Autoencoder-Based Techniques for Improved Classification in Settings with High Dimensional and Small Sized Data

A thesis submitted in fulfillment of the requirements for the degree

of

Doctor of philosophy

at

The University of Waikato

by

Maisa Daoud

in

*The **Machine Learning** group, department of **Computer Science***

September 15, 2020

Acknowledgements

By the name of the Lord the most gracious the most merciful.

Allow me to start by thanking the GOD acknowledging and the favor that both of my parents have bestowed upon me since the first moment of my life .. life keeps us busy but what I am sure of is " their care and love have never been conditional to a time or place" ... I am in debt to both of you!

To my children, Hazar, Fahmi, Siwar and Danah .. I was not the ideal mum through out the past four years .. I had a big sense of commitment to the project I started and my timetable was very tight. Sorry for the assemblies that I missed and the camps that I couldn't attend .... it has never been easy for me, as to any working parent, to say No or I can't .. it hurts us as much as, or even more than it did for you :(.. I should also add that having you guys involved in my study added a lot of joy and laughter. To my husband Mohammad! thanks for your support and help, it couldn't have been done without it.

Dr. Mike Mayo, my chief supervisor, thank you for your supervision, advice, reading my work and giving me feedback during the four years of study. My co-supervisors Dr.Sally Jo Cunningham and Dr.Tony Smith I appreciate all of your emotional support, encouragement, revisions and compliments these meant a lot.

To my siblings, my life-long friends (the Daoud's); brothers Mohammad, Daoud and Ameen and my sisters Sana, Laila and Lina thank you for the support and encouragement!

Throughout my hilly journey, I met many precious people who made the mission successfully doable. Thanks for every smile, to everyone who passed me with a greeting or a gesture. There are many of you to mention individually and I hope you know who you are. I was truly lucky to meet all the academic, the non academic staff members and students.. everyone, without exception, is precious and made an adorable memory .. I would particularly like to thank the ones who used to approach me to ask how I was doing. Dr. Eibe Frank .. It's been truly well said that "Eibe is the father of the ML students!" And that's what I've experienced. A special thank goes to you for providing me with the GPUs to complete my experiments. Jacqui and Tim Elphick, thank you for the invitations that made me feel involved, thank you for every single moment and bit which all formed an indescribable feeling. Dr. Simon Laing thank you for your humble nature and smiles :). Khadija Bahiss, Dahook Azzam, AttuAllah Sahito, Chen Zheng and the labs colleagues thank you for the unforgettable moments.

Last but not least, I want to thank the department of Computer Science, and Dr. David Bainbridge in particular for approving on funding my conferences and the journal fees. A special thank also to the School of Graduate Studies for granting me the University of Waikato Doctoral scholarship which made this undertaking financially viable.

Maisa Daoud

To the ones whom I miss and remember every hour. Mother (*Elham*) and Father (*Taher*).

Abstract

Neural network models have been widely tested and analysed using large sized high dimensional datasets. In real world application problems, the available datasets are often limited in size due to reasons related to the cost or difficulties encountered while collecting the data. This limitation in the number of examples may challenge the classification algorithms and degrade their performance. A motivating example for this kind of problem is predicting the health status of a tissue given its gene expression, when the number of samples available to learn from is very small.

Gene expression data has distinguishing characteristics attracting the machine learning research community. The high dimensionality of the data is one of the integral features that has to be considered when building predicting models. A single sample of the data is expressed by thousands of gene expressions compared to the benchmark images and texts that only have a few hundreds of features and commonly used for analysing the existing models. Gene expression data samples are also distributed unequally among the classes; in addition, they include noisy features which degrade the prediction accuracy of the models. These characteristics give rise to the need for using effective dimensionality reduction methods that are able to discover the complex relationships between the features such as the autoencoders.

This thesis investigates the problem of predicting from small sized high dimensional datasets by introducing novel autoencoder-based techniques to increase the classification accuracy of the data. Two autoencoder-based methods for generating synthetic data examples and synthetic representations of the data were respectively introduced in the first stage of the study. Both of these methods are applicable to the testing phase of the autoencoder and showed successful in increasing the predictability of the data.

Enhancing the autoencoder's ability in learning from small sized imbalanced data was investigated in the second stage of the project to come up with techniques that improved the autoencoder's generated representations. Employing the radial basis activation mechanism used in radial-basis function networks, which learn in a supervised manner, was a solution provided by this thesis to enhance the representations learned by unsupervised algorithms. This technique was later applied to stochastic variational autoencoders and showed promising results in learning discriminating representations from the gene expression data.

The contributions of this thesis can be described by a number of different methods applicable to different stages (training and testing) and different autoencoder models (deterministic and stochastic) which, individually, allow for enhancing the predictability of small sized high dimensional datasets compared to well known baseline methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autoencoders are multi-layer perceptron feed forward neural networks. They learn the representations of the data using unsupervised learning algorithm i.e. without considering the class outcome of the training examples. Autoencoders' learning algorithm can be framed as a classical optimization problem which goal is to find the set of parameter values that minimise a reconstruction based function such as the Mean Squared Error (MSE). Examples of well known autoencoders are the Stacked Denoising Autoencoder (SDA), adversarial autoencoders and the sparse autoencoders.

The general goal of training an autoencoder is to allow the neurons at every layer to learn abstract representations of the data which are, in ideal cases, discriminating to every cluster of the data. Trained autoencoders have been widely used for two general purposes: (i) initializing deep neural network architectures in which layers are treated as a stack of autoencoders [Baldi, 2012]. (ii) generating low dimensional representations of the data at the middle layer of the network to overcome the curse of dimensionality, sparsity, problem challenging the machine learning classifiers and analysis tools when used with the original high dimensional data. Autoencoders have proved remarkably successful in reducing the dimen-

sionality of the data and generating low dimensional representations compared to other linear dimensionality reduction methods such as the Principal Component Analysis (PCA) which have limitations in capturing the complex relationships between the data examples [Shiokawa *et al.*, 2018].

Nevertheless, reducing the dimensionality of the data using neural network methods, which have outperformed other methods [Tan & Eswaran, 2010], is not an easy goal to achieve. This technique requires careful selection of the network's parameters, choosing proper initialization methods and applying appropriate pre-processing techniques [Bengio *et al.*, 2013]. Autoencoder-based dimensionality reduction techniques have been investigated widely in literature, which is rich nowadays with methods designed to improve the tendency of capturing good representations of the data [Bengio *et al.*, 2013; Goodfellow *et al.*, 2014; Kingma & Welling, 2013; Vincent *et al.*, 2010b]; for instance, noising the data is a simple yet efficient technique applied in the SDA [Masci *et al.*, 2011] to make it robust to the noise. However, most of the available methods have been designed and empirically tested on large scale datasets. It has been also proved that learning representations from small sized data will potentially allow the network to overfit the training data and perform poorly in reconstructing or predicting future testing observations as a result [Vincent *et al.*, 2010b].

This thesis focuses on introducing autoencoder-based techniques for improving the classification accuracy of high dimensional small sized datasets. Generating good representations for small sized datasets is a significant challenge because of the need for a powerful dimensionality reduction method while the limitation in size poses a problem to neural network-based methods. We specifically chose to test the proposed autoencoder-based techniques on the gene expression datasets because of their distinguishing characteristics which include (i) the limited number of examples (a few hundred at most), (ii) the imbalanced distribution of

the examples between classes and (iii) the number of features per example is very high (1000-10000s). We also applied the techniques to benchmark images datasets such as the MNIST to comply with the common practice and preferences that we found in literature.

The remainder of this chapter is organized as follows: Section 1.1 presents the motivation. Section 1.2 presents the research questions and the proposed methods. Section 1.3 lists the published and unpublished works derived for this thesis and Section 1.4 sets the structure of this thesis.

1.1 Motivation

In this thesis, we are interested in finding autoencoder-based techniques that improve the the classification accuracy of the target data in two main folds: (i) exploiting the trained autoencoder models for solving the data related problems, e.g. class imbalance. (ii) improving the autoencoder's ability in learning from small sized datasets. The latter goal was achieved by applying techniques that proved successful in improving the locality of response to similar patterns of the data such as the radial-activations employed by the Radial Basis Functions RBF networks which were designed to learn from small sized datasets in a supervised manner. Most of the suggested autoencoder-based dimensionality reduction methods have been designed for and tested on benchmark datasets with large quantity of training and testing examples. Despite their powerful performance in most cases, the effectiveness of these methods degenerates sometimes when they are applied to small sized data. Even in cases where autoencoders successfully learned discriminating representations, the use of these models for solving problems such as the class imbalance, for example, have got limited if no attention. As a result, questions around improving or exploiting the available

autoencoding methods in solving real world application problems with limited number of examples is an open question requiring further investigation.

Transfer learning is one of the most popular techniques to overcome the problem of learning from small sized datasets. The method which basically depends on reusing the representations of the data that were previously learned for a different task in initializing a network that is specifically designed for a problem under investigation proved efficient in cases where the applications are sufficiently related. It has also achieved preliminary success in transferring the natural images' features to a new task [Ching *et al.*, 2018]. In fact, most of the available transfer-learning based methods were also designed for the image [Shin *et al.*, 2016] and electronic health record-based domains [Ching *et al.*, 2018]. Nonetheless, learning representations from phenotype data has received limited if no attention [Ching *et al.*, 2018]. Hence, it becomes a necessity to look for solutions to increase the prediction accuracy of this data, which we are considering in this study.

1.2 Research Questions

Having the problems of (i) predicting the class of small sized high dimensional data examples, (ii) the need for generating good discriminating representations for these examples to improve their classification accuracy, (iii) the need for solving other related problems such as the class imbalance problem as current opened problems led us to propose methods for enhancing the predictability of these datasets. Some of the introduced methods are applicable at the testing stage of the autoencoder's while the others are proposed to enhance the autoencdoer's ability in learning from small sized high dimensional data.

However, the researches done for this thesis aimed at answering the following questions:

(i) *Given a pre-trained autoencoder, can we generate different variations of a given image?* This question is related to the problem,mentioned in Chapter 3, of generating visually appealing testing examples using poorly trained autoencoders. The quality of the autoencoders' output relies on the training hypermeters which are set by trial-and-error techniques or using some heuristic methods which does not guarantee learning good representations or generating good reconstructed versions of the data. This thesis tries to answer this question by introducing an optimization-based technique that generates reconstructed version for an input test images. The idea is to see if the optimized low dimensional solutions generated by an optimization method can motivate the decoder's neurons to generate better reconstructed versions than the ones generated by the low dimensional representations generated by the encoder part of the network.

(ii) *What are the stat-of-the-art neural network based cancer prediction models?* Gene expression data is a well known example to small sized high dimensional datasets which are the settings that this thesis focuses at. Hence, before going further in designing techniques that enhance the predictability this data. This thesis will answer this question by showing, in Chapter 4, a complete study to the current neural network-based cancer prediction models, present the preprocssing tools, the networks architectures and the analysis tools.

(iii) *Can we use a pretrained autoencoder to generate synthetic gene expressions form the minority classes?* This question is related to question 1 and both of these questions are applicable at the testing stage of the autoencoder learning. This thesis attempts at solving the class imbalance problem which degrades the classifiers performance by introducing, in Chapter 5, a new autoencoder-based minority class oversampling technique to decrease the majority to minority class ratio and reduce the bias towards perfectly predicting one of the two classes.

Rather than looking for other off-line methods that exploit the learned rep-

resentations as initial seeds, We were motivated by the idea of enhancing the "learning" from small sized datasets in the second stage of the project which encouraged us to ask the following question.

(iv) *How can we enhance the autoencoder's ability in learning from samll sized dataset?* This question is tied directly with the problem of learning from small sized high dimensional datasets and will be investigated in Chapter 6 which will introduce the problem in details, provide mathematical foundation to the success of the supervised Radial Basis Function (RBF) networks which are commonly used with these settings and shows how RBF activations can be integrated withing the unsupervised learning autoencoders.

(v) *Can variational autoencoders be improved to learn from small sized imbalanced datasets?* Variational autoencoders are probabilistic networks that show a tendency to learn from a single cluster of the data and a degradation in its performance when used with small sized data. This question is answered in Chapter 7 by introducing a new network architecture and an alternative loss function to encourage learning from different clusters of the data and to overcome the above mentioned problems.

1.3 Objectives and Contributions

This section highlights the contributions of this thesis which are connected according to Figure 1.1 and itemized as follows:

- Introducing a technique for generating good reconstructed images from a poorly trained autoencoder

  M. Doaud and Mayo, M. Using swarm optimization to enhance autoencoder's images. In V. Torra, Y. Narukawa, A. Honda, & S. Inoue (Eds.), USB Proceedings of 14th International Conference on Modeling Decisions for Artificial Intelligence. pp. 118–131.

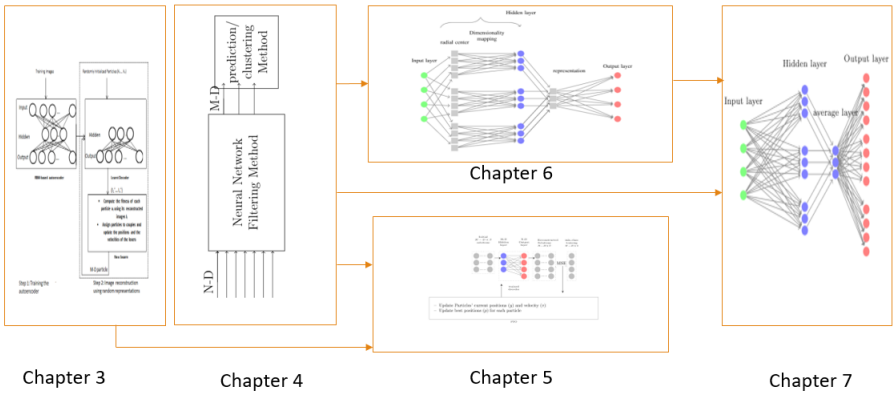


Figure 1.1: Thesis contribution map

2017 Kitakyushu, Japan.

- Surveying the state-of-the art neural network-based models for cancer prediction from microarray data.

  M. Daoud and M. Mayo, "A survey of neural network-based cancer predictionmodels from microarray data,"Artificial Intelligence in Medicine, 2019.

- Introducing a new synthetic minority class over-sampling approach that utilizes the decoder part of a previously trained autoencoder and the PSO.

  M. Daoud and M. Mayo, "A novel synthetic over-sampling technique for imbalanced classification of gene expressions using autoencoders and swarm optimization," in Australasian Joint Conference on Artificial Intelligence,pp. 603–615, Springer, 2018.

- Introducing the RBF autoencodrs, which utilizes the capability of the RBF networks in learning from small-sized data.

  M. Daoud, M. Mayo and S.J. Cunningham, RBFA: Radial Basis Function Autoencoders. In 2019 IEEE Congress on Evolutionary Computation (CEC) pp. 2966-2973, IEEE, 2019.

- Introducing a multi modal framework for training the Variational Autoencoder (VAE) to overcome the problem of singularity in traditional VAE.

M. Daoud, M. Mayo and E. Frank, Multi-Modal framework for training variational autoencoders. (to be submitted to the journal of Machine Learning)

1.4 Thesis Structure

This thesis is organised as follows: Chapter 2 presents a technical background for the tools forming the backbone of the introduced methods. Chapters 3-7 include our published works an unpublished work formatted. Each of these chapters is named by the publication's title. They are: "Using Swarm Optimization To Enhance Autoencoders Images" (Chapter 3), "A Survey of Neural Network-Based Cancer Prediction Models From Microarray Data (Chapter 4)", "A Novel Synthetic Over-sampling Technique for Imbalanced Classification of Gene Expressions using Autoencoders and Swarm Optimization" (Chapter 5), "RBFA: Radial Basis Function Autoencoders "(Chapter 6) and "Multi modal approach for training variational Autoencoders" (Chapter 7). Finally, Chapter 8 concludes the thesis and suggests future works.

Chapter 2

Literature Review

This chapter introduces an overview of the tools and techniques that are fundamental to the solutions proposed in the thesis. Each of the attached published and unpublished works includes its background section with detailed discussion about related state-of-art comparing methods. However, we chose to include a brief technical background about neural networks, PSO algorithms and microarray data to provide the reader with an initial understanding before going further in the thesis.

2.1 Multi Layer Perceptron Neural Networks

Multi layer perceptrons are one type of neural networks consisting of a number of interconnected neurons initialized with weight values and organized in layers to model the complex relationships of the data. A layer in most multi layer perceptron networks can be mathematically represented by a matrix mapping the input examples into an output of different cardinality using normal matrix multiplications rules. Non linear activation functions are commonly employed to activate the layers output to allow for approximating (learning) the nonlinear

relationships between the inputs and the target outputs. The output of every layer in the network is passed to the next level in the hierarchy in the multi layered architecture in order to generate an output at the output layer.

Multi layer perceptron networks are used as general function approximators so, with a large amount of data and careful selection to the network's parameters, they can approximate a continuous function representing the objective of learning [Chen *et al.*, 1991]. Hence, the algorithm employs a non linear optimization algorithm to optimize the network parameters to generate optimal outputs. The difference between the generated output and the target output is quantified and a backpropagation algorithm is used to propagate the error derivatives back through the network layers to fine-tune the neuron weights.

Multi layer perceptron networks have been applied to a variety of applications. For instance, it is used to predict the class of the un-labeled data by directly learning the features-label relationship in a supervised manner using the labeled training examples or to learn the input-output relationships using unsupervised learning algorithms. The latter approach is commonly used as a starting point for initializing other supervised learning-based networks as a mechanism to avoid random initialization which can lead to bad solutions [Vincent *et al.*, 2010b]. Alternatively, unsupervised learning approachs can learn low dimensional abstract representations of the inputs, this basic mechanism will be used in this project and will be presented in the next subsection. Generative models such as the adversarial networks [Goodfellow *et al.*, 2014] and the variational autoencoders [Kingma & Welling, 2013] are other examples to multi layer perceptron generative networks trained in unsupervised manner to generate new examples of the data.

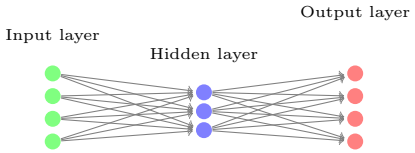


Figure 2.1: A simple autoencoder

2.1.1 Autoencoders

Autoencoders are simple feed forward networks belonging to the multi layer perceptron family of networks. The simplest form of the autoencoder consists of an input layer, hidden layer and an output layer stacked in two parts called *encoder* and *decoder* as shown by Figure 2.1. The layer separating both of these parts is usually defined with the minimum number of neurons to learn low dimensional codes of the data in the middle level of the network. These representations show better performance when used with classification, clustering and data analysis tools compared to the original high-dimensional data. A stacked and more complicated form of the autoencoder can be built by adding more layers with different architectures (e.g convolutional layers) to both sides of the network.

