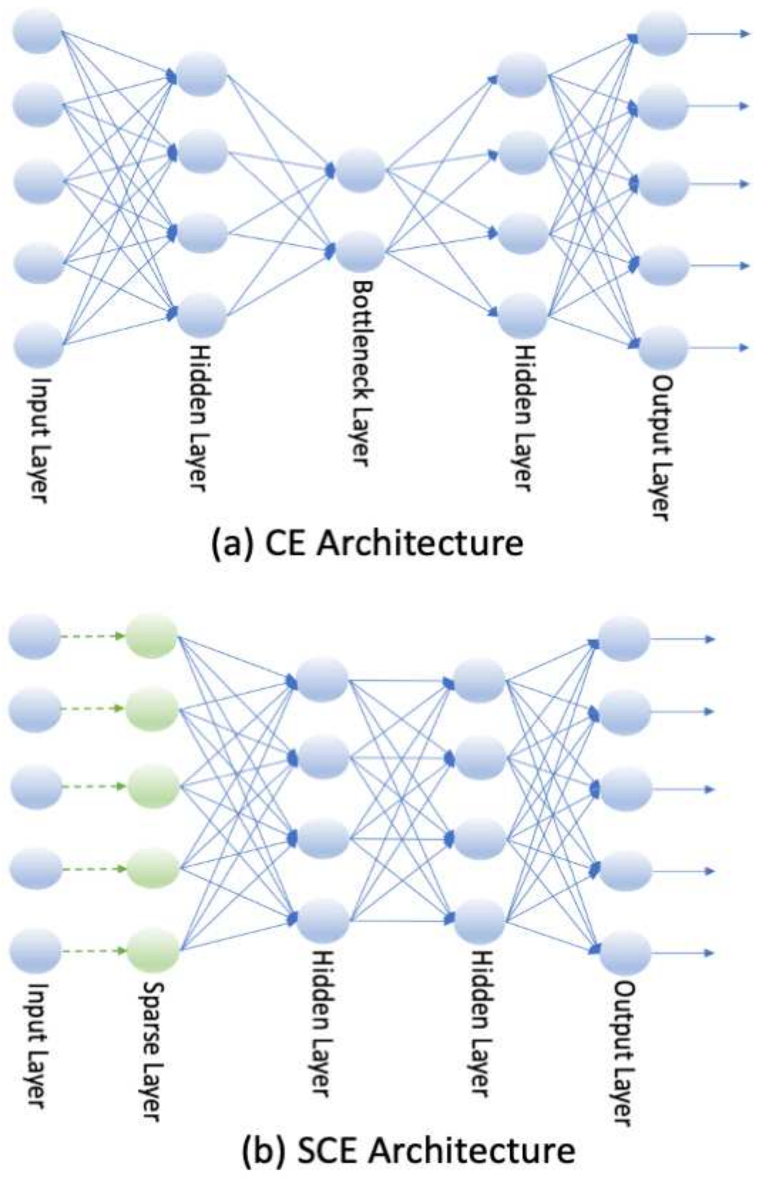


Figure 5.1: The architecture of Centroid-Encoder and Sparse Centroid-Encoder. Notice the Centroid-Encoder uses a bottleneck architecture which is helpful for visualization. In contrast, the Sparse Centroid-Encoder doesn't use any bottleneck architecture; instead, it employs a sparse layer between the input and the first hidden layer to promote feature sparsity.

Let θ_{spl} is the parameters (weights) of the SPL and θ is the parameters of the rest of the network. The cost function of sparse centroid-encoder is given in Equation 5.1 where λ is the hyper-parameter which controls the sparsity. A higher value of λ will promote higher sparsity resulting more near-zero weights in SPL. In other words λ will control the number of features selected from input.

$$\mathcal{L}_{sce}(\theta) = \frac{1}{2N} \sum_{j=1}^{M} \sum_{i \in I_j} \|c_j - f(x^i; \theta))\|_2^2 + \lambda \|\theta_{spl}\|_1 \tag{5.1}$$

Like centroid-encoder we trained sparse centroid-encoder using error back propagation, which requires the gradient of the cost function of Equation 5.1. As ℓ_1 function is not differentiable at 0, we used sub-gradient.

Feature Cut-off

The ℓ_1 penalty of the sparse layer (SPL) drives a lot of weight to near zero. Often hard thresholding or a ratio of two consecutive weights is used to pick the nonzero weight [39]. We take a different approach. After training SCE, we arrange the absolute value of the weights of the SPL in descending order. And then find the elbow of the curve. We measure the distance of each point of the curve to the straight line formed by joining the first and last points of the curve. The point with the largest distance is the position (P) of the elbow. We pick all the features whose absolute weight is greater than that of P; see panel (c), (d) of Figure5.2 and panel (b), (c) of Figure 5.5.

5.1.1 Empirical Analysis of SCE

In this section we present an empirical analysis of our model. The results of feature selection for the digits 5 and 6 from the MNIST set are displayed in Figure 5.2. In panel (a), we compare the two terms that contribute to Equation 5.1, i.e., the CE and ℓ_1 costs, weighted with different values of λ. As expected, we observe that the CE cost monotonically decreases with λ, while the ℓ_1 cost increases as λ decreases. For larger values of λ, the model focuses more on minimizing the ℓ_1-norm of the sparse layer, which results in smaller values. In contrast, the model pays more attention to minimizing the CE cost for small λs; hence we notice smaller CE cost and higher ℓ_1 cost. Panel (b) of Figure 5.2 shows the accuracy on a validation set as a function of λ; the validation accuracy reached its peak for $\lambda = 0.001$. In panels (c) and (d), we plotted the magnitude of the feature weights of the sparse layer in descending order. The sharp decrease in the magnitude of the weights demonstrates the promotion of sparsity by SCE. The model effectively ignores features by setting their weight to approximately zero. Notice the model produced a sparser solution for $\lambda = 0.1$, selecting only 32 features compared to 122 chosen variables for $\lambda = 0.001$. Figure 5.3 shows the position of the selected features, i.e., pixels, on the digits 5 and 6. The intensity of the color represents the feature's importance. Dark blue signifies a higher absolute value of the weight, whereas light blue means a smaller absolute weight.

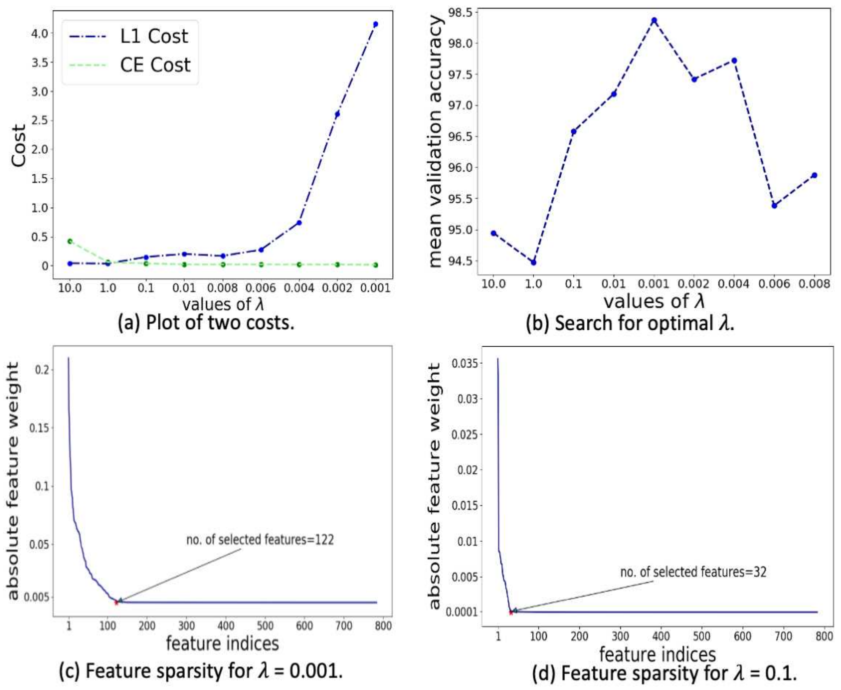


Figure 5.2: Analysis of Sparse Centroid-Encoder. (a) Change of the two costs over λ. (b) Change of validation accuracy over λ. (c) Sparsity plot of the weight of W_{SPL} for $\lambda = 0.001$. (d) Same as (c) but $\lambda = 0.1$.

Our next analysis shows how SCE extracts informative features from a multi-modal dataset, i.e., data sets whose classes appear to have multiple clusters. In this case, one center per class may not be optimal, e.g., ISOLET data. To this end, we trained SCE using a different number of centers per class where the centers were determined using standard k-Means algorithm [120, 121]. After the feature selection, we calculated the validation accuracy and plotted it against the number of

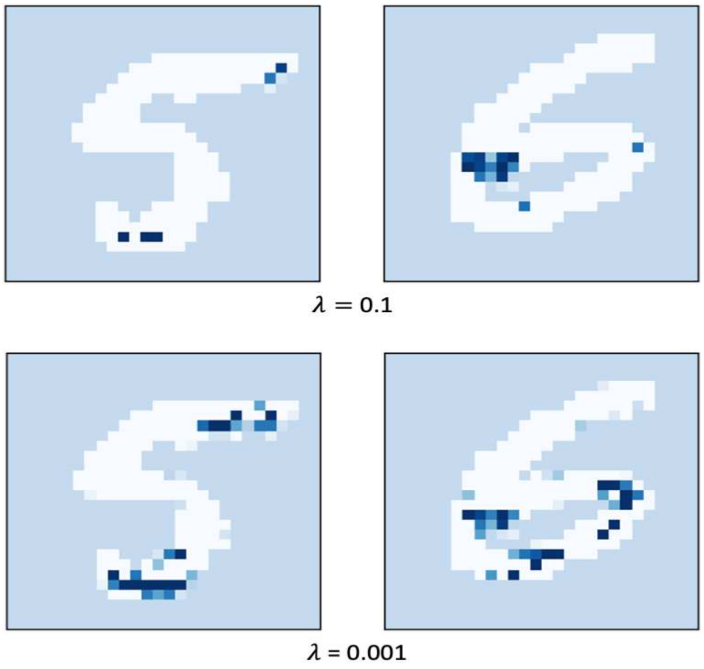


Figure 5.3: Demonstration of the sparsity of Sparse Centroid-Encoder on MNIST digits 5 and 6. The digits are shown in white, and the selected pixels are marked using blue—the darkness of blue indicates the relative importance of the pixel to distinguish the two digits. We showed the selected pixels for two choices of λ. Notice that for $\lambda = 0.1$, the model chose the lesser number of features, whereas it picked more pixels for $\lambda = 0.001$. The parameter λ is the knob which controls the sparsity of the model.

centers per class in Figure 5.4. The validation accuracy jumped significantly from one center to two centers per class. The increased accuracy indicates that the speech classes are multi-modal, further validated by the two-dimensional PCA plot of the three classes shown in panel (b)-(d).

Our last analysis sheds light on the feature selection stability of the SCE on MNIST digits as shown in Figure 5.5. In panel (a), we present the position of the selected pixels over two runs (194 and 198 respectively with 167 overlapping ones). Most of the selected pixels reside in the middle of the image, making sense as the MNIST digits lie in the center of a 28 x 28 grid. Notice that the non-overlapping pixels of the two runs are neighbors, making sense as the neighboring pixels perhaps contain similar information about the digits. On the SMK_CAN dataset, which has over 19,000 features, the ℓ_1 penalty of the sparse layer makes most of the variables to zero/near zero, selecting only 570 and 594 biomarkers. We didn't use hard thresholding to induce sparsity. Our feature cut-off technique, mentioned in Section 5.1, picks the non-zero biomarkers. Also,

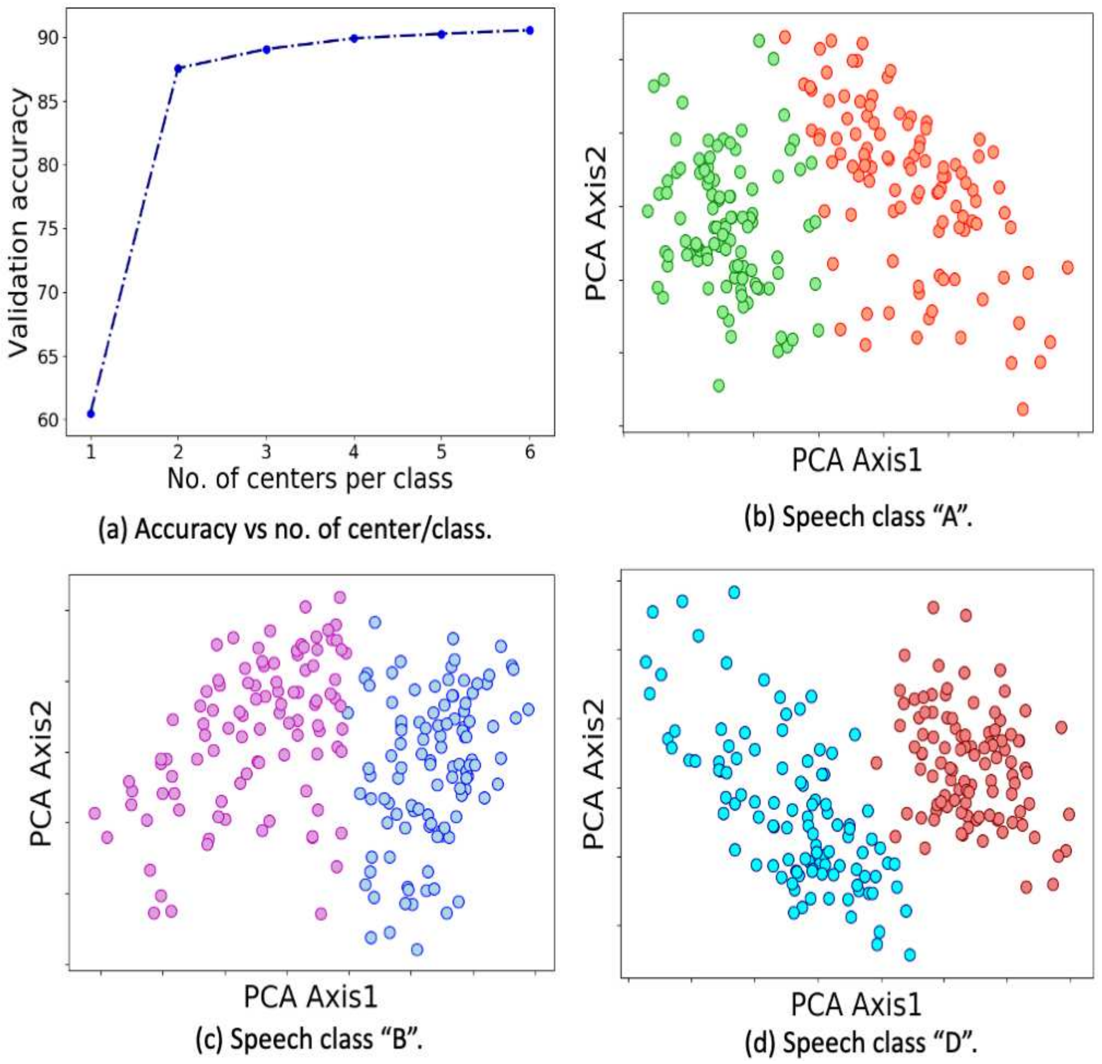


Figure 5.4: Sparse Centroid-Encoder for multi-modal data set. Panel (a) shows the increase in validation accuracy over the number of centroids per class. Panel (b)-(d) shows the two-dimensional PCA plot of the three speech classes.

notice the absolute weight of the selected biomarkers, which suggests that the ℓ_1 didn't shrink all parameters of the sparse layers.

5.1.2 Feature Selection Workflow Using Sparse Centroid-Encoder

By design, sparse methods identify a small number of features that accomplish a classification task. If one is interested in *all* the discriminatory features that can be used to separate multiple classes, then one can repeat the process of removing good features. This section describes how

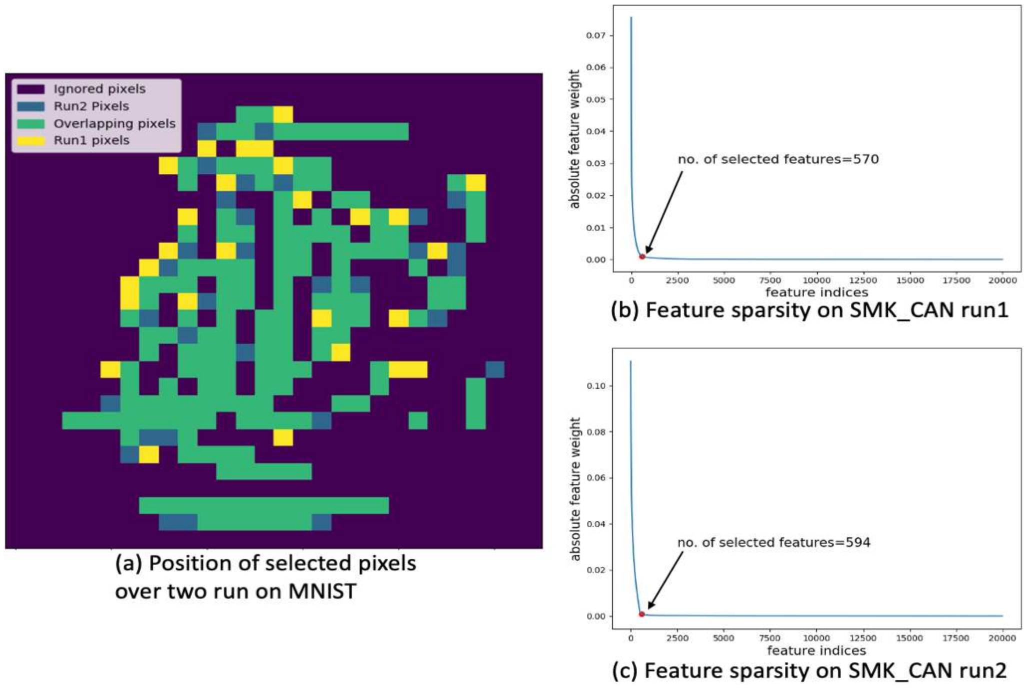


Figure 5.5: Sparse Centroid-Encoder on MNIST (all ten classes) and high dimensional (#features 19993) SMK_CAN data. Panel (a) shows position of the selected pixels over two run ($\lambda = 0.0002$). Panel (b)-(c) shows the sparsity plot ($\lambda = 0.0002$) of the SPL layer over two run on SMK_CAN training data.

sparse centroid-encoder (SCE) can be used iteratively to extract all discriminatory features from a data set; see [39] for an application of this approach to sparse support vector machines.

SCE is a model based on neural network architecture; hence, it's a non-convex optimization. As a result, multiple runs will produce different solutions, i.e., different feature sets on the same training set. These features may not be optimal given an unseen test set. To find out the robust features from a training set, we resort to frequency-based feature pruning. In this strategy, first, we divide the entire training set into k folds. On each of these folds, we ran the SCE and picked the top N (user select) number of features. We repeat the process T times to get $k \times T$ feature sets. Then we count the number of occurrences of each feature and call this number the frequency of a feature. We ordered the features based on the frequency and picked the optimum number from a validation set. We present the feature selection work flow in Figure 5.6. We trained SCE using Scaled Conjugate Gradient Descent [123].

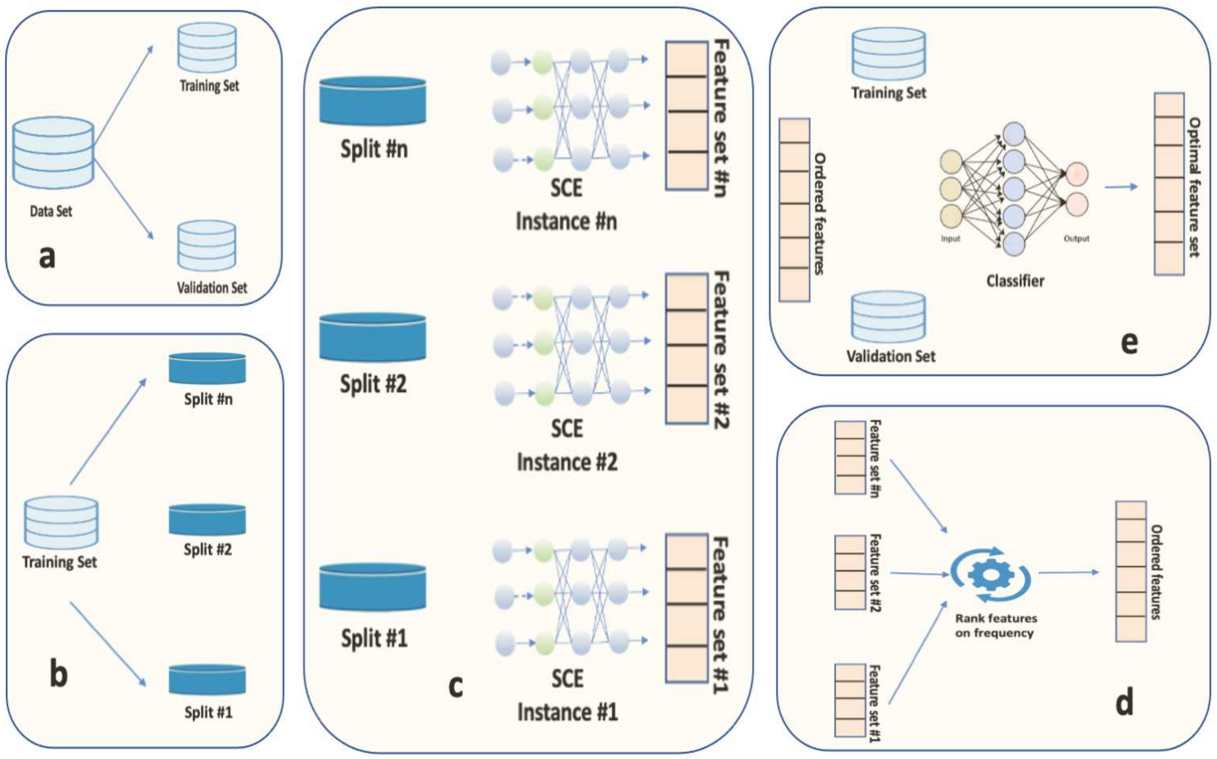


Figure 5.6: Feature selection workflow using Sparse Centroid-Encoder. **a** First, the data set has been partitioned into training and validation. **b** We further partitioned the training set into *n* splits. **c** On each of the training splits, we ran Sparse Centroid-Encoder to get *n* feature sets. **d** We calculated the occurrence of each feature among the *n* sets and called it the frequency of the feature. We ranked features from high to a low frequency to get an ordered set. **e** At last, we picked the optimum number of features using a validation set.

5.2 Experimental Results

This section presents the experiments we did on various data sets and the results compared to other feature selection models. We begin with a brief description of data sets.

5.2.1 Data sets

GM12878 It's a single cell data set. The samples were collected from the annotated DNA region of lymphoblastoid cell line. Each sample is represented by a 93 dimensional features sampled from three classes: active enhancer (AE), active promoter (AP) and background (BG) where each class contains $2,156$ number of samples. The data set is split equally into a separate training, validation and test sets.