Despite their simple architecture, autoencoders form the back-bone of neural networks unsupervised learning paradigm. They learn using the same mechanism mentioned above for training the multi layer perceptron networks. SDA [Vincent *et al.*, 2010b] are simple autoencoders with the same architecture as normal autoencoders but trained to denoise corrupted versions of the inputs. The main motivation behind learning through decorruption was to move the learning concentration from finding perfect copies of the inputs, which maximize the mutual information, into learning interesting representations that capture useful structure of the data. This mechanism shows successful in learning robust high level representations (at the output layer) which is not affected by corrupted inputs. Vincent *et al.* [2010b] suggested three corruption techniques: (i) Gaussian

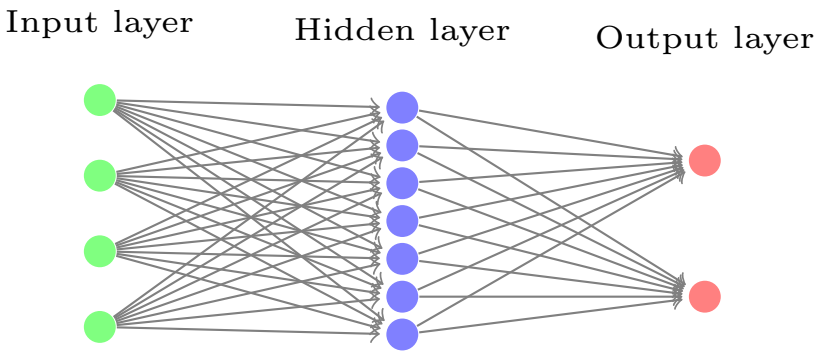


Figure 2.2: RBF network

noise, (ii) setting a fraction of randomly chosen features to zero and (iii) adding salt-and-pepper noise.

SDAs showed surprisingly good results in detecting edges from natural image patches [Vincent *et al.*, 2010b] and proficiency in learning representations of the data for different domain applications [Gu *et al.*, 2018; Scheithe *et al.*, 2019].

2.1.2 Radial Basis Functions Networks

Radial Basis Function networks (RBF) are simple networks with a single hidden layer that are recognized as a good alternative to multi layered networks which have significant amount of parameters [Chen *et al.*, 1991]. A typical RBF network consists of three layers: an input, a hidden and an output layer. The hidden layer in the middle of the network consists of center points; Each of which receives the input example and interpolates it using a radial function into a new space. The output of the hidden layer activates the weighted neurons, in the last layer, to generate an output indicating the class of the input example. The network depicted in Figure 2.2 represents an RBF network with an input layer of 4 neurons, 7 center points in the hidden layer and 2 neurons in the output to learn the class of binary-labeled examples.

The performance of the RBF networks depends upon the number of cen-

troids in the hidden level layer which can be initialized to arbitrarily chosen data points or by using the centroids found by a clustering algorithm [Que & Belkin, 2016]. The response of these centroids monotonically decreases with the distance between them and the input's features, which allows for capturing local representations of the data. However, using every training example as a center point would allow for overfitting; hence the overall error is increased. Overfitting is a problem accrues when the learning algorithm fits the training data too well but performs bad with the testing data. It is related to the bias/variance trade off explained by Geman *et al.* [1992] as a phenomena that allows the variance to be too large, if the algorithm fits the data too well.

It was theoretically proven that the choice of the radial function makes no crucial difference to its performance [Chen *et al.*, 1991]. However, the thin-plate-spline (Equation 2.1), Gaussian (Equation 2.2), multiquadratic (Equation 2.3) and inverse-multiquadratic Equation (2.4) functions are examples to the most commonly used RBF functions which are usually adopted based on the nature of the data.

$$\phi(v) = v^2 log(v) \tag{2.1}$$

$$\phi(v) = exp(\frac{-v^2}{\beta^2}) \tag{2.2}$$

$$\phi(v) = (v^2 + \beta^2)^{\frac{1}{2}} \tag{2.3}$$

$$\phi(v) = (v^2 + \beta^2)^{\frac{-1}{2}} \tag{2.4}$$

where v is the Euclidean norm $||x - c_i||$ between the data example x and the centroid c_i. β is a constant.

Recent works in RBF network have been suggested to solve real world application problems [Teng, 2018], to improve the performance of the network and reduce the complexity emerging with large scale data we can use methods to speed up the interpolation in large scale data such as using space division based techniques suggested in [Smolik & Skala, 2018].

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [Eberhart & Kennedy, 1995] is a stochastic evolutionary computation algorithm inspired by the swarm behaviour in bird flocking and is often used for optimizing continuous non linear functions with high dimensional space.

PSO was implemented using a straight forward algorithm. It starts with initializing a swarm of I potential solutions called particles. Initialization mechanisms that guarantee the convergence of the algorithm's fitness function into stable points were extensively provided by empirical studies. However, such guidelines can be only considered with careful attention as the PSO parameters are problem dependent [Van den Bergh & Engelbrecht, 2006].

Each particle i in the PSO is initialized at $t = 0$ with a position $y_{i,0}$ and velocity $v_{i,0}$ vectors updated in every iteration $t_1, ..., t_T$ of the algorithm to return the best fitted particle at the end. The update, given by the following equations, in the particle's related vectors is a function of the best position p_g found by the particles in the neighborhood region and the particle's personal best position p_i found so far:

$$v_{i,t} = v_{i,t-1} + U_{i,t}(0, \phi_1) \times (p_i - y_{i,t-1}) + U_{i,t}(0, \phi_2) \times (p_g - y_{i,t-1}) \tag{2.5}$$

$$y_{i,t} = v_{i,t} + y_{i,t-1} \tag{2.6}$$

where $U(0, \phi_1)$ and $U(0, \phi_2)$ or the *cognitive components* and the *social component* respectively are weighting vectors of randomly generated numbers uniformly distributed between $[0, \phi_1]$ and $[0, \phi_2]$ respectively for each particle i at each iteration t. The purpose of these vectors is to control the tendency of each particle to return to its best environment, and its tendency to follow other particles' best potential locations. It is a common practice to define p_g as a global parameter representing the best location found by the entire swarm as opposed to using a neighborhood function to define the neighbors of every particle and choosing the best location found by the neighbors as p_g for every particle [Parsopoulos & Vrahatis, 2002]. PSO evaluates the suitability of the particles based on a predefined fitness (objective) [1] function and updates the values of the p_i and p_g accordingly.

Despite its simplicity and applicability to wide rage of applications including the DNA compression [Zhu *et al.*, 2011] and resource allocation [Gong *et al.*, 2012], PSO showed poor performance in problems with large number of local minima [Poli *et al.*, 2007]. Studies in the PSO algorithm tried to improve the performance of the PSO by defining new velocity update formulas [Shi *et al.*, 2001] and restricting the growth of the velocity from exceeding the boundary of the search space, but these techniques increased the computational complexity of the algorithm [Suganthan, 1999]. However, Competitive Swarm Optimization (CSO) is a new variation to the PSO which omits the influence of the p_i and p_g parameters to reduce the algorithms complexity [Cheng & Jin, 2014].

[1] PSO Fitness and objective terms are used interchangeably in this thesis following the original paper [Poli *et al.*, 2007]

2.2.1 Competitive Swarm Optimization

CSO [Cheng & Jin, 2014] is a simple optimizer inspired by the PSO with some conceptual differences. In CSO, the competition is done in pairs where particles are randomly grouped into couples which are evaluated using a predefined fitness function. The algorithm only updates the velocity and the location of the loser particles based on their winning partners, which are not updated, before passing them in to the next generation of the swarm $t + 1$. CSO does not define the p_i nor the p_g parameters which reduces the number of comparisons and updates per generation to be $\frac{I}{2}$ compared to the original PSO. Hence, the update in the loser's position is only influenced by its winning partner according to the following:

$$v_{l,t} = r_{1,t}v_{l,t-1} + r_{2,t}(y_{w,t} - y_{l,t-1}) + \gamma r_{3,t}(\overline{y}_{t-1} - y_{l,t-1}) \tag{2.7}$$

and the position can be updated accordingly based on the following Equation:

$$y_{l,t} = y_{l,t-1} + v_{l,t} \tag{2.8}$$

where $r_1, r_2, r_3 \in [0, 1]$ are random vectors. $\overline{y}_t$ is the mean position of the swarm, γ is a parameter that controls the influence of $\overline{y}_t$ and y_w, y_l are the winner and the loser partners respectively.

CSO was empirically tested on six benchmark functions to study the influence of its parameters [Cheng & Jin, 2014]. It has also outperformed the-state-of-the-art algorithms which were designed for solving large-scale problems.

2.3 Microarray Datasets

Microaray technology is a laboratory tool for quantifying the amount of protein produced by the genes (expression level). The technology simply hybridizes two

Deoxyribonucleic Acid (DNA) (reverse transcript of mRNA) strands collected from two samples, e.g. diseased and healthy, by mixing them into a single microarray and scanning them after that with an appropriate light source. The intensities of the resulting array of features, i.e. image, represents the average difference between the matches and the mismatches and can be related to the amount of the mRNA, i.e expression, presents in the diseased tissue. Microarray data allows for measuring the expression levels in one assay which speeds the process of finding patterns in the gene expression.

Cancer prediction models are a set of methods collaborating to statistically analyse and predict the existence of cancer or the cancer type of the samples. The high-dimensionality and the limited size of the microarray datasets are crucial factors affecting their analysis. Feature selection or creation (projection) techniques have been used widely to: (i) select a subset of informative features from the data or (ii) generate new lower dimensional representations of the data.

Autoencoders are commonly used neural networks-based dimensionality reduction methods with the microarray data [Doersch *et al.*, 2015; Dosovitskiy *et al.*, 2014]. They are commonly used by replacing the output layer with a new layer which have a number of neurons equivalent to the number of classes or, alternatively, by using the generated representations with other classification algorithms [Yousefi *et al.*, 2016]

The imbalanced distribution of the samples between different classes (i.e. normal and cancer classes) is another challenge affecting the accuracy of the data analysis and statistical tools. Synthetic Minority Oversampling technique (SMOTE) and its new variations are popular methods used for solving the class imbalance problem.

Cancer prediction models might also include a neural network method to directly learn the class type of the pre-processed or the original samples. However,

our survey [Daoud & Mayo, 2019] of the most recent neural network cancer models presented in Chapter 4 revealed that the shallow networks, which consist of only one hidden layer, is the most commonly used architecture in recent works [Chen *et al.*, 2015b; Kumar & Ramakrishnan, 2014]. In addition, most classification models (e.g. [Chen *et al.*, 2014; Mandal & Banerjee, 2015]) used a few number of genes as inputs to the network models. The genes were chosen based on the recommendations found in previous works or well-known online repositories such as the Online Mendelian Inheritance in Man (OMIM) [OMIM repository]. The survey paper also presents feature selection and creation methods that have been recently used with the gene expression data.

2.4 Minority Class Over Sampling Methods

Real world datasets are often predominately imbalanced with majority to minority class ration is high. This imbalance distribution challenges the classification algorithms which become more predictable to the majority class examples than to the minority class examples; as they do not find enough examples of the minority class to learn from especially when the size of the dataset is too small.

The problem of imbalance classification is encountered in many application domains and started getting more emphasise by the research community in the last a few years. In this section, we will give an overview to the well known used minority class oversampling techniques used in the medical field with gene expression data.

SMOTE: is a synthetic oversampling technique suggested by Chawla *et al.* [2002] in 2002. It generates a new minority class example in the line segment between a selected example and its K nearest neighbor minority class examples.

$$Synth = s + random(0,1) \times (s - nn) \tag{2.9}$$

where s is a selected minority example and nn is its nearest k neighbor.

BorderlineSMOTE: was developed in 2005 by Han *et al.* [2005] as a new variant of SMOTE. The algorithm tries to learn the border line of every class as exactly as possible. The algorithm only considers the border line samples for over sampling in order to strengthen this area. The minority class examples, in BorderlineSMOTE, are categorized into one of the three following sets:

- *noise*: consists of the examples which K nearest neighbors are all belonging to the majority class.
- *danger*: all the examples which have the majority (more than half) of their K nearest neighbors are belonging to the majority class.
- $safe$: all examples which majority of the K nearest neighbors are belonging to the minority class.

Only those minority examples defined as *danger* examples will be considered for oversampling. Hence, for every $s \in danger$, find the K nearest minority examples, select a random example from the nearest set, call it nn. Finally, generate a $Synth$ using the formula given in Equation 2.9

Density Based SMOTE (DBSMOTE): In 2012, [Bunkhumpornpat *et al.*, 2012] introduced the DBSMOTE method which is inspired by the BorderlineSMOTE mentioned above and relies on the DBSCAN [Ester *et al.*, 1996] clustering algorithm introduced by . DBSMOTE does not only sample from the overlap region like the BorderlineSMOTE but it also consider sampling from the $safe$ set to improve the minority detection rate. To generate a synthetic example $Synth$ based on s minority example, the algorithm works according to the following:

- For every cluster C_i of the data found by DBSCAN, construct a directly density-reachable graph G connecting border line instances with the core instances where each edge of this graph is connected to a core instance and two border line instances are not directely reachable.
- determine the pseudo centroid c as the nearest instance to the mean of the center C_i.
- find the shortest path from s to c, call it $path$
- select a random edge point e located on $path$. and find two data points v_1 and v_2 connected to e.
- generate $Synth$ based on the following equation:

$$Synth = v_1 + random(0, 1)(v_1 - v_2) \tag{2.10}$$

So far, we have presented an overview to the methods and techniques that are either used in the introduced solutions or as base line methods for comparison purpose. More details will be found about these methods in the following chapters which will also explicitly define the problems and present experimental results.

Chapter 3

Using Swarm Optimization to Enhance the Autoencoder's Generated Images

Autoencoders are trained by encoding the input examples into low dimensional representations and decoding the representations back into their original dimension. This process of encoding and decoding is done for a set of forward and backward passes which respectively compute the neurons activations and back-propagate the error derivatives. Nevertheless, learning good representations of the data that are able to efficiently encode and decode it back is not an easy task and requires a careful selection to vast amount of networks parameters.

A random or optimized selection of the network's parameters does not guarantee learning discriminating representations of the data or solving the problem of generating different variations of an input [Vincent *et al.*, 2010b]. This chapter, however, introduces a method that uses the CSO algorithm's particles to trigger the learned weights of a Restricted Boltzmann Machine (RBM)-based autoencoder to generate different variations of a given example. The intention is

to generate new versions of a given input with better qualitative characteristics compared to the output generated by a trained encoder's representation. This optimization-based method activates the decoder part of the network using a swarm of codes to generate a number of outputs equals to the number of particles in the swarm. Each of these versions is compared to a given input example to update the position and the velocity of the particles according to the best solution found in the current iteration and the best solution found so far. The updated particles are used as inputs to the decoder in the second iteration of the algorithm. In the same way, the process is repeated for a set of iterations or until finding an output which quality matches a certain quality measure.

The method introduced in this chapter generates better versions of a given input and show successful performance in cases where the learned autoencoder is not effectively able to reconstruct the input examples. We will show later in this thesis that this method is applicable for solving real world application problems such as the class imbalance problem and prove that the generated optimized codes are discriminating and leading to increase the classification accuracy of different classifiers compared to other methods.

Using Swarm Optimization to Enhance Tthe Autoencoder's Generated Images

Maisa Daoud *mtdd1@students.waikato.ac.nz*
Michael Mayo *michael.mayo@waikato.ac.nz*
Department of Computer Science
University of Waikato, Hamilton. NZ

Abstract

We investigate how to improve the accuracy of a previously trained autoencoder results by answering the following question, Given a pre-trained autoencoder, a test image, and using a real parameter optimizer, can we generate better quality reconstructed image than the one generated by the autoencoder? Autoencoders learn data representations through reconstruction. Robust training is the key factor affecting the autoencoders learning and, consequently, the accuracy of their results. Previous works suggested methods for deciding the optimal autoencoder configuration which allows for proper training. Nevertheless, improving the performance of previously trained autoencoder has got limited, if no, attention. The proposed method was applied by combining the decoder part of a pre-trained Restricted Boltzman Machine (RBM) autoencoder with the Competitive Swarm Optimization algorithm. Experiments show that it is possible to reconstruct images using pre-trained decoder from randomly initialized representations. Results also show that our approach reconstructed better quality images than the autoencoder in most of the test cases, indicating that the proposed approach could be used for improving the performance of an RBM-based pre-trained autoencoder if it does not give satisfactory results.

keywords: representations, optimization, autoencoders, reconstruction

3.1 Introduction

Autoencoders are powerful Neural Network (NN) models that learn representations of the data with multiple levels of abstraction. Based on the application and the objective measure, the abstract representations are described as "good" if they are useful for addressing tasks of interest [Vincent *et al.*, 2010a]. Examples of these tasks are speech recognition, visual object recognition, image reconstruction and classification.

Learning good discriminative representations using NN-based approaches is very challenging and requires robust training. Factors affecting the training can be related to the network's configuration, the number of training examples [Bengio *et al.*, 2007a], the lack of proper data preprocessing (engineering) [Bengio *et al.*, 2013]. However, deciding the optimal network configuration is still determined by trail-and-error techniques i.e. by searching through a vast space of possible hypermeter combinations.

Our experiments show that it is possible to reconstruct interesting images using a pre-trained decoder and the CSO from randomly initialized representations. Moreover, our evaluations proved the ability of the proposed approach in generating good quality reconstructed versions and allowed the decoder to detect some fine details.

The rest of the paper is organized as follows. In the next section, we cover background about the image representations and briefly present some related works. In the section after that, we describe our proposed new approach. Next, we outline our evaluations. The last two sections conclude our work and present the limitations.

3.2 Technical Background

3.2.1 Images Representations and Reconstruction

Images can be generally represented using manually extracted low level representations such as color and shape properties, and high level representations such as NN representations.

Low level images representations can be extracted using different techniques such as Bag of Words (BoW) [Bosch *et al.*, 2006], Fisher Kernel (FK) [Perronnin & Christopher, 2007] and Vector Of Locally Aggregated Descriptors (VLAD) [Hervé *et al.*, 2006]. High level NN representation are automatically generated using the features learned during the training process and can be extracted, after training, by activating a certain layer or a set of layers and concatenating [Babenko *et al.*, 2014; Sharif Razavian *et al.*, 2014] or pooling them [Babenko & Lempitsky, 2015] which is a common mechanism in Convolutional Neural Network (CNN)-based approaches. Such representations are useful for image classification [Leung & Jitendra, 2001],visual object recognition [Li *et al.*, 2014], retrieval [Philbin *et al.*, 2007] and reconstruction [Hinton & Ruslan, 2006].

Images representation and reconstruction are highly related in the NN world. Dosovitskiy & Brox [2016], for examples, used a deconvolutional-based approach to reconstruct images from representations learned by a pre-trained deep Convolutional Neural Network (CNN) autoencoders and proved their efficiency in learning deep images representations. Other autoencoder-based techniques such as RBM-based autoencoders [Hinton & Ruslan, 2006], discussed in the following subsection, and Stacked Denosing Autoencoders [Vincent *et al.*, 2010a] were also introduced to extract robust low dimensional images representations through reconstruction and to reconstruct test images from low dimensional representations.

Generating good image representations which give satisfactory results is still

a challenge as it is highly dependent on the robustness of the training process [Ciancio *et al.*, 2016]. Neural networks are blackbox predictive tools and designing them require extensive knowledge engineering and careful parameter and configuration decisions [Walczak & Cerpa, 1999]. Walczak & Cerpa [1999] suggested a set of heuristic principles for designing networks with optimal output performance. Most recently, Ciancio *et al.* [2016] suggested four different heuristic approaches to increase the generalization abilities of a neural network. These methods are based, respectively, on the use of genetic algorithms, Taguchi, tablu search and decision trees. The parameters taken into account are the training algorithm, the number of hidden layers, the number of neurons and the activation function of each hidden layer.

3.2.2 RBM-based Autoencoder

RBM-based autoencoders were first introduced by Hinton & Ruslan [2006] as a non linear generalization of Principal Component Analysis (PCA). The network consists of an "encoder" part which transforms the high dimensional input data into a low-dimensional representation (the code), and a "decoder" part which recovers the data (image) from the code.

The autoencoder consists of two layer RBM network which has stochastic visible and hidden binary units arranged in sublayers using symmetrically weighted connections (weights), Figure 1 depicts an illustration of this. The joint configuration of the visible and hidden units has an energy given by

$$E(v,h) = -\sum_{i \in pixels} b_i v_i - \sum_{j \in features} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{3.1}$$

where v_i and h_j are the binary states of (visible) pixel i and (hidden) feature j respectively; b_i and b_j are their biases; and w_{ij} is the weight between them.

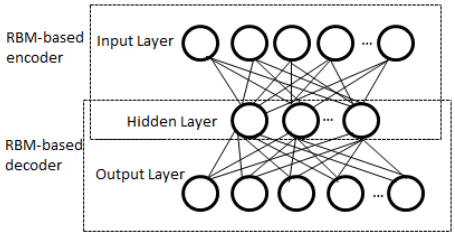


Figure 3.1: RBM autoencoder

Starting with random weights, the state of hidden node h_j is set to be 1 with a probability defined as :

$$p(h_j = 1|v) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{3.2}$$

where $\sigma(x)$ is the logistic function $1/[1 + exp(-x)]$.

Once the binary states have been chosen for the hidden nodes, the network sets the visible states by:

$$p(v_i = 1|h) = \sigma(b_i + \sum_j h_j w_{ij}) \tag{3.3}$$

where b_i is the bias of i.

The states of the hidden nodes are then updated once more so that they represent features of the confabulation. The network tries iteratively to minimize the discrepancy between the input and output data using an optimization algorithm such as gradient descent and propagating the error derivatives through the decoder and then through the encoder networks to fine-tune the weights for optimal reconstruction.

To use the RBM with real valued images, Ruslan *et al.* [2007] suggested pre-training the network to replace the binary states by stochastic activities. After pre-training, the model unfolds to produce encoder and decoder networks and fine

tunes the weights to replace the stochastic activities by deterministic, real valued probabilities. Backpropagation is also used through the whole autoencoder to fine tune the weights for optimal reconstruction.

3.2.3 Competitive Swarm Optimization (CSO)

CSO [Cheng & Jin, 2015] is fundamentally inspired by Particle Swarm Optimization (PSO) algorithm [Kennedy & Eberhart, 1995], which is a conceptually simple optimization algorithm that has attracted considerable research interests so far [Kennedy & Mendes, 2002; Suganthan, 1999]. PSO defines a swarm of n particles, each of which has a position and velocity flying in an m-dimensional search space. Each particle evaluates the objective function at its current position and iteratively updates its velocity and position according to the particle's best position, personal best, and the entire swarm global best position found so far. CSO was suggested to overcome the PSO's poor performance in solving high dimensional problems and problems with large number of local optima.

In CSO, the swarm's n particles are randomly allocated into $\frac{n}{2}$ couples and a competition occurs between the two particles in each couple. Only the ones with the better fitness, the winners, are passed to the next generation (iteration), indexed as $t + 1$, of the swarm. Each loser particle updates its position and velocity by learning from its winning competitor, and the updated loser is passed to generation $t + 1$ after that. Hence, the total number of comparisons occur per generation is $\frac{n}{2}$ and only the velocity and speed for $\frac{n}{2}$ particles are updated per generation. A modified pseudocode of the CSO (with modifications explained in the next section) is given by Algorithm 1.

3.3 Using the CSO For Enhancing the Autoencoders' Images

Our main goal is to introduce a novel approach that improves the accuracy of previously trained autoencoder results. The idea of the approach is to use the decoder part of a pre-trained autoencoder and a real parameter optimizer to reconstruct given test images. To apply the approach, we trained an RBM-based autoencoder and used its decoder with the CSO optimizer. The following subsections presents the methodology and Figure 2 depicts it.

3.3.1 Training the RBM-based Autoencoder

To train an RBM-based autoencoder according to [Hinton & Ruslan, 2006], the network has to be configured by setting the number of input and output layers' nodes equal to the number of pixels in the training images. Weights and biases are initialized and the network starts learning by feeding the training images through all of its layers to generate output images (reconstructed images) . The discrepancy between the input training image batches and their reconstructed versions are calculated and the error derivatives are propagated through the decoder and the encoder parts to fine tune the weights and biases. After a finite number of iterations, the autoencoder will learn the images' features, and its encoder part will be ready to generate low dimensional representations for test images that are fed through its input layer. These representations can be used as inputs to the decoder part to reconstruct the input test images or used for other applications.

3.3.2 Image Reconstruction Using Trained RBM-Decoder and CSO

The second step of the approach representing our main contribution in reconstructing an image using a pre-trained decoder with the CSO algorithm. We will refer to our method as decoder+CSO(m) to indicate the length m of the used representational vector (Algorithm 1).

Contrary to studies that discard the decoder part and use the network's upper part as a fast image dimensionality reduction method, our novel approach only uses the decoder part of the trained autoencoder and discards the encoder. To generate a reconstructed version I' for test image I (Equation 3.4), we feed a swarm $P(t)$ of n randomly initialized low dimensional vectors $(X_1, .., X_n)$ through the trained decoder's layers to get a set of output images I'_n at its output layer as depicted by Figure 3.2 step 2. Note that each particle X_n is an m-dimensional vector and m equals to the number of nodes in the decoder's first layer i.e. the autoencoder's bottleneck layer.

$$I' = Decoder(X) \tag{3.4}$$

We used the Euclidean norm (given by Equation 3.5) to calculate the difference between each output image $I'_{1..n}$ and the target test image I and to identify the fittest individuals.

$$F(X) = ||(I - I')||_2 \tag{3.5}$$

Where I' is the result of decoding representation X.

Particles are randomly allocated into $\frac{n}{2}$ couples and the position and velocity of the individuals losing each competition are updated according to their winning partner's existing velocity and the swarm's mean velocity as shown in Equations

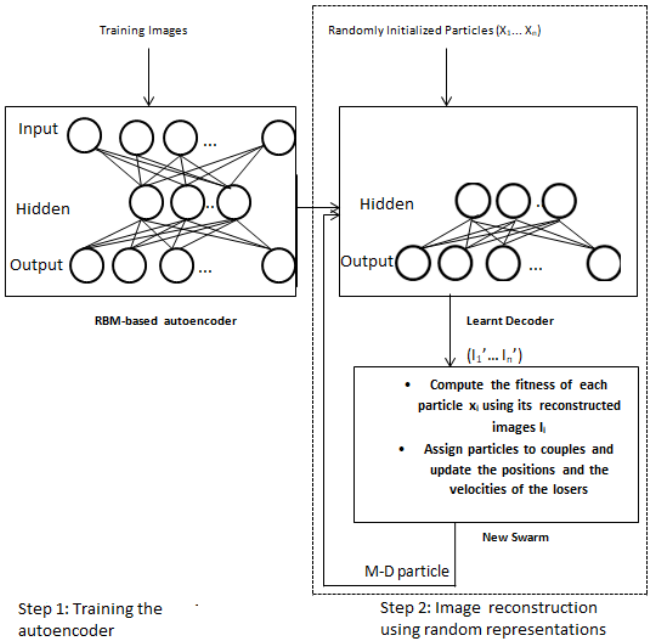


Figure 3.2: RBM-based autoencoder training and image reconstruction using pre-trained decoder+CSO (m). See Algorithm 1 for better understanding

3.7 & 3.7) and cited in [Cheng & Jin, 2015].

$$V_{l,k}(t+1) = R_1(k,t)V_{l,k}(t) + R_2(k,t)(X_{w,k}(t) - X_{l,k}(t)) + \varphi R_3(k,t)(\bar{X}_k(t) - X_{l,k}(t)) \tag{3.6}$$

$$X_{l,k}(t+1) = X_{l,k}(t) + V_{l,k}(t+1). \tag{3.7}$$

where $X_{w,k}$, $X_{l,k}$,$V_{w,k}$ and $V_{l,k}$ are the position and velocity of the winner and loser in of generation t respectively, $K \in [1, \frac{n}{2}]$ and $R_1(k,t)$, $R_2(k,t)$, $R_3(k,t) \in [0,1]^n$ are three randomly generated vectors after the k-th round of the competition and learning process in generation t, $\bar{X}_k(t)$ is the mean position value of the relevant

particles and φ is a parameter that controls the influence of $\bar{X}_k$ (t).

The process of feeding the particles through the decoder part, calculating the fitness function and updating each particle's positions and velocities are continued for a set of iterations to find the best reconstructed version I' for I.

Algorithm 1: Image Reconstruction using pre-trained decoder+CSO (m)

$t = 0$
$count = 0$
$I = targettestimage$;
$Decoder(X)$, the decoder part of a pre-trained autoencoder
$P(t) = X_1, X_2, .., X_n$ randomly initialized m-dimensional particles
while *terminal condition is not satisfied* **do**
  feed each element of $P(t)$ through the trained decoder to generate $I'_1, I'_2, .., I'_n$ using $X_1, X_2, .., X_n$ according to Equation 4
  calculate $F(X)$ between I and $I'_1, I'_2..I'_n$ using Equation 5
  $U = P(t), O(t+1) = \emptyset$
  while $U \neq \emptyset$ **do**
    randomly remove two particles $X_{rand1}(t)$, $X_{rand2}(t)$ from U
    if $F(X_{rand1}(t)) \leq F(X_{rand2}(t))$ **then**
      $X_w(t) = X_{rand1}(t), X_l(t) = X_{rand2}(t)$;
    else
      $X_w(t) = X_{rand2}(t), X_l(t) = X_{rand1}(t)$;
    add $X_w(t)$ into $P(t+1)$
    update $X_l(t)$ using (5) and (6)
    add the updated $X_l(t+1)$ to $P(t+1)$
  $t = t + 1$

Experimental Study

An open source deep learning library called DeepLearning4j (DL4j) (version 0.7.0) with the MNIST and the OlivettiFaces Roweis [2012] datasets were used to perform a set of experiments.

As can be observed by Table 1, the MNIST dataset consists of 60,000 (28×28)

training for all (0-9) digits. The OlivettiFaces dataset contains ten (64×64) images for each of forty different people. We constructed a training dataset by rotating (-90 to 270) and subsampling, to improve the training, images of 30 people to get 10800 (22×22) images (i.e. 30 people $\times$ 10 images per person $\times$ 36 rotations). The 500 OlivettiFaces test images were created in the same way using 10 images of different people. All training and testing images of both datasets were normalized using min-max normalizer to get values in [0-1] range.

Table 3.1: Description of the datasets and network architectures used.

Dataset	dimension	no. training examples	no. testing examples	model architecture
MNIST	28×28	60,000	500	784-30-784
	28×28	60,000	500	784-250-784
	28×28	60,000	500	7841000-500-256-30 -256-500-1000-784
OlivettiFaces	22×22	10,800 (30 people)	500 (14 people)	484-30-484
	22×22	10,800 (30 people)	500 (14 people)	484-300-484

To test the proposed approach, we compared the accuracy of the images reconstructed by the decoder+CSO(m) (optimized) representations with the ones reconstructed by the encoder's (non-optimized) representations. The trained encoder part of the RBM-based autoencoder was used to generate a low dimensional representation. This representation is fed through the decoder part to generate a reconstructed version of the input test image. Hereafter, we will denote this method by autoencoder(m) to indicate the length m of the used representational vector.

In all test cases, we set the number of input and output layer's nodes equals to the training images' size. All the network's nodes were initialized using DL4j's default XAVEIR initialization method where weights are drawn uniformly in the range $[\frac{-2}{\sqrt{In_l+Out_l}}, \frac{2}{\sqrt{In_l+Out_l}}]$ such that " In_l" and "Out_l" are the number of nodes

sending input to and receiving output from the layer (l) to be initialized (see Glorot & Bengio [2010a]). The network was activated by the sigmoid function, and back propagated using the Gradient Descent Optimization algorithm (GDS).

Every trained decoder was used to reconstruct 500 test images using the CSO's particles. The swarm size was set to 100 randomly initialized [0-1] individuals and the algorithm was run for 100 iterations, which was good enough to evolve interesting results. We also performed experiments using a larger swarm size (500) and more generations (200), but results were qualitatively very similar to the ones obtained using 100 particles and 100 iterations, so we are only presenting the results of this one.

3.3.3 Experimental Results and Evaluation

Five experiments, three trained on MNIST dataset and the other two on the OlivettiFaces dataset, were performed using different network models. The aim was to test the accuracy of the proposed approach in reconstructing images from different datasets and using different configurations.

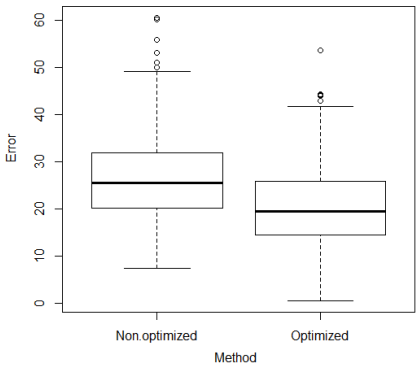


Figure 3.3: Comparing the performance of autoencoder(30) v.s. decoder+CSO(30) representations in reconstructing MNIST test images.

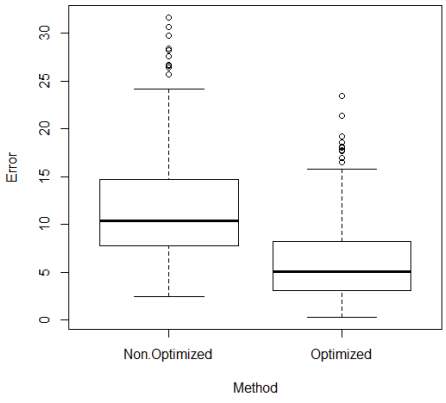


Figure 3.4: Comparing the performance of autoencoder(250) v.s. decoder+CSO(250) representations in reconstructing MNIST test images.

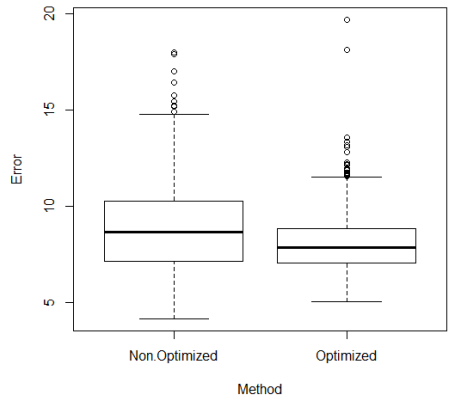


Figure 3.5: Comparing the performance of autoencoder(30) v.s. decoder+CSO(30) representations in reconstructing OlivittiFaces test images.

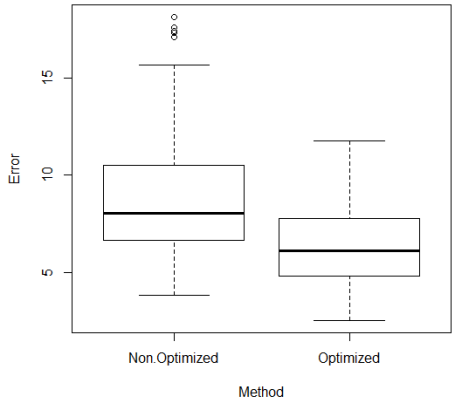


Figure 3.6: Comparing the performance of autoencoder(300) v.s. decoder+CSO(300) representations in reconstructing OlivittiFaces test images.

Figures 3.1-3.7 depict a set of box-and-whisker plots showing the distribution of the error between target original images and reconstructed versions resulting from pre-trained autoencoder(m) and decoder+CSO(m). Comparing the performance of the two methods indicates that the trained decoder+CSO(m) reconstructed better quality images with remarkably less reconstruction error than the trained encoder (m) in most test cases. The decoder+CSO(m) clearly achieved superior results with lowest overall and median reconstruction errors when using three-layered network models for both MNIST datasets (Figures 3.3-3.4) and OlivettiFaces dataset (Figure 3.5-3.6).

We were also interested in testing our approach using multi layer networks. For fast training, we chose a Multi Layer Percptron (MLP) network configured using nine layers with the following sizes 784-1000-500-256-30-256-500-1000-784. As shown by Figure 3.7, the optimization has only added slight little improvement to the reconstructed images compared to the accuracy of the multi layer autoencoder images.

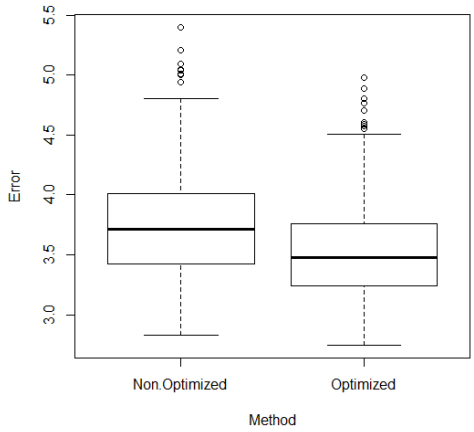


Figure 3.7: Comparing the performance of multi layer autoencoder(30) v.s. Multi layer decoder+CSO(30) representations in reconstructing MNIST test images.

Qualitative Comparison

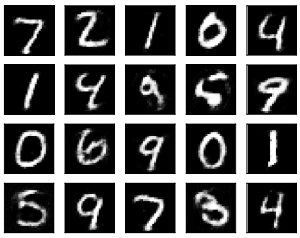


Figure 3.8: Reconstruction of MNIST using the decoder+CSO(250) representations.

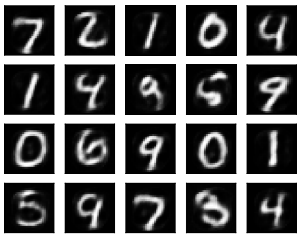


Figure 3.9: Reconstruction of MNIST using the autoencoder(250) representations.

Finally we performed qualitative comparisons between all related test cases. As shown by Figures 3.8-3.17, the decoder+CSO(m) method reconstructed better quality, less blurry and distorted, MNIST images compared to the autoencoder(m). However, best results obtained, from both methods, when the length of the representational vector was 250.

Experiments on the OlivettiFaces dataset show that the autoencoder(m) method, trained using our parameter settings, was able to learn the orientation of faces

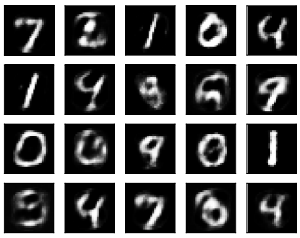


Figure 3.10: Reconstruction of MNIST using decoder+CSO(30) representations.

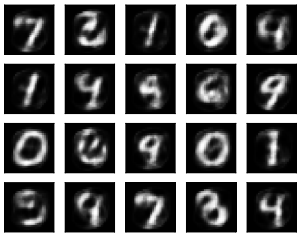


Figure 3.11: Reconstruction of MNIST using autoencoder(30) representations.

but not their fine details (see Figures 3.12-3.15). However, using the optimizer helped in revealing more of the face details.

Results generally indicated that using the CSO real-parameter optimizer enabled the pre-trained network to detect (filter) more of the test images features. Indicating that the general CSO randomly initialized population was able to activate some of the network's feature detectors better than the encoder's representations.

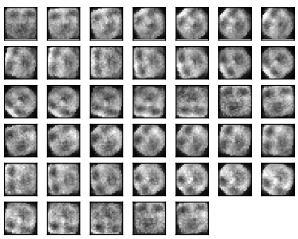


Figure 3.12: Reconstruction of OlivittiFaces using decoder+CSO(30) representations.