Forest Cover It's a data set of seven forest cover types (e.g. ponderosa pine) where each sample is represented by a vector of 54 elements which were extracted from cartographic data. There are about half million samples. The data is available in UCI repository.

MNIST This is a widely used collection of digital images of handwritten digits (0..9)[10] with separate training (60,000 samples) and test set (10,000 samples). Each sample is a grey level image consisting of 1-byte pixels normalized to fit into a 28 x 28 bounding box resulting in *vecced* points in $\mathbb{R}^{784}$.

Indian Pines The hyper-spectral Indian Pine data set consists of $145x145$ pixels by 220 bands from 0.4 to 2.4 μm distributed among the sixteen classes. Often time the water absorption bands 104-108, 150-163, and 220 are discarded. But in our experiment, we did not exclude these bands with the hope that the sparse centroid-encoder should reject these bands. In our experiment, we included the pixels which had label information.

GSE73072 This microarray data set is a collection of gene expressions taken from human blood samples as part of multiple clinical challenge studies ([124]) where individuals were infected with the following respiratory viruses HRV, RSV, H1N1, and H3N2. In our experiment we excluded the RSV study. Blood samples were taken from the individuals before and after the inoculation. RMA normalization ([125]) is applied to the entire data set, and the LIMMA ([126]) is used to remove the subject-specific batch effect. Each sample is represented by 22,277 probes associated with gene expression. The data is publically available on the NCBI GeneExpression Omnibus (GEO) with identifier GSE73072.

5.2.2 Experimental Details

We did three bench-marking experiments to compare the sparse centroid-encoder with other state-of-the-art feature selection methods. To make the evaluation objective, we compared the classification accuracy on the unseen data using the selected features of different models. All the three experiments share the following workflow:

[10]The data set is available at http://yann.lecun.com/exdb/mnist/index.html.

- Using the training samples run sparse centroid-encoder to select optimal number of features. Table 5.1 shows the ℓ_1 penalty used for the data sets.
- Build K classification models with these features on the training set. We used centroid-encoder as the classification model [23].
- Compute the accuracy on the sequestered test set using the K trained models and report the mean accuracy with standard deviation.

Table 5.1: The penalty term λ used for the four data sets.

Data set	Penalty term (λ)
GM12878	0.01
MNIST	0.001
Forest Cover	0.01
Indian Pine	0.01

We follow this common evaluation strategy in the three bench-marking experiments so that our CE results are comparable to the published results. We describe details of each experiment including how the training and test partitions were selected below.

Experiment 1: The first bench-marking experiment is done on the single-cell GM12878 data set. We took the published result of deep feature selection (DFS), shallow feature selection, LASSO, and Random Forest (RF) from the work of Li et al. [6] to evaluate our model. The data set has separate training validation and a test set of equal size. We used the validation set to tune hyper-parameters and to pick the optimal number of features. After the feature selection step, we merged the training and validation set and trained $K = 10$ centroid-encoder classifiers with the selected features, and reported classification accuracy on the test set.

Experiment 2: We conducted the second experiment on the widely studied MNIST data set along with the Forest Cover data. Following the experimental setup of Scardapane et al. [7], each data set was randomly partitioned into a training and test set with a ratio of 75:25. We used 20% of the training sample as a validation set to select the optimum number of features and hyper-parameters. After the feature selection, a centroid-encoder is trained to predict the class label of the unseen test cases. We repeated the process $K = 25$ times and reported the mean accuracy with standard deviation.

Experiment 3: Our third bench-marking experiment compared SCE with the Sparse Support Vector Machine (SSVM) on the Indian Pine data set. For a fair comparison, we followed the same experimental protocol as done by Chepushtanova et al. in their work [8]. The entire data set is split in half into a training and test set. Because of the small size of the training set, we did a 5-fold cross-validation on the training samples to tune hyper-parameters. After the feature selection on the training set, we took top $n = 1, 2, 3, 4, 5, 10, 20, 40, 60, 80$ features to build a centroid-encoder classifier on the training set to predict the class labels of the test samples. For each n we repeat the classification task $K = 10$ times. In the article [8], the authors used spatial smoothing, which significantly improved their classification result. We compared the efficacy of SCE and SSVM features without spatial smoothing as our goal is not to report high accuracy.

Experiment 4: We conducted our last experiment on the GSE73072 human respiratory infection data where the goal is to predict the classes control, shedders, and non-shedders at the very early phase of the infection, i.e., at time bin spanning hours 1-8. Controls are the pre-infection samples, whereas shedders and non-shedders are post-infection samples picked from the time bin 1-8 hr. Shedders actually disseminate virus while non-shedders do not. We considered six studies, including two H1N1 (DEE3, DEE4), two H3N2 (DEE2, DEE5), and two HRV (Duke, UVA) studies. We used 10% training samples as a validation set—the training set comprised all the studies except for the DEE5, which was kept out for testing. We did a leave-one-subject-out (LOSO) cross-validation on the test set using the selected features from the training set. In this experiment we compared SCE with Random Forest (RF).

5.2.3 Quantitative and Qualitative Analysis

Now we present the results from a comprehensive quantitative and qualitative analysis across all the data sets.

GM12878

Table 5.2 presents the classification accuracy on the test data using the features selected by various methods. As with Li et al., we used the top 16 features to report the mean accuracy of the test samples. Besides that, we report the test accuracy using the top 48 features picked from the validation set. The classification using the top 48 SCE features is the best in the lot. When restricted to the top 16, we see that the SCE features outperform all the other models. Note that the accuracy of the deep DFS features is 2.66% lower than our model. We also reported the accuracy of SCE without the feature selection workflow. In this case, we ran SCE and took the top 16 features to classify the test samples. We repeated the step 10 times and presented the average result. Note that the SCE features without the framework performed better than the DFS by approximately 2%.

Table 5.2: Classification accuracies using the top 16 features by various techniques. Results of Deep DFS, Shallow DFS, LASSO, and Random Forest are reported from [6]. We present accuracy with the top 48 features which were selected from a validation set.

Feature Selection Method	No. of Features	Accuracy
SCE	**48**	**89.40 ± 0.24**
SCE	**16**	**88.33 ± 0.13**
SCE w/o framework	**16**	**87.51 ± 0.89**
DFS	16	85.67
Shallow DFS	16	85.34
LASSO	16	81.86
Random Forest	16	88.21

Among all the models, LASSO features exhibit the worst performance with an accuracy of 81.86%. This relatively low accuracy is not surprising, given LASSO is a linear model.

The classification performance gives a quantitative measure that doesn't reveal the biological significance of the selected genes. We surveyed the functionality of the top genes selected by sparse centroid-encoder model and provided the description below from the literature. We see that many of these genes are related to the proliferation of the lymphoblastoid cancer cells, e.g., POL2, NRSF/REST, GCN5, PML, etc. Some genes play an essential role in transcriptional activation, e.g., H4K20ME1, TAF1, H3K27ME3, etc. Gene H3K27AC plays a vital role in separating active enhances from inactive ones. This survey confirms the biological significance of the selected genes.

- **POL2(POLR2A)**: It's a subunit of RNA polymerase II, which interacts with nuclear CD26 using a chromatin immunoprecipitation assay. This interaction led to transcriptional repression of the POLR2A gene, resulting in a proliferation of cancer cells [127].

- **H4K20ME1** This gene has been implicated in transcriptional activation. Recent studies showed a strong correlation between H4K20me1 and gene activation in the regions downstream of the transcription start site [128].

- **NRSF(REST)** NRSF/REST is highly expressed in non-neuronal tissues like the lung. The findings of Kreisler et at. [129] support that NRSF/REST may act as an essential modulator of malignant progression in small-cell lung cancer.

- **TAF1** It's the largest integral subunit of TFIID, initiates RNA polymerase II-mediated transcription. Wang et al. discovered a critical promoter-binding function of TAF1 in transcription regulation [130].

- **H3K27AC** This gene distinguishes active enhancers from inactive/poised enhancer elements containing H3K4me1 alone [131].

- **GCN5** GCN5 functions as a transcriptional coactivator of E2f1 target genes. In small-cell lung cancer, E2F1 recruits GCN5 to acetylate H3K9, facilitating transcription of E2F1, CYCLIN E, and CYCLIN D1 (39) all of which promote cellular proliferation and tumor growth [132].

- **PML** The PML gene provides instructions for a protein that acts as a tumor suppressor, which means it prevents cells from growing and dividing too rapidly or in an uncontrolled way [133].
- **RUNX3** This gene binds to the core DNA sequence 5'-PYGPYGGT-3' found in several enhancers and promoters. It also interacts with other transcription factors. It functions as a tumor suppressor, and the gene is frequently deleted or transcriptionally silenced in cancer [134].
- **ZZZ3** It's prortein binding gene which oftens promotes gene activation [135].
- **H3K27ME3** This gene can function as silencers to regulate gene expression [136].

MNIST and Forest Cover

Now we present the results of Experiment 2, where we compare our sparse model with [7] on MNIST and Forest Cover data.

Table 5.3: Classification result using the top features by various models on the MNIST data set. Results of SG-L1-NN, L1-NN, and L2-NN are reported from [7].

Feature Selection Method	Average no. of Features	Accuracy
Sparse Centroid-Encoder	**355.32**	**98.44 $\pm$ 0.08**
SG-L1-NN	581.8	97.00
L1-NN	658.2	97.00
L2-NN	676.4	98.00

Table 5.3 shows the classification accuracy using the features selected by four methods. The features of SCE produce the best accuracy beating the other sparse models SG-L1-NN and L1-NN by a margin of 1.44%. Note that our classification accuracy is achieved using 355 number of features on average, which is approximately 45% of the total number of pixels of an MNIST image. On the other hand, SG-L1-NN, L1-NN, and L2-NN used 74%, 84%, and 86% of the total number of features, respectively. We further analyze the features from a qualitative point of view.

We present a visual representation of the selected variables of the SCE model in Figure 5.7 where we show the spatial location of the selected pixels in a 28 x 28 grid.

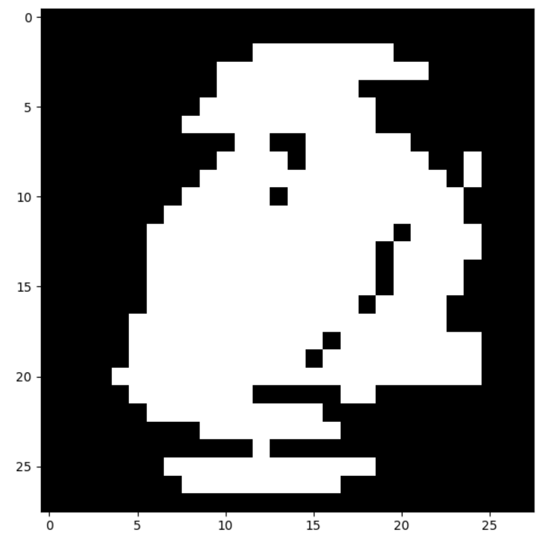


Figure 5.7: Locations of selected features of MNIST image shown in a 28 x 28 grid. The selected pixels are marked in white, and the ignored pixels are marked in black.

Observe that most of the selected pixels are located in the middle of the grid, making sense as most MNIST digits are placed in the middle of the 28 x 28 bounding box. This fact establishes the robustness of the features of the Sparse Centroid-Encoder.

The classification accuracy of the four models on the Forest Cover data is presented in Table 5.4. The result shares a similar trend to the MNIST set. The average test accuracy is better using the SCE features. At the same time, the number of SCE features is considerably less than the other sparse models of Scardapane et al. [7]. Note that the mean accuracy of our model is higher by a margin of $3\% - 4\%$. It's noteworthy that the models of Scardapane et al. required most of the features for classification. The data set lives in $\mathbb{R}^{54}$ in the ambient space. Out of these 54 variables, SG-L1-NN, L1-NN, and L2-NN utilized $52.7, 53$, and 54 features, respectively. In contrast, our model only used 38 features on average. The high test accuracy with a relatively small feature set establishes the value of SCE as a robust variable selection technique.

Table 5.4: Classification results using the top features by various models on the Forest Cover data set. Results of SG-L1-NN, L1-NN, and L2-NN are reported from [7].

Feature Selection Method	Average no. of Features	Accuracy
Sparse Centroid-Encoder	**38.1**	**87.37 $\pm$ 0.36**
SG-L1-NN	52.7	83.00
L1-NN	53.0	83.00
L2-NN	54.0	84.00

Indian Pines

Here we discuss the result of our third experiment, which we did on the Indian Pine data set. We chose a well know feature selection model, Sparse Support Vector Machine (SSVM) [8] to compare with SCE. The task is to identify the bands which are essential to assign a test sample correctly to one of the sixteen classes. We took all the 220 bands in the feature selection step, including the twenty water absorption bands. Note, in the literature these noisy water absorption bands are often excluded before the experiments [137, 138]. We wanted to check whether our model was able to reject them. In fact, our model did discard them; we didn't see any of the water absorption bands in the top 100 features. It appeared that SSVM included some of these noisy bands [8].

Figure 5.8 presents the accuracy on the test data using the top n bands ($n = 1, 2, 3, 4, 5, 10, 20$ $40, 60, 80$) which were calculated on the training set. Note we didn't use spatial smoothing as done in [8]. Classification using SCE features generally produces better accuracy. Notice that SCE features yield better classification performance using fewer bands. In particular, the accuracy of the top SCE feature (band 13) is at least 15% higher than the top SSVM feature (band 1).

We listed the top ten features from each model in Table 5.5. We have included the WaLuMI + SSVM model, where SSVM is applied to the WaLuMI features to prune the set further. There is no common feature among these three sets.

Respiratory Infections in Humans

Now we present the results of the last experiment on GSE73072 data set. The results on this data set in shown in Table 5.6. The top 35 features of SCE produce the best Balanced Success

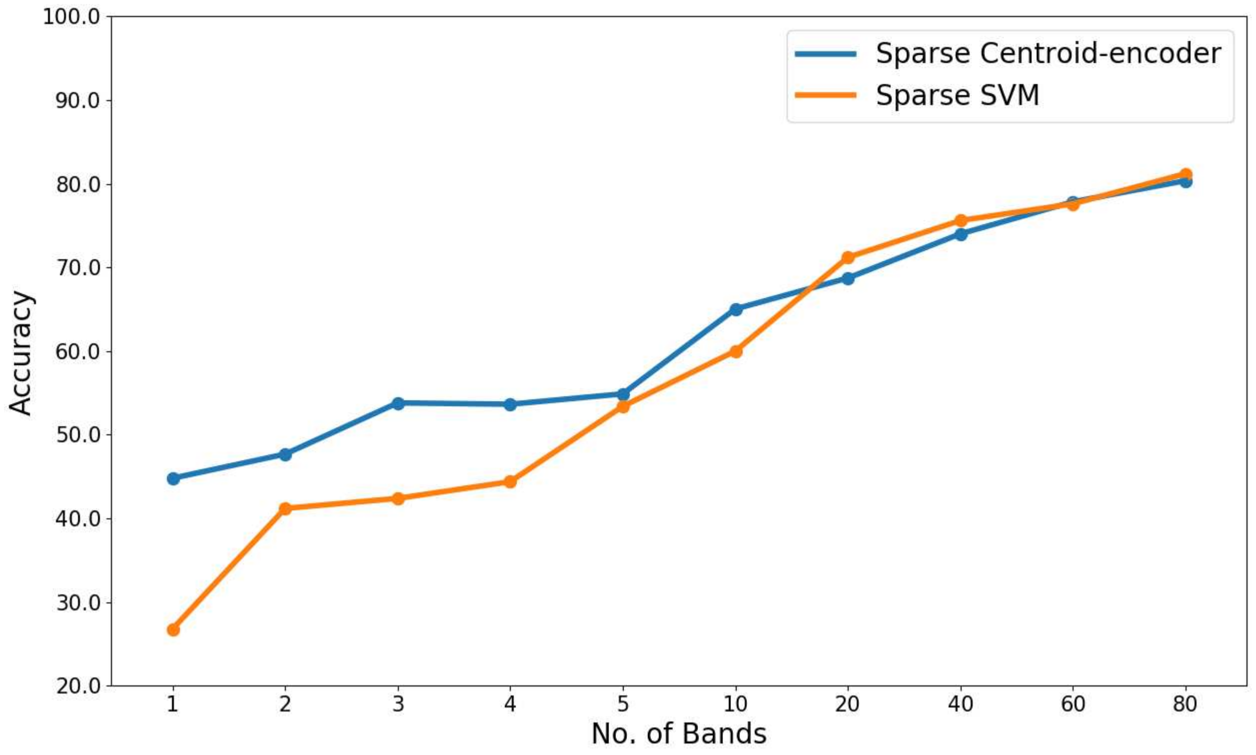


Figure 5.8: Comparison of classification accuracy using SCE and SSVM features on Indian Pine data set.

Table 5.5: List of top ten bands from each model. Bands selected by SSVM and SSVM + WaLuMI are reported from [8].

SCE	SSVM	WaLuMI + SSVM
13,148,51,47,49	1,34,2,3,29	5,25,100,55,183
43,45,17,134,44	32,41,39,28,42	129,79,52,68,88

Rate (BSR) of 90.61% on the test study DEE5. For the Random Forest model, the best result is achieved with 30 features. We also included the results with 35 biomarkers, but the BSR didn't improve. Note both the models picked a relatively small number of features, 30 and 35 out of the 22,277 genes, but SCE features outperform RF by a margin of 7%. Although RF selects features with multiple classes using a single model, it weighs a single feature by measuring the decrease of out-of-bag error. In contrast, SCE looks for a group of features while minimizing its cost. We think the multivariate approach of SCE makes it a better features detector than RF.

Time Bin	Model	No. of Features	BSR
$1-8$	SCE	35	90.61 ± 2.38
	RF	30	83.05 ± 2.42
	RF	35	82.65 ± 2.51

Table 5.6: Balanced success rate (BSR) of LOSO cross-validation on the DEE5 test set. The selected features from training set is used to predict the classes of control, shedder, and non-shedder.

5.3 Comparative Analysis between SCE and DFS

In this section, we conducted several experiments to further compare and contrast the SCE and DFS models. We used the code provided by the authors to run the experiments with DFS. DFS minimizes the cross-entropy loss and the ℓ_1-norm of the sparse layer. On the other hand, SCE minimizes the CE loss along with the ℓ_1-norm of the sparse layer. The first experiment analyzed how the costs change over λ for each model. Figure 5.9 showed the plots for DFS and SCE for different values of λ. In panel (b), we compare the two terms that contribute to Equation 5.1, i.e., the centroid-encoder and ℓ_1 costs, weighted with different values of λ. As expected, we observe that the CE cost monotonically decreases with λ, while the ℓ_1 cost increases as λ decreases. For larger values of λ, the model focuses more on minimizing the ℓ_1-norm of the sparse layer, which results in smaller values. In contrast, the model pays more attention to minimizing the CE cost for small λs; hence we notice smaller CE cost and higher ℓ_1 cost.

In contrast, the DFS model exhibited a completely different pattern of the two costs over λ, see panel (a). The cross-entropy for the network remains constant for $\lambda = 100, 10, 1, 0.1$, and after that the value decreases. The ℓ_1-norm of the sparse layer, i.e., ℓ_1 cost, changes abruptly with λ. For example, the ℓ_1 loss for $\lambda = 100$ should be lower than $\lambda = 10$; but the values are exactly the opposite. We see the same pattern for $\lambda = 0.1, 0.01$. This experiment suggests that the costs change expectedly for SCE, whereas the behavior of DFS is not robust.

Now we compare the ability to promote sparsity of these two models. We plotted the absolute values of weights in the sparse layer in descending order for the six values of λ and called this plot a sparsity plot. Figure 5.10 and Figure 5.11 presents the sparsity plot of DFS and SCE respectively.

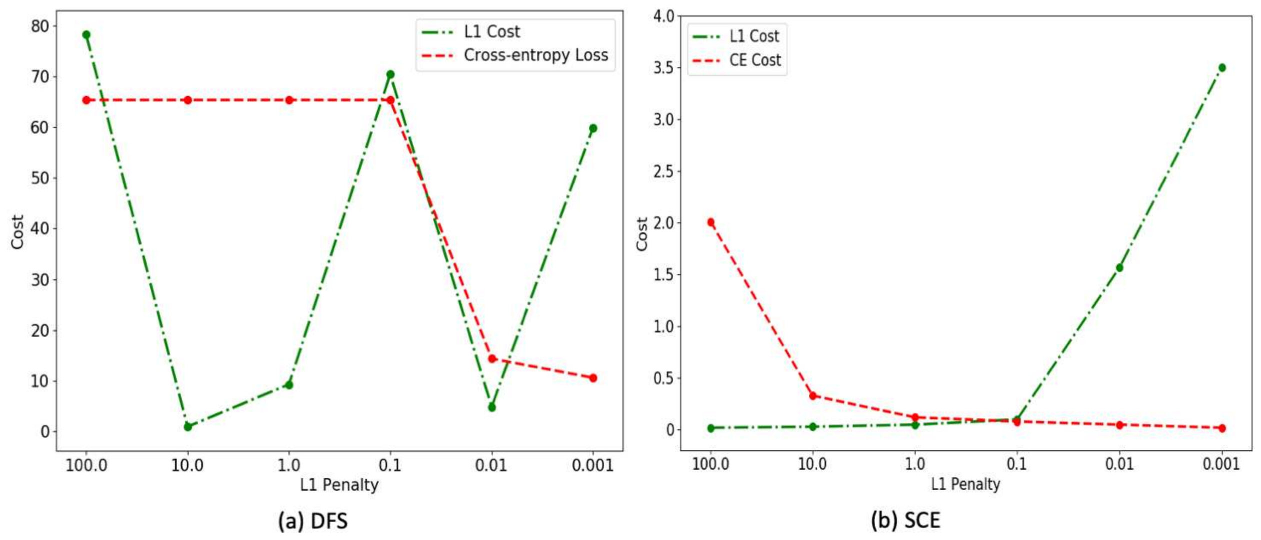


Figure 5.9: (a) Change of the Cross-entropy loss and ℓ_1 cost of DFS over different values of λ. (b) Change of the Centroid-Encoder loss and ℓ_1 cost of SCE over different values of λ. The experimrnt is done on GM12878 data set.