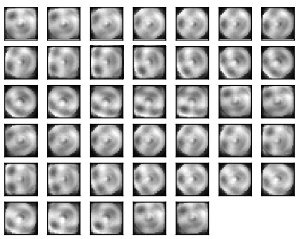


Figure 3.13: Reconstruction of OlivittiFaces using autoencoder(30) representations.

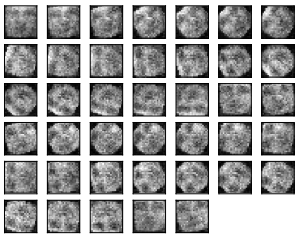


Figure 3.14: Reconstruction of OlivittiFaces using decoder+CSO(300) representations.

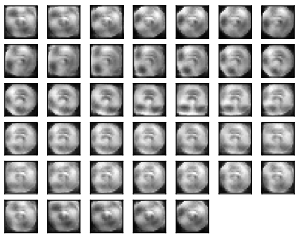


Figure 3.15: Reconstruction of OlivittiFaces using autoencoder(300) representations.

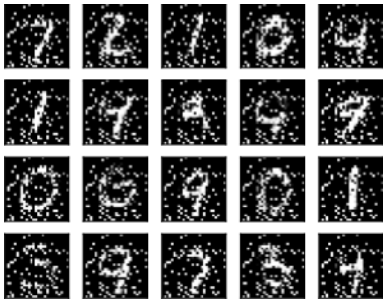


Figure 3.16: Reconstruction of MNIST using multi layer decoder+CSO(30) representations.

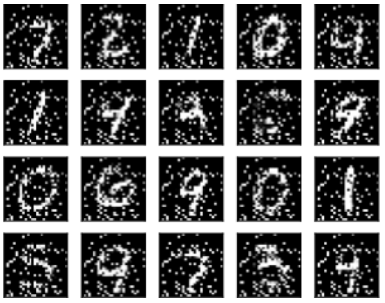


Figure 3.17: Reconstruction of MNIST using multi layere autoencoder(30) representations.

3.4 Limitations

Our approach is proposed to improve the accuracy of pre-trained autoencoders results. Reasons affecting the networks performance can be related to the networks' learning parameters settings and configuration which are determined by of trials-and-errors. Using the approach with robustly trained or un-trained networks will add no improvement to the results. The approach is helpful when the network's neurons are able to detect most of the images features but more accuracy is required. In this case, the optimizer, with its random population, could extract more information from the pre-trained network and increase its ability in detecting fine details and improving the images.

3.5 Chapter Summary

This chapter has described a novel approach for improving the performance of previously trained autoencoders. The approach was applied by combining a pre-trained RBM-based decoder with the CSO algorithm and was used to reconstruct a set of test images. Experiments have shown that the suggested approach is able to reconstruct images using randomly initialized representations. The optimization helped in producing sharper images, detecting finer details and improving the accuracy of pre-trained autoencoder results.

The approach can be applied using other types of autoencoders and/ or real-parameter optimizers especially when no satisfactory results obtained from the trained network. Chapter 5 will present a new application where optimized representations can be used to solve real world biomedical applications problems.

Chapter 4

A Survey of Neural Network-Based Cancer Prediction Models from Microarray Data

This chapter presents a published journal paper that surveys the works published between 2013-2018 in neural network-based cancer prediction models from microarray gene expression data. Based on the neural network's objective of training, the approaches were categorized into: 1. Filtering approaches. 2. Prediction approaches and 3. Clustering approaches.

Filtering approaches are trained to learn the representations of the gene expressions using unsupervised algorithms by optimizing an objective function that is not directly related to the prediction goal. The learned representations are used with other machine learning tools for analysing, visualizing or predicting the class of the gene expressions. Predicting and clustering methods, on the other hand, are trained to extract the representations that directly maximize

the prediction accuracy or best group of the genes or samples according to their mutual similarities into sets.

The paper presented the state-of-the-art studies which included one or more neural network models as main method/s for predicting cancer from microarray datasets in particular. We described the pre-processing tools and the networks architectures including the number of layers, neurons per layer, objective function and regularization methods. This paper highlights some practical issues that have to be considered when building cancer prediction models such as the class imbalance problem and provides a detailed discussion to some best practices.

A Survey of Neural Network-based Cancer Prediction Models from Microarray Data

Maisa Daoud *mtdd1@students.waikato.ac.nz*

Michael Mayo *michael.mayo@waikato.ac.nz*

Department of Computer Science

University of Waikato, Hamilton. NZ

Abstract

Neural networks are powerful tools used widely for building cancer prediction models from microarray data. We review the most recently proposed models to highlight the roles of neural networks in predicting cancer from gene expression data. We identified articles published between 2013-2018 in scientific databases using keywords such as cancer classification, cancer analysis, cancer prediction, cancer clustering and microarray data. Analyzing the studies reveals that neural network methods have been either used for filtering (data engineering) the gene expressions in a prior step to prediction; predicting the existence of cancer, cancer type or the survivability risk; or for clustering unlabeled samples. This paper also discusses some practical issues that can be considered when building a neural network-based cancer prediction model. Results indicate that the functionality of the neural network determines its general architecture. However, the decision on the number of hidden layers, neurons, hypermeters and learning algorithm is made using trail-and-error techniques

4.1 Introduction

Microarray technology is one of the most widely used tools for analyzing genetic diseases. Standardized microarray dataset consists of thousands of gene

expressions and a few hundred of samples. Each expression measures the level of activity of genes within a given tissue. Hence, comparing the genes expressed in abnormal cancerous tissues with those in normal tissues gives a good insight into the disease pathology and allows for better diagnosis and predictions for future samples.

The high dimensionality of the gene expression profiles is a crucial issue when building a cancer predictive model. This problem affects the accuracy of the model and increases the computational time [Chandrashekar & Sahin, 2014; El-Bakry, 2010]. Two general approaches have been suggested to reduce the dimensionality of the gene expressions and to overcome its consequent problems. These are: (i) feature selection methods, which select the most relevant discriminating features and eliminate the non relevant dependent features; and (ii) feature creation methods, which generate new low dimensional features (codes) that best represent the original high dimensional features (as indicated in [Xue *et al.*, 2016]).

Neural networks are powerful machine learning methods that are used to learn data representations at multiple levels of abstractions. These representations are useful for many applications such as reconstruction, classification, clustering and recognition. Predictive models such as cancer prediction models use the generated features for classifying, clustering or applying statistical analysis on the samples.

Based on our analysis of the most recent studies on cancer prediction models, we categorize the current neural network methods according to their functionality into: (i) filtering (preprocessing) methods, (ii) predicting (classification) methods, and (iii) clustering methods. Neural network filtering methods are used for extracting representations that best describe the gene expressions without any direct consideration to the prediction goal, such as the networks used in [Fakoor *et al.*, 2013; Tan *et al.*, 2015]. Alternatively, predicting and clustering methods extract the representations that, respectively, maximize the prediction accuracy

[Chaudhary *et al.*, 2017; Danaee *et al.*, 2016; Macías-García *et al.*, 2017] or best divide the genes or samples according to their mutual similarities into groups [Borkowska *et al.*, 2014; Yu *et al.*, 2017, 2015].

In this study, we provide a review to the most recent neural network-based cancer prediction models by presenting the data pre-processing tools and the adopted architectures. We also provide a brief discussion that highlights some important issues that can be considered when building new cancer prediction models. This work is distinguished from others by presenting neural networks models that were specifically designed for predicting cancer using gene expression data. Previous works such as [Leung *et al.*, 2016] focused on applications of deep learning in different bioinformatic related fields. Deep learning applications in regulatory genomics and biological image analysis was introduced in [Angermueller *et al.*, 2016] and gave practical points to start with deep architectures. Ravi et al Ravı *et al.* [2017] also highlighted computational biology problems in a way that is accessible to machine learning researchers.

The rest of the paper is organized as follows. In Section 2. we present a basic background to neural networks and cancer prediction models. Section 3. presents our methodology including a review of the most commonly used neural network models for preprocessing, filtering, prediction and clustering gene expressions. Section 4. summarizes the reviewed studies. The discussion is presented in Section 6. and the conclusion in Section 7.

4.2 Background

This section presents a basic introduction to neural networks and cancer prediction models.

4.2.1 Neural Networks

Neural networks are powerful tools capable of solving non linear complex problems and discovering universal input-output mappings [Lam *et al.*, 2014]. To get a better understanding of the concept, consider a fully connected feedforward Multi-Layer Perceptron (MLP) network with L layers ordered as: input layer, a sequence of hidden layers, and an output layer. The layers are indexed as $l = \{0, ..., L-1\}$ and each layer has a number of neurons equal to n_l. We will denote each input training example x as $I(x) = [I_1, I_2, ..., I_{n_0}]$ and its output $O(x) = [O_1, O_2, ..., O_{n_{L-1}}]$.

The network is trained by feeding the inputs forward to calculate the activation value for every neuron. At the output layer, neuron activation values are calculated and aggregated to get $O(x)$.

The difference between $O(x)$ and a desired output, i.e. the error, is calculated using a predefined objective function. Using a backpropagation algorithm, the objective function is optimized by propagating the error derivatives back through the network to fine tune the weights for optimal error value. The discussed feed forward layered networks and backpropagation mechanism are one of the most commonly used architectures and training algorithms. We focus on them as they are widely used with gene expression data. Prieto *et al.* [2016] presented a comprehensive overview of modelling, simulation and implementation of neural networks with some examples to models used for solving real world problems.

In the next sections, we review previous works proposing neural network-based cancer prediction models. These models adopt the MLP, convolutional neural network or generative adversarial network architectures for learning the gene expression features. All methods use a similar training algorithm to the one described above with some differences in the number of neurons and networks architectures. Convolutional neural networks [Lawrence *et al.*, 1997], for example,

perform feature extraction by scanning a set of weight matrices across the input; these weight matrices learn to recognize the relevant patterns. Another example is generative adversarial networks [Radford *et al.*, 2015] which consist of a generative network and a discriminator network. The generative network learns to generate output samples, given random noise as input, while the discriminator network learns to discriminate the true data samples from the generated fake data samples.

4.2.2 Cancer Prediction Models

Cancer is a serious worldwide health problem usually associated with genetic abnormalities. These abnormalities can be detected using microarray techniques which measure the expression and the activity of thousands of genes. Generating a microarray involves hybridizing two DNA strands collected from two samples, e.g. diseased and healthy tissues. Each of these samples is originally a reverse transcript of mRNA and labeled with a dye. The two samples are mixed into a single microarray and scanned with an appropriate source of light to provide an image with an array of features. The intensity of each spot or the average difference between matches and mismatches can be related to the amount (expression) of the mRNA presents in the tissue, i.e the amount of protein produced by the gene corresponding to the given feature. The ultimate goal of this genomic analysis technology is to get better insight into the disease and to improve cancer diagnosis [Bashiri *et al.*, 2017].

Cancer prediction models consist of one or more methods working collaboratively to achieve high prediction accuracy. Statistical and machine learning methods have been widely used for building cancer prediction models that help physicians to provide more accurate prognosis, individualized treatments, and reduce the cost per patient.

The accuracy of cancer prediction models is affected by the characteristics

of the input data. Gene expressions are high dimensional and include irrelevant noisy features which degrades classification accuracy. They also exhibit spatial structure and, hence, incorporating information about this may increase the model's discriminating ability Tang *et al.* [2014].

Table 4.1: Classification metrics, TP is true positive, TN is true negative, FP is false positive and FN is false negative.

Metric Name	Definition
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
Sensitivity, Recall	$\frac{TP}{TP+FN}$
Specificity	$\frac{TN}{TN+FP}$
Precision	$\frac{TP}{TP+FP}$
Negative Prediction Rate	$\frac{TN}{TN+FN}$
Matthews Correlation Coefficient (MCC)	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
F1	$\frac{2 \times precision \times recall}{(precision+recall)}$

The predictive performance of cancer prediction models can be measured by different metrics such as the accuracy, recall, specificity, precision, negative prediction rate, Matthew correlation coefficient and F1 (shown in Table 1). Receiver Operating Characteristic (ROC) curve [Fawcett, 2006] is another measure represented as a plot of the true positive (sensitivity) rate against the false positive rate (specificity) [Ciatti *et al.*, 2015; Huang *et al.*, 2017]. It reflects the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. A good prediction model is the one achieving good balance between sensitivity and specificity [Ciatti *et al.*, 2015]. Statistically, this corresponds to ROC ≥ 0.7 [Ciatti *et al.*, 2015]. However, the ROC value should be paired with the confidence interval value to test its validity.

Survival analysis is a different field concerned with predicting the time until a medical condition occurs. From a machine learning perspective, survival analysis is a ranking problem in which data points are ranked on their survival times rather

than predicting the actual survival times [Steck *et al.*, 2008]. The Concordance Index (CI) is one of the standard performance measuring tools for assessing the quality of ranking in survival analysis studies. CI can be interpreted as the probability of concordance between the predicted and the observed survival where a value close to 0.7 indicates a good model and a value close to 0.5 means random concordance. Brier score is another metric measuring the mean of the difference between observed and estimated survival over a certain time. Brier score ranges between 0 and 1 and a larger score indicates higher inaccuracy [Chaudhary *et al.*, 2017].

4.3 Neural Network-based Cancer Prediction Models

An extensive search relevant to neural network-based cancer prediction was conducted using Google scholar and two other electronic databases namely PubMed and Scopus. Search was performed using keywords such as "Neural Networks" AND "Cancer classification", "Neural Networks" AND "Gene expressions", "Neural Networks" AND "Microarray", "Neural Networks AND Cancer Prediction" and "Neural Networks AND gene expression clustering". Only articles published between 2013-2018, publicly available for free, and including one or more neural network models in their approach were considered. The chosen papers covered cancer classification, discovery, survivability prediction, and statistical analysis models. Papers using imaging or text (record) inputs were excluded.

Fig. 4.1 shows a graph representing the number of citations for each of the considered papers. The papers are grouped according to their functionality and ordered chronologically by the year of publication. Considered papers have reasonable number of citations ranges between 5-120, however, papers published by

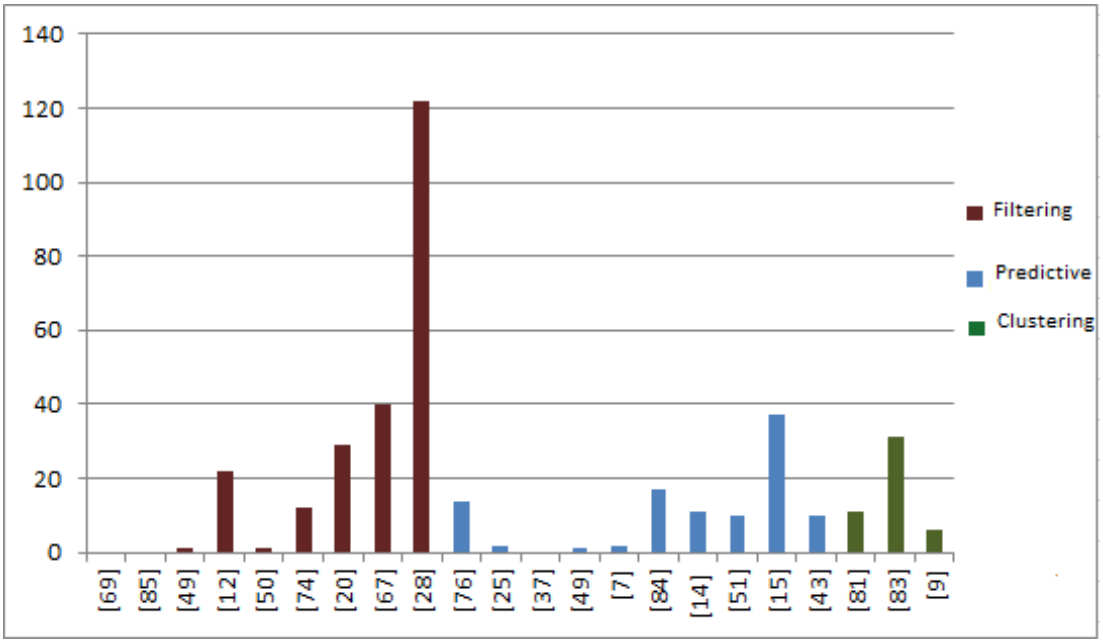


Figure 4.1: Number of citation for each of the considered datasets

2018 are the least cited ones, some of them have zero citation, but considered to present some state-of-the-art approaches in the field.

In the next subsections, we review the chosen papers by presenting the adopted preprocessing techniques and the proposed models configuration.

4.3.1 Datasets and Preprocessing

Most studies investigating automatic cancer prediction and clustering used publicly available datasets such as the TCGA[Tomczak *et al.*, 2015], UCI [Dheeru & Karra Taniskidou, 2017], NCBI Gene Expression Omnibus (GEO) [NCBI repository] and Kentridge biomedical [Kentridge, 2017] databases. These repositories were used by [Danaee *et al.*, 2016; Hou *et al.*, 2018; Mandal & Banerjee, 2015; Titus *et al.*, 2018], [Bhat *et al.*, 2017; Mandal & Banerjee, 2015; Zhang *et al.*, 2018], [Chen *et al.*, 2014, 2015b; Hou *et al.*, 2018; Zhang *et al.*, 2018] and [Kumar & Ramakrishnan, 2014] respectively. In rare cases such as [Borkowska *et al.*,

2014; Macías-García *et al.*, 2017], studies are conducted using a specific dataset that was collected and prepared for a problem under investigation.

Removing the genes that have zero expression value across all samples is one of the simple and most straight forward preprocessing technique and is used by [Chaudhary *et al.*, 2017; Danaee *et al.*, 2016; Yuan *et al.*, 2016]. Chaudhary *et al.* [2017] followed this step with removing the samples that have 20% of the features removed.

Normalization is also an essential technique in some cases. MAS5.0 affymatrix normalization was used by [Chen *et al.*, 2014, 2015b; Titus *et al.*, 2018]. Mapping to Entrez Gene ID and averaging were used by [Zhang *et al.*, 2018]. Fragments per kilobase per million (FPKM) normalization was used by [Xiao *et al.*, 2018]. Funnorm normalization to remove the unwanted technical variation in methylation arrays and filtering with $pvalue > 10^{-05}$ was used in [Titus *et al.*, 2018]. Datasets should be also normalized for training purposes using methods such as the zero mean one unit variance normalization which was used in [Chaudhary *et al.*, 2017; Kumar & Kumar, 2018; Way & Greene, 2017]. Other kinds of transformations such as log-fold change transformation and log_2 transformations can be also used such as in [Macías-García *et al.*, 2017] and [Chaudhary *et al.*, 2017] respectively.

Different techniques were adopted to reduce the dimensionality of the gene expressions by selecting a subset of genes. Statistical methods were applied by [Danaee *et al.*, 2016; Way & Greene, 2017]. Choosing a subset of related genes as indicated by other studies or data repositories such as the Online Mendelian Inheritance in Man (OMIM) [Hamosh *et al.*, 2005] is another simple feature selection technique adopted by [Chen *et al.*, 2014, 2015b; Mandal & Banerjee, 2015]. Chen *et al.* [2014, 2015b] applied, in both of their studies, the same preprocessing methods including Chi-square feature selection, selecting the top-10 ranked lung

cancer-related gene signatures and combining them with T-stage and N-stage clinical data. Instead of selecting a subset of genes, Tan *et al.* [2015] set the expression of a randomly chosen number of genes to zeros. Xiao *et al.* [2018] applied the DESeq method to select a set of differentially expressed genes, most informative genes, based on their read count.