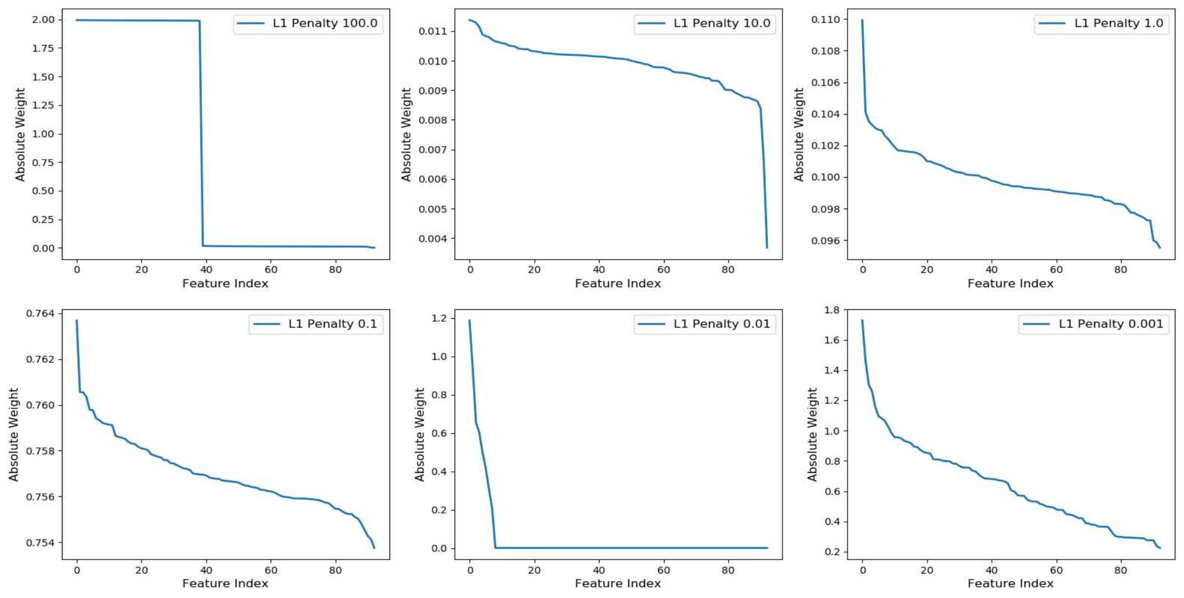


Figure 5.10: Sparsity plot of DFS on GM12878 data set.

In general, SCE is a more robust approach for weight-sparsity promotion. Notice that a higher value of λ doesn't promote sparsity; rather, it shrinks the weights of the sparse layer. This phenomenon is known as the shrinkage problem of the ℓ_1 norm [139–141]. Note that the DFS model provides a sparse solution only for $\lambda = 0.01$, which shows the sensitivity of the DFS to the choice of λ. Our finding is consistent with the current literature where [10] Yamada et al. also reported

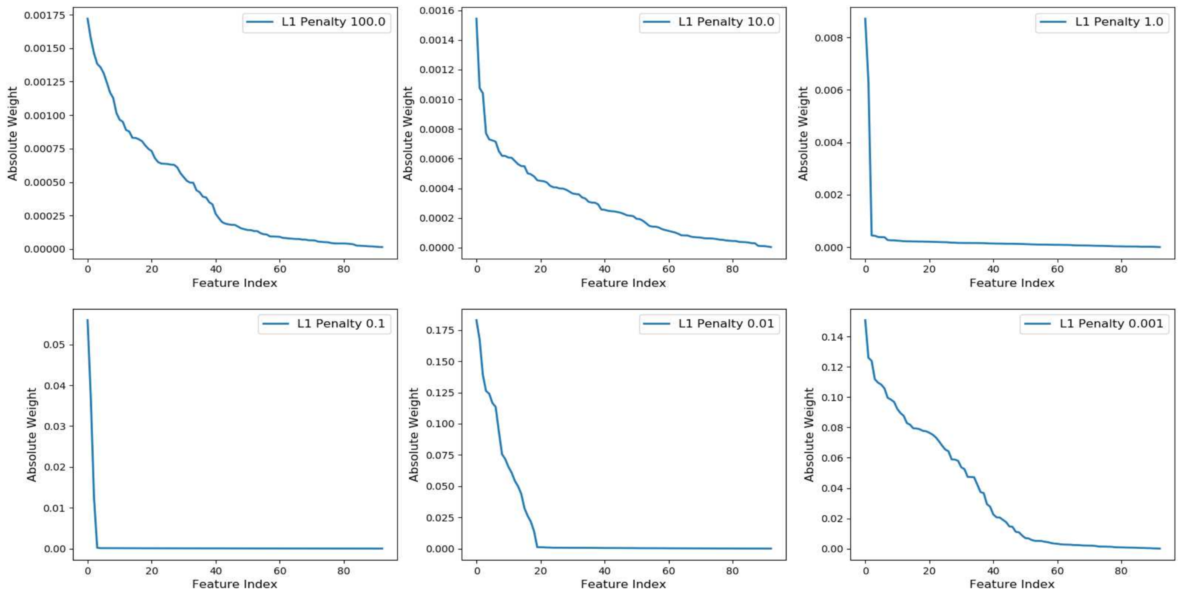


Figure 5.11: Sparsity plot of SCE on GM12878 data set.

the lack of sparsity of DFS. Note for $\lambda = 100$, the model picks around 40 features, which gives a classification accuracy of 33.33% on the test set; clearly, these features are not discriminatory.

5.4 Challenges of Minimizing 1-norm using Stochastic Optimization

The previous section shows that the DFS is more sensitive to parameter tuning and fails to induce robust feature-sparsity when compared to SCE. The optimization of the parameters of DFS is done using stochastic optimization (stochastic gradient descent) on the mini-batches. On the other hand, SCE uses scaled conjugate gradient descent (SCG [123]) on the entire training set. One of the advantages of SCG is that it can find out a suitable stepsize/learning rate at each iteration. On the other hand, stochastic optimizations like Adam [49] or stochastic gradient descent (SGD) require hyper-parameters, e.g., learning rate, mini-batch size, momentum, etc. Calculating the gradient on a random subsample (mini-batches) of a training set might add noise that may affect ℓ_1-norm minimization [49]. This section investigates the effect of stochastic optimization on ℓ_1-norm optimization. We did an array of experiments to evaluate the dependencies of the hyper-parameters on ℓ_1-norm minimization.

All the experiments in this section use Sparse Centroid-Encoder on MNIST data set. This time we use Adam to optimize the network parameters over mini-batches. We used one hidden layer with 500 ReLU activation units with a learning rate and λ set to 0.01 and 0.0001, respectively. In Figure 5.12, we present the result of the first experiment, where we show the effect of the size of the mini-batch. The three columns (A, B, and C) show results for a specific choice of mini-batch, i.e., 512, 1024, and 5000. For each column, the upper panel shows the position of the top 200 selected pixel, and the lower panel shows the absolute weight of the sparse layer in descending order.

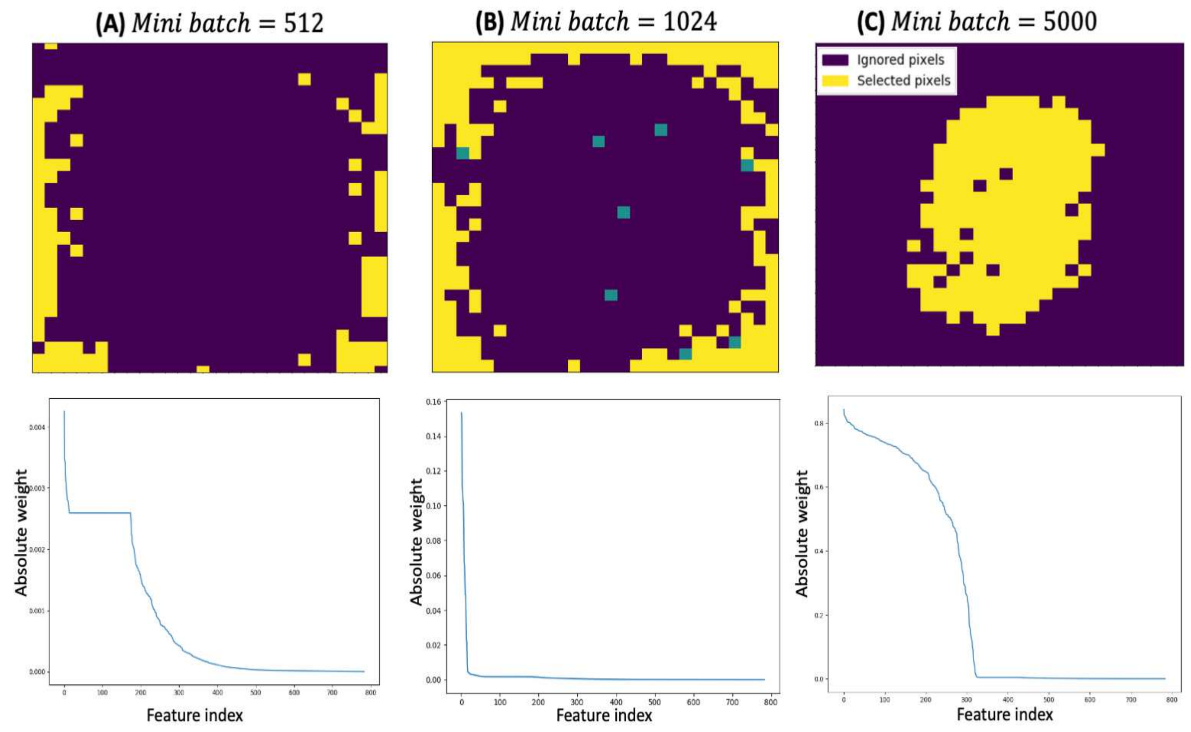


Figure 5.12: Effect of the size of mini-batch on ℓ_1-norm minimization using SCE for three choices of mini-batches- 512 in (A), 1024 in (B), and 5000 in (C). For each case, the upper panel shows the position of the selected pixels in a 28 x 28 grid, and the lower panel presents the absolute weight of the sparse layer in descending order.

Minimizing the ℓ_1-norm with smaller mini-batches (512) doesn't induce sparsity. Surprisingly, the ℓ_1-norm of the sparse layer put higher weight on the pixels around the border, ignoring the pixels in the center of the image. The model selects only 8 pixels (color teal) for mini bath 1024; among them, four pixels reside at the image's border. The position of the pixels and the sparsity plot improves significantly for mini-batch size 5000, selecting around 300 pixels from the center of the picture. Also, notice that the scale of the absolute weight increases with the size of the min-batch. We saw similar observations while working with a hyperbolic tangent ('tanh') activation function, i.e., the relation between the mini-batch size and the sparsity doesn't change if we switch from ReLU to tanh.

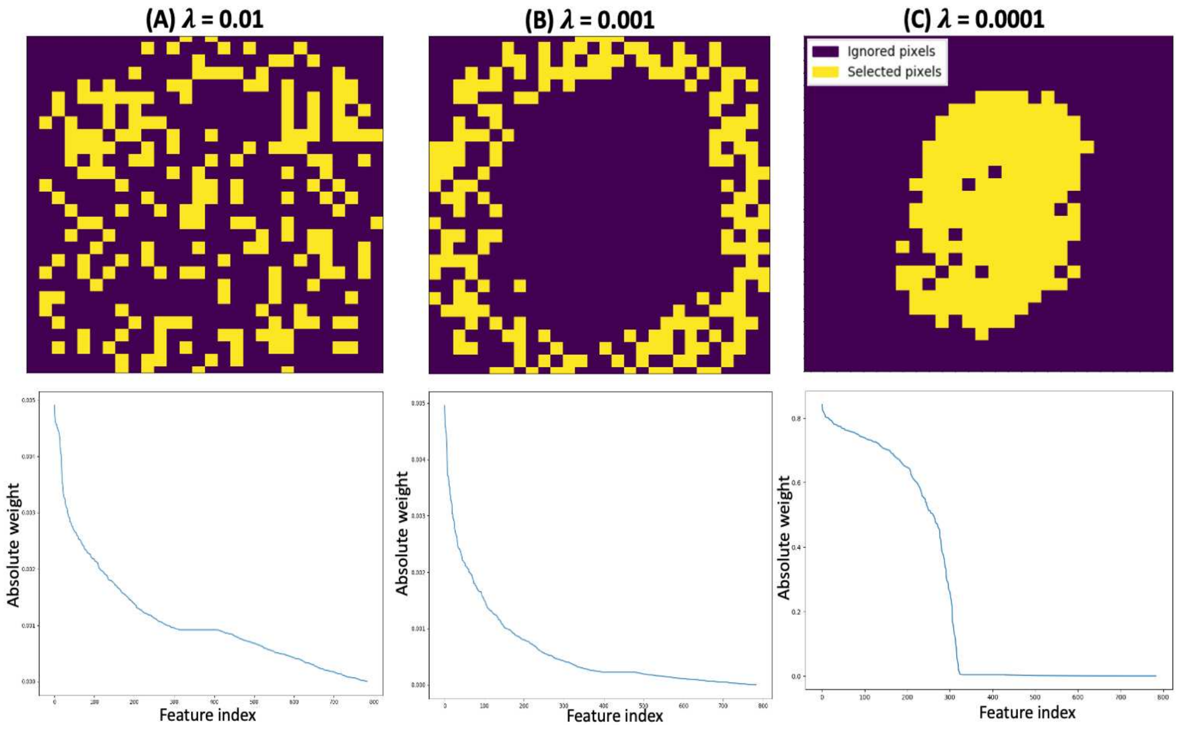


Figure 5.13: Effect of λ on ℓ_1-norm minimization using SCE for three values 0.01 in (A), 0.001 in (B), and 0.0001 in (C). For each case, the upper panel shows the position of the selected pixels in a 28 x 28 grid, and the lower panel presents the absolute weight of the sparse layer in descending order.

Figure 5.13 shows the result of the second experiment where we study the effect of penalty term λ for three different values 0.01 (panel A), 0.001 (panel B), and 0.0001 (panel C) for a fixed mini-batch size of 5000 and learning rate of 0.01. We used one hidden layer of 500 ReLU units.

Notice that the model didn't promote sparsity for $\lambda = 0.01, 0.001$. The ℓ_1-norm of the sparse layer selects pixels from all over the image when $\lambda = 0.01$; in contrast, $\lambda = 0.001$ ignores the middle of the images and picks pixels from the boundary. Interestingly, the selected pixels for $\lambda = 0.001$ form a circle. Clearly, these two values of λ won't pick the most informative features from an MNIST image. On the other hand, we see a sparser solution for $\lambda = 0.0001$, selecting around 325 features from the middle of the 28 x 28 grid. The position of the selected pixels also makes sense as the digits lie in the center of the grid.

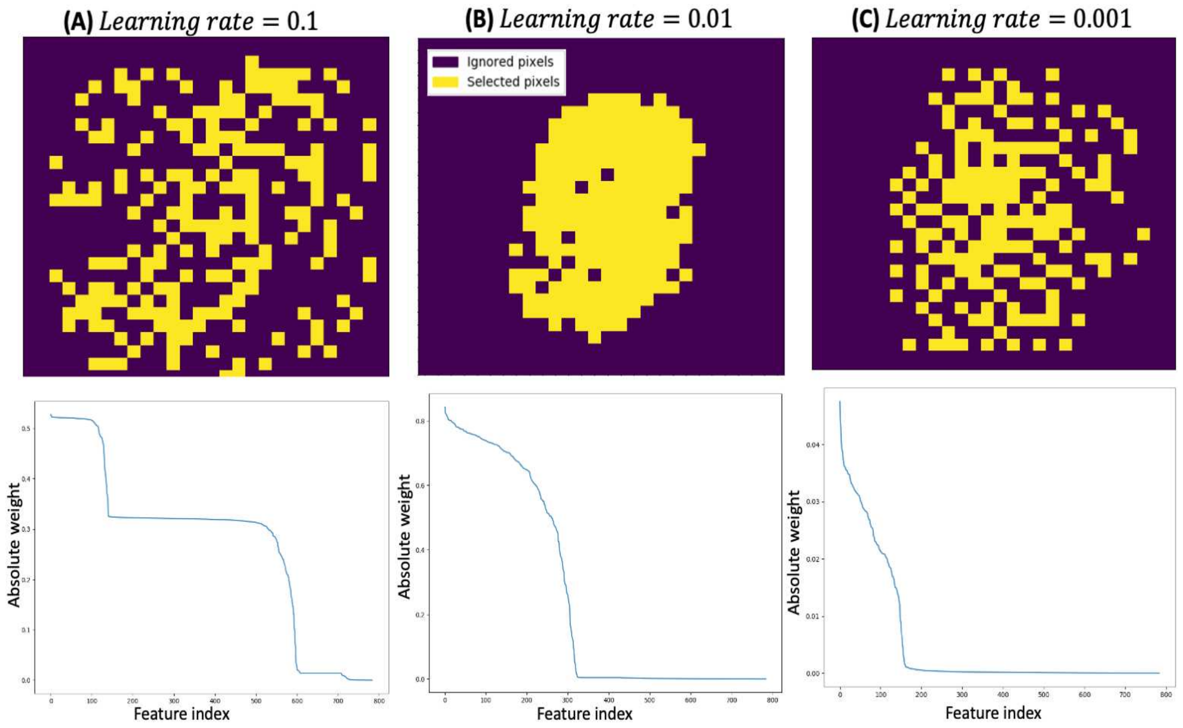


Figure 5.14: Effect of learning rate on ℓ_1-norm minimization using SCE for three values 0.1 in (A), 0.01 in (B), and 0.001 in (C). For each case, the upper panel shows the position of the selected pixels in a 28 x 28 grid, and the lower panel presents the absolute weight of the sparse layer in descending order.

In the last set of experiments of this section, we show the effect of the learning rate/step size on the model's sparsity for $\lambda = 0.0001$ and mini-batch size of 5000. Figure 5.14 shows the results. For a relatively larger step size (0.1), the model didn't induce sparsity, and a lot of selected pixels (top 200) lie on the border of the image, suggesting the presence of noise. In contrast, the model produces sparse solutions for learning rates of 0.01 and 0.001. In both cases, the selected pixels also reside in the middle of the image.

Stochastic optimizations have been successfully used in deep neural networks, especially for image data classification, where feature selection is not the primary objective. To the best of our knowledge, Scardapane et al. [7] and Li et al. [6] used ℓ_1-norm in their work. But the authors didn't present a detailed analysis of sparsity in their work. Recently, Yamada et al. [10] reported that the work of Scardapane et al. for feature selection didn't promote sparsity on high-dimensional biological data sets. Although, the authors didn't investigate the reason. The in-depth analysis of this Section reveals an essential aspect of stochastic optimization when minimizing the ℓ_1-norm. We have observed that the hyper-parameters of stochastic optimization play a crucial role. Smaller minibatch size and higher λ don't promote feature sparsity, and the selected features perhaps contain noise. The learning rate also dictates the sparsity when other hyper-parameters are kept constant. These challenges can be overcome by carefully tuning the hyper-parameters using a validation set. So, in summary, minimizing ℓ_1-norm using stochastic optimization is challenging; and requires a careful selection of hyper-parameters to induce feature sparsity.

5.5 Application of SCE on High Dimensional Biological Data set

The experimental results in Section 5.2 show the efficacy of SCE features on four benchmarking data sets while comparing to other methods. These data sets have more samples than the number of features. Now, we present applications of SCE on high-dimensional biological data sets where the number of cases is much less than the number of features. To this end, we picked three infectious disease data sets, namely Salmonella, and Ebola. In the following sections, we present the data sets' descriptions and experimental setups, and results.

5.5.1 Feature Selection of Salmonella Data set

Here we analyzed the transcriptomic data (RNAseq) associated with the Collaborative Cross Mice model of Salmonella. The data was collected by the biologists of the Texas A&M team to investigate the mechanisms of tolerance to Salmonella. Samples were taken from the Liver and Spleen of mice. Each instance is represented by a vector of 55471 elements(genes) that belong to either tolerant, susceptible, or resistant class. With this data set, we ran four experiments to extract discriminative features. We further validated the discriminatory power of the selected features using classification on unseen data and visualization. Table 5.7 shows the sample statistics of the data set.

Table 5.7: Number of samples in each class of Salmonella data.

Organ	Category		
	Tolerant	Resistant	Susceptible
Liver	75	57	72
Spleen	75	57	71

Experimental Setup: We ran the following experiments:

- Tolerant vs Resistant vs Susceptible using Liver Samples
- Tolerant vs Resistant vs Susceptible using Spleen Samples
- Tolerant vs Susceptible using Liver Samples
- Tolerant vs Susceptible using Spleen Samples

For each of these experiments, we use a separate test set, which was not used in the feature selection process, to test the efficacy of the selected features. After the feature selection process, we trained a model on the training set with the chosen features; then, we tested the model with the sequester data set. To create the sequester test set, we split the original data set into training and test sets. We took 80% of total samples from each class into the training set and the rest of 20% in the test set. Table 5.8 shows the count of samples of each class in training and test sets. We kept 10% of training samples as a validation set to find the optimal number of features.

Table 5.8: Number of samples per class in training and test set for the three class problem.

Organ	Train			Test		
	Tolerant	Resistant	Susceptible	Tolerant	Resistant	Susceptible
Liver	60	45	57	15	12	15
Spleen	60	45	56	15	12	15

Results: In Table 5.9 we present the number of features extracted by SCE in each experiments.

Table 5.9: Number of features extracted by SCE for different classification experiments.

Experiment	Organ	No. of features
Tolerant vs Resistant vs Susceptible	Liver	17
Tolerant vs Resistant vs Susceptible	Spleen	27
Tolerant vs Susceptible	Liver	9
Tolerant vs Susceptible	Spleen	5

Once the feature extraction is done, we trained a Linear SVM model on the training set using the extracted features. Then we predicted each test sample (test samples are represented with the selected features) using the trained model and reported the balanced success rate in Table 5.10. In two of the four cases, the selected features predicted the test samples with 100% accuracy, and in the other two cases, the balanced success rate is over 95%. The high prediction accuracies prove the efficacy of the selected features. The 100% accuracy on the tolerant vs. susceptible experiment indicates a possible tolerant mechanism of the mice model.

We further support the high classification accuracy using the three-dimensional visualization of PCA. Figure 5.15 shows the PCA embedding of the three classes of liver and spleen samples. The projections were calculated using all the $55,471$ genes. The classes are not separable in the 3D PCA space.

Table 5.10: BSR on test set for different classification experiments.

Experiment	Organ	BSR
Tolerant vs Resistant vs Susceptible	Liver	95.56%
Tolerant vs Resistant vs Susceptible	Spleen	97.78%
Tolerant vs Susceptible	Liver	100.00%
Tolerant vs Susceptible	Spleen	100.00%

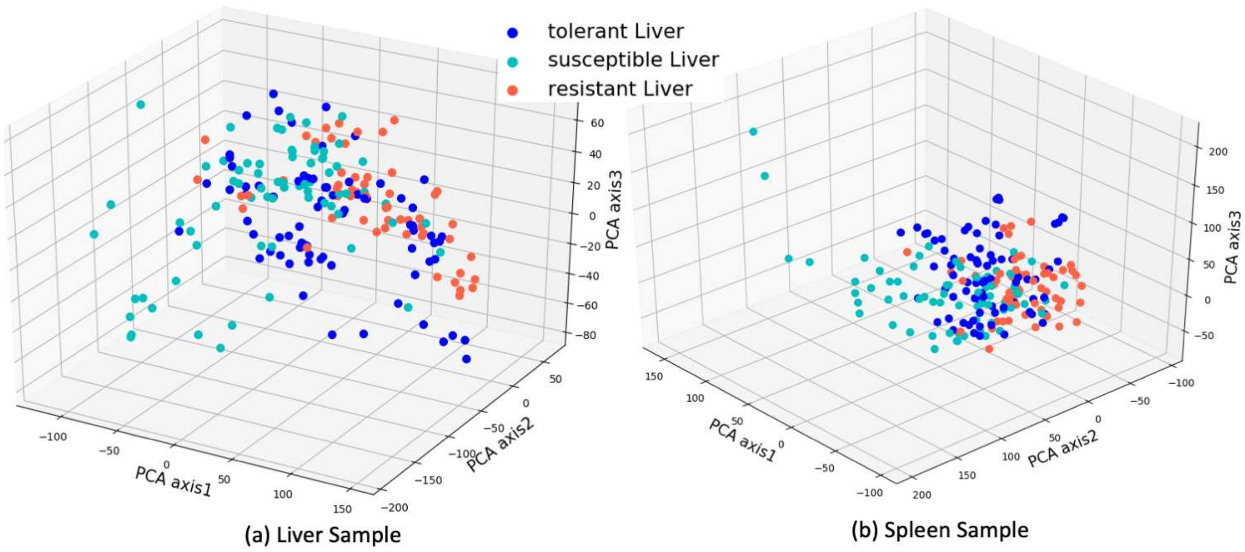


Figure 5.15: Three dimensional PCA projection of liver and spleen samples using all 55471 features.

We calculated the PCA projection using the SCE features. In Figure 5.16 we present the three dimensional projection of liver and spleen samples of training and test cases. The separation among the three classes is readily visible, with some occasional overlap. The test samples are closely mapped to the training data of the respective class. Note PCA is an unsupervised method, i.e.; it doesn't use the class label to build the projection. The separation among the tolerant, resistant, and susceptible classes in PCA embedding emphasizes that SCE pulled the discriminative features from the original gene set. The visualization also supports the high classification accuracy of the test data.

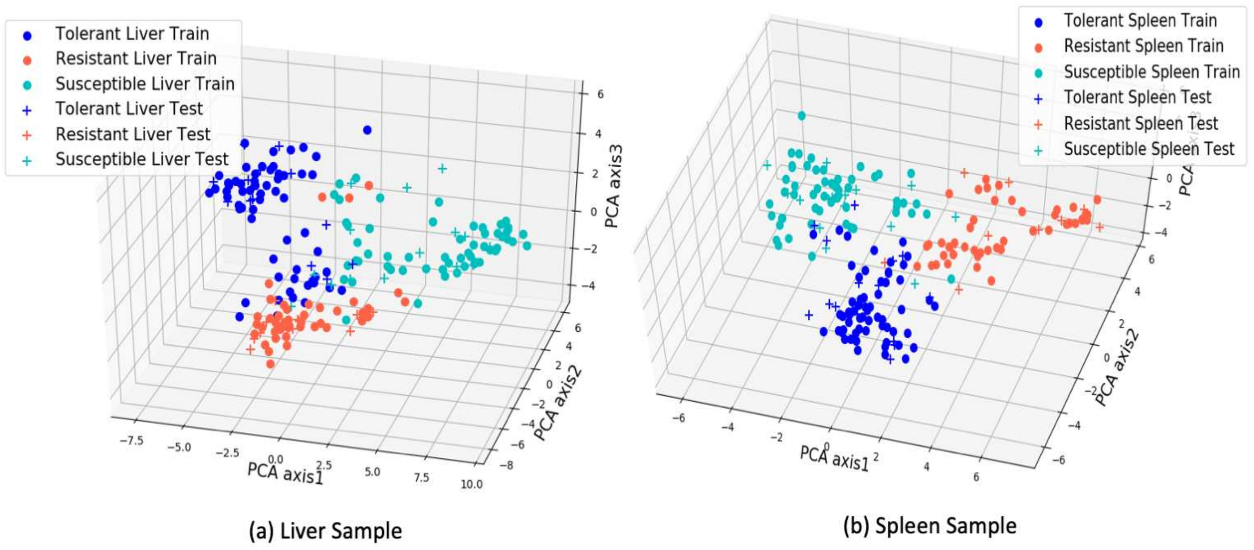


Figure 5.16: Three dimensional PCA projection of liver and spleen samples using 17 and 27 SCE features respectively.

Figure 5.17 presents the three dimensional PCA plot of tolerant and susceptible classes using liver and spleen samples. The embedding was built using the top 9 and top 5 SCE features, respectively. In both cases, there are two clusters for each class.

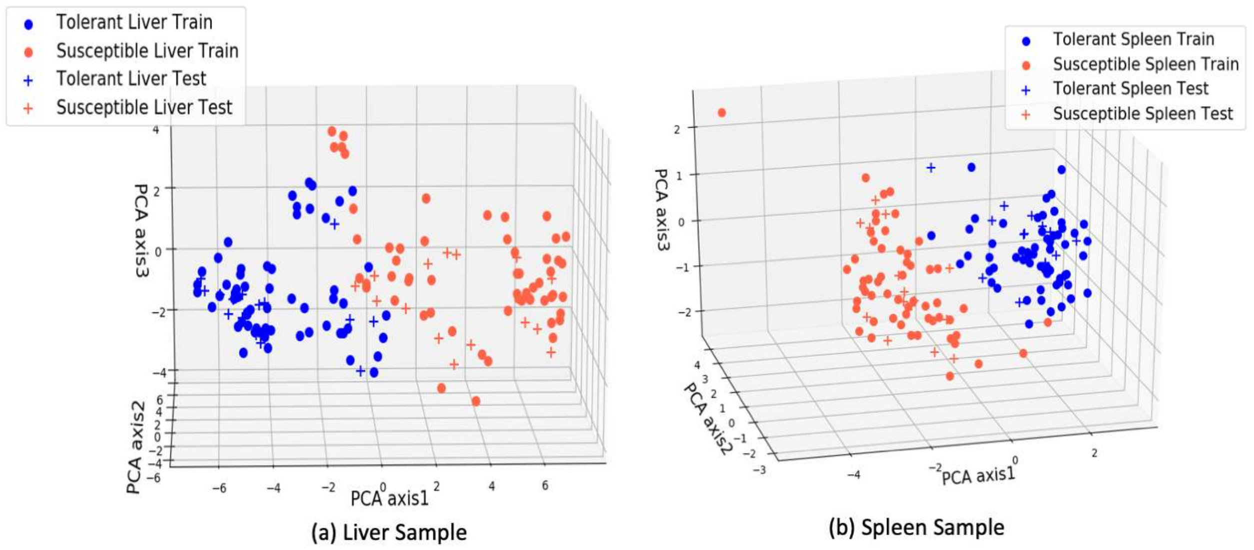


Figure 5.17: Three dimensional PCA projection of liver and spleen samples using 9 and 5 SCE features respectively.

The clusters of the liver samples are more dispersed than the spleen. This fact suggests that the gene expression of spleen samples may give more information to understand Salmonella's tolerance mechanism. Again the separation in 3D space supports the high classification rate on the test samples.

5.5.2 Feature Selection of Ebola Data set

We analyzed the second biological data set of Collaborative Cross (CC) mice data of Ebola disease. The biologist of Columbia University compiled the data set where each sample in represented by $55,471$ genes. This data set has two groups, mock mice, which were not infected by Ebola Virus, and the Ebola mice infected by the virus. Each mock and Ebola group has tolerant and lethal mice. The tolerant group survived the infection, whereas the lethal group didn't. As in Salmonella data, the samples were collected from the tissues of the liver and spleen. Table 5.11 presents the sample count of each group.

Table 5.11: Number of samples per class for various combinations in Ebola data set.

	Mock	Ebola
Liver	Tolerant: 56 Lethal: 38	Tolerant: 53 Lethal: 35
Spleen	Tolerant: 51 Lethal: 39	Tolerant: 55 Lethal: 36

Experimental Setup: We ran the following experiments:

- Tolerant Mock vs Lethal Mock using Liver Samples
- Tolerant Mock vs Lethal Mock using Spleen Samples
- Tolerant Ebola vs Lethal Ebola using Liver Samples
- Tolerant Ebola vs Lethal Ebola using Spleen Samples

In each of these experiments, we partitioned the samples into training and test sets into 80:20 ratio. We further kept 10% of training samples as a validation set to find the optimal number of features. We ran feature selection using Sparse Centroid-Encoder (SCE) on the training set. The feature selection step produced an ordered list of genes. We chose $n = 1, 2, ...$ features from the ranked gene list to train a linear SVM on the training set and monitor the accuracy of the validation set. We stopped the process once the validation accuracy didn't improve. Once we got the top n features, we merged the training and validation set and built a linear SVM model with those n features. At last, we used the trained model to predict the test samples. Table 5.12 shows the partition of training and test samples.

Table 5.12: Number of samples per class in training and test set.

Organ	Training		Test	
	Mock	Ebola	Mock	Ebola
Liver	Tolerant: 44 Lethal: 30	Tolerant: 42 Lethal: 28	Tolerant: 12 Lethal: 8	Tolerant: 11 Lethal: 7
Spleen	Tolerant: 40 Lethal: 31	Tolerant: 44 Lethal: 28	Tolerant: 11 Lethal: 8	Tolerant: 11 Lethal: 8

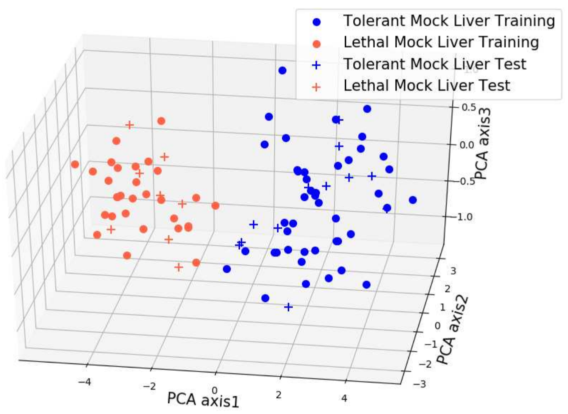
Results: Now we present the results from a quantitative and qualitative perspective.

Table 5.13: Number of features extracted by SCE for different classification experiments on the Ebola data set.

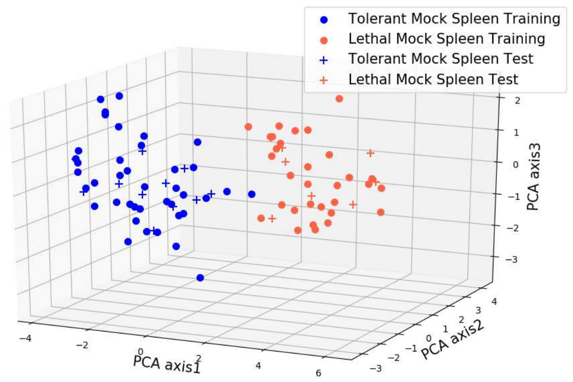
Experiment	Organ	No. of features
Tolerant Mock vs Lethal Mock	Liver	6
Tolerant Mock vs Lethal Mock	Spleen	11
Tolerant Ebola vs Lethal Ebola	Liver	6
Tolerant Ebola vs Lethal Ebola	Spleen	7

Table 5.14: BSR on test set for different classification experiments on the Ebola mice data set.

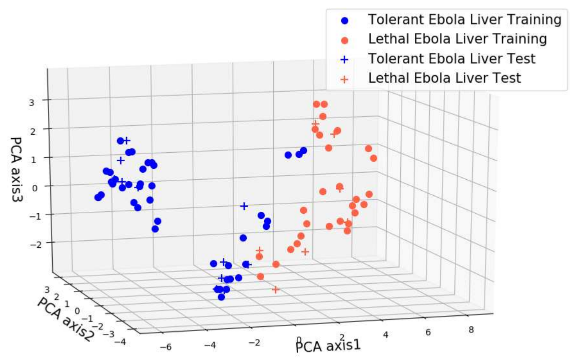
Experiment	Organ	BSR
Tolerant Mock vs Lethal Mock	Liver	100.00%
Tolerant Mock vs Lethal Mock	Spleen	100.00%
Tolerant Ebola vs Lethal Ebola	Liver	100.00%
Tolerant Ebola vs Lethal Ebola	Spleen	100.00%



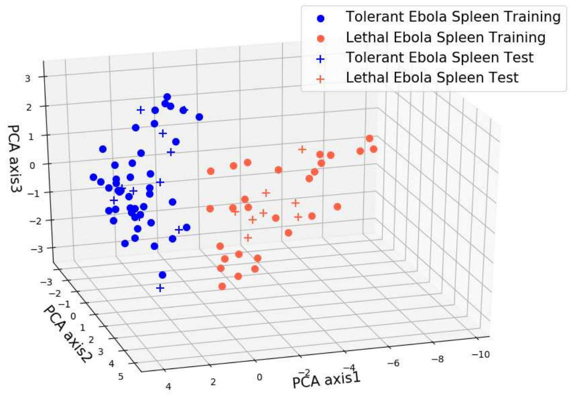
(a) Tolerant Mock vs Lethal Mock Liver



(b) Tolerant Mock vs Lethal Mock Spleen



(c) Tolerant Ebola vs Lethal Ebola Liver



(d) Tolerant Ebola vs Lethal Ebola Spleen

Figure 5.18: Three-dimensional PCA projection of Lethal and Mock samples of the four experiments. In each of the cases, PCA is applied on the training data where each sample consists of the features selected by sparse centroid-encoder as shown in Table 5.9. The first three eigenvectors were used to project the training and test samples.

In Table 5.13 we show the number of optimum features selected by our model. Surprisingly, the number of features for each experiment is considerably low, given the total number of features in the data set is over 55,000. Our sparse model picked about $0.01\% - 0.02\%$ of total features for the classification task. In table 5.14 we show the classification result for the four experiments. Notice that the classification accuracies of the four experiments are 100% on the sequester test data. To understand the high accuracies, we analyzed the features from a qualitative visual perspective. Figure 5.18 shows the three-dimensional visualization of training and test samples of the four experiments using PCA. The separation between the two classes is easily visible in all four cases. Notice that the test samples are mapped close to the training, demonstrating the robustness of selected features. The visual analysis using PCA further supports the high test accuracies.

5.6 Comparison of SCE with Current State-of-the-art Methods

In this section, we compared SCE with recent state-of-the-art neural network-based techniques on twelve data sets as shown in Table 5.15. We did two bench-marking experiments using these data sets. We didn't run SCE with the feature selection workflow for a fair comparison.

Table 5.15: Descriptions of the data sets used for benchmarking experiments.

Dataset	No. Features	No. of Classes	No. of Samples	Domain
ALLAML	7129	2	72	Biology
GLIOMA	4434	4	50	Biology
SMK_CAN	19993	2	187	Biology
Prostate_GE	5966	2	102	Biology
GLI_85	22283	2	85	Biology
GM12878	93	3	6468	Biology
Mice Protein	77	8	975	Biology
COIL20	1024	20	1440	Image
Isolet	617	26	7797	Speech
Human Activity	561	6	5744	Accelerometer Sensor
MNIST	784	10	70000	Image
FMNIST	784	10	70000	Image

In our first set of experiments, we compared SCE with LassoNet [9] and Stochastic Gate (STG) [10] on six publicly data sets taken from different domains, including image (MNIST, Fashion MNIST, COIL-20), speech (ISOLET), activity recognition (Human Activity Recognition Using Smartphones), and a biological data set (Mice Protein data). These data sets have been used in the literature for benchmarking [9, 105]. Following the experimental protocol of Lemhadri et al., we randomly partitioned each data set into a 70:10:20 split of training, validation, and test set. We normalized the training partition by subtracting the mean and dividing each feature by its corresponding standard deviation. We used the mean and standard deviation of the training to standardize the test samples. We used the training set for feature selection and the validation set for hyperparameter tuning. After the feature selection, we used a one hidden layer neural network classifier to predict the class label on the test set. We ran the classifier ten times ($K = 10$) and presented the mean accuracy with standard deviation in Table 5.16. Apart from showing the accuracy with top-50 features, we also give classification results with an optimum number of features selected using the validation set.

Table 5.16: Classification results using LassoNet, STG, and SCE features on six publicly available data sets. The column '#Centers for SCE' denotes how many centroids per class are used to train SCE. Numbers for LassoNet and STG are reported from [9] and [10] respectively. All the reported accuracies are measured on the test set. NA means the result has not been reported.

Data set	Top 50 features			#Centers	All features
	LassoNet	STG	SCE	for SCE	ANN
Mice Protein	95.8	NA	**98.4**	1	100.00
MNIST	87.3	91.0	**93.8**	3	97.60
FMNIST	80.0	NA	**84.7**	3	90.16
ISOLET	88.5	85.0	**91.1**	5	96.96
COIL-20	99.1	97.0	**99.3**	1	98.87
Activity	84.9	NA	**89.4**	4	92.81

As we can see from Table 5.16, the features of the Sparse Centroid-Encoder produce better classification accuracy than LassoNet and STG in all the cases. Especially for Mice Protein, Activity, Isolet, FMNIST, and MNIST, our model has better accuracy by $2.5\% - 4.5\%$. The results for Stochastic Gates (STG) in [10] are not in a table form, but our eyeball comparison of classification accuracy with the top 50 features on ISOLET, COIL20, and MNIST suggests that stochastic gate is not more accurate than SCE. For example, using the top 50 features, STG obtains approximately 85% accuracy on ISOLET while SCE obtains 91.1%; STG obtains about 97% on COIL20 while SCE obtains 99.3%; on the dataset, MNIST STG achieves approximately 91% while SCE 93.8%. In this experiment, we ran SCE with multiple centroids per class and observed an improved prediction rate than one center per class on Isolet, Activity, MNIST, and FMNIST. The observation suggests that the classes are multi-modal, providing a piece of valuable information. The optimum number of centers was picked using the validation set, see Figure 5.4.

In Table 5.17 we present the comparison of SCE with FsNet [11] and supervised Concrete Autoencoders (SCAE) [105]. These experiments comprise five high-dimensional real-world biological data sets available at ASU's feature selection database[11].

Table 5.17: Comparison of mean classification accuracy of FsNet, SCAE, and SCE features on five real-world high-dimensional biological data sets. The prediction rates are averaged over twenty runs on the test set. Numbers for FsNet and SCAE are being reported from [11].

Data set	Top 10 features			Top 50 features			All features
	FsNet	SCAE	SCE	FsNet	SCAE	SCE	ANN
ALLAML	91.1	83.3	**92.5**	92.2	93.6	**95.9**	89.9
Prostate_GE	87.1	83.5	**89.5**	87.8	88.4	**89.9**	75.9
GLIOMA	62.4	58.4	**63.2**	62.4	60.4	**69.0**	70.3
SMK_CAN	**69.5**	68.0	68.6	64.1	66.7	**69.4**	65.7
GLI_85	87.4	**88.4**	84.1	79.5	82.2	**85.5**	79.5

Following the experimental protocol of Singh et al., we split each data set into a 50-50 ratio of training and test partition. We ran our feature selection framework on the training set and then predicted the test samples using the selected features. We repeated the experiment on each data set 20 times and calculated the average accuracy, as shown in Table 5.17. Apart from the results using a subset (10 and 50) of features, we also provide the prediction using all the features. In most cases, feature selection helps improve classification performance. Generally, SCE features perform better than SCAE and FsNet; out of the ten classification tasks, SCE produces the best result on eight. Notice that the top fifty SCE features give a better prediction rate than the top ten in all the cases. Interestingly, the accuracy of SCAE and FsNet drop significantly on SMK_CAN and GLI_85 using the top fifty features.

[11] http://featureselection.asu.edu/datasets.php

5.7 Discussion and Conclusion

In this chapter, we presented Sparse Centroid-Encoder as an efficient feature selection tool for binary and multi-class problems. The benchmarking results span thirteen diverse data sets and six methods providing evidence that the features of SCE produce better generalization performance than other state-of-the-art models. We compared SCE with FsNet, mainly designed for high-dimensional biological data, and found that our proposed method outperformed it in most cases. The comparison also includes Supervised CAE, which is not more accurate than SCE. On the data sets, where the no. of observations are more than the no. of variables, SCE produces the best classification results than LassoNet, Stochastic Gate, and DFS in each case. These experiments involve image, speech, and accelerometer sensor data. Moreover, the survey of the sixteen SCE genes of GM12878 indicates plausible biological significance. The empirical evaluation using an array of diverse data sets establishes the value of the Sparse Centroid-Encoder as a nonlinear feature detector.

We have also demonstrated that our feature selection algorithm often selected fewer features than other models—the MNIST, Forest Cover experiments show this. These experiments include the comparison to some neural network-based models where group-sparsity is applied for variable selection. In addition to extracting the most robust features, the model shows the ability to discard noisy features. On the Indian Pine data set, our model didn't pick any of the water absorption bands considered noisy.

Our model has the advantage over the class of linear techniques, where a binary feature selection method is used as a multi-class method by one-against-one(OAO) or one-against-all(OAA) class pairs. For example, Chepushtanova et al. used ${}^{16}C_2 = 120$ binary class SSVM models on the Indian Pine data set. Similarly, Lasso needed three models for the GM12878 data. These models will suffer a combinatorial explosion when the number of classes increases. In contrast, SCE uses a single model to extract features from a multi-class data set. The result of the Indian Pine data also shows that the features of sparse centroid-encoder have more discriminatory signatures than the

OAO SSVM model. We also showed empirically that the SCE features captured more statistical variance than SSVM and WaLuMI features.

SCE compares favorably to the neural network-based model FsNet, SCAE DFS, LassoNet, and STG, where samples are mapped to class labels. SCE employs multiple centroids to capture the variability within a class, improving the prediction rate of unknown test samples. In particular, the prediction rate on the ISOLET improved significantly from one centroid to multiple centroids suggesting the speech classes are multi-modal. The two-dimensional PCA of ISOLET classes further confirms the multi-modality of data. We also observed an enhanced classification rate on MNIST, FMNIST, and Activity data with multiple centroids.

In contrast, single-center per class performed better for other data sets (e.g., COIL-20, Mice Protein, GM12878, etc.). Hence, apart from producing an improved prediction rate using features that capture intra-class variance, our model can provide extra information about whether the data is unimodal or multi-modal. This aspect of sparse centroid-encoder distinguishes it from the techniques which do not model the multi-modal nature of the data.

Chapter 6

Principal Centroid Component Analysis

Let $X \in \mathbb{R}^{d \times n}$ is the data matrix where n is the total number of samples and d is the dimension of each sample $x_i \in \mathbb{R}^d$. Assume X has M classes $\{C_j\}_{j=1}^M$ where the set of pattern indices of class C_j is denoted by I_j. We define centroid of each class as

$$c_j = \frac{1}{|C_j|} \sum_{i \in I_j} x_i \tag{6.1}$$

where $|C_j|$ is the cardinality of class C_j. Define a matrix $\tilde{C} \in \mathbb{R}^{d \times n}$ which contains the corresponding c_j's for each sample x_i. Note $\tilde{C}$ will have non-unique entries. For example, consider the data set $X = \{x_1, x_2, x_3, x_4, x_5\}$ which has two classes, say, C_1, C_2 where $I_1 = \{1, 3, 5\}$; $I_2 = \{2, 4\}$ and c_1, c_2 are the corresponding centroids. In this case $\tilde{C} = \{c_1, c_2, c_1, c_2, c_1\}$. With this set up, we present the formulation of Principal Centroid Component Analysis (PCCA).