Even though neural networks are used for extracting the datasets features by reducing the dimensionality of the data, Principle Component Analysis (PCA) can be used as an initial preprocessing step [Fakoor *et al.*, 2013; Zhang *et al.*, 2018]. The PCA method linearly transforms the dataset features into a lower dimensional space without capturing the complex relationships between the features. Therefore, a good technique can be adopted by merging the PCA components with a random number of raw features to allow the networks to capture further useful relationships [Fakoor *et al.*, 2013; Zhang *et al.*, 2018].

In cases where researchers are interested in studying datasets from different sources such as [Tan *et al.*, 2015; Zhang *et al.*, 2018]. Dataset features can be forced to have the same dimensionality for training and testing purposes, [Zhang *et al.*, 2018] padded the features with zero values, but this technique might increase the sparsity of the data. However, [Tan *et al.*, 2015] overcame the problem by simply removing the genes that were not measured by the other datasets.

The class imbalance problem has only been considered once by Danaee *et al.* [2016] who used Synthetic Minority Class Over Sampling (SMOTE) method to generate synthetic minority class samples. Liu *et al.* [2017] had considered the problem of small sized dataset and proposed an oversampling technique that simply duplicates 20% of the data and used it for training. The duplication rate was proportional to the dimensionality of the data and, to force variability in the generated samples, they set a number of randomly chosen features into zeros. However, this technique can lead to sparse matrix problem.

Clustering was also applied in some studies for labeling the data by grouping the samples into high-risk, low-risk groups, or more [Chen *et al.*, 2015b]. In [Yuan *et al.*, 2016] a clustered gene filtering technique was used based on the mutation occurrence frequency of the gene data and to reduce the sparsity and the dimensionality of the data they proposed indexed sparsity reduction (ISR) procedure (see [Yuan *et al.*, 2016] for detailed description of both methods).

Table 4.2 presents the dataset used by each of the considered references, the applied normalization technique, the cancer type and the dimensionality of the datasets.

4.3.2 Building Neural Networks-based Approaches for Gene Expression Prediction

This section surveys a group of recently published studies to highlight the basic role that neural networks perform in cancer prediction. Our classification of the studies was based on investigating the learning algorithm, the objective function and its relationship with the prediction goal.

Based on our investigation of the related studies, we categorized the networks into filtering methods, prediction methods and clustering methods. Filtering methods learn how to generate representative codes with dimensionality $M \leq N$ (where N is the dimensionality of the input) that can be used with other machine learning algorithms such as naïve Bayes and k-means for prediction or clustering purposes. These methods can be proceeded with preprocessing methods, but used to apply further processing on the data and learn in a purely unsupervised way with no direct relationship with prediction (Figure 4.2). Predictive neural networks are trained using a supervised learning algorithm to maximize classification accuracy. Clustering methods, on the other hand, are trained using an unsupervised algorithm to set similar samples or genes in groups (Figure 4.3).

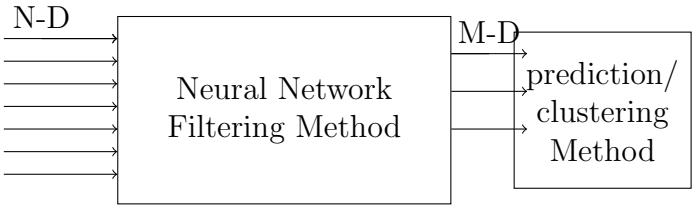


Figure 4.2: Neural networks for filtering the gene expressions in cancer prediction models.

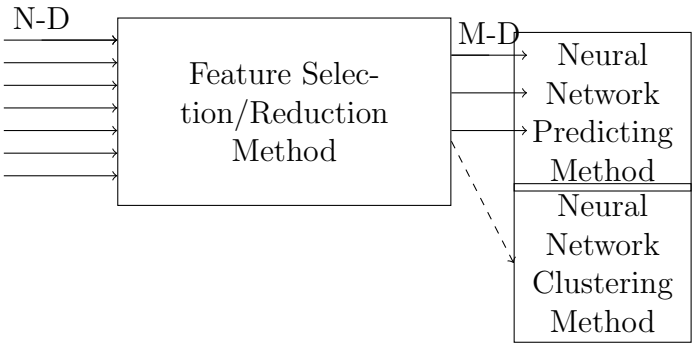


Figure 4.3: Neural networks for predicting or clustering the gene expressions in cancer prediction models.

Prediction and clustering approaches are usually proceeded by dimensionality reduction methods but the goal of training them is to increase the network's capability to classify or to find the most similar group to a new testing instance with unknown label respectively.

4.3.2.1 Neural Network Filters for Cancer Prediction

There is a growing interest in using autoencoders to extract generic genomic features in a preprocessing step to classification, clustering and statistical analysis [Doersch *et al.*, 2015; Dosovitskiy *et al.*, 2014; Girshick *et al.*, 2014; Wang & Gupta, 2015]. The autoencoder, in its simplest form, consists of three layers: input layer, hidden layer and output layer, divided into two parts: the encoder

part to learn the mapping between high dimensional unlabeled input data $I(x)$ and low dimensional representations (in the bottleneck layer), and the decoder part which learns the mapping from the middle layer's representation to the high-dimensional reconstructed output $O(x)$. More complicated deep architectures are built by adding more hidden layers to both halves of the architecture.

Autoencoder-based approaches learn how to reconstruct the input examples by optimizing an objective function such as the Root Mean Squared Error (RMSE) (shown in Equation 1):

$$RMSE = \sqrt{\frac{\sum(I(x) - O(x))^2}{n}} \tag{4.1}$$

or the Logloss function (Equation 2):

$$Logloss = \sum(I(x)log(O(x)) + (1 - I(x))(log(1 - O(x)))) \tag{4.2}$$

where the number of neurons at the input layer n_0 equals to the number of neurons at the output layer n_{L-1} and L is the total number of layers.

Different types of autoencoders such as stacked denoising autoencoders used by [Danaee *et al.*, 2016; Tan *et al.*, 2015], contractive autoencoders [Macías-García *et al.*, 2017], sparse autoencoders [Fakoor *et al.*, 2013], regularized autoencoders [Chaudhary *et al.*, 2017] and variational autoencoders [Titus *et al.*, 2018; Way & Greene, 2017] have been recently used for filtering microarray gene expressions (see Table 3). The architecture of the networks varied in depth (shallow and deep architecture), loss function and other parameters. A single hidden layer architecture and sigmoid activation function was used in [Fakoor *et al.*, 2013; Tan *et al.*, 2015]. [Tan *et al.*, 2015] used a cross-entropy similarity function and stochastic gradient descent optimization. In 2016, [Danaee *et al.*, 2016] used a deep architecture of four layers (15,000, 10,000, 2,000, and 500 neurons respectively) with

stochastic gradient descent optimization algorithm. Both Way & Greene [2017] and Titus *et al.* [2018] used the same variational autoencoder consisting of three hidden layers; two of the layers were in the same hierarchal level such that they receive their inputs from the input layer and send their activations forward to the third hidden layer in the higher level of the network. Titus *et al.* [2018] used the same implementation used by [Way & Greene, 2017] except that they adapted the model to take 300,000 inputs instead of 5000. Both studies set the batch size to 50, learning rate to 0.0005 and epochs to 50. A simple network of two stacked hidden layers (500, 100 neurons) autoencoder was used in [Chaudhary *et al.*, 2017] and trained for 10 epochs. Zhang *et al.* [2018] used an interesting yet simple approach for predicting the the clinical outcomes of breast cancer patients. They used two hidden layers 64, 32 and added a number of neurons to the input layer which has 13698 neurons for the PCA components which were extracted in a preprocessing step. The layers were activated with the exponential linear unit activation function and optimized using Adam optimizer for backpropagation. Learning rate was set to 0.001, batch size 64 , epochs 10000 and each layer was initialized with uniform distribution. Note that we are counting the number of hidden layers in the encoder side. The decoder part is a reflection to the encoder.

Overfitting is one of the major challenges affecting the efficiency of the extracted features. To overcome this problem, Chaudhary *et al.* [2017] used regularization technique. Dropout was used by [Chaudhary *et al.*, 2017; Danaee *et al.*, 2016], and [Fakoor *et al.*, 2013; Zhang *et al.*, 2018] added sparsity penalty to the similarity functions.

However, the autoencoders features were used by different statistical methods and classifiers to solve both binary and multi-class classification problems. Fakoor *et al.* [2013] applied softmax regression for binary classification. Macías-García *et al.* [2017] applied Cox regression model analysis and showed that autoencoders

are valuable statistical tool for noise reduction in breast cancer data. They indicated that this result could be generalized for other biomedical data. Danaee *et al.* [2016] used Support Vector Machine (SVM) and a shallow neural network classifier with a softmax layer to classify the breast samples into two classes. K-means clustering was used in Chaudhary *et al.* [2017] and results showed that clustering the resulted autoencoder features into 2 clusters was the best to give optimum survival analysis results. They next used the resulting two clusters as labels to build an SVM classifier. Results demonstrated that using cluster labels is robust to predict survival-specific clusters better than the PCA extracted components. The overall performance was measured using CI and Brier error values tools. Zhang *et al.* [2018] used an AdaBoot classifier to classify breast cancer patients into good prognosis and poor prognosis groups according to whether distant metastasis had occurred within 5 years or not. Titus *et al.* [2018] applied t-SNE on the extracted representations for further dimensionality reduction to 3-D, 2-D and 1-D spaces. Logistic regression using a "one v.s rest" multi-class approach was used for subtype cancer classification and results showed that the 3-D features were significantly better than the others. Tan *et al.* [2015] applied sample characteristics classification, transcription factor enrichment, survival analysis, and pathway analysis both on binary and multi-class levels. In Tan *et al.* [2015] the author tested the performance of independent hidden nodes in discriminating tumors from normal samples and in Danaee *et al.* [2016] they used another simple neural network classifier consisting of an input layer connected directly to an output layer and compared its performance with an SVM. Most of the studies used 10-fold [Chaudhary *et al.*, 2017; Fakoor *et al.*, 2013; Tan *et al.*, 2015] and 5-fold [Danaee *et al.*, 2016] cross validation for estimating the classification error.

To conclude, neural network filtering methods were used for three different purposes: 1. to learn low dimensional representations; 2. as a transformation

function that removes the noise from the input, and 3. to initialize a neural network classifier by replacing the autoencoder's output layer with a new output layer and re-training the classifier using the previously learned weights and biases, this leads to a better generalization performance. Table 4.3 lists the type of the autoencoder used by each reference, overfitting elimination technique, the number of hidden layers, the predictor type (classification/clustering algorithm), the number of classes and the used evaluation metric.

4.3.2.2 Neural Networks Prediction Methods for Cancer Prediction

Neural network-based prediction (classification) involves building a network that maps the input features to an output with one or two neurons (binary classification) or multiple neurons (multi-class classification). A different approach for solving the multi-class classification problem is by using multiple independent binary neural networks. However, a predictive network can be configured using an input layer, one or more hidden layers and an output layer with $n_{L-1} = k$ where k equals to the number of classes that the training inputs belong to. Before training, a binary string C'_k of length k (called the "codeword") is assigned to each training example such that the codeword for the j-th class is assigned by setting the bit in the codeword at the j-th position equals to 1, and the remaining bits to 0. For example, a 4-class problem can be modeled by assigning the (0,0,1,0) codeword to all training examples belonging to the third class of the problem, this technique is called "one-hot encoding".

A neural network can learn by feeding the labeled (or coded) inputs, calculating the neurons activations and passing them forward through the network. At the output layer, the activation function sums together the contributions of all sending units. This sum is then further modified by adjusting the activation sum to a value between 0 and 1 (in binary-class problems) or by setting the activation

value at a specific position to zero unless a threshold level is reached. Through this process, the network iteratively learns the mapping between each input example $I(x)$ and its class codeword by minimizing an objective function such as the Mean Squared Error (MSE) (Equation 3). In the testing phase, a sample is assigned to the j-th class if the network output $C_k^{'}$ (predicted class) at the j-th position has the highest confidence value.

$$MSE = E[C_j(x) - C_j^{'}(x)]^2 \tag{4.3}$$

Hence, the MSE estimates the posterior probability function for the classification problem.

Neural network-based cancer classifiers have been used with both binary-class and multi-class problems to identify cancerous/non-cancerous samples, a specific cancer type, or the survivability risk.

Our analysis revealed that the architecture of the recent predictive neural networks range between deep MLP models [Chen *et al.*, 2015b; Hou *et al.*, 2018; Mandal & Banerjee, 2015; Yuan *et al.*, 2016] and single-layered networks Bhat *et al.* [2017]; Chen *et al.* [2014]; Dwivedi [2018]; Kumar & Ramakrishnan [2014]. In 2014, Chen *et al.* [2014] predicted the survival risk of lung cancer patients. The decision on the model configuration was made based on trail-and-error and best results achieved with one hidden layer and eight hidden nodes. The input to the neural network was formed by combining six gene expressions, T-stage and N-stage data to form 14-D inputs. The approach achieved high prediction accuracy in classifying the patients into low-risk and high-risk groups. The same MLP model was adopted by Chen *et al.* [2015b] to improve the accuracy of lung cancer survival multi-class prediction where patients were classified into five classes (very low, low, normal, high and very high). Labels were assigned to the samples using a clustering technique in a preprocessing step. The network was used to learn these

labels and tested in terms of its ability to predict the survivability class of new testing samples. The model achieved superior classification accuracy compared to Bayesian network, SVM, and K-Nearest Neighbor (KNN).

Mandal & Banerjee [2015] used another MLP network and experimentally tested the model using two different datasets for breast and lung cancer. Only the results of the latter dataset are considered in this review, as the former consisted of 10 non-genome features. Mandal & Banerjee [2015] used 15 genes that are responsible for the lung cancer, as indicated by an earlier study Mandal *et al.* [2013]. The best accuracy level (94.0%) and was achieved when the number of hidden layers was 3.

In 2016, Bhat *et al.* [2017] suggested a deep generative model called DeepCancer for binary (cancerous/non-cancerous) classification. DeepCancer uses a convolutional GAN with an input dense layer, four hidden layers in the generator side, three hidden layers and an output layer with two hidden nodes in the discriminator side. Interestingly, Bhat *et al.* [2017] also used two other RBM-based regression models, called RBM-SVM and RBM-logistic, as baselines. In both models they used one hidden layer and experimented different number of hidden nodes to get the best classification precession. No feature selection method was applied before classification, and the average performance of DeepCancer was better than the baseline models. This indicates that the generator learned how to accurately represent the features of the datasets.

DeepGene [Yuan *et al.*, 2016] is a convolutional feed forward model for somatic point mutation-based cancer type classification. DeepGene's approach includes converting the gene data into indexes of its non-zero elements using Indexed Sparsity Reduction (ISR), and feeding the output into a deep neural network classifier. The approach was tested on classifying twelve types of cancers and compared with other machine learning approaches such as SVM, KNN and naïve

Bayes. The model achieved the best level of accuracy when using four hidden layers, ReLU activation function, softmax output layer and logarithmic loss function.

A new approach was suggested by [Kumar & Ramakrishnan, 2014] using Extreme Learning Machines (ELM). ELM utilizes the concept of generalized single hidden layer feedforward neural networks. In this network, neurons are generated using an analytical approach so the hidden layer does not require any tuning. The model was tested using five diseases framed as binary class problems. The correlation coefficient mechanism was used to select 2-4% relevant genes and the model achieved high classification accuracies.

Convolutional networks are most commpnly used with images datasets, however, Liu *et al.* [2017] introduced one dimensional convolutional framework that is applicable with the 1-D gene expression dataset. The model, named SE1DCNN after the sample expansion method used to oversample the training split of the data and the 1-D convolutional model which consists of 7 layers (input, 2 convolutional layers, 2 max pooling layers, one fully connected layer and an output layer)

Dwivedi [2018] compared the performance of six machine learning classifiers including: neural networks, SVM, logistic regression, naïve Bayes, classification trees and KNN, in classifying a lukemia dataset into two groups. He used an MLP architecture (one hidden layer with 20 neurons, 30 epochs) and proved that neural network outperformed other considered classifiers.No preprocessing, filtering or feature selection was applied on the 6817 genes which were obtained from a previous study [Golub *et al.*, 1999].

Hou *et al.* [2018] integrated genetic algorithms with artificial neural network in a model called GA-ANN. The model consisted of 3 layers, 1000 nodes as the input layer, and one node in the output layer. The learning rate was 0.1 and the

goal was to classify the clinical phenotype into binary groups. The role of the genetic algorithm was to select the best number of input variables that maximizes the classification accuracy. The population size for the genetic algorithm was set to 100, the maximum evolutionary generations was 50, and the algorithm selected 15 candidate input variables.

Xiao *et al.* [2018] applied deep learning to an ensemble approach that incorporates five different machine learning models. Informative gene selection was applied on differentially expressed genes. Then, a deep learning method was employed to ensemble the outputs of the five classifiers KNN, SVM, decision trees, random forests, and gradient boosting decision trees. The model of five hidden layers was used for binary (tumor/normal) classification hence one output unit was used in the output layer, ReLU activation functions and SGD optimizer was used to minimize the MSE similarity function; regularization, to constrain the magnitudes of the weights, was added to the function to overcome the overfitting problem. This paper used neural networks for discovering the relationship between the 5 different classifiers which were used to classify the training sample into tumor/normal class. The approach built a new dataset consisting to $m \times 5$ items where m is the number of training samples and 5 denotes the binary label predicted by a specific classifier. The new dataset was used as input to the neural network model to avoid using the weighted averaging and majority voting algorithms which is widely used in general ensemble strategies.

Table 4.4 lists the models' name, reference, the type of the used neural network, overfitting prevention technique, the number of hidden layers, and the number of classes i.e. the number of output layer's nodes and the used evaluation metric . The researchers experimentally tested different neural network configurations and parameters but we only listed the number of hidden layers that

achieved best results. We also listed [1]the number of nodes in the output layer to indicate the category of the problem (binary-class/multi-class). We want to note that some of the listed, parenthesized, approaches have been used as baselines for comparison purposes and we considered them here to show their configuration.

4.3.2.3 Neural Network Clustering Methods in Cancer Prediction

Clustering is an unsupervised learning technique which divides the input data examples based on their feature similarity into groups.

Neural networks, particularly SOM, are traditional model-based clustering techniques that are widely used with gene expression data. SOM [Kohonen, 1998] is a single-layered neural network projecting high dimensional data inputs onto a grid space. SOM's output layer neurons are organized in a two or three-dimensional map where each neuron represents a cluster and similar clusters are placed near each other using a simple neighborhood function. SOM associates each of its output neurons with a reference vector, learned during training, and each data point is mapped to the neuron with the closest reference vector.

SOM learns using a pure unsupervised algorithm with unlabeled data and without backpropagation mechanism. Its accuracy can be measured using various evaluation matrices such as the Rand Index (RI, defined by Equation 4.4) which is one of the most commonly used techniques to asses clustering in gene expression datasets. [Yu *et al.*, 2017, 2015].

$$RI = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

Adjusted Random Index (ARI) and Normalized Mutation Information (NMI) are improved variations of RI metric, equations can be found in [Yu *et al.*, 2015].

[1]Refer to the paper in its original format to view a complete version of the tables, https://www.sciencedirect.com/science/article/pii/S0933365717305067

In terms of cancer related applications, clustering is an analysis tool used to divide the samples or the genes into groups. Generally, gene expression clustering approaches separately group the genes or the samples by either considering the relevance, probability, of a sample belonging to a cluster (samples-to-cluster assignment) or the relevance of a gene belonging to a cluster (gene-to-cluster assignment). However, recent approaches tried to improve the quality of clustering by taking both kinds of assignments into account.

The high dimensionality of gene expression samples is a problem facing clustering algorithms. Traditional clustering techniques such as k-means separate the samples (or genes) based on a distance function but this approach is not suitable for high dimensional datasets as the distance between the samples is isometric [Steinbach *et al.*, 2004]. Neural network-based approaches are able to discover sample-cluster mapping and, consequently, they improve clustering accuracy. However, these methods may suffer from noisy genes and improper setting of parameters. Two solutions have been suggested to overcome the high dimensionality problem: (i) clustering ensembles by repeatedly running a single clustering algorithm such as SOM or Neural Gas (NG) with different initializations or numbers of parameters, and (ii) projective clustering by only considering a subset of the high dimensional features.

In 2014, Borkowska *et al.* [2014] evaluated the molecular events that characterize high- and low-grade bladder cancer pathways in bladder cancer patients. Low-grade and high-grade tumors are two distinct pathways of urothelial carcinogenes resulting from the deletion and mutation of some markers. However, many tumors have mutual aspects of low- and high-grade biology. The aim of the study was to discriminate future tumor behavior using molecular alterations. The authors collected samples from 104 random patients and measured the expression of ten genes which were proved to be related to bladder cancer as indicated by

previous studies. The number of the network's inputs was ten and the output was set to two-dimensional grid of 16 (4×4) neurons. SOM was compared to other statistical methods and proved its ability to group the patients into 4 groups, each of which consisted of 4 clusters namely $X1, X2, Y1, Y2$. Patients with the worst prognosis were set in $X1$ group. The highest abnormal TP53 expression and heterozygocity loss for 9, 13 and 17 chromosome loci were grouped in $X2$. Samples with negative UroVysion test and high FGFR3 mutation ratio were grouped in $Y1$ and the ones with positive UroVysion test and had FGFR3 gene mutation were grouped in $Y2$. These results were hard to be obtained using classic statistical models which require explicit assumption of certain relationships within the data that are often unproven.

In 2015, Yu *et al.* [2015] introduced two ensemble clustering frameworks, respectively, Random Double Clustering-based Cluster Ensembles (RDCCE) and Random Double Clustering-based Fuzzy Cluster Ensembles (RDCFCE). Both frameworks select a basic clustering algorithm such as SOM or NG to project high dimensional genes into a low grid dimension. As a result, a set of representative features, corresponding to the centers of the clusters, is generated. A new dataset is generated after that based on a subset of representative features and the process repeats for a specific number of times to get a number of clustering solutions B. A consensus matrix is then constructed using the set of clustering solutions $C'^1, C'^2, ..., C'^B$. Finally, a normalized cut algorithm is used to partition the consensus matrix, and obtains the final result. Both RDCCE and RDCFCE work in the same way except that RDCFCE extends the model by incorporating the fuzzy algorithm to improve the performance of the framework. The models achieved high accuracy clustering results but they only considered sample-to-cluster assignment and ignored the gene-to-cluster assignment. Clustering performance was measured by RI and Purity measure (more details are

provided in [Yu *et al.*, 2015]).

As shown by Table 4.5, Yu *et al.* [2015] also used another neural network-based clustering approach as a baseline which is: (i) double SOM-based Clustering Ensemble Approach (SOM^2CE) Yu *et al.* [2012] and (ii) double NG-based Clustering Ensemble Approach (NG^2CE) Chen *et al.* [2012]. Both obtain satisfactory results on most testing datasets and proved that they are robust to noisy genes.

In 2017, Yu *et al.* [2017] suggested Projective Clustering Ensemble (PCE), which combines the advantages of both projective clustering and ensemble clustering, and they compared its performance with a set of clustering techniques including SOM and RDCFCE. Experiments on synthetic datasets showed that the accuracy of SOM decreases with the increase of injected noise. This indicates that SOM can not distinguish clusters in the presence of noisy genes. Results also showed that the PCE outperformed the RDCFCE as it assigns the irrelevant genes weight that explicitly reduces the interference.

4.4 Summary

Cancer is a world wide genetic related disease which imposes significant mortality and cost. Analyzing gene expression data is essential for discovering genes abnormalities and increasing survivability as a consequence.

Neural network methods are the backbone of most recent cancer prediction models. Their ability to discover complex input-output relationships assists in obtaining more accurate sample-class (or sample-cluster) assignments than using the traditional machine learning tools which relays on distance functions or statistical assumptions. Our analysis to the most recent research in gene expression analysis tools and cancer prediction models reveals that neural networks

are essentially used for filtering the gene expressions, predicting their class, or clustering them.

Neural network filtering methods, more specifically the autoencoders, were used as data engineering methods in a prior step to prediction. Examples of autoencoder-based approaches are contractive autoencoders, regularized autoencoders, sparse autoencoders and stacked denoising autoencoders which was the most widely used one. Most of the suggested filtering approaches have been experimentally tested using shallow architectures. However, deep architectures are more recommended for best practice as they combine many non linearities. On the other hand, it has been proven that shallow networks are inefficient in terms of the required training examples and the number of hidden nodes [Bengio *et al.*, 2007a].

The extracted features have been used to train different machine learning tools such as SVM [Danaee *et al.*, 2016], neural network classifiers [Yousefi *et al.*, 2016], k-means clustering[Chaudhary *et al.*, 2017] or statistical analysis [Macías-García *et al.*, 2017]. Generally, filtering methods were used to reduce the dimensionality of the gene expressions, to transform the gene expressions into a different form, with the same length, or to learn gene expressions' representations and use the learned weights and biases to initialize a predicting network.

Neural network prediction methods have been used for both binary and multi-class problems. In binary problems, the classifier learns how to diagnose a given sample as cancerous, or non-cancerous or to discriminate one type of cancer from another. In this case, the network architecture can only have one [Krishnaiah *et al.*, 2013; Kumar & Ramakrishnan, 2014] or two [Bhat *et al.*, 2017; Chen *et al.*, 2015b] output neurons in its output layer. In multi-class problems the network learns how to discriminate between multiple types of cancer or to predict the survivability risk; so the network's output layer has a number of neurons equal

to the number of classes that the training data belong to [Yuan *et al.*, 2016], or alternatively multiple binary classifiers can be used. However, the decision on the number of hidden layers and nodes is usually made based on trail-and-error technique. Statistical preprocessing have been also applied in some cases [Chen *et al.*, 2015b]. Deep architecture with convolution layers was the most recently used model, especially with multiclass problems, and proved efficient capability in predicting cancer subtypes as it captures the spatial correlations between gene expressions. However, full Details on the networks configuration, overfitting elimination technique and learning parameters were not provided in most studies. Results generally show that neural network-based approaches outperformed other machine learning tools in classifying gene expression samples in most of the studies.

Clustering is another analysis tool that is used to divide the gene expressions into groups. Neural network-based clustering approaches have been applied as an alternative to traditional clustering techniques, which use a distance function to measure the similarity between gene expressions. Nevertheless, neural network approaches such as SOM and NG are not able to distinguish the noisy genes, so samples are clustered based on both the relevant and irrelevant genes. Ensembling clustering and projective clustering are two general approaches suggested to overcome the high dimensionality related problems. A hybrid approach combining both the ensembling clustering and projective clustering, such as [Yu *et al.*, 2017], has proved higher accuracy than using single-point clustering algorithm such as SOM.

Considered studies used different preprocessing techniques, datasets, analysis and classification tools and they were used to solve different problems (binary and multi-class classifications). These differences make the decision on the best network configuration and performance hard to be made and unfair. This was

also noted by [Svensson, 2016] who indicated that there is no machine learning algorithm consistently outperforms any of the other as the nature of the dataset seem to be of a major influence on the performance of the algorithm. There is an opportunity here for a future work to develop a "benchmark" toolkit for gene expression data mining so that modern NN algorithms can be compared in a uniform way. However, we are discussing in the next section some practical issues that can be considered for future models.

4.5 Discussion

In this paper, we reviewed recent works on neural network based cancer prediction models. Here we discuss some technical problems that can be considered for building new models.

Overfitting: gene expression datasets are high dimensional and have a relatively small number of samples. While most of the studies on cancer prediction models focused on learning good representations that increase the models predictability, some ignored the problem of overfitting which is caused by using small number of training examples as indicated by [Srivastava *et al.*, 2014]. Overfitting problem occurs when the network is over trained on the training examples so it properly fits the training examples but not the validation and the testing examples due to the lack of generalization capability. Overfitting can be avoided by: (i) adding weight penalties using regularization [Srivastava *et al.*, 2014]; (ii) models combination by averaging the predictions from many models trained on different datasets [Srivastava *et al.*, 2014] ; (iii) dropout [Srivastava *et al.*, 2014].

Model configuration and training: choosing neural network configuring and setting its parameters is a crucial issue for extracting good information that

achieve high prediction accuracy. While there is no rule of thumb for setting the networks parameters, there are some recommended issues that can be considered to reduce both the computational and memory expenses for better performance such as: (i) proper initialization: poor initialization cause slow convergence, local minima and model uncertainties problems [Alhamdoosh & Wang, 2014]. (ii) pruning the unimportant connections by removing the neurons that have zero values. Pruning is a good approach for reducing the memory and computational complexities. (iii) using ensemble learning framework by training different models using different parameter settings or using different parts of the dataset for each base model. (iv) class imbalance is a data-related problem that had little to get attention to date when building and analyzing cancer prediction models. This problem makes the classifiers biased toward the classes that has the majority of the data, and consequently poorly classifies the samples belonging to minor classes. Danaee *et al.* [2016] used SMOTE for dealing with class imbalance on the high dimensional level. however, oversampling methods can be applied to generate synthetic representations to increase classifiers accuracy.

Model evaluation: Using 10-fold and 5-fold cross validation to estimate the error of classifying small data size leads to severely inaccurate conclusions as proved by [Braga-Neto & Dougherty, 2004]. Braga-Neto & Dougherty [2004] investigated, in their simulation study the performance of cross-validation, substitution and bootstrap methods and revealed that cross-validation displayed excessive variance and, hence, it is unreliable for small size data. The bootstrap method proved more precise and reliable predictability.

Study reproducibility: Reproducibility of the studies is another important issue that has to be highlighted here. A study is reproducible when others are able to replicate the results using the same algorithms data and methodology. Reproducibility enhances research reliability and requires the authors to publish

the used data and to clearly document their methodology. Researchers using genomic databases such as TCGA Tomczak *et al.* [2015] should at least state the query used for downloading their experimental dataset in supplementary material.

4.6 Conclusion

This paper reviewed the most recent neural network-based cancer prediction models and gene expression analysis tools. The considered papers were published between 2013-2018 and used gene expression datasets for cancer classification and clustering. This review presented some commonly used architectures, datasets, and the accuracies of each suggested model.

Analysing the considered papers indicated that neural network methods are able to serve as filters, predictors and clustering methods. Neural network filtering methods are used to reduce the dimensionality of the gene expressions and remove their noise. MLP and convolutional neural network classifications methods have been used with binary-class and multi-class classification problems while the number of the networks' hidden layers and hidden nodes have been decided by trial-and-error. A hybrid approach combining both ensembling clustering and projective clustering is the best to achieve high clustering accuracy.

Deciding the network architecture is one of the challenges facing the cancer prediction model designers, as there is no specific rule to guarantee high prediction accuracy. Most of the studies determined the number of hidden layers and neurons based on trail and error. However, this study indicated that the role of the neural network determines its general architecture.

This study has summarized most recent approaches and their related neural network architectures. We also highlighted some critical points that have to be considered when building a neural network-based prediction model such as over-

fitting and class imbalance. More powerful neural network-based approaches can be suggested in future by choosing different network's parameters or combining two or more of the presented approaches.

Conflict of Interest Statement

The authors declare that they have no conflict or competing interests.

Table 4.2: Datasets and Preprocessing. Dimension column shows the number of $features \times samples$ used for prediction, "-" indicates a missing value.

Dataset	normalization	cancer type	dimensions
Golub *et al.* [1999]	AML	-	7129×46
TCGA	FPKM	lung	1385×162
		stomach	801×271
		breast	934×878
TCGA, GEO and others	-	prostate	15×466
TCGA	Funnorm	breast	$300,000 \times 862$
GEO	MAS 5.0	breast	$881 \times 13698 + 69/75$
TCGA	unit-variance	liver	$- \times 365$
Woodward *et al.* [2013]	unit-variance	breast	30006×20
Alon *et al.* [1999]		lukemia	12600×60
Cheok *et al.* [2003]		brain	2000×62
TCGA	unit variance	33 different cancer	$5000 \times 10,459$
Hoshida *et al.* [2007]	-	breast	1213×49
Hoshida *et al.* [2007]		DLBCLA	661×141
Monti *et al.* [2003]		leukemia	985×248
Monti *et al.* [2003]		Multi-tissue	1000×103
de Souto *et al.* [2008]		brain	1379×142
de Souto *et al.* [2008]		multi-tissue	1363×190
de Souto *et al.* [2008]		endometrium	1771×42
de Souto *et al.* [2008]		multi-tissue	1571×174
home made	-	breast	21×222
TCGA	-	12 different cancer	$3122 \times 22,834$
TCGA	unit variance	breast	40992×20
		prostate	12600×136
GEO	-	breast cancer	$1210 \times -$
Curtis *et al.* [2012]	-	METABRIC	2136×2520
TCGA		breast cancer	547×2520
Mills *et al.* [2009]	-	AML	2341×54613
Fujiwara *et al.* [2012]		adenocarcinoma	193×34749
Woodward *et al.* [2013]		breast cancer	1047×30006
Klein *et al.* [2009]		leukemia	2284×54675
Cheok *et al.* [2003]		leukemia	658×12600
Yagi *et al.* [2003]		AML	625×12625
Wang *et al.* [2005]		breast cancer	2301×22277
Gashaw *et al.* [2005]		seminoma	618×12625
Petricoin *et al.* [2002]		ovarian cancer	153×15154
Alon *et al.* [1999]		colon Cancer	32×2000
Pomeroy *et al.* [2002]		medulloblastoma	30×7129
Singh *et al.* [2002]		prostate Cancer	102×12600
Verhaak *et al.* [2009]		leukemia	2389×54613
Armstrong *et al.* [2002]	-	leukemia	2194×72
Dyrskjøt *et al.* [2003]		bladder	1203×40
Risinger *et al.* [2003]		endometrial	1771×42
Yeoh *et al.* [2002]		leukemia	248×2526
Bhattacharjee *et al.* [2001]		lung	1543×203
Chowdary *et al.* [2006]		breast	182×104
Gordon *et al.* [2002]		lung	1626×181
Laiho *et al.* [2007]		colorectal	2202×37

Table 4.3: Neural network filtering methods used in cancer prediction models.

neural network type	overfitting	features	h layers	predictor	class	Metric
Variational Autoencoder	KL divergence	100	2	Logistic regression	5	accuracy
Stacked Autoencoder	noising sparsity	64	4	AdaBoot	2	accuracy specifity sensitivity MCC
Stacked Denoising Autoencoder	noising	50 30 100	2	neural network	2 4 2	accuracy
Regularized Autoencoder	regularization dropout	37	2	K-means SVM	2 2	c-index $log-rank$ $p-vlaue$ Brier score
Contractive Autoencoder	Frobenius norm	22	1	statistical analysis	-	cox regression Friedman test Holm step- down
Variational Autoencoder	KL divergence	100	2	statistical analysis	-	$p-value$
Stacked Denoising Autoencoder	noising dropout	500	5	NN classifier SVM classifier	2	accuracy sensitivity specificity precision F1
Stacked Denoising Autoencoder	noising	100	1	characteristics classification	2 5	accuracy
Sparse Autoencoder	noising sparsity	183 28 1047 125 60 27 143 20 100 30 30 34 230	1	softmax regression classifier	2	accuracy

Table 4.4: Neural network predicting methods used in cancer prediction models

Model name	neural network type	overfitting	no.layers	no. hidden nodes	no.output	metric
MLP	MLP	regularization	5	-	1	AUC
MLP	MLP	-	1	20	1	F1
GA-ANN	MLP	-	3	-	1	AUC
SE1DCNN	convolutional	-	7	21,4,21,4,-	2 4 2	accuracy
DeepCancere (RBM-Logistic) (RBM-SVM)	convolutional GAN	-	7	-	2	precesion recall F1
DeepGene	convolutional network	-	4	-	12	accuracy
Cancer adjuvant chemotherapystrategic classification	MLP	-	1	8	2	acccuracy sensitivity specificity
MLP	MLP	-	3	-	1	accuracy
MLP	MLP	-	1	8	5	accuracy
ELM-based classifier	ELM	-	1	-	1	accuracy

Table 4.5: Neural network clustering methods used in cancer prediction models.

Model Name	neural network type	No. Clusters	metric
PCE (SOM) (RDCFCE(SOM))	SOM	4 3 6 4 5 14 4 10	RI Purity
RDCCE RDCFCE (SOM^2CE) (NG^2CE)	SOM / NG	3 3 4 6 5 2 2 2	AR ARI NMI
SOM	SOM	16	-

Chapter 5

A Novel Synthetic Over-sampling Technique for Imbalanced Classification of Gene Expressions Using Autoencoders and Swarm Optimization

Class imbalance is a problem emerges when the examples of the data are distributed unequally between its different classes; such that the majority to minority class ratio is significantly large. This imbalanced distribution of the data allows the classifiers to be biased toward the majority class examples which degrades their general performance.

This research was done as part of the whole study to introduce autoencoder-based techniques to solve problems affecting the classification accuracy of high dimensional small sized data. As mentioned earlier, gene expression datasets are high-dimensional, have limited number of examples, and exhibit the class imbal-

ance problems. In addition, the classes in gene expression data are unseparable [Okun & Priisalu, 2007] which makes traditional solutions to the class imbalance problem such as the SMOTE to be ineffective. These solutions rely on generating new examples of the data on the line segment between neighbouring minority examples allowing for increasing the overlap region between the classes.

The research presented in this chapter exploits the representations learned by a previously trained autoencoder and the PSO algorithm to generate synthetic minority class representations and decrease the majority to minority class ratio as a result. This oversampling technique improved the classification accuracy of four different classifiers compared to SMOTE and a newer version of it called Density-Based Synthetic Minority Over-sampling TEchnique (DBSMOTE) [Bunkhumpornpat *et al.*, 2012].

The technique proposed in this chapter motivates the tuned (learned) neurons in the decoder part of the network using random vectors at the initial state of the optimization. This forward pass to the random vectors generates high-dimensional outputs at the output layer of the decoder, the resulting outputs are compared to a given training minority class example using a distance function to quantify the similarity between them and update the position and velocity of the swarms particles which are used as inputs for the second iteration of the algorithm.

This method generates synthetic minority class representations for the training data at the testing phase of the network life cycle. However, the experiments done for this paper motivated the author to introduce new techniques that improve "learning" from small-sized high dimensional datasets, which will be discussed in the next chapter.