6.1 Formulation

Given the transformation vector $\mathbf{a} \in \mathbb{R}^d$, consider the following optimization problem

$$\begin{aligned} \underset{\mathbf{a}}{minimize} \ \ & \|\tilde{C} - \mathbf{a}\mathbf{a}^T X\|_F^2 \\ subject\ to\ & \mathbf{a}^T\mathbf{a} = 1 \end{aligned} \tag{6.2}$$

The Lagrangian of Equation (6.2) is

$$\mathcal{L}(\mathbf{a}, \lambda) = \|\tilde{C} - \mathbf{a}\mathbf{a}^T X\|_F^2 - \lambda(\mathbf{a}^T\mathbf{a} - 1) \tag{6.3}$$

where λ is the Lagrangian multiplier. Notice that, setting $\frac{\partial \mathcal{L}}{\partial \lambda} = 0$ implies $\mathbf{a}^T\mathbf{a} = 1$. Further simplifying the equation 6.3,

$$\mathcal{L} = Tr[(\tilde{C} - \mathbf{a}\mathbf{a}^T X)^T(\tilde{C} - \mathbf{a}\mathbf{a}^T X)] - \lambda(\mathbf{a}^T\mathbf{a} - 1) \tag{6.4}$$

$$\mathcal{L} = Tr(\tilde{C}^T\tilde{C}) - Tr(\tilde{C}^T\mathbf{a}\mathbf{a}^T X) - Tr(X^T\mathbf{a}\mathbf{a}^T\tilde{C}) + Tr(X^T\mathbf{a}\mathbf{a}^T\mathbf{a}\mathbf{a}^T X) - \lambda(\mathbf{a}^T\mathbf{a} - 1) \tag{6.5}$$

$$\mathcal{L} = Tr(\tilde{C}^T\tilde{C}) - Tr(\tilde{C}^T\mathbf{a}\mathbf{a}^T X) - Tr(X^T\mathbf{a}\mathbf{a}^T\tilde{C}) + (\mathbf{a}^T\mathbf{a})Tr(X^T\mathbf{a}\mathbf{a}^T X) - \lambda(\mathbf{a}^T\mathbf{a} - 1) \tag{6.6}$$

Note that, $Tr(\tilde{C}^T\mathbf{a}\mathbf{a}^T X) = Tr(\mathbf{a}^T X\tilde{C}^T\mathbf{a})$, $Tr(X^T\mathbf{a}\mathbf{a}^T\tilde{C}) = Tr(\mathbf{a}^T\tilde{C}X^T\mathbf{a})$, and $Tr(X^T\mathbf{a}\mathbf{a}^T X) = Tr(\mathbf{a}^T XX^T\mathbf{a})$. Using these equalities, we get

$$\mathcal{L} = Tr(\tilde{C}^T\tilde{C}) - Tr(\mathbf{a}^T X\tilde{C}^T\mathbf{a}) - Tr(\mathbf{a}^T\tilde{C}X^T\mathbf{a}) + (\mathbf{a}^T\mathbf{a})Tr(\mathbf{a}^T XX^T\mathbf{a}) - \lambda(\mathbf{a}^T\mathbf{a} - 1) \tag{6.7}$$

$$\mathcal{L} = \tilde{C}^T\tilde{C} - \mathbf{a}^T\tilde{X}\tilde{C}^T\mathbf{a} - \mathbf{a}^T\tilde{C}X^T\mathbf{a} + (\mathbf{a}^T\mathbf{a})(\mathbf{a}^T XX^T\mathbf{a}) - \lambda(\mathbf{a}^T\mathbf{a} - 1) \tag{6.8}$$

Taking the derivative of Equation 6.8 w.r.t. **a** and setting it to 0,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}} = -2X\tilde{C}^T\mathbf{a} - 2\tilde{C}X^T\mathbf{a} + 2(\mathbf{a}^T\mathbf{a})XX^T\mathbf{a} + 2(\mathbf{a}^T XX^T\mathbf{a})\mathbf{a} - 2\lambda\mathbf{a} = 0 \tag{6.9}$$

Using $\mathbf{a}^T\mathbf{a} = 1$, we get,

$$-X\tilde{C}^T\mathbf{a} - \tilde{C}X^T\mathbf{a} + XX^T\mathbf{a} + (\mathbf{a}^T XX^T\mathbf{a})\mathbf{a} - \lambda\mathbf{a} = 0 \tag{6.10}$$

Rearrenging terms,

$$(XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{a} = (\lambda - \mathbf{a}^T XX^T \mathbf{a})\mathbf{a} \tag{6.11}$$

Setting $\mu = (\lambda - \mathbf{a}^T XX^T \mathbf{a})$

$$(XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{a} = \mu\mathbf{a} \tag{6.12}$$

Given, $(XX^T - X\tilde{C}^T - \tilde{C}X^T)$ is symmetric the solution of Equation 6.12 can be found by eigen-decomposition of $(XX^T - X\tilde{C}^T - \tilde{C}X^T)$. This optimization is similar to Principle Component Analysis [24], except that the principle directions are governed by the pre-computed centroids of the data set.

To solve for the second projection direction **b** we require,

$$\begin{aligned} &\underset{\mathbf{a}}{minimize} \;\; \|\tilde{C} - \mathbf{b}\mathbf{b}^T X\|_F^2 \\ &subject\ to\ \mathbf{b}^T\mathbf{b} = 1 \quad \mathbf{b}^T\mathbf{a} = 0 \end{aligned} \tag{6.13}$$

The Lagrangian of the above constraint optimization is

$$\mathcal{L} = \|\tilde{C} - \mathbf{b}\mathbf{b}^T X\|_F^2 - \alpha(\mathbf{b}^T\mathbf{b} - 1) - \beta(\mathbf{b}^T\mathbf{a}) \tag{6.14}$$

where α and β are the Lagrangian multipliers. Taking the partial derivative of Equation 6.14 w.r.t. **b** and setting it to 0,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = 2(XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{b} - 2\alpha\mathbf{b} - \beta\mathbf{a} = 0 \tag{6.15}$$

Hitting the Equation 6.15 by $\mathbf{a}^T$ and using $\mathbf{a}^T\mathbf{a} = 1$,

$$2\mathbf{a}^T(XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{b} - 2\alpha\mathbf{a}^T\mathbf{b} - \beta = 0 \tag{6.16}$$

From Equation 6.12, it's easy to verify that $\mathbf{a}^T(XX^T - X\tilde{C}^T - \tilde{C}X^T) = \mu\mathbf{a}^T$. Using these results,

$$2\mu\mathbf{a}^T\mathbf{b} - 2\alpha\mathbf{a}^T\mathbf{b} - \beta = 0 \implies \beta = 0 \tag{6.17}$$

Using $\beta = 0$ in Equation6.15 we get,

$$\begin{aligned} (XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{b} - \alpha\mathbf{b} = 0 \\ (XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{b} = \alpha\mathbf{b} \end{aligned} \tag{6.18}$$

We see that **b** is the second eigenvector of the symmetric matrix $(XX^T - X\tilde{C}^T - \tilde{C}X^T)$ and from the constraint **b** is orthogonal to **a**.

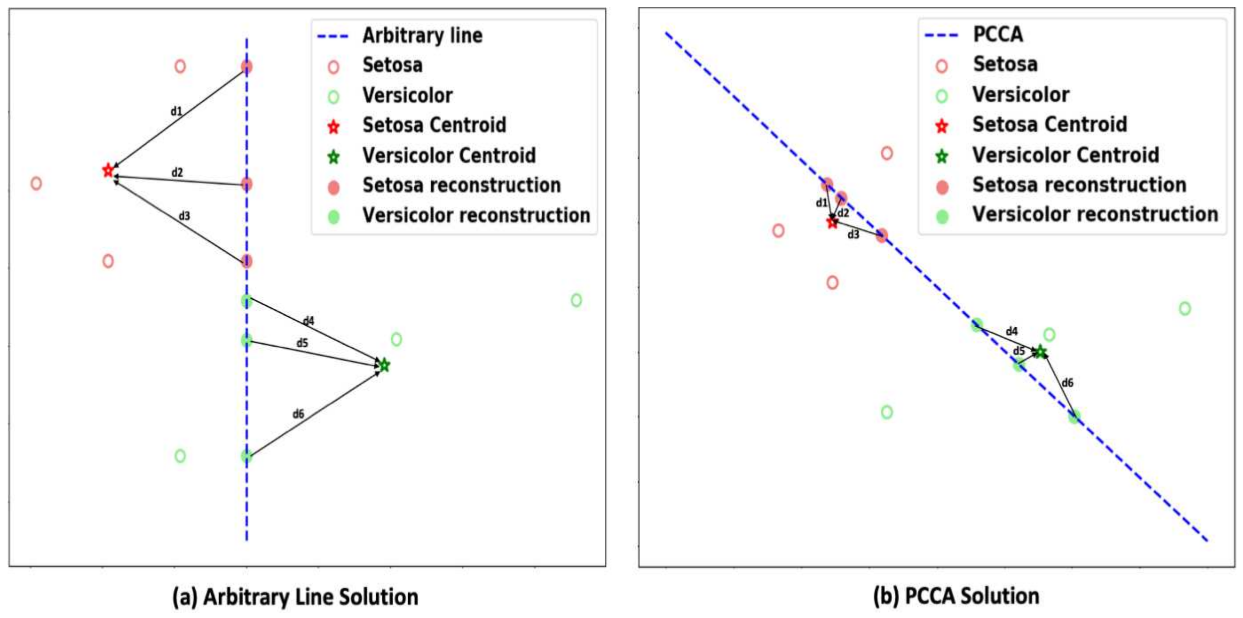


Figure 6.1: The geometric intuition of the PCCA algorithm using toy data. In panel (a), we used an arbitrary line to compute the cost in Equation6.2. The original samples were reconstructed using the line, and then the distances from the corresponding class centroid were denoted using the black lines. On the other hand, panel (b) shows the reconstruction of all the samples using the PCCA solution and the distances from the corresponding class centroids. Notice that the sum of the distances ($d_1, ..., d_6$) using PCCA is less than using an arbitrary line; PCCA explicitly searches for a line that minimizes the sum of the distances.

6.1.1 Relation Between the Eigenvalue and the Objective

Hitting both side of equation 6.12 by $\mathbf{a}^T$,

$$\mathbf{a}^T(XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{a} = \mathbf{a}^T\mu\mathbf{a} \tag{6.19}$$

$$\mathbf{a}^T(XX^T - X\tilde{C}^T - \tilde{C}X^T)\mathbf{a} = \mu \tag{6.20}$$

$$\mathbf{a}^T XX^T\mathbf{a} - \mathbf{a}^T X\tilde{C}^T\mathbf{a} - \mathbf{a}^T\tilde{C}X^T\mathbf{a} = \mu \tag{6.21}$$

$$Tr(XX^T\mathbf{a}\mathbf{a}^T) - Tr(X\tilde{C}^T\mathbf{a}\mathbf{a}^T) - Tr(\tilde{C}X^T\mathbf{a}\mathbf{a}^T) = \mu \tag{6.22}$$

Now adding $Tr(\tilde{C}^T\tilde{C})$ to the both side of Equation 6.22

$$Tr(\tilde{C}^T\tilde{C}) + Tr(XX^T\mathbf{aa}^T) - Tr(X\tilde{C}^T\mathbf{aa}^T) - Tr(\tilde{C}X^T\mathbf{aa}^T) = \mu + Tr(\tilde{C}^T\tilde{C}) \tag{6.23}$$

$$Tr(\tilde{C}^T\tilde{C}) - Tr(\tilde{C}^T\mathbf{aa}^TX) - Tr(X^T\mathbf{aa}^T\tilde{C}) + Tr(X^T\mathbf{aa}^TX) = \mu + Tr(\tilde{C}^T\tilde{C}) \tag{6.24}$$

$$Tr(\tilde{C}^T\tilde{C}) - Tr(\tilde{C}^T\mathbf{aa}^TX) - Tr(X^T\mathbf{aa}^T\tilde{C}) + Tr(X^T\mathbf{aa}^T\mathbf{aa}^TX) = \lambda + Tr(\tilde{C}^T\tilde{C}) \tag{6.25}$$

Using the linearity property of trace,

$$Tr(\tilde{C}^T\tilde{C} - \tilde{C}^T\mathbf{aa}^TX - X^T\mathbf{aa}^T\tilde{C} + X^T\mathbf{aa}^T\mathbf{aa}^TX) = \mu + Tr(\tilde{C}^T\tilde{C}) \tag{6.26}$$

$$Tr(\tilde{C}^T\tilde{C} - \tilde{C}^T\mathbf{aa}^TX - X^T\mathbf{aa}^T\tilde{C} + X^T\mathbf{aa}^T\mathbf{aa}^TX) = \lambda + Tr(\tilde{C}^T\tilde{C}) \tag{6.27}$$

$$Tr[(\tilde{C} - \mathbf{aa}^TX)^T(\tilde{C} - \mathbf{aa}^TX)] = \mu + Tr(\tilde{C}^T\tilde{C}) \tag{6.28}$$

$$\|\tilde{C} - \mathbf{aa}^TX\|_F^2 = \mu + Tr(\tilde{C}^T\tilde{C}) \tag{6.29}$$

The above equation establishes the relationship between the cost of the objective and the eigenvalue.

In Figure 6.2, we demonstrated it visually. Equation 6.29 explains higher value of cost compared to the eigenvalue. Note that the centroid reconstruction cost with the first eigenvector, which

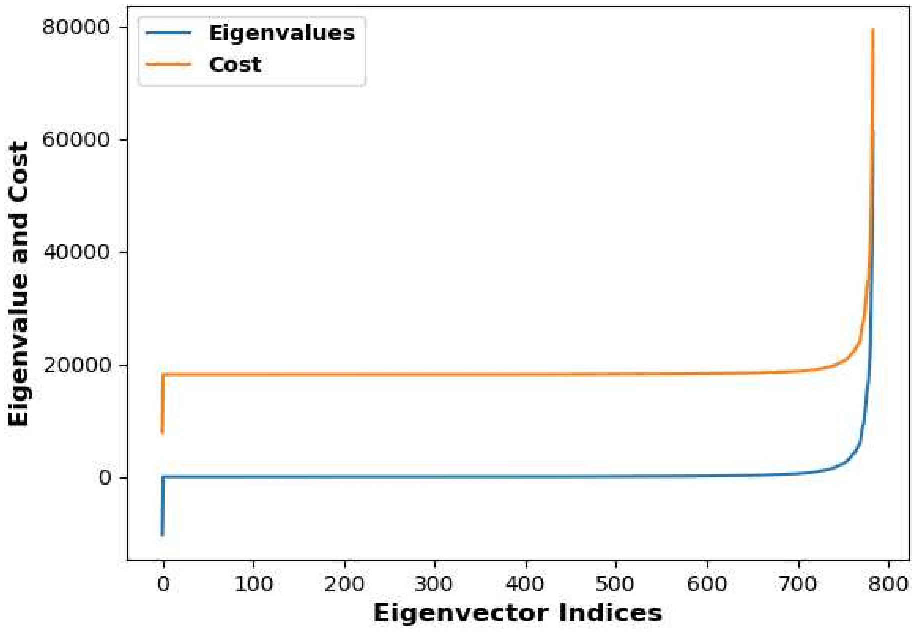


Figure 6.2: We show how the one-dimensional reconstruction cost (Equation 6.2) changes for each eigenvector along with the corresponding eigenvalue. The experiment is run using MNIST digits 4 and 9.

has the smallest eigenvalue, is also the smallest. The eigenvector associated with the largest eigenvalue produces the highest reconstruction cost.

6.1.2 Connection with PCA

PCA can be derived from the reconstruction perspective as follows,

$$\begin{aligned} \underset{\mathbf{a}}{minimize} \ \ & \|X - \mathbf{a}\mathbf{a}^T X\|_F^2 \\ subject\ to\ & \mathbf{a}^T\mathbf{a} = 1 \end{aligned} \tag{6.30}$$

where **a** is the transformation vector. Comparing equation 6.30 with 6.2 we can say PCA reconstructs each samples, whereas PCCA reconstructs the centroid of a class. As PCCA uses the class labels to calculate the centroids, hence it can be thought of as supervised PCA.

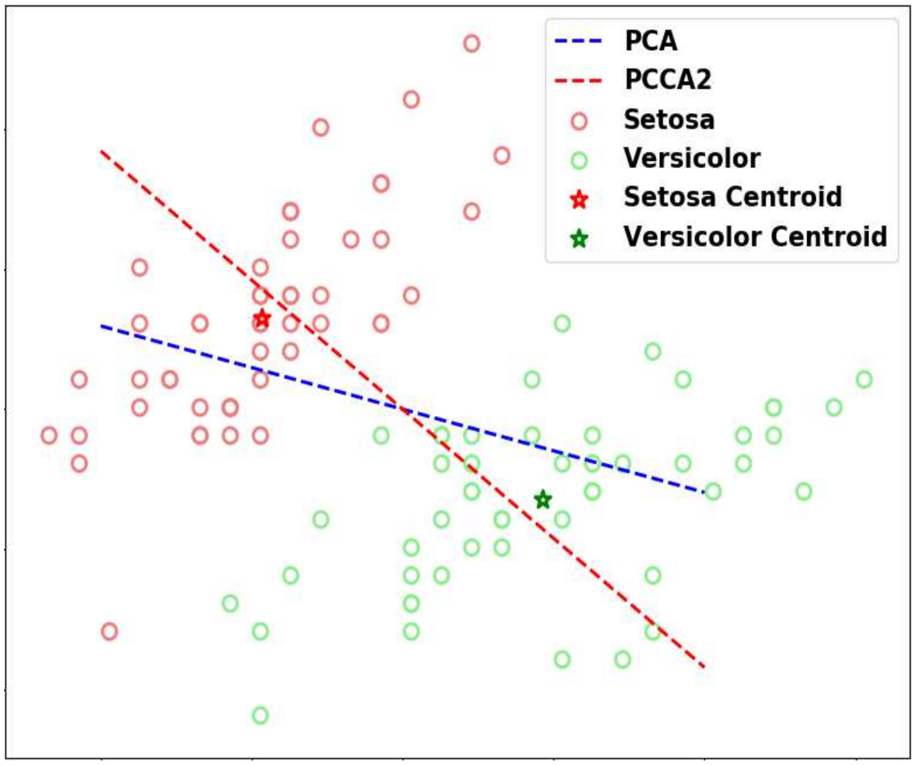


Figure 6.3: Comparison of PCA and PCCA solution on toy data.

6.1.3 Connection with Centroid-Encoder

The nonlinear mapping of centroid-encoder (CE) [142] is defined as,

$$\underset{\theta}{\text{minimize}} \quad \sum_{j=1}^{M} \sum_{i \in I_j} \|c_j - f(x^i; \theta))\|_2^2 \tag{6.31}$$

where $f(\cdot) = h(g(\cdot))$, composition of dimension reducing *encoder* mapping g followed by a dimension increasing *decoder* mapping h with θ being the parameter set for the mapping function. Both PCCA and CE reconstruct a class-centroid from the samples belonging to that class, but unlike CE, PCCA incorporates a linear mapping with orthogonality constraints. Hence PCCA can be thought of as a linear counterpart of CE.

6.2 Visualization and Classification Results

We present some visualization and classification experiments on five real-world data: USPS, MNIST, Human Activity, Mice Protein, and Arcene. These data sets have been used in the literature for benchmarking [9, 143]. We compare our method with PCA and a supervised version of PCA (SPCA) [50]. The comparison is made using a 5-NN classifier on reduced 2D space. We split each data set into a ratio of 80:20 of training and test partition, except for MNIST, which has a separate test set. After fitting all three models using the training set, we project the training and test samples using the first two eigenvectors; and then predict the test cases using a 5-NN classifier. We repeat the process 25 times and report the average accuracy with standard deviation.

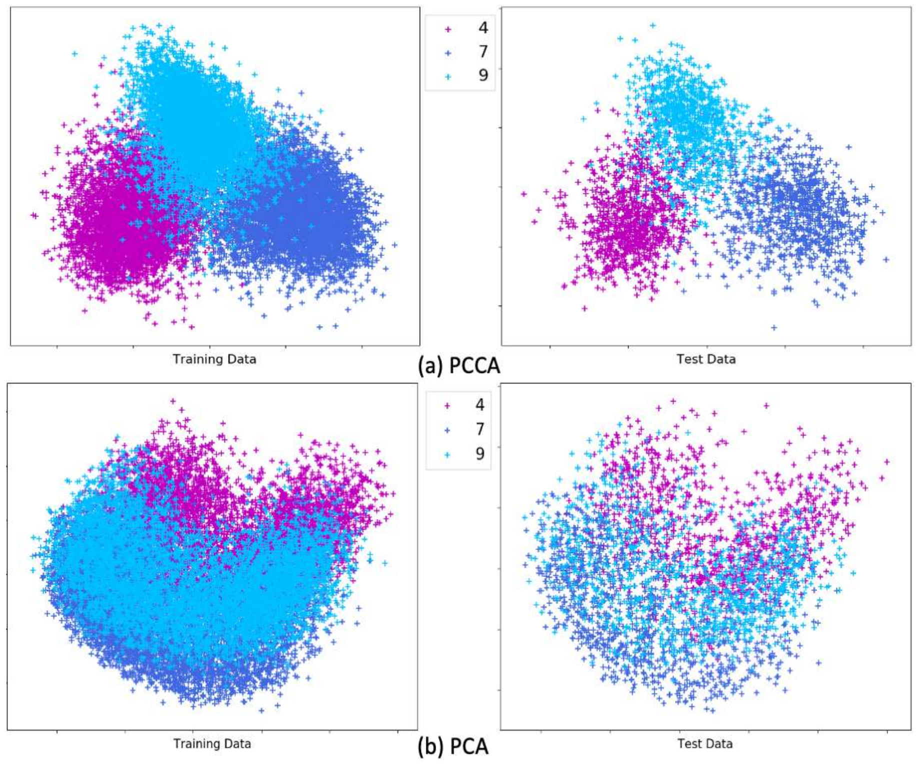


Figure 6.4: Comparison of PCCA(a) and PCA(b) projection on MNIST digits 4,7, and 9.

Before analyzing the results, we show the visualization on MNIST digits 4, 7, and 9 in Figure 6.4. These three digits are not separable in two-dimensional PCA space, and we want to verify whether PCCA can produce better visualization. In panel (a), we present the 2D PCCA projection. The three test classes are clearly separated, creating three blobs for each category.

Now we turn our attention to the low-dimensional classification results comparing PCCA, PCA, and SPCA as presented in Table6.1. Not surprisingly, the PCCA and SPCA perform better than

Table 6.1: Classification accuracies ($\%$) of 5-NN classifier on the 2D embedded data by various dimensionality reduction techniques.

Dataset	Model		
	PCCA	PCA	SPCA
USPS	**55.64** $\pm$ **0.95**	39.54 ± 0.85	46.27 ± 0.93
MNIST	**45.74** $\pm$ **0.00**	42.43 ± 0.00	44.17 ± 0.00
Activity	**65.20** $\pm$ **0.87**	51.83 ± 1.46	63.26 ± 1.26
Mice Protein	**66.17** $\pm$ **2.83**	44.91 ± 2.82	51.54 ± 3.59
Arcene	**85.37** $\pm$ **5.02**	65.59 ± 8.09	63.71 ± 7.91

PCA, which doesn't use the label. Note, on Arcene, the accuracy of PCA is better than SPCA. Our model provides the best 5-NN accuracy across all the data sets. The standard deviation on MNIST is 0 for all the models, which is not surprising as MNIST has a fixed training and test partition.