A Novel Synthetic Over-sampling Technique for Imbalanced Classification of Gene Expressions Using Autoencoders and Swarm Optimization

Maisa Daoud *mtdd1@students.waikato.ac.nz*
Michael Mayo *michael.mayo@waikato.ac.nz*
Department of Computer Science
University of Waikato, Hamilton. NZ

Abstract

A new synthetic minority class over-sampling approach for binary (normal/cancer) classification of microarray gene expression data is proposed. The idea is to exploit a previously trained autoencoder in combination with the Particle Swarm Optimisation algorithm to generate new synthetic examples of the minority class for solving the class imbalance problem. Experiments using two different autoencoder representation sizes (500 and 30) and two base classifiers (Support Vector Machine and naïve Bayes) show that the proposed method is able to generate discriminating representations that outperformed state-of-the-art methods such as Synthetic Minority Class Over-sampling Technique and Density-Based Synthetic Minority Class Over-sampling Technique in many test cases.

Keywords: class imbalance, cancer prediction, autoencoders, classification.

5.1 Introduction

A labeled dataset for classification purposes is considered imbalanced when samples are distributed unequally among different classes. This kind of dataset poses

a challenge to machine learning classifiers as it becomes difficult to model the minority class samples.

The problem of imbalanced datasets exists in many real-world applications such as text classification [Zheng *et al.*, 2004], detection of fraudulent telephone calls [Fawcett & Provost, 1997], information retrieval and filtering tasks [Weiss, 2004]. In this research we are interested in high dimensional cancer prediction applications where using a dimensionality reduction method such as an autoencoder is a necessity for reducing noise and increasing classification accuracy.

Microarrays produce high dimensional matrices consisting of thousands of gene expressions for only a few hundred samples. The shortage in the number of samples is a result of the cost and time needed for sequencing the biological samples. Microarray datasets are usually imbalanced, where cancer samples form the majority of the dataset, and have non linearly separable classes [Lin & Chen, 2012]. The high dimensionality of the datasets is a major challenge to machine learning classifiers and reducing it would help in decreasing computational complexity and increasing classifier's accuracy [Dong *et al.*, 2017].

Autoencoders are commonly used to reduce the dimensionality of data and to remove the noise. However, the suggested approach utilizes the representations learnt by a pre-trained autoencoder for generating new synthetic minority class data. To achieve this, we employed the Particle Swarm Optimisation (PSO) algorithm to optimize the generated data to be as much similar (close) to the original source data as possible. The aim of the approach is to find a variation of a desired minority class representation that will make a good synthetic example to be added to the dataset.

The approach was experimentally tested by training different autoencoders using the training split of each of the considered microarray datasets. We tried different number of hidden nodes, 500 and 30 respectively, in the network's bot-

tleneck layers to test the suitability of the approach in generating representations with different dimensionalities. The approach showed promising results when compared to two other methods called Synthetic Minority Class Over-sampling Technique (SMOTE) and Density-Based Synthetic Minority Over-sampling Technique (DBSMOTE). We used naïve Bayes and Support Vector Machine (will be abbreviated by SMO after the Sequential Minimization Optimization algorithm which is used for training it) classifiers to classify the data generated by each oversampling method. Generally, naïve Bayes and the suggested method's representations achieved the highest prediction accuracy overall compared to the SMO and other oversampling methods. However, the SMO classifier performed better with our method's representations than SMOTE's. Best results for the SMO were achieved with DBSMOTE representations.

The rest of the paper is organized as follows. In Section 2 we present a basic background to autoencoders, PSO and the class imbalance problem. Section 3 presents our motivation, Section 4 shows the methodology, Section 5 discusses the experiments and results. Finally, the conclusion is presented in Section 6.

5.2 Background

This section presents a basic introduction to autoencoders, PSO and the class-imbalance problem.

5.2.1 Autoencoders

Autoencoders are neural network models that have been successfully used for extracting low dimensional representations from high dimensional data. In its simplest form, the autoencoder consists of two fully connected feedforward layers called the encoder and the decoder and this results in a three-layered network.

In the general (deep) case, both the encoder and the decoder may have multiple layers. The number of outputs at every layer L_l equals the number of inputs at the next layer L_{l+1}, and the number of outputs from the encoder equals to the number of inputs to the decoder.

The autoencoder learns from examples $x_1, x_2, \ldots$ by feeding the training input vectors forward through the network to calculate the activation values of its neurons at every layer subsequent to the input layer. The activation values at the output layer are calculated and aggregated to get an output $O(x_i)$ for each x_i. Since the aim of an autoencoder is to reconstruct the input perfectly, then the difference between $O(x_i)$ and x_i is calculated using a predefined error function such as the Mean Squared Error (MSE). The network tries to minimize the objective function using a backpropagation learning algorithm which propagates the error derivatives back through the network to fine-tune the weights for optimal reconstruction.

Stacked Denoising Autoencoders (SDA) [Vincent *et al.*, 2010b] are an improved variation of the ordinary stacked autoencoder which is trained to denoise corrupted versions of the inputs. Different methods can be used for corrupting the initial input such as Gaussian noise, mask noise, and salt-and-pepper noise Vincent *et al.* [2010b]. SDAs use the same training algorithm that is used by normal autoencoders but achieve significant improvements over the latter as they are better able to learn useful structures of the data such as Gabor-like edge detectors from natural images [Vincent *et al.*, 2010b].

5.2.2 Particle Swarm Optimization

PSO [Poli *et al.*, 2007] is an optimization algorithm ideally used for optimizing continuous nonlinear functions. The algorithm can be simply implemented by randomly initializing the location (vector) and velocity (vector) for a number of

particles in a flock. On each iteration of the algorithm, the fitness of each particle's position is evaluated using the problem's objective function. Each particle i updates its velocity based on its current position y_i and its previous best position p_i. The velocity is also influenced by the position p_i found by the best neighbor p_g. New points are found, for the subsequent iteration, by adding the velocity coordinates to y_i according to the following equations:

$$v_i = v_i + U(0, \phi_1) \times (p_i - y_i) + U(0, \phi_2) \times (p_g - y_i) \tag{5.1}$$

$$y_i = y_i + v_i \tag{5.2}$$

where $U(0, \phi_1)$ is a vector of random numbers generated for each particle at each iteration and uniformly distributed over $\left[0, \phi_i\right]$, ϕ_1 and ϕ_2 are acceleration factors and v_i has to be kept within the range $[V_{min}, V_{max}]$ [Poli *et al.*, 2007].

Typically, PSO will continue iterating until either the best particle with the best position (representing a perfect solution) is found, or until a maximum number of iterations has been reached.

5.2.3 Class Imbalance

Different approaches have been suggested to improve the prediction accuracy of imbalanced test samples. Majority class random under-sampling is one of the direct and most straight forward suggested approaches [Kubat *et al.*, 1997]. Focused under-sampling, which targets the samples that are further away from the classes' borders, has been investigated by Japkowicz [2000] who observed that using sophisticated under-sampling techniques did not add an advantage to the results in the considered domain.

Minority over-sampling, on the other hand, is a different approach that can be

either applied using random duplication method or more sophisticated synthetic methods. SMOTE [Chawla *et al.*, 2002] is a synthetic over-sampling method which generates samples on the line segment joining a chosen minority class sample and its neighbors from the same class. DBSMOTE is a new variation of SMOTE that discovers an arbitrary shaped cluster and over-samples it. Majority Weighted Minority Over-sampling (MWMOTE) [Barua *et al.*, 2014] is another synthetic over-sampling clustering-based technique which generates new samples from weighted minority class samples.

5.3 Motivation

Most of the suggested minority over-sampling methods are efficient and effective when generating synthetic samples, with the resulting datasets increasing classification accuracy [Barua *et al.*, 2014; Han *et al.*, 2005]. However, we found that existing methods work ideally when the boundaries between the classes are not ambiguous, a situation that is not true for all kinds of datasets.

Real world datasets, especially medical datasets, are high dimensional, complex, noisy and often have limited number of samples [Lin & Chen, 2012]. In such kind of datasets, majority and minority class samples are not separable and they frequently overlap [Lin & Chen, 2012]. See Figure 5.1 for an illustration of this. Classifying high-dimensional datasets requires reducing the dimensionality of the data using either a dimensionality reduction method or a feature selection method to remove the noise. Both of these solutions depend on trial-and-error techniques for generating or selecting a number of discriminating features that increase the classifier's accuracy. Hence, reducing the dimensionality of the data does not necessarily solve the class overlap and borderline-ambiguity problems.

K-nearest neighbor-based methods such as SMOTE generate synthetic sam-

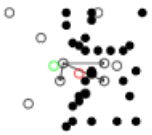


Figure 5.1: Border line ambiguity in 2-D space. white circle: minority class samples, black dots: majority class samples, red circle: SMOTE generated sample and green circle: a synthetic sample generated by the proposed approach.

ples on the line segments between sample s and its K nearest neighbors. This tends to increase the density of the minority-class samples generated within the majority class region. For example, in Figure 5.1, a synthetic sample (the red sample) is generated from two positive (white) examples. Since the synthetic example lies on the line segment between two positive examples, it ends up being very close to several negative (black) examples. As a consequence, the classifier's performance in discriminating both majority and minority samples may well decrease because the synthetic example is effectively noise. The problem becomes more complicated as the dimensionality of the data increases. Blagus & Lusa [2012] proved, based on a set of evaluations, the poor performance of SOMTE in generating high dimensional gene expression data.

Based on this observed weakness of the SMOTE family of algorithms, our proposed new method aims at overcoming this problem by generating synthetic examples that are variations of a single real example as opposed to being a function of multiple real examples. In this way, the synthetic examples should be in theory better represent the true region of the feature space occupied by the minority class we are attempting to model. In Figure 5.1, such an example is illustrated by green example.

Our approach is not a random oversampling-by-replication or duplication approach. Instead, we use an optimization approach that utilizes the trained decoder part of an autoencoder to generate a new example that is similar to an existing minority class sample. The autoencoder is trained on all of the training examples, both minority and majority, but the optimisation step focuses only on the minority class training samples since these are ones being generated. In this respect, our approach shares a commonality with SMOTE: both approaches consider all minority class training samples to generate the synthetic data. Therefore we chose SMOTE and a newer version DBSMOTE as suitable baseline methods for comparison.

5.4 Minority Over-Sampling Using Autoencoder and Optimization

Our proposed approach integrates both the decoder part of a trained SDA and the PSO algorithm for generating synthetic minority class samples. The algorithm is presented in detail in the following two sub-sections.

5.4.1 Training the Autoencoder

To configure the autoencoder, we first decide on the number of layers and size of the layers. The size of the input and output layers must be fixed and equal to the dimensionality of the dataset, but the size of the hidden layers can be varied.

We also must select parameters for the neural network training algorithm, which include an initialization method for the initial network weights, a noising method such as salt-and-pepper and an objective function to compute the difference between training examples (both minority and majority class examples)

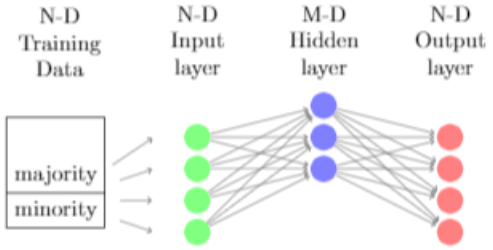


Figure 5.2: Training a single layer autoencoder.

and their reconstructed versions. The optimization algorithm for minimising the reconstruction error must also be chosen for the training procedure.

After training the network for a set of iterations as shown in Figure 5.2, the autoencoder is expected to learn the training data's abstract representations which can be obtained by propogating an input example forward through the encoder portion of the network and "reading off" the final activations at the bottleneck layer. To generate a compact representation j for a data instance J we simply perform this process using J as an input example. If the bottleneck layer consists of M neurons, ($M << N$, where N is the example's dimensionality) then the representation j will be M dimensional while the original examples J is N dimensional. This process of encoding can be encapsulated as a function $j = Encoder(J)$.

Our approach uses a trained encoder to generate low-dimensional training and testing datasets, $R_{training}$ and $R_{testing}$ respectively, by activating the encoder part using the low dimensional training and testing datasets. The compact training representations $R_{training}$, since they are labeled with the same classes as the original training data, can be used for building a machine learning model for classification, while $R_{testing}$ is going to be used for evaluating the accuracy of the model.

5.4.2 Generating Optimized Representations

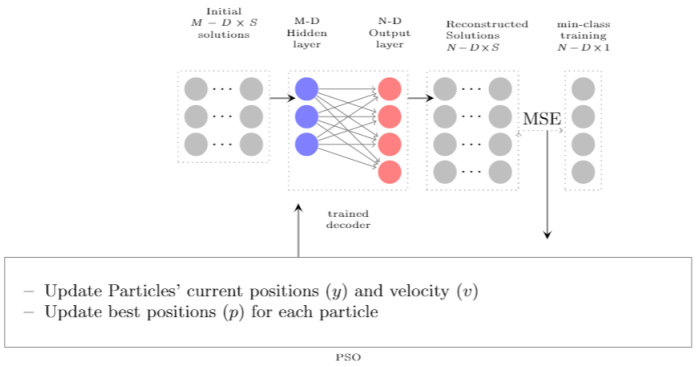


Figure 5.3: Generating minority class over-sampling using trained decoder and PSO.

We used the decoder part of the trained autoencoder in combination with the PSO algorithm to generate new synthetic examples. This procedure is described in Algorithm 1 and by Figure 5.3. Given one real minority class sample, multiple synthetic examples can be generated by repeatedly applying the algorithm.

To generate a single optimized minority class representation, we randomly initialize the location and velocity (each of which is an M-D vector) for a swarm $y(t = 0)$ of S particles to small numbers between [0,1]. The algorithm iterates over the initial particles $y_1, y_2, ..., y_s$ to feed them separately through the decoder part of the trained autoencoder and to generate $Y'_1, Y'_2, .., Y'_s$ at the output layer.

The MSE (difference) between each generated version Y'_i (at the output layer) and a desired minority training sample Y is computed to determine the particle's fitness. PSO determines each particle's p_g to update the positions and the velocities according to Eq. 1 and Eq. 2. Updated particles are fed again through the decoder part for a subsequent iteration and the algorithm runs until a termination condition is satisfied. The algorithm returns the most fitted particle (representation) y which inverse Y' is the most similar to the original sample Y.

Algorithm 2: Minority-Class Over-Sampling using PSO

```
t = 0 ;
Y =target minority-class training sample;
y(t) = y_1, y_2, .., y_S /*Swarm with S randomly initialized M-D particles*/ ;
p_1, p_2, .., p_S =y_1, y_2, .., y_S ;
while terminal condition is not satisfied do
    for i = 1...S do
        /*feed y_i through the trained decoder to generate the reconstructed
          versions */;
        Y'_i = decoder(y_i) ;
        /* evaluate the fitness f(y_i) */ ;
        f(y_i)= MSE(Y,Y'_i) ;
        if f(y_i) < f(p_i) then
            p_i = y_i;
        determine p_g ;
        update y_i using Eq(1) and Eq(2);
    t = t + 1;
```

5.5 Experiments

We used 10 different datasets downloaded from The Cancer Genome Atlas (TCGA) Data Portal (https://portal.gdc.cancer.gov/) to experimentally test the proposed approach. The TCGA datasets are genomic data made available to public community and researchers to improve the prevention, diagnosis, and treatment of cancer. An R/Bioconductor package called TCGAbiolinks [Colaprico *et al.*, 2015] was used to download and prepare the RNA-seq gene expression data. Table 5.1 shows detailed characteristics of the experimental datasets which were chosen based on the number of samples and imbalance ratios.

5.5.1 Experimental Setup

To build our proposed approach, we used an open source machine intelligence library called TensorFlow r.1.4 [Abadi *et al.*, 2015] to configure two different autoencoders as follows:

Table 5.1: The microarray datasets and the distribution of the samples across normal and cancerous groups. "No" denotes normal samples and "Ca" is for cancer samples.

Dataset	cancer type	no(train/test)	ca(train/test)	imb.ratio	length(N)
BLCA	Bladder Urothelial Carcinom	15/4	326/82	0.047	31209
BRCA	Breast Invasive Carcinoma	90/13	877/218	0.094	3506
ESCA	Esophageal Carcinoma	8/3	147/37	0.060	4438
HNSC	Head & Neck Squamous Cell Carcinoma	35/9	416/102	0.085	3459
KICH	Kidney Chromophobe	13/5	73/18	0.198	4293
LUAD	Lung Adenocarcinoma	47/12	412/103	0.115	3849
LUSC	Lung Squamous Cell Carcinoma	40/11	401/101	0.102	5354
READ	Rectum Adenocarcinoma	8/2	75/19	0.106	3649
STAD	Stomach Adenocarcinoma	29/7	331/83	0.087	2439
THCA	Thyroid Carcinoma	48/13	400/100	0.122	2149

1. An $input - 1000 - 500 - 1000 - output$ autoencoder, with bottleneck layer size $N = 500$; and

2. An $input - 1000 - 30 - 1000 - output$ autoencoder, with bottleneck layer size $N = 30$.

In each case, the size of the input and output layers was determined by the dimensionality of the individual datasets. The reason for considering two different autoencoder architectures was so that the effectiveness of the method in generating discriminative minority-class representations with different dimensionalities (i.e. small and large) could be assessed.

We used the MSE as an objective function. The Stochastic Gradient Descent (SGD) [Bottou, 2010] back propagation algorithm was used to optimise the network weights. The learning rate was set to 0.001, the batch size was 25 and the

number of epochs was 250. We also tried to train the autoencoder for more (500) and less (100) iterations but results were comparable to the 250 iteration case, so we only considered this number of iterations in our experiments.

For optimization, we used a python PSO library called `pyswarm` `https://pythonhosted.org/pyswarm/` to generate synthetic samples from minority class representations by utilizing the decoder part of the trained autoencoder. The PSO library requires the user to specify the lower-bound and upper-bound vectors that bound the swarms' locations between them. Hence, we defined the lower-bound as an M-D vector of zeros, and the upper-bound as an M-D vector of ones. The number of particles and iterations have also experimentally specified and were set to 200 and 2 respectively.

For the baseline comparison, we used SMOTE and DBSMOTE methods from an R package called `smotefamily` [Siriseriwan, 2016] to over-sample the minority-class representations extracted using the trained autoencoder. SMOTE method requires the user to specify K, the number of neighbors to be considered, which was set to 5 as indicated by the original paper [Chawla *et al.*, 2002].

Each of the configured autoencoders was separately trained and tested ten times per dataset so that mean expected performance metrics of the algorithm could be calculated. In each run, we randomly selected 80% of the cancer samples and 80% of the normal samples to form a training set, and the remaining 20% were reserved for testing. After training, we extracted the training and testing datasets representations, and then over-sampled the minority class training representations to varying degrees. The exact amount of over-sampling was determined by the size of the majority class. We tried over-sampling the minority class until it was 25%, 35%, 50%, 75%, 85% or 100% of the majority class size. This was done for each of the ten repetitions per dataset, and was done using both our new proposed method (referred to hereafter as Opt/Decoder), SMOTE and

DBSMOTE. Therefore three versions of each training dataset for each amount of over-sampling was generated. Finally, we built a classifier using the training samples and tested it using the testing representations. Note that we did not over-sample the test data.

5.5.2 Classification Algorithms

The goal of this research is to solve the class-imbalance problem which degrades the performance of classification algorithms. In order to evaluate this, we chose two classifiers: specifically, SVM (abbreviated SMO, after the Sequential Minimization Optimisation algorithm) and naïve Bayes. Both of these classifiers are simple and produce interpretable models. We used `WEKA` package [Eibe Frank] implementations for both classifiers with their default parameters.