6.3 Conclusions

This chapter presented a linear formulation for Centroid-Encoder, called Principle Centroid Component Analysis (PCCA). Unlike the iterative method of Centroid-Encoder, PCCA has a closed-form solution using eigendecomposition. Both CE and PCCA are nonconvex, but the solution of PCCA can be ordered on the eigenvalues. We have shown the connection between the eigenvalues and the PCCA cost. The proposed optimization of PCCA has similarities to PCA; where PCA reconstructs each sample using a linear transformation, PCCA reconstructs the corresponding centroid using the class labels. We have compared our method with standard PCA and a supervised PCA, and the experimental results favor our approach. The current formulation of PCCA doesn't employ any repulsive force on the nearby classes to push them apart; thus, if two classes are close in ambient space, PCCA will keep them close in reduced space, increasing misclassification. In the future, we will explore these limitations.

Chapter 7

Centroid Component Retrieval

This chapter presents a supervised dimensionality reduction technique called Centroid Component Retrieval (CCR). Unlike Centroid-Encoder, CCR is convex and uses a linear transformation. The proposed model optimizes a multi-objective cost function by balancing two complementary terms. The first term pulls the samples of a class towards its centroid by minimizing a sample's distance from its class centroid in low dimensional space. The second term pushes the classes far apart by maximizing the scattering volume via the logarithm of the determinant (*log det*) of the outer product matrix formed by the class-centroids in embedded space. Using the negative of the *log det*, we pose the final cost as a minimization problem, which balances the two terms. Although the design principle of the proposed model is similar to LDA (Linear Discriminant Analysis), unlike multi-class LDA, CCR is convex. We have also presented an initialization technique using eigendecomposition, which allows faster convergence of CCR. Our experimental results show that CCR has a performance advantage over LDA on several high-dimensional data sets, including, COIL20, Yale Faces, and ORL.

7.1 Problem Formulation

Let $X \in \mathbb{R}^{d\times n}$ be the data matrix where n is the total number of samples and d is the dimension of each sample $x_i \in \mathbb{R}^d$. Assume X has M classes $\{C_j\}_{j=1}^{M}$ where the set of sample indices associated with class C_j is denoted by I_j. We define centroid of each class as

$$c_j = \frac{1}{|C_j|}\sum_{i\in I_j} x_i \tag{7.1}$$

where $|C_j|$ is the cardinality of class C_j. Define a matrix $\tilde{C} \in \mathbb{R}^{d\times n}$ which contains the corresponding c_j's for each sample x_i. Note $\tilde{C}$ will generally have non-unique entries. Define another matrix $\hat{C}$ which only contains the unique centroids c_j's of each class. More precisely, each column of $\hat{C}$ is a distinct centroid c_j. For example, consider the data set $X = \{x_1, x_2, x_3, x_4, x_5\}$ which has two classes, say, C_1, C_2 where $I_1 = \{1, 3, 5\}$; $I_2 = \{2, 4\}$ and c_1, c_2 are the corresponding centroids. In this case $\tilde{C} = \{c_1, c_2, c_1, c_2, c_1\}$ and $\hat{C} = \{c_1, c_2\}$. Under this set up we seek a transformation matrix $A \in \mathbb{R}^{d\times p}$ to achieve the following two goals in the reduced space:

- each point should be mapped approximately to its class centroid
- the centroids should be maximally scattered in the reduced space

To achieve the first goal we minimize the quantity $\|A^T(\tilde{C} - X)\|_F^2$ where $\|.\|_F$ denotes the Frobenius norm. Minimizing this quantity will map each sample $x_i \in C_j$ close to its corresponding class centroid c_j in the reduced space. It is useful to use the fact $\|A^T(\tilde{C} - X)\|_F^2 = Tr[(\tilde{C} - X)(\tilde{C} - X)^T AA^T]$. To achieve the second goal, we maximize $\log\det(A^T\hat{C}(A^T\hat{C})^T + \gamma I)$ where γ is small positive number (in our experiments we keep this value very small). The quantity $\det(A^T\hat{C}(A^T\hat{C})^T)$ gives the square of the scattering volume of the hyper-ellipsoidal formed by the low dimensional centroids [144]. Maximizing this volume will scatter the centroids maximally in low dimensions. Taken together, we are proposing the following minimization problem over the transformation matrix A:

$$\begin{aligned} \underset{A}{minimize} \;\; & Tr[(\tilde{C} - X)(\tilde{C} - X)^T AA^T] \\ & -\lambda \log\det(A^T\hat{C}\hat{C}^T A + \gamma I) \end{aligned} \tag{7.2}$$

where λ is a positive quantity which is used to balance the two terms. The crucial property of this multi-objective optimization is that it's a convex function of the matrix A.

7.2 Proof of Convexity

We write the original objective as a sum to two parts, $\mathcal{L}(A) = \mathcal{L}_1(A) + \lambda\mathcal{L}_2(A)$ where

$$\mathcal{L}_1 = Tr[(\tilde{C} - X)(\tilde{C} - X)^T AA^T]$$

$$\mathcal{L}_2 = -\log\det(A^T\hat{C}(A^T\hat{C})^T + \gamma I)$$

First, observe that the domain, the set of $d \times p$ matrices, is a convex set. Now we will show that $\mathcal{L}_1$ is convex. Differentiating $\mathcal{L}_1$ w.r.t. A gives

$$\Delta \mathcal{L}_1 = 2(\tilde{C} - X)(\tilde{C} - X)^T A \tag{7.3}$$

Differentiating again

$$\Delta^2 \mathcal{L}_1 = 2(\tilde{C} - X)(\tilde{C} - X)^T \tag{7.4}$$

Observe that the Hessian of $\mathcal{L}_1$ is the sample co-variance matrix therefore it's positive semidefinite. Hence by the second-order condition of convexity [45], $\mathcal{L}_1$ is a convex function.

Now we show $\mathcal{L}_2 = -\log\det(A^T \hat{C}\hat{C}^T A + \gamma I)$ is convex. Observe that the matrix $M = (A^T \hat{C}\hat{C}^T A + \gamma I)$ is a positive definite matrix when $\gamma > 0$. Consider the value of $\mathcal{L}_2$ on an arbitrary line segment given by $M = P + tQ$ where P and Q are positive definite matrices. Note that the domain of positive definite matrices is also convex. Now define a function g as

$$g(t) = -\log\det(P + tQ) \tag{7.5}$$

As P is positive definite, we can write $P = P^{1/2}P^{1/2}$ where $P^{1/2}$ is also positive definite. Hence we can write,

$$g(t) = -\log\det(P^{1/2}(I + tP^{-1/2}QP^{-1/2})P^{1/2}) \tag{7.6}$$

Using the fact $\det(AB) = \det A \det B$ and the properties of logarithms we conclude

$$g(t) = -\log\det P - \log\det(I + tP^{-1/2}QP^{-1/2}) \tag{7.7}$$

The $\det(I + tP^{-1/2}QP^{-1/2})$ is the product of the eigenvalues of $(I + tP^{-1/2}QP^{-1/2})$. Let $\sigma_1, \sigma_2, ..., \sigma_n$ are the eigenvalues of $tP^{-1/2}QP^{-1/2}$. Hence,

$$g(t) = -\log\det P - \log\prod_{i=1}^{n}(1 + t\sigma_i) \tag{7.8}$$

$$g(t) = -\log\det P - \sum_{i=1}^{n}\log(1 + t\sigma_i) \tag{7.9}$$

Differentiating w.r.t. t twice gives

$$\Delta^2 g(t) = \sum_{i=1}^{n}\frac{{\sigma_i}^2}{1 + t{\sigma_i}^2} \geq 0$$

Since g is convex we conclude $\mathcal{L}_2$ is convex. So the cost is a summation of two convex functions over a convex set, hence it's also convex function[12].

7.3 Calculation of Gradient

The gradient of the cost function Equation 7.2 is given by

$$\frac{\partial\mathcal{L}}{\partial A} = 2(\tilde{C} - X)(\tilde{C} - X)^T A - 2\lambda(\hat{C}\hat{C}^T A)(A^T\hat{C}\hat{C}^T A + \gamma I)^{-1} \tag{7.10}$$

7.4 Initialization Techniques

The model parameters which are the elements of matrix A can be initialized in two ways.

First, we used a uniform random distribution from the range $(-1/\sqrt{p}, +1/\sqrt{p})$ to initialize the matrix A.

Alternatively, we may solve an eigenvector problem and use the solution to initialize the matrix A. Consider the set up of the problem described at the beginning of chapter 7 where $X \in \mathbb{R}^{d\times n}$ is the data matrix with n samples and M classes, c_j is the centroid of each class, and the matrix $\tilde{C} \in \mathbb{R}^{d\times n}$ contains the centroids c_j corresponding to each sample x_i. We seek a transformation

[12]The sum of two convex functions is also convex [45]

vector **a** that solves

$$\begin{aligned} &\underset{\mathbf{a}}{minimize} \;\; \|\mathbf{a}^T\tilde{C} - \mathbf{a}^T X\|_2^2 \\ &subject\ to\ \mathbf{a}^T\mathbf{a} = 1 \end{aligned} \tag{7.11}$$

The Lagrangian of Equation (7.11) is

$$\mathcal{L} = \|\mathbf{a}^T\tilde{C} - \mathbf{a}^T X\|_2^2 - \beta(\mathbf{a}^T\mathbf{a} - 1) \tag{7.12}$$

where β is the Lagrangian multiplier. After differentiation of the Lagrangian we conclude that

$$(\tilde{C} - X)(\tilde{C} - X)^T\mathbf{a} = \beta\mathbf{a} \tag{7.13}$$

As $(\tilde{C} - X)(\tilde{C} - X)^T$ is symmetric, the solution can be obtained from the eigen decomposition of the matrix $(\tilde{C} - X)(\tilde{C} - X)^T$. This optimization is similar to Principle Component Analysis [24], except that the principle directions are governed by the pre-computed centroids of the data set. After initializing the the model by the solution (eigenvectors) of the optimization in Equation 7.11, we fine tune the model using the gradient as shown in Equation 7.10. Note that in the fine tuning step the orthogonality constraint is not obeyed anymore.

7.5 Experiments to Analyze the CCR Model

In this section we describe the details of our numerical experiments.

7.5.1 Experiment to analyze the effect of Lambda

The CCR model requires the tuning of the hyper parameter λ which is used to control the separation between the classes. A bigger value will make the classes more separated compared to a smaller one. To demonstrate this we conducted experiment with the following values of λ: $0.1, 1.0, 10.0, 100.0$ on a subset of MNIST. We took all the samples from digit class 0, 1 and 2 and trained the CCR to get a two-dimensional representation of each sample. Then we calculated the

Hausdorff distances ($d_h(A,B) = \underset{a \in A}{max}\ \underset{b \in B}{min}\ dist(a,b)\ where\ A, B\ are\ two\ non-empty\ sets$) of each pair of classes and present the result in Table 7.1.

Table 7.1: The effect of λ on a subset of MNIST. The 3-dimensional data is represented in 2D by CCR model3. Hausdorff distances among the class pairs are measured for each value of λ.

Hausdorff Dist.	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 10.0$	$\lambda = 100.0$
Digit 0 vs Digit 1	10.9114	13.4852	17.9494	44.3374
Digit 1 vs Digit 0	0.5410	1.6782	5.2770	16.6911
Digit 0 vs Digit 2	9.6558	11.7688	11.9953	28.9347
Digit 2 vs Digit 0	343.7246	473.9643	475.3801	498.0760
Digit 1 vs Digit 2	0.4137	1.3104	4.1407	13.0951
Digit 2 vs Digit 1	353.1745	485.9086	483.7695	503.1173

It's clear that as the value of λ increases the Hausdorff distances also increases thus increasing the class separation.

7.5.2 Trade-off of the Optimization

The experiments in Section 7.5.1 demonstrates that the class separation increases with the increase of λ. But increasing λ will also increase the scatter of each classes. The explanation lies in the objective function. The term $\mathcal{L}_1$ brings the samples of a class near to its centroid while $\mathcal{L}_2$ serves to increase the separation among the classes. As we increase λ the gradient of the second part of the objective function will start to dominate and the model will focus more on class-separation. As a result the scatter of each class will start to increase. To examine this further we ran an experiment on a subset of MNIST data as mentioned in Section 7.5.1. To objectively quantify the class-scatter and class-separation, we define two measures. The scatter of a class C_j is defined as :

$$CS = \frac{1}{|C_j|} \sum_{i \in I_j} dist(c_j, x_i) \tag{7.14}$$

Now we define the *Mean Set Distance (MSD)* to measure the separation among two class C_1 and C_2

$$MSD = \frac{1}{|C_1|} \sum_{i \in I_1} \frac{1}{|C_2|} \sum_{j \in I_2} dist(x_i, x_j) \tag{7.15}$$

In Table 7.2 we present the CR score for each digit class over different values of λ. Higher values of λ increase the scatter. We also observe in Table 7.3 that increasing λ also increases the pair-wise class separation.

Table 7.2: The effect of λ on class-scatter on a subset of MNIST digits. First, the 784 dimensional data is represented in 2D by CCR model. After that the CR and MSD are measured on 2D space.

Class Scatter	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 10.0$	$\lambda = 100.0$
Digit 0	2.8746	4.0491	6.6547	11.7280
Digit 1	5.0957	9.0666	16.1205	28.6672
Digit 2	13.1986	15.5978	18.2859	27.1962

Table 7.3: The effect of λ on class-separation on a subset of MNIST digits. First, the 784 dimensional data is represented in 2D by CCR model. After that the CR and MSD are measured on 2D space.

Class Separation	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 10.0$	$\lambda = 100.0$
Digit 0 vs Digit 1	1.2339	2.1895	3.8876	6.9136
Digit 0 vs Digit 2	1.1300	1.9907	3.5181	6.2386
Digit 1 vs Digit 2	1.0188	1.7988	3.1827	5.6430

We have also plotted the sub-costs $\mathcal{L}_1$ and $\mathcal{L}_2$ for different values of λ in Figure 7.1. We see that the sub-cost $\mathcal{L}_1$ increases as we increase the value of λ. Increase of $\mathcal{L}_1$ means that the samples of a class are not tightly clustered, which is not desirable. On the other hand we observe that $\mathcal{L}_2$ decreases as we increase λ. The reduction of $\mathcal{L}_2$ indicates that the classes are being separated as we expect. This is the trade-off of the model. So the λ balances the two terms. For the purposes of supervised visualization, we can pick the value of λ which will produce low generalization error (e.g. k-NN error on embedding space) on a validation set.

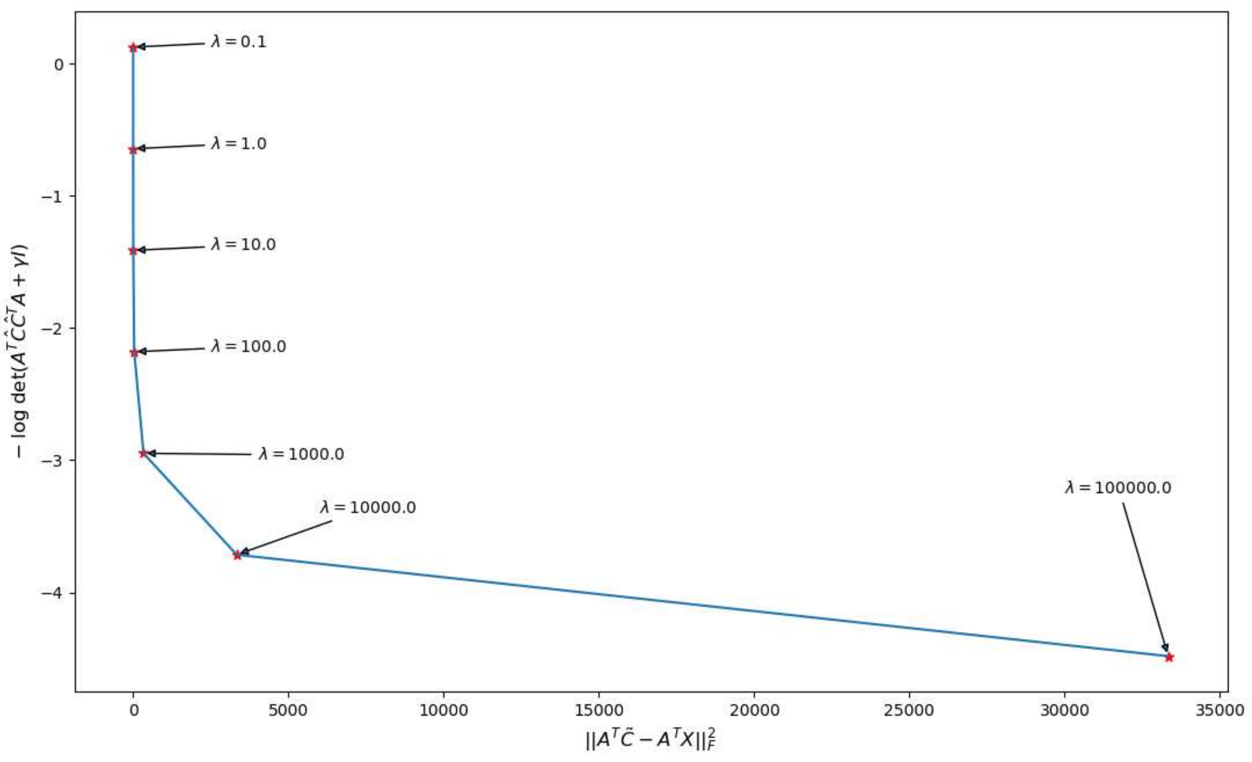


Figure 7.1: Plot of $\mathcal{L}_1$ and $\mathcal{L}_2$ across λ.

7.5.3 Comparison between Random vs Deterministic Initialization

In Section 7.4 we mentioned two types of initialization. As the optimization is convex, the choice of the initialization is not critical since the algorithm converges to the final solution. However, we found that random initialization may take a long time to converge. To this end, we ran an experiment on a subset of MNIST (only considering digit class 0,1, and 2) with both types of initialization. We train the model to project the data on 2D space. The training was done until the absolute difference of the cost of two consecutive iteration reaches the threshold of 10^{-6}. For each type of initialization, we repeat the process ten times. The random initialization took about 208 iteration to converge, whereas the deterministic one reaches the optimum with just 94 iteration.

7.6 Classification and Visualization Experiments

In this section we describe the low dimensional classification and visualization experiments.

7.6.1 Data Sets

Here we provide a brief description the data sets used in the bench-marking experiments.

MNIST Digits: This is widely used collection of digital images of handwritten digits (0..9)[13] with separate training (60,000 samples) and test set (10,000 samples). Each sample is a grey level image consisting of 1 byte pixels normalized to fit into a 28 x 28 bounding box resulting in *vecced* points in $\mathbb{R}^{784}$.

USPS Data: A data set of handwritten digits $(0 \ldots 9)$[14], where each element is a 16x16 image in gray-scale resulting in vecced points in $\mathbb{R}^{256}$. Each of the ten classes has 1100 digits for a total of 11,000 digits.

Iris Data: The UCI Iris data set is comprised of three classes and is widely used in the Machine Learning literature. Each class has 50 samples and each sample has four features.

Wine: This data set contains 178 samples across three types of wines. Each samples is a 13 dimensional vector. It's available in UCI repository.

Ionosphere (Ion): The data was collected to detect structure of ionosphere using radar signal. There are 351 samples divided into two classes, 'good' and 'bad'; and each sample is represented by a 34 dimensional vector. The data is also available in UCI repository.

Balance Scale: The data was collected to conduct psychological experiment by measuring four attributes. A total of 625 samples were put together which were spread across three classes: balanced, left and right and it's available in UCI repository.

Sonar: This UCI data set has two classes: mine and rock. Each sample is represented by a 60 dimensional vector. The mine class has 111 patterns whereas rock has 97 samples.

Yale Face: The data set contains 165 gray scale images of 15 individuals[15]. Images were taken under different lighting condition, facial expression and with/without glass. Original images are cropped to fit into a square box of size 32×32 which is represented by a $\mathbb{R}^{1024}$ dimensional vec-

[13]The data set is available at http://yann.lecun.com/exdb/mnist/index.html

[14]The data set is available at https://cs.nyu.edu/ roweis/data.html

[15]For details see http://vision.ucsd.edu/content/yale-face-database

tor.[16]

ORL: The data set contains 400 face images (32 x 32) from 40 subjects, and each subject represents a class that includes ten images in grayscale. Some pictures have different facial expressions (e.g., open/closed eyes, smiling / not smiling) and facial details (glasses / no glasses). Images were taken with a dark background with the subjects in an upright, frontal position. The data set can be found here: The data set is available at https://jundongl.github.io/scikit-feature/datasets.html.

COIL20: The data set is a collection of $32x32$ grayscale images from 20 toy objects (e.g., car, duck, cup, box, etc.). Pictures were taken 5 degrees apart for each object, resulting in 72 (360/5) images for each class. The data set is available at https://jundongl.github.io/scikit-feature/datasets.html.

7.6.2 Supervised Models

With these data sets, we compare our model with the following supervised models:

- Linear Discriminant Analysis (LDA) [27]
- Neighborhood Component Analysis (NCA) [145]
- Maximally Collapsing Metric Learning (MCML) [67]
- Supervised PCA (SPCA) [50]

Apart from our own model, we have also implemented NCA and SPCA in Python. We used the package pyDML [146] to run LDA and MCML.

7.6.3 Generalization Performance

Given we are comparing supervised methods, we restrict our attention to the generalization performance as measured by a test data set. To this end we calculated the class prediction accuracy as defined below:

$$Accuracy\ (\%) = \frac{100}{N}\sum_{i=1}^{N} I[l_i = f(\tilde{x}_i)] \tag{7.16}$$

[16]A processed version is available at http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

Here N is total number of test samples, l_i is the true label of the i^{th} test sample, and f is a classification function which returns the predicted label of the embedded test sample $\tilde{x}_i$. Here I denotes the indicator function, which has the value 1 if the argument is true (label correct), and 0 otherwise (label incorrect).

Experiment 1

For the first experiment we used data sets where number of samples are greater than the sample-dimension. In this bench-marking experiment, we split each data set into a separate training and test set by a ratio of 80:20 (except for MNIST as it has separate training and test sets), and then trained each model on the training set[17]. Then we used the trained model to represent both the training and test samples into low dimensional space (2D). After that, we used $k-$NN (k=5) classifier to calculate the accuracy on test data in low dimensional space. We repeat the process 25 times and present the average test accuracy with standard deviation in Table 7.4.

Table 7.4: Classification accuracies ($\%$) of 5-NN classifier on the 2D embedded data by various dimensionality reduction techniques. We don't have the result on MNIST and USPS for MCML as it was taking too long to complete.

Dataset	Model				
	CCR	NCA	MCML	LDA	SPCA
Balance	**90.83 ± 1.96**	**90.88 ± 1.86**	88.57 ± 3.89	89.43 ± 2.56	89.97 ± 2.26
Ion	**85.92 ± 3.47**	**85.96 ± 1.95**	84.85 ± 3.97	84.79 ± 3.33	82.99 ± 4.12
Wine	**99.14 ± 1.47**	96.51 ± 2.57	93.30 ± 4.78	98.59 ± 2.18	71.27 ± 4.91
Iris	**97.07 ± 2.55**	96.18 ± 2.70	98.00 ± 2.83	96.67 ± 3.40	94.93 ± 2.22
Sonar	**74.79 ± 5.33**	**75.94 ± 5.34**	68.93 ± 5.42	73.12 ± 6.58	67.88 ± 7.06
USPS	**65.42 ± 0.94**	59.09 ± 4.93		64.87 ± 0.56	46.27 ± 0.93
MNIST	53.72 ± 0.00	**55.08 ± 5.59**		52.31 ± 0.00	44.17 ± 0.00

In all of the cases CCR outperforms SPCA, particularly in Wine, Sonar, USPS and MNIST. CCR also outperforms MCML in most of the data sets, especially in Sonar. We note that for relatively bigger data sets, e.g. USPS and MNIST, we didn't get the results for MCML as they

[17]A 5-fold internal cross-validation on the training set is used to find the optimal λ for CCR.

were taking too long to finish. The performance of LDA and CCR is almost identical except for the the data sets USPS and MNIST. On USPS data set, CCR performed slightly better in 5-NN classification by a margin of 0.55%. On the MNIST set, the generalization performance of CCR is better than LDA by a margin of 1.41%. It's noteworthy that the variance of the classification result for LDA, CCR and SPCA is 0 as these are deterministic models so the solution is fixed for the fixed MNIST training set. Now we compare our model with NCA which is a non-convex optimization problem. NCA produced better classification performance in Balance, Ion, Sonar and MNIST data whereas CCR did better on Wine, Iris and USPS data. It's noteworthy that in USPS, CCR outperforms NCA by a margin of 6.33%. In each of these cases the variance of the result is higher in NCA compared to CCR. Similar to USPS, NCA has higher variance in MNIST as well. The higher variances suggest that the model often stuck to a poor local minimum in training.

Table 7.5 presents the average training time taken by each model on seven data sets. We used an "HP-Z440-XeonE5-1650v4" machine with 6x3.6 GHz processor and 32 Gb RAM. Amongst all the models, MCML is the slowest. We don't have the average training time for MCML on USPS and MNIST data as the model took too long to finish. After MCML, NCA is the second slowest model. In Balance, Ion, and Wine, SPCA took longer time to train than CCR and LDA, whereas, in Iris, Sonar, USPS, and MNIST, CCR took more time than SPCA LDA. Among all the models, LDA turns out to be the fastest algorithm. SPCA, LDA are methods that depend on eigendecomposition; the package we used uses fast code for that. That's why we see the relatively quicker training time for those two methods. The models NCA and MCML are slower because both require the pair-wise distance calculation, which prohibits them from scaling up as the data set size increases.

Now we show the visualization of two dimensional embedding of CCR on Wine, Iris, Ion and USPS sets. The clustering of the three classes in Wine (Figure 7.2) and Iris (Figure 7.3) is prominent which supports the high classification accuracy of these two data sets in two dimensional space.

Table 7.5: Average training time in seconds for each model on different data sets. We don't have the time for MCML on USPS and MNIST data as the model was taking too long to finish.

Dataset	Average Training Time				
	CCR	NCA	MCML	LDA	SPCA
Balance	0.02571	2.08642	189.91499	0.00177	0.08842
Ion	0.02572	0.31745	53.39456	0.00674	0.02905
Wine	0.00685	0.10010	14.82072	0.00111	0.00723
Iris	0.00647	0.37909	4.91297	0.00093	0.00494
Sonar	0.10282	0.23940	18.05541	0.00220	0.01066
USPS	26.36772	862.40248		0.22244	16.97416
MNIST	1330.10100	121644.25098		5.30934	909.81591

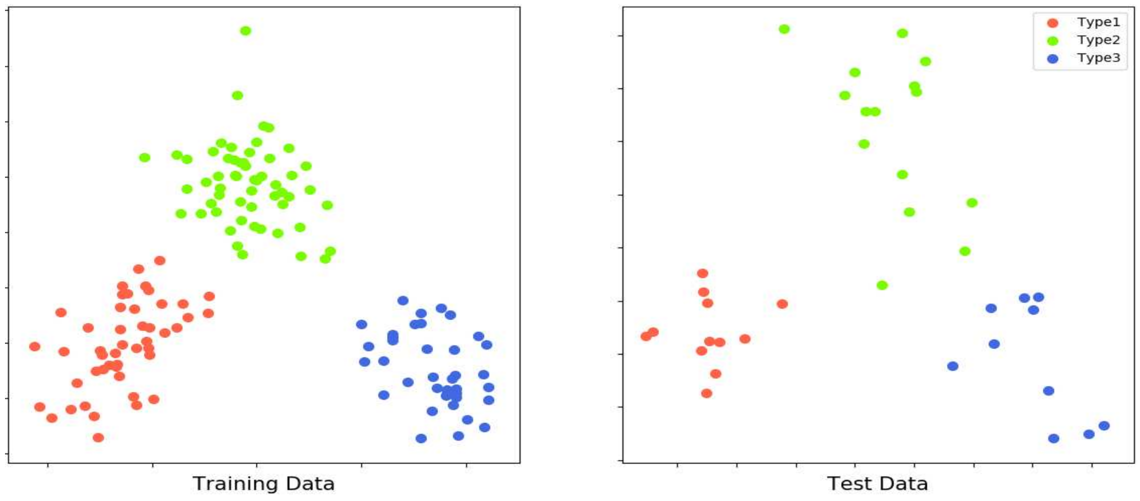


Figure 7.2: Two dimensional plot of Wine samples training and test digits using CCR.

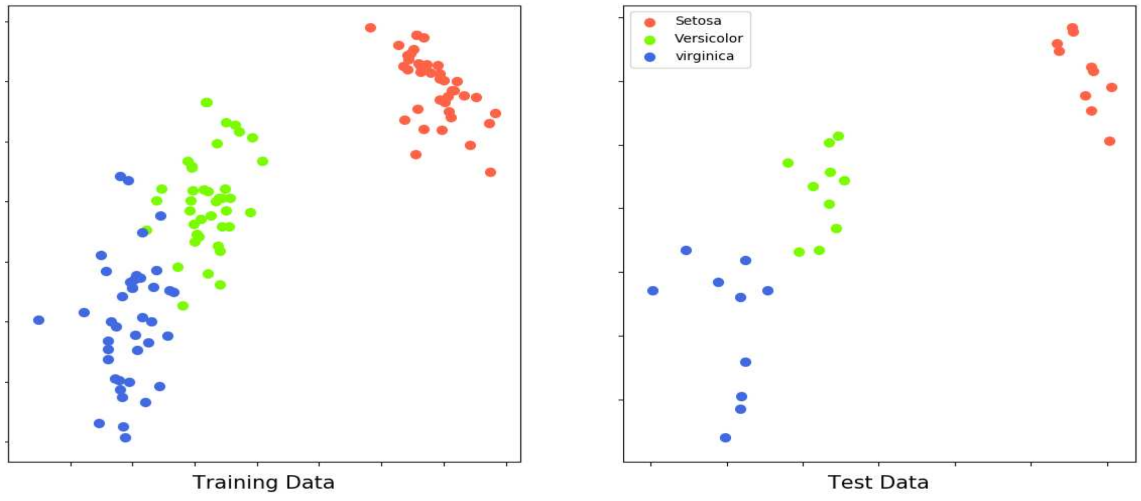


Figure 7.3: Two dimensional plot of training and test samples of Iris data set using CCR.

We present the visualization of Ion in Figure 7.4. The 'Bad' samples are tightly clustered whereas the 'Good' ones are scattered a lot. There is some overlap between the two classes which affects the classification accuracy in 2D space. At last the USPS data is shown in Figure 7.5.

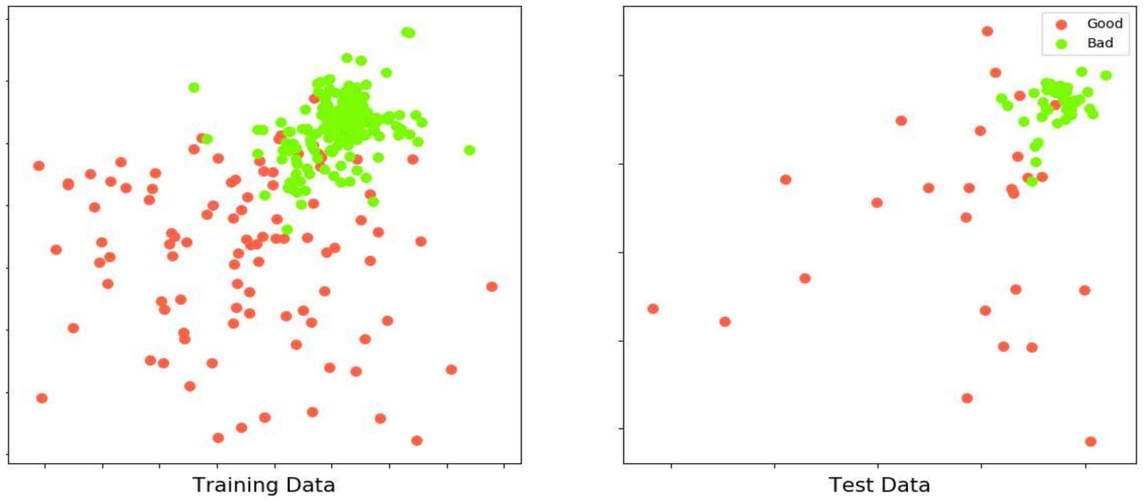


Figure 7.4: Two dimensional plot of training and test samples of Ionosphere data set using CCR.

The digit classes '3', '8' and '6' are clustered at the left in both training and test set. Perhaps the similarities among these digits are making them neighbors in 2D space. Digits '5' and '9' are placed at the top right corner. There is a lot of overlap among the classes '7', '4', '2' and '1'. Digit '0' is put next to the digits '3', '8' and away from digit '1' which is dissimilar to '0'.

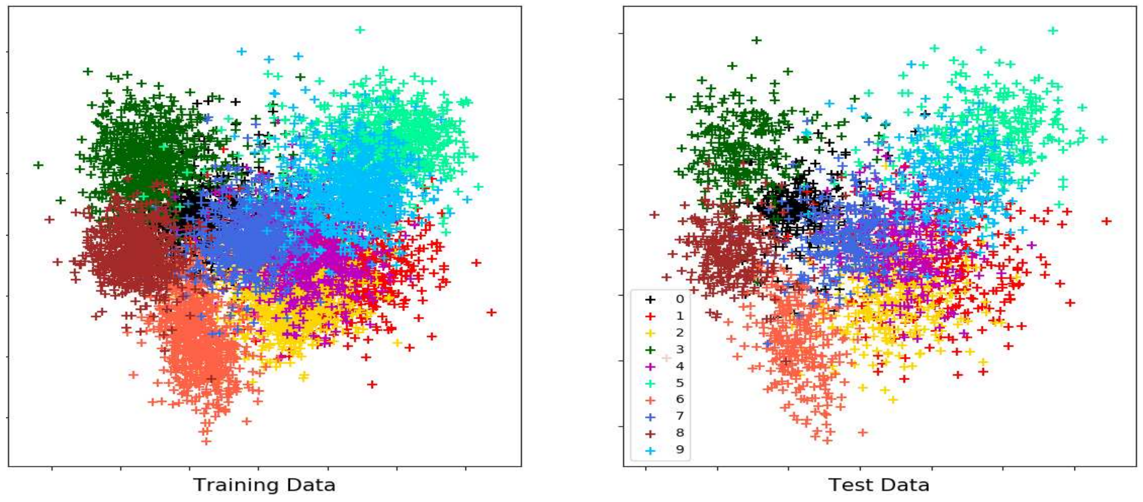


Figure 7.5: Two dimensional plot of USPS training and test digits by CCR.

Experiment 2

We did our last classification experiment with three high dimensional data sets Yale Face, ORL, and COIL20. Our goal is to compare the models by performing classification on different

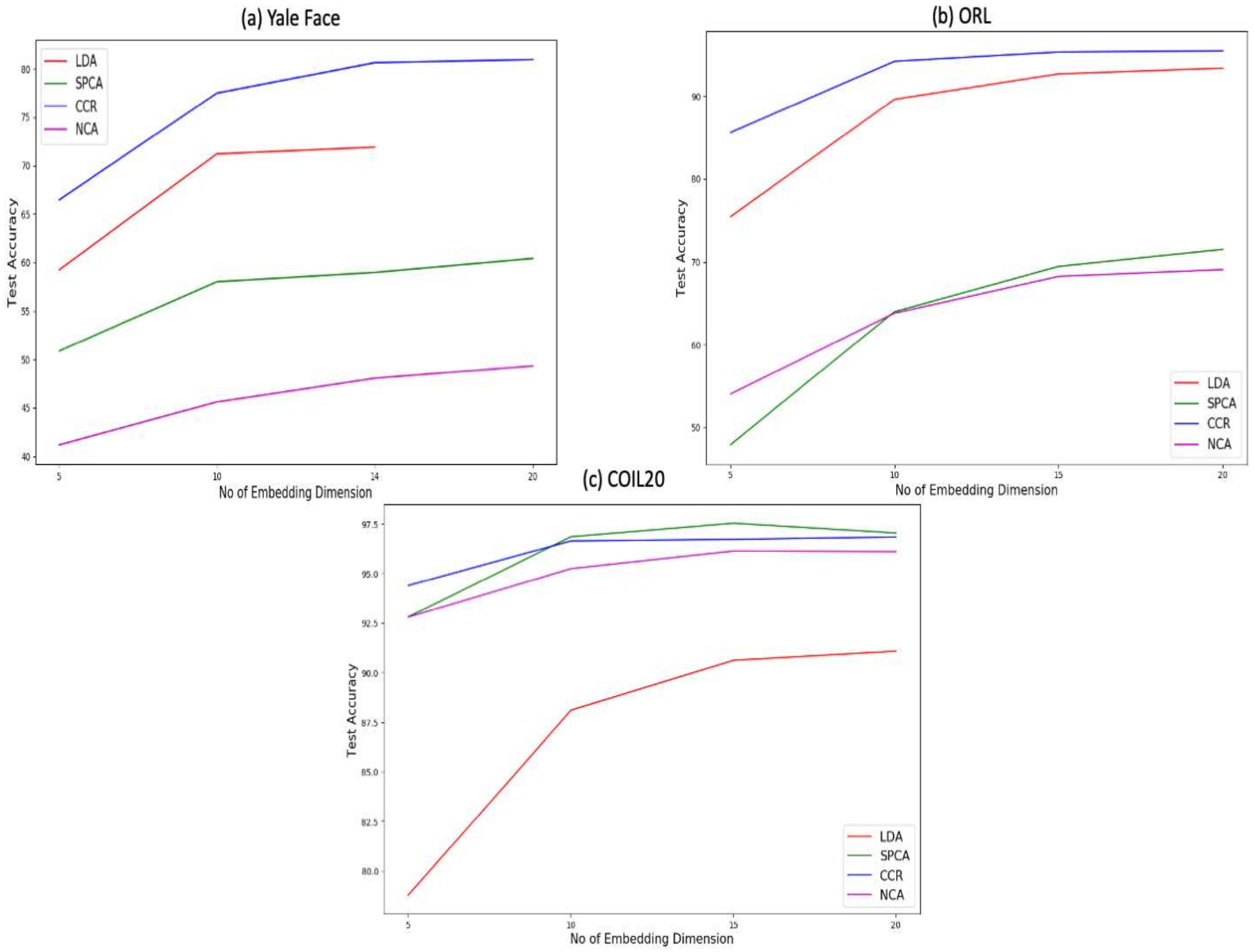


Figure 7.6: Classification accuracy of different DR methods on (a) Yale Face (b) ORL and (c) COIL20 data sets. Classification is performed on 5, 10, 15 and 20 dimensional embedding space. As Yale Face has 15 classes so the maximum embedding dimension for LDA is 14 ($\#class - 1$).

embedding dimensions 5, 10, 15, and 20. We split each dataset into training and test sets by taking 50% random samples from each class into the training set and the rest into the test set. After that, we trained each model on the training set and projected the training and test cases on different embedding dimensions to run a $5-$NN classification to measure the accuracy of the test set. We repeated the steps 25 times and reported the average test accuracy. In Figure 7.6 we plotted the test accuracy across the embedding dimensions for the three data sets. We see that in each model, the accuracy increases with the increase of the projection dimension across the three data sets. In Yale Face and ORL data, CCR outperformed all the models across different projection dimensions. In the COIL20 data, the performance of our model is comparable to SPCA. The high accuracy of CCR in most cases can be attributed to the class separation $\mathcal{L}_2$ terms in the optimization. The models SPCA and NCA learn an embedding by keeping the samples of a class close together. These two models don't have any mechanism to push the classes from each other. Therefore if two classes are close in ambient space, SPCA and NCA won't make any effort to separate them in low dimensional space; and as a result, the k-NN classification will suffer. On the other hand, LDA incorporates class separation by maximizing the volume of the between-class scatter matrix along with minimizing the within-class scatter. Although for high dimensional data sets, the within-class scatter matrix becomes singular, which deteriorates the performance of LDA [51]. Please note that CCR won't suffer from the matrix singularity problem; see Section 7.7 for more details.

7.7 Difference between CCR and LDA

While our algorithm has similarities to LDA, there are three fundamental differences between the formulation of classical LDA and CCR that we now describe.

- The objective of multi-class LDA is the ratio of the between-class scatter to within-class scatter where the scatters are calculated using determinant [144]. In contrast, our proposed objective is the sum of two parts, where the first part $\mathcal{L}_1$ controls the within-class scatter and the second part $\mathcal{L}_2$ controls the between-class scatter. We used trace to measure the within-class scatter.
- The optimization problem of multi-class LDA is not convex while the optimization of CCR is convex. Note, Kim et al. [147] have shown that the two-class LDA can be solved under a convex optimization problem.

- The generalized eigenvalue solution of LDA assumes that the within-class scatter matrix (S_w) is non-singular. But researchers have reported that when the dimension of input data is larger than number of samples, the S_w becomes singular [51] which results in poor performance. We observed that in the last experiment with Yale Face, ORL and COIl20 data. In contrast, CCR doesn't suffer from this problem. As a result the classification performance of CCR is significantly higher than LDA on Yale Face data.

7.8 Conclusions

This section introduces a new supervised linear dimensionality reduction technique called Centroid Component Retrieval (CCR). We showed that the proposed method is convex. We also presented a novel initialization technique that makes the convergence fast. We demonstrated that in most cases, the 2D embedding of CCR yields better classification accuracy than other methods. Although the classification performance of CCR is very similar to LDA, CCR has shown significant improvement when the input dimension of data is relatively higher than the number of samples, e.g., COIL20, Yale Face, and ORL. The advantage of CCR over other data visualization techniques, e.g., NCA and MCML, is that it does not require the pairwise distance calculation. The only overhead of CCR is the calculation of centroids of each class, but this is linear with the number of classes. Therefore our method scales easily compared to NCA and MCML, which becomes prohibitively slow with the increase of data.

Chapter 8

Conclusion and Future Research

In this dissertation, we proposed several new tools for dimensionality reduction, visualization, and feature selection using convex and nonconvex optimization problems. We formulated the following nonconvex models Centroid-Encoder (CE), Bottleneck Centroid-Encoder (BCE), Sparse Centroid-Encoder (SCE), Principal Centroid Component Analysis (PCCA), and one convex model called Centroid Component Retrieval (CCR). Note that PCCA is a nonconvex model, but it can be solved in closed-form using eigendecomposition; hence the solution can be ordered, unlike the iterative nonconvex methods, e.g., CE, SCE, and BCE. The empirical evaluation suggests the potential utility of these models.

Centroid-Encoder (CE), a deep neural network-based model designed for supervised dimensionality reduction, is an implementation of nonlinear supervised principal component analysis. The experimental results show that CE often produces better, if not optimal, generalization performance than other methods. The algorithms with the most similar prediction errors compared to CE require the computation of distances between all pairs of points. Our examples illustrate that CE captures the topological structure, i.e., class neighborhoods of the data, comparable to Laplacian Eigenmaps (LE) without calculating the neighborhood graph. In addition, CE exploits data labels to minimize the within-class variance to improve the localization of the mapping such that points are mapped more faithfully to their Voronoi regions in low dimensions. Unlike many other methods, e.g., Laplacian Eigenmaps and t-SNE, CE gives a mapping that can be applied to the new data. CE is well suited to visualize very large data sets, e.g., over a million data points, such as the SUSY analysis. We also showed empirically that the model globally captures high statistical variance relative to optimal linear transformations, i.e., more than PCA. We have also extended the CE by adding constraints in the bottleneck layer, called Bottleneck Centroid-Encoder (BCE), which improves the class localization and separability in the low-dimensional space compared to CE.

In the future, we plan to extend these models to the setting of unsupervised and semi-supervised learning.

We have demonstrated how Centroid-Encoder can be turned into an effective nonlinear feature selection tool by adding the 1-norm into the objective on CE to promote sparsity. The resulting model, Sparse Centroid-Encoder (SCE), extracts discriminatory features in groups by minimizing the 1-norm along with the centroid-encoder loss. The benchmarking results with six methods provide evidence that the features of SCE produce better generalization performance than other state-of-the-art models. We compared SCE with FsNet, mainly designed for high-dimensional biological data, and found that our proposed method outperformed it in most cases. The comparison includes neural network-based current state-of-the-art, e.g., Supervised CAE, LassoNet, Stochastic Gate, and DFS. SCE features consistently outperform these models on data sets where number of samples is significantly lower than number of features and cases where the number of observations is more than the number of variables. The empirical evaluation using an array of diverse data sets establishes the value of the Sparse Centroid-Encoder as a nonlinear feature detector. Our analysis of the Sparse Centroid-Encoder in Section 5.1.1 demonstrates that the 1-norm induces good feature sparsity. We chose the λ from the validation set from a wide range of values and saw that smaller values work better for classification. The visualization of the MNIST pixels (panel (a) of Figure 5.5) provides a qualitative justification for a high prediction rate. SCE selected most of the pixels from the central part of the image, ignoring the border, making sense as the digits lie in the center of a 28 x 28 grid. The ℓ_1 penalty on the SPL layer induces sharp sparsity on the SMK_CAN data (panel (b) and (c) of Figure 5.5) without shrinking all the variables. Our feature cut-off technique correctly demarcates the important features from the rest.