We used the extracted training representations from both classes with the, appended, generated minority-class samples for building each classifier.

The metric used for assessing the performance of each algorithm was Area Under the Receiver Operating Characteristics graph (AUROC, or alternatively ROC). An AUROC value of close to 0.5 indicates random prediction performance while a value close to 1.0 indicates near-perfect prediction performance.

5.5.3 Results Using the TCGA Datasets

We compared the average AUROC (over 10 runs) of the SMO and naïve Bayes (NB) classifiers which were used for predicting Opt/Decoder, SMOTE and DBSMOTE representations with different minority/majority ratios (25%, 35%, 5%, 75%, 85%, 100%) and representation size $N = 500$ and $N = 30$ respectively.

The visual inspection, based on the number of winning cases per method, of the graphs, listed in the Appendix section, shows that NB and Opt/Decoder (NB+Opt/Decoder)(indicated by the blue line) representations has generally

achieved the highest AUROC in most test cases compared to other methods and classifiers.

Comparing the performance of the NB+Opt/Decoder with NB+SMOTE and NB+ DBSMOTE indicates that our method's (500D) representations were better classified than SMOTE (500D) and DBSMOTE(500D) respectively. Another interesting thing to be noted is the slight increase of NB+Opt/decoder (500) AURC with the increase in the number of generated minority samples. NB+Opt/decoder (500) has also proved success in cases where the number of training samples is relatively small (BLCA, ESCA, KICH, LUSCA, READ and STAD). The NB+Opt/Decoder (30D) had a comparable behavior to NB+SMOTE(30D) but had a slight improvement over and NB+DBSMOTE(30D).

The performance of the SMO classifier, on the other hand, was generally lower than the NB. However, our method (represented by the green line) showed a slight improvement over SMO+SMOTE in 500D case. A comparable behavior of the SMO classifier in detecting both Opt/Decoder ad DBSMOTE data was noted in the case of 500D but SMO+DBSMTE won the competition in 30D case.

To statistically prove the results and to get a general overview, across all datasets, about the performance of our method, we followed the suggestion by [Demšar, 2006] and applied Wilcoxon signed-rank test [Gibbons & Chakraborti, 2011]. The test results shown in table 5.2 indicates that the overall performance of the NB+Opt/Decoder was better than NB+SMOTE and NB+DBSMOTE with better positive rank (R+) sum than other methods. SMO+Opt/Decoder had also a general advantage over SMO+SMOTE using both representation lengths (500D) and (30D). However, the SMO classifier had better overall detection accuracy to DBSMOTE's representations than our method's.

Table 5.2: Wilcoxon signed rank test for the average AUROC over the 10 runs, different minority/majority ratios, representations length and classifiers.

Method	classifier	R+	R-	p-val
SMOTE30	SMO	1243	587	0.007956
	NB	1761	68	2.32E-10
SMOTE500	SMO	1797	32	4.59E-11
	NB	1645	184	3.93E-08
DBSMOTE30	SMO	11	1819	1
	NB	1234	596	0.04348
DBSMOTE500	SMO	639	1191	0.01715
	NB	1340	490	1.21E-09

5.6 Conclusion

This paper presented a different method for generating training data using the features learnt by a pre-trained autoencoder. Results on 10 different microarray datasets show that our method frequently outperforms SMOTE and DBSMOTE in generating better discriminative minority class 500D representations, leading to increased the AUCROC for both SMO and naïve Bayes classifiers in most test cases. The opposite was true in the 30D experiments, however, with SMO+DBSMOTE generally performing the best overall. This leads to the question of whether SMOTE family of algorithm can scale to higher dimensional data.

Several issues are left for future work, such as the consideration of other optimisation algorithms for both training the autoencoder and generating the representations. Applying this method to both very small datasets and imbalanced multi-class datasets is another possibility.

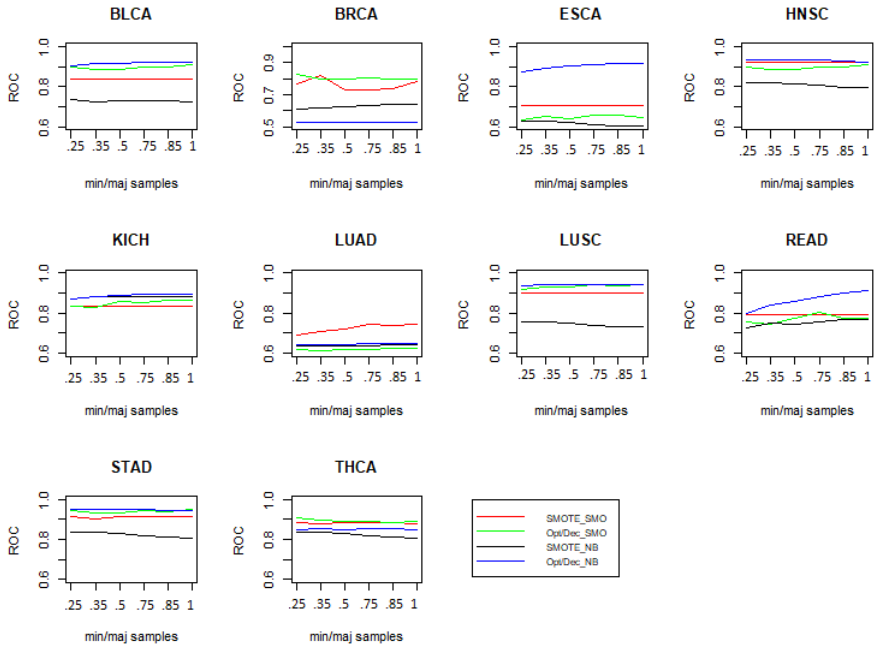


Figure 5.4: Experimental results for AUROC evaluated by classifying the (500D) SMOTE and Opt/Decoder representations using SMO and naïve Bayes.

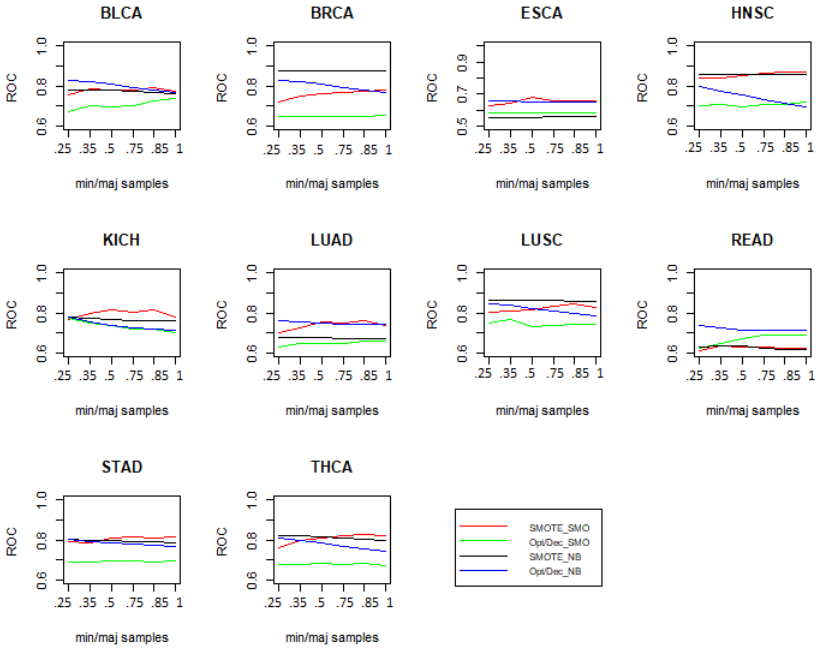


Figure 5.5: Experimental results for AUROC evaluated by classifying the (30D) SMOTE and Opt/Decoder representations using SMO and naïve Bayes.

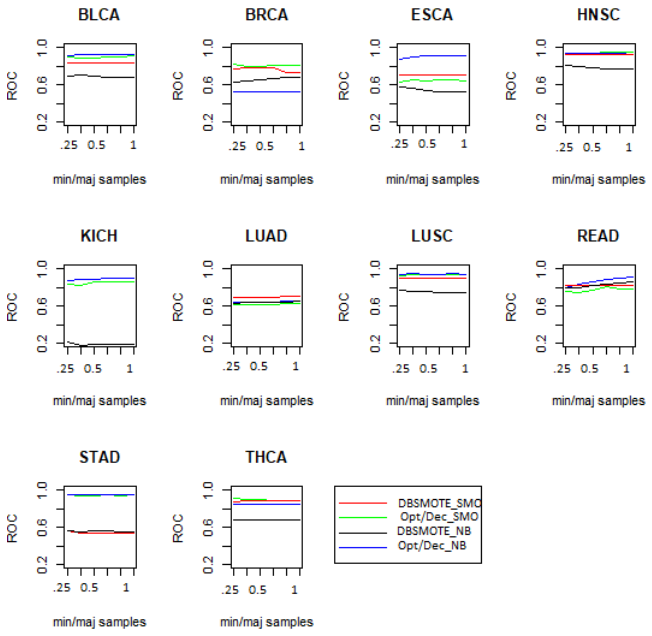


Figure 5.6: Experimental results for AUROC evaluated by classifying the (500D) DBSMOTE and Opt/Decoder representations using SMO and naïve Bayes.

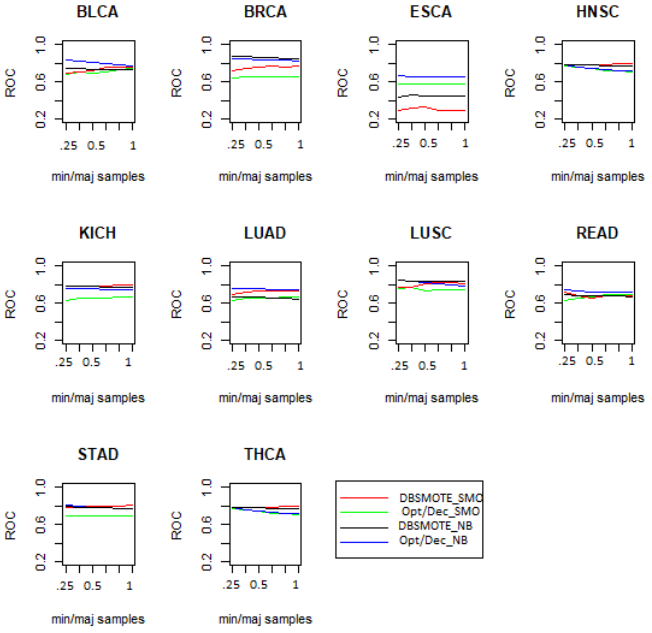


Figure 5.7: Experimental results for AUROC evaluated by classifying the (30D) DBSMOTE and Opt/Decoder representations using SMO and naïve Bayes.

5.7 Further Discussion and Analysis

In this section, we will present more empirical results comparing the performance of the method introduced in this chapter with other recently used over-sampling techniques.

To generate the optimized minority class representations, we used the open source python library pyswarm which takes the following parameters: particle velocity scalar factor w, $c1$ and $c2$ scalar factors to balance the contribution of the p_{best} and p_g respectively. r_1 and r_2 are randomely generated values from $[0, 1]$ range and a maximum velocity term v_{max} in order to limit oscillation which will allow for re-writing the velocity update for iteration $t + 1$ given by equation in Equation 5.1 according to the following:

$$v_i^{t+1} = wv_i^t + c_1 r_1 \times (p_i - y_i^t) + c_{2r2}(p_g - y_i^t). \tag{5.3}$$

Table 5.3 presents the parameters' settings of our experiments which were constant across the used datasets. The PSO parameters have been used before in [Van den Bergh, 2001]. However, we tried different number of iterations to make a decision on the best choice.

Making decision on the autoencoder's parameters is not an easy task. Nevertheless, we noticed in the survey presented in Chapter 4 that shallow networks, which consists of 1 or 3 hidden layers, are the most frequently used type of autoencoders, for this reason we decided to run our experiments using 2 network architectures that differ in the size of the middle layer to test the ability of the model to generate discriminating representations with different lengths. The

Table 5.3: Parameter Settings for the Autoencoder and PSO

Method	parameter	value
Autoencoder	learning rate	0.001
	optimizer	SGD
	activation	sigmoid
	noising	Gaussian noise
	batch size	25
	epochs	250
	layers	1000,500,1000 1000,30,1000
PSO	c_1	1.49618
	$c2$	1.49618
	maximum velocity	6
	w	0.729844
	Swarm size	200
	iteration	1 2 3 5 10 20

learning rate was set to 0.001 based on a set of experiments we ran and presented in Figure 5.8.

As can be noted from Table 5.4 logistic regression classifier was better able to discriminate the Opt/Decoder synthetic representations than other methods' synthetic representations. A better performance was noted for Opt/Decoders' representations compared to random duplication, SMOTE and KMeansSMOTE methods (wins in 10 and similar in 3 cases). It won the competition in 8 cases and had an equal performance to BorderLineSMOTE in 8 cases out of 20. Opt/Decoder is related to random duplication in that it treats the randomly generated representations as candidate examples but differs in some other aspects as it uses the similarity criterion as a condition on the validity of the candidates such that the closest candidate is ended up to be chosen as a synthetic examples. This

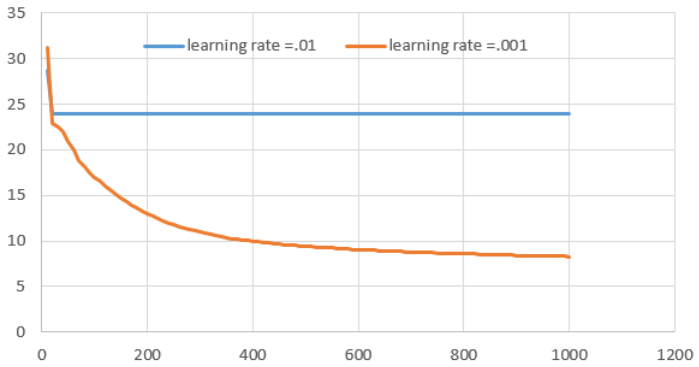


Figure 5.8: Comparison between the decrease in the loss function for two different learning rate values

similarity-based selection criterion explains why our method outperforms better than the random duplication method in some cases.

5.8 Chapter Summary

In this chapter, we presented a new synthetic oversampling technique which can be used in cases where the dataset is high dimensional and autoencoders is the adopted dimensionality reduction method. In this case, an optimization method such as the PSO can be utilized to return optimized solutions that can be mapped, using the decoder, to high dimensional examples which are similar to the original examples. The generated optimized low dimensional representations are found to be discriminating and can be used to solve the class imbalance related problems.

This method showed comparable, if not better, performance to other recently used methods in many test cases included datasets exhibiting different characteristics which indicates that the method is applicable to real world problems and can be found as a good alternative to existing solutions.

Table 5.4: Comparing the performance of PSO/Decoder, Random duplication, SMOTE, BorderLineSMOTE, KmeansSMOTE using the AUROC of random forest, SMO, naïve Bayes and logistic regression

Dataset	method	random forest	SMO	naïve Bayes	logistic reg
Leukemia	Opt/Decoder	**0.863** $\pm$ 0.186	0.500 $\pm$ 0.000	0.500 $\pm$ 0.000	0.858 $\pm$ 0.135
	Random	0.795 $\pm$ 0.177	0.585 $\pm$ 0.225	0.614 $\pm$ 0.205	0.858 $\pm$ 0.099
	SMOTE	0.833 $\pm$ 0.129	**0.660** $\pm$ 0.234	**0.673** $\pm$ 0.236	0.858 $\pm$ 0.099
	BLineS	0.820 $\pm$ 0.142	0.523 $\pm$ 0.233	0.598 $\pm$ 0.251	0.849 $\pm$ 0.097
	KmeansS	0.807 $\pm$ 0.159	0.603 $\pm$ 0.186	0.703 $\pm$ 0.201	**0.874** $\pm$ 0.108
GLI	Opt/Decoder	**0.799** $\pm$ 0.158 •	0.500 $\pm$ 0.000	0.544 $\pm$ 0.158	**0.769** $\pm$ 0.162
	Random	0.770 $\pm$ 0.138	0.727 $\pm$ 0.203	**0.782** $\pm$ 0.185	0.747 $\pm$ 0.149
	SMOTE	0.770 $\pm$ 0.138	**0.781** $\pm$ 0.200	0.679 $\pm$ 0.253	0.761 $\pm$ 0.165
	BLineS	0.770 $\pm$ 0.138	0.713 $\pm$ 0.177	0.714 $\pm$ 0.203	0.747 $\pm$ 0.149
	KmeansS	**0.799** $\pm$ 0.146 •	0.721 $\pm$ 0.193	0.653 $\pm$ 0.263	0.739 $\pm$ 0.135
Prostate-GE	Opt/Decoder	0.818 $\pm$ 0.101	**0.554** $\pm$ 0.099	**0.554** $\pm$ 0.000	**0.803** $\pm$ 0.120
	Random	0.798 $\pm$ 0.108	**0.554** $\pm$ 0.099	**0.554** $\pm$ 0.099	0.795 $\pm$ 0.158
	SMOTE	0.828 $\pm$ 0.119	**0.554** $\pm$ 0.099	**0.554** $\pm$ 0.099	0.795 $\pm$ 0.147
	BLineS	**0.852** $\pm$ 0.117	**0.554** $\pm$ 0.099	**0.554** $\pm$ 0.099	**0.803** $\pm$ 0.156
	KmeansS	0.817 $\pm$ 0.108	**0.554** $\pm$ 0.099	**0.554** $\pm$ 0.099	0.789 $\pm$ 0.145
SMK-CAN	Opt/Decoder	0.693 $\pm$ 0.111	**0.601** $\pm$ 0.082	0.531 $\pm$ 0.047	**0.733** $\pm$ 0.107
	Random	0.705 $\pm$ 0.108	0.593 $\pm$ 0.079	**0.604** $\pm$ 0.085	0.710 $\pm$ 0.107
	SMOTE	0.700 $\pm$ 0.109	0.595 $\pm$ 0.082	**0.604** $\pm$ 0.085	0.710 $\pm$ 0.118
	BLineS	**0.735** $\pm$ 0.079	0.593 $\pm$ 0.079	**0.604** $\pm$ 0.085	0.679 $\pm$ 0.123
	KmeansS	0.716 $\pm$ 0.100	0.601 $\pm$ 0.093	**0.604** $\pm$ 0.085	0.731 $\pm$ 0.108
ALLAML	Opt/Decoder	**0.846** $\pm$ 0.104	**0.767** $\pm$ 0.079	**0.788** $\pm$ 0.132	**0.879** $\pm$ 0.093
	Random	0.835 $\pm$ 0.085	0.699 $\pm$ 0.121	**0.788** $\pm$ 0.083	0.876 $\pm$ 0.086
	SMOTE	0.831 $\pm$ 0.103	0.699 $\pm$ 0.121	0.780 $\pm$ 0.036	0.855 $\pm$ 0.105
	BLineS	0.830 $\pm$ 0.084	0.643 $\pm$ 0.118	0.643 $\pm$ 0.162	0.848 $\pm$ 0.083
	KmeansS	0.826 $\pm$ 0.095	0.621 $\pm$ 0.072	0.791 $\pm$ 0.087	0.843 $\pm$ 0.121

Chapter 6

RBFA: Radial Basis Function Autoencoders

Multi layer perceptron networks effectively learn good representations of the data under some conditions including the availability of large amount of data; a requirement that is not applied to all domains [Srivastava