SCE compares favorably to the neural network-based model FsNet, SCAE DFS, LassoNet, and STG, where samples are mapped to class labels. SCE employs multiple centroids to capture the variability within a class, improving the prediction rate of unknown test samples. In particular, the prediction rate on the ISOLET improved significantly from one centroid to multiple centroids suggesting the speech classes are multi-modal. The two-dimensional PCA of ISOLET classes further confirms the multi-modality of data. We also observed an enhanced classification rate on MNIST, FMNIST, and Activity data with multiple centroids. In contrast, single-center per class performed better for other data sets (e.g., COIL-20, Mice Protein, GM12878, etc.). Hence, apart from producing an improved prediction rate using features that capture intra-class variance, our model can provide extra information about whether the data is unimodal or multi-modal. This aspect of sparse centroid-encoder distinguishes it from the techniques which do not model the multi-modal nature of the data.

In the final two chapters of this dissertation, we presented two linear models for dimensionality reduction. The Centroid Component Retrieval (CCR) shares a similar design principle to Linear Discriminant Analysis, but unlike multi-class LDA, CCR is a convex optimization problem. We have presented an initialization technique leveraging eigendecomposition and empirically showed that the initial condition significantly improves convergence time. Our experimental results favor CCR over LDA on high-dimensional data sets, outperforming LDA in k-NN classification in all cases. We have also observed that for those data sets, Yale Faces, ORL, and COIL20, CCR is consistently more accurate than other supervised techniques. Although CCR and LDA have a similar design principle, we haven't studied whether their solution is equivalent in theory. In the future, we would like to investigate this avenue.

We proposed Principle Centroid Component Analysis (PCCA) as a linear Centroid-Encoder. The objective function of PCCA is similar to PCA; thus, it can also be thought of as a new formulation of Supervised PCA. The solution of PCCA is in closed form using eigendecomposition, and we established a connection between the eigenvalue and the final cost. At last, we evaluated this model with five real-world data sets and found that it performed better than PCA and a version of supervised PCA. In the current formulation, the embedding is driven by the position of class centroids in the ambient space. If two centroids are close in the original space, the solution will put those two classes nearby in reduced space, increasing the misclassification. In the future, we will explore different ideas to push the nearby classes apart in low-dimensional space.

8.0.1 Limitations

In this section, we comment on the limitations of our work. We discuss some of the limitation of each of the five models.

Centroid-Encoder

Although the results show that Centroid-Encoder produces better visualization with relatively high accuracy compared to other models, it has some limitations which we elaborate below.

1) Data where class-centroids coincide: Centroid-encoder runs on the assumption that the mean of each class is different, therefore mapping all the samples to its class-centroid will capture the discriminative features among the classes. Our model will work as long as the centroids of each class do not overlap in the ambient space. For example, let's consider a synthetic data of concentric circles as shown in Figure 8.1. Each color represents a class. The position of the centroids of each class is the same which is the center of the circles. In this case, centroid-encoder will map all the points from different classes to the same location. In practice, it's very rare to have a data set where the class-centroids coincide.

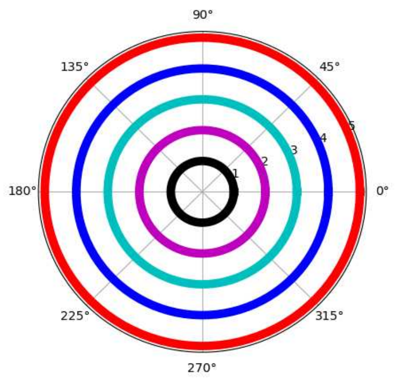


Figure 8.1: Concentric circles.

2) Non-convexity: The nature of the optimization of our model is strictly non-convex. Therefore the model runs with the risk of stopping in a local minimum. Because of this, multiple runs on the same data set can produce different embeddings. In this case one can pick the solution with lowest error.

Bottleneck Centroid-Encoder

Like Centroid-Encoder, Bottleneck Centroid-Encoder (BCE) is a nonconvex optimization problem. Therefore, we recommend running the model multiple times and picking the solution with minimum training error. If a validation set is available, then the best model should be selected, which gave the minimum validation error. We also comment that the BCE incorporates a repulsive force to separate the classes in low-dimensional space, and this force may change the geometric structure of classes. Therefore it's recommended to study the BCE embedding along with a CE embedding to explore the geometric relationships among classes.

Sparse Centroid-Encoder

Sparse Centroid-Encoder uses ℓ_1-norm to promote sparsity. Note, ℓ_1-norm shrinks the variables when used with a large value of λ. In our experiments, we used small values and didn't observe the shrinkage problem. So, running the model with a small λ is recommended. The SCE model uses the Centroid-Encoder cost, and the model may not pick discriminatory features if two class centroids are close in the ambient space.

Principal Centroid Component Analysis

Being a line technique, the model won't capture nonlinear relationships. PCCA uses eigendecomposition of the data matrix, which becomes significantly large with the number of features. In these scenarios, it may take a long time to get the solution.

Centroid Component Retrieval

Like PCCA, the model won't capture the nonlinear relationships.

Bibliography

[1] Andre Stuhlsatz, Jens Lippel, and Thomas Zielke. Feature extraction with deep neural networks by a generalized discriminant analysis. *IEEE transactions on neural networks and learning systems*, 23(4):596–608, 2012.

[2] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *AISTATS*, volume 5 of *JMLR Proceedings*, pages 384–391. JMLR.org, 2009.

[3] Martin R Min, Laurens Maaten, Zineng Yuan, Anthony J Bonner, and Zhaolei Zhang. Deep supervised t-distributed embedding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 791–798, 2010.

[4] Martin Renqiang Min, Hongyu Guo, and Dongjin Song. Exemplar-centered supervised shallow parametric data embedding. *arXiv preprint arXiv:1702.06602*, 2017.

[5] Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *J. Mach. Learn. Res.*, 11:451–490, March 2010.

[6] Yifeng Li, Chih-Yu Chen, and Wyeth W Wasserman. Deep feature selection: theory and application to identify enhancers and promoters. *Journal of Computational Biology*, 23(5):322–336, 2016.

[7] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[8] Sofya Chepushtanova, Christopher Gittins, and Michael Kirby. Band selection in hyperspectral imagery using sparse support vector machines. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XX*, volume 9088, page 90881F. International Society for Optics and Photonics, 2014.

[9] Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021.

[10] Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *International Conference on Machine Learning*, pages 10648–10659. PMLR, 2020.

[11] Dinesh Singh, Héctor Climente-González, Mathis Petrovich, Eiryo Kawakami, and Makoto Yamada. Fsnet: Feature selection network on high-dimensional biological data. *arXiv preprint arXiv:2001.08322*, 2020.

[12] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June 2003.

[13] Gordon Bell, Tony Hey, Alex Szalay, et al. Beyond the data deluge. *Science*, 323(5919):1297–1298, 2009.

[14] Andrew Bevan. The data deluge. *Antiquity*, 89(348):1473–1484, 2015.

[15] Richard G Baraniuk. More is less: Signal processing and the data deluge. *Science*, 331(6018):717–719, 2011.

[16] Michael C Schatz and Ben Langmead. The dna data deluge. *Ieee Spectrum*, 50(7):28–33, 2013.

[17] A C Pease, D Solas, E J Sullivan, M T Cronin, C P Holmes, and S P Fodor. Light-generated oligonucleotide arrays for rapid dna sequence analysis. *Proceedings of the National Academy of Sciences*, 91(11):5022–5026, 1994.

[18] Dari Shalon, Stephen J Smith, and Patrick O Brown. A dna microarray system for analyzing complex dna samples using two-color fluorescent probe hybridization. *Genome research*, 6(7):639–645, 1996.

[19] Michael L Metzker. Sequencing technologies—the next generation. *Nature reviews genetics*, 11(1):31, 2010.

[20] Jason A Reuter, Damek V Spacek, and Michael P Snyder. High-throughput sequencing technologies. *Molecular cell*, 58(4):586–597, 2015.

[21] Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review. *Data Clustering*, pages 29–60, 2018.

[22] DM Deepak Raj and R Mohanasundaram. An efficient filter-based feature selection model to identify significant features from high-dimensional microarray data. *Arabian Journal for Science and Engineering*, 45(4):2619–2630, 2020.

[23] M Aminian, T Ghosh, A Peterson, AL Rasmussen, S Stiverson, K Sharma, and M Kirby. Early prognosis of respiratory virus shedding in humans. *Scientific reports*, 11(1):1–15, 2021.

[24] I.T. Jolliffe. *Principal Component Analysis*. Springer, New York, 1986.

[25] T. Kohonen. Boosting the computing power in pattern recognition by unconventional architectures. In *Proceedings of the World Congress on Neural Networks*, volume IV, pages 1–4, Portland, OR, 1993.

[26] V. de Silva . B. Tenenbaum and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[27] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[28] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Phil. Mag. S.*, 2(11):559–572, 1901.

[29] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, September, 1933.

[30] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[31] Sofya Chepushtanova, Elin Farnell, Eric Kehoe, Michael Kirby, and Henry Kvinge. Dimensionality reduction. In *Data Science for Mathematicians*, pages 291–337. Chapman and Hall/CRC, September 2020.

[32] Garrison W Cottrell and Paul Munro. Principal components analysis of images via back propagation. In *Visual Communications and Image Processing'88: Third in a Series*, volume 1001, pages 1070–1077. SPIE, 1988.

[33] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.*, 37(2):233–243, 1991.

[34] Mark A. Kramer. Autoassociative neural networks. *Comput. Chem. Engng.*, 16(4):313–328, 1992.

[35] E. Oja. Data compression, feature extraction, and autoassociation in feedforward neural networks. In T. Kohonen, K. Mäkisara., O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 737–745, NY, 1991. Elsevier Science.

[36] Geoffrey E Hinton. Connectionist learning procedures. In *Machine learning*, pages 555–610. Elsevier, 1990.

[37] M. Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. Wiley, 2001.

[38] Kun Wang, Vineet Bhandari, Sofya Chepustanova, Greg Huber, Stephen O' Hara, Corey S O' Hern, Mark D Shattuck, and Michael Kirby. Which biomarkers reveal neonatal sepsis? *PloS one*, 8(12):e82700, 2013.

[39] Stephen O'Hara, Kun Wang, Richard A Slayden, Alan R Schenkel, Greg Huber, Corey S O'Hern, Mark D Shattuck, and Michael Kirby. Iterative feature removal yields highly discriminative pathways. *BMC genomics*, 14(1):1–15, 2013.

[40] Tomojit Ghosh, Xiaofeng Ma, and Michael Kirby. New tools for the visualization of biological pathways. *Methods*, 132:26 – 33, 2018. Comparison and Visualization Methods for High-Dimensional Biological Data.

[41] Tomojit Ghosh and Michael Kirby. Supervised dimensionality reduction and visualization using centroid-encoder, 2020.

[42] Gen Li, Yuantao Gu, and Jie Ding. <inline-formula><tex-math notation="latex">ℓ_1</tex-math></inline-formula> regularization in two-layer neural networks. *IEEE Signal Processing Letters*, 29:135–139, 2022.

[43] Gen Li, Yuantao Gu, and Jie Ding. The efficacy of l_1 regularization in two-layer neural networks. *arXiv preprint arXiv:2010.01048*, 2020.

[44] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[45] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.

[46] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[47] T. Kohonen. Adaptive, associative, and self-organizing functions in neural computing. *Applied Optics*, 26(23):4910–4918, 1987.

[48] Xiaofeng Ma, Michael Kirby, and Chris Peterson. Self-organizing mappings on the flag manifold. In *International Workshop on Self-Organizing Maps*, pages 13–22. Springer, June 2019.

[49] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[50] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recogn.*, 44(7):1357–1371, July 2011.

[51] Daniela M Witten and Robert Tibshirani. Penalized classification using fisher's linear discriminant. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):753–772, 2011.

[52] S. Watanabe. Karhunen–Loève expansion and factor analysis. In *Trans. 4th. Prague Conf. on Inf. Theory, Statist. Decision Functions, and Random Proc.*, pages 635–660, Prague, 1965.

[53] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, 43:59, 1982.

[54] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[55] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[56] Geoffrey E Hinton and Sam T. Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT Press, 2003.

[57] Killan Q. Weinberger and Lawrence K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1683–1686. AAAI Press, 2006.

[58] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[59] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[60] Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Technical report, Stanford University, 2004.

[61] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Willey & Sons, New York, 1973.

[62] Eric Bair, Trevor Hastie, Debashis Paul, and Robert Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137, 2006.

[63] Le Song, Alexander J. Smola, Karsten M. Borgwardt, and Arthur Gretton. Colored maximum variance unfolding. In *NIPS*, 2007.

[64] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 513–520, Cambridge, MA, USA, 2004. MIT Press.

[65] Tomoharu Iwata, Kazumi Saito, Naonori Ueda, Sean Stromsten, Thomas L. Griffiths, and Joshua B. Tenenbaum. Parametric embedding for class visualization. *Neural Comput.*, 19(9):2536–2556, September 2007.

[66] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419, 2007.

[67] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *Advances in neural information processing systems*, pages 451–458, 2006.

[68] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[69] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[70] Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. *Advances in neural information processing systems*, 31:107–117, 2018.

[71] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[72] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[73] Jian Cheng, Can Cheng, and Yi-nan Guo. Supervised isomap based on pairwise constraints. In *Proceedings of the 19th International Conference on Neural Information Processing - Volume Part I*, ICONIP'12, pages 447–454, Berlin, Heidelberg, 2012. Springer-Verlag.

[74] Shi-qing Zhang. Enhanced supervised locally linear embedding. *Pattern Recogn. Lett.*, 30(13):1208–1218, October 2009.

[75] B. Raducanu and F. Dornaika. A supervised non-linear dimensionality reduction approach for manifold learning. *Pattern Recogn.*, 45(6):2432–2444, June 2012.

[76] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: an empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *arXiv preprint arXiv:2012.04456*, 2020.

[77] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[78] Cosmin Lazar, Jonatan Taminau, Stijn Meganck, David Steenhoff, Alain Coletta, Colin Molter, Virginie de Schaetzen, Robin Duque, Hugues Bersini, and Ann Nowe. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(4):1106–1119, 2012.

[79] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[80] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[81] Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24(1):175–186, 2014.

[82] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine learning research*, 5(9), 2004.

[83] Naoual El Aboudi and Laila Benhlima. Review on wrapper feature selection approaches. In *2016 International Conference on Engineering & MIS (ICEMIS)*, pages 1–5. IEEE, 2016.

[84] Chun-Nan Hsu, Hung-Ju Huang, and Stefan Dietrich. The annigma-wrapper approach to fast feature selection for neural nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(2):207–212, 2002.

[85] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. *Machine Learning*, 1988.

[86] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[87] Thomas Navin Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. Embedded methods. In *Feature extraction*, pages 137–165. Springer, 2006.

[88] Valeria Fonti and Eduard Belitser. Feature selection using lasso. *VU Amsterdam Research Paper in Business Analytics*, 30:1–25, 2017.

[89] R Muthukrishnan and R Rohini. Lasso: A feature selection technique in predictive modeling for machine learning. In *2016 IEEE international conference on advances in computer applications (ICACA)*, pages 18–20. IEEE, 2016.

[90] Yongdai Kim and Jinseog Kim. Gradient lasso for feature selection. In *Proceedings of the twenty-first international conference on Machine learning*, page 60, 2004.

[91] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[92] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[93] Ben J Marafino, W John Boscardin, and R Adams Dudley. Efficient and sparse feature selection for biomedical text classification via the elastic net: Application to icu risk stratification from nursing notes. *Journal of biomedical informatics*, 54:114–120, 2015.

[94] Li Shen, Sungeun Kim, Yuan Qi, Mark Inlow, Shanker Swaminathan, Kwangsik Nho, Jing Wan, Shannon L Risacher, Leslie M Shaw, John Q Trojanowski, et al. Identifying neuroimaging and proteomic biomarkers for mci and ad via the elastic net. In *International Workshop on Multimodal Brain Image Analysis*, pages 27–34. Springer, 2011.

[95] Artem Sokolov, Daniel E Carlin, Evan O Paull, Robert Baertsch, and Joshua M Stuart. Pathway-based genomics prediction using generalized elastic net. *PLoS computational biology*, 12(3):e1004790, 2016.

[96] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[97] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[98] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[99] Seong Gon Kim, Nawanol Theera-Ampornpunt, Chih-Hao Fang, Mrudul Harwani, Ananth Grama, and Somali Chaterji. Opening up the blackbox: an interpretable deep neural network-based classifier for cell-type specific enhancer predictions. *BMC systems biology*, 10(2):243–258, 2016.

[100] Debaditya Roy, K Sri Rama Murty, and C Krishna Mohan. Feature selection using deep neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2015.

[101] Kai Han, Yunhe Wang, Chao Zhang, Chao Li, and Chao Xu. Autoencoder inspired unsupervised feature selection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945. IEEE, 2018.

[102] Aboozar Taherkhani, Georgina Cosma, and T Martin McGinnity. Deep-fs: A feature selection algorithm for deep boltzmann machines. *Neurocomputing*, 322:22–37, 2018.

[103] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[104] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[105] Muhammed Fatih Balın, Abubakar Abid, and James Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International conference on machine learning*, pages 444–453. PMLR, 2019.

[106] Adriana Romero, Pierre Luc Carrier, Akram Erraqabi, Tristan Sylvain, Alex Auvolat, Etienne Dejoie, Marc-André Legault, Marie-Pierre Dubé, Julie G Hussin, and Yoshua Bengio. Diet networks: thin parameters for fat genomics. *arXiv preprint arXiv:1611.09340*, 2016.

[107] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[108] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, pages 1–7. Austin, TX, 2010.

[109] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[110] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 153–160, Cambridge, MA, USA, 2006. MIT Press.

[111] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[112] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD dissertation, Harvard University, August 1974.

[113] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantization design. *IEEE transactions on Communications*, 28(1):84–95, January 1980.

[114] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[115] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[116] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[117] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[118] Roland Memisevic and Geoffrey Hinton. Multiple relational embedding. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 913–920, Cambridge, MA, USA, 2004. MIT Press.

[119] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

[120] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[121] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[122] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[123] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[124] Tzu-Yu Liu, Thomas Burke, Lawrence P Park, Christopher W Woods, Aimee K Zaas, Geoffrey S Ginsburg, and Alfred O Hero. An individualized predictor of health and disease using paired reference and target samples. *BMC bioinformatics*, 17(1):1–15, 2016.

[125] Rafael A Irizarry, Bridget Hobbs, Francois Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003.

[126] Matthew E Ritchie, Belinda Phipson, DI Wu, Yifang Hu, Charity W Law, Wei Shi, and Gordon K Smyth. limma powers differential expression analyses for rna-sequencing and microarray studies. *Nucleic acids research*, 43(7):e47–e47, 2015.

[127] Kohji Yamada, Mutsumi Hayashi, Hiroko Madokoro, Hiroko Nishida, Wenlin Du, Kei Ohnuma, Michiie Sakamoto, Chikao Morimoto, and Taketo Yamada. Nuclear localization of cd26 induced by a humanized monoclonal antibody inhibits tumor cell growth by modulating of polr2a transcription. *PloS one*, 8(4):e62304, 2013.

[128] Artem Barski, Suresh Cuddapah, Kairong Cui, Tae-Young Roh, Dustin E Schones, Zhibin Wang, Gang Wei, Iouri Chepelev, and Keji Zhao. High-resolution profiling of histone methylations in the human genome. *Cell*, 129(4):823–837, 2007.

[129] A Kreisler, PL Strissel, R Strick, SB Neumann, U Schumacher, and CM Becker. Regulation of the nrsf/rest gene by methylation and creb affects the cellular phenotype of small-cell lung cancer. *Oncogene*, 29(43):5828–5838, 2010.

[130] Hui Wang, Elizabeth C Curran, Thomas R Hinds, Edith H Wang, and Ning Zheng. Crystal structure of a taf1-taf7 complex in human transcription factor iid reveals a promoter binding module. *Cell research*, 24(12):1433–1444, 2014.

[131] Menno P Creyghton, Albert W Cheng, G Grant Welstead, Tristan Kooistra, Bryce W Carey, Eveline J Steine, Jacob Hanna, Michael A Lodato, Garrett M Frampton, Phillip A Sharp, et al. Histone h3k27ac separates active from poised enhancers and predicts developmental state. *Proceedings of the National Academy of Sciences*, 107(50):21931–21936, 2010.

[132] Yan-Wei Yin, Hong-Jian Jin, Wenjing Zhao, Beixue Gao, Jiangao Fang, Junmin Wei, Donna D Zhang, Jianing Zhang, and Deyu Fang. The histone acetyltransferase gcn5 expression is elevated and regulated by c-myc and e2f1 transcription factors in human colon cancer. *Gene expression*, 16(4):187, 2015.

[133] Paolo Salomoni and Pier Paolo Pandolfi. The role of pml in tumor suppression. *Cell*, 108(2):165–170, 2002.

[134] Gui-Ping Yu, Yong Ji, Guo-Qiang Chen, Bin Huang, Kai Shen, Song Wu, and Zhen-Ya Shen. Application of runx3 gene promoter methylation in the diagnosis of non-small cell lung cancer. *Oncology letters*, 3(1):159–162, 2012.

[135] Wenyi Mi, Yi Zhang, Jie Lyu, Xiaolu Wang, Qiong Tong, Danni Peng, Yongming Xue, Adam H Tencer, Hong Wen, Wei Li, et al. The zz-type zinc finger of zzz3 modulates the atac complex-mediated histone acetylation and gene activation. *Nature communications*, 9(1):1–9, 2018.

[136] Yichao Cai, Ying Zhang, Yan Ping Loh, Jia Qi Tng, Mei Chee Lim, Zhendong Cao, Anandhkumar Raju, Erez Lieberman Aiden, Shang Li, Lakshmanan Manikandan, et al. H3k27me3-rich genomic regions can function as silencers to repress gene expression via chromatin interactions. *Nature communications*, 12(1):1–22, 2021.

[137] R Reshma, V Sowmya, and KP Soman. Dimensionality reduction using band selection technique for kernel based hyperspectral image classification. *Procedia Computer Science*, 93:396–402, 2016.

[138] Xianghai Cao, Cuicui Wei, Jungong Han, and Licheng Jiao. Hyperspectral band selection using improved classification map. *IEEE geoscience and remote sensing letters*, 14(11):2147–2151, 2017.

[139] Dimitris Bertsimas, Martin S Copenhaver, and Rahul Mazumder. The trimmed lasso: Sparsity and robustness. *arXiv preprint arXiv:1708.04527*, 2017.

[140] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 14(5):877–905, 2008.

[141] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

[142] Tomojit Ghosh and Michael Kirby. Supervised dimensionality reduction and visualization using centroid-encoder. *Journal of Machine Learning Research*, 23(20):1–34, 2022.

[143] Alexander Ritchie, Clayton Scott, Laura Balzano, Daniel Kessler, and Chandra S Sripada. Supervised principal component analysis via manifold optimization. In *2019 IEEE Data Science Workshop (DSW)*, pages 6–10. IEEE, 2019.

[144] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

[145] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 513–520, Cambridge, MA, USA, 2004. MIT Press.

[146] Juan Luis SuÃ¡rez, Salvador GarcÃa, and Francisco Herrera. pydml: A python library for distance metric learning. *Journal of Machine Learning Research*, 21(96):1–7, 2020.

[147] Seung-Jean Kim, Alessandro Magnani, and Stephen Boyd. Robust fisher discriminant analysis. *Advances in neural information processing systems*, 18, 2005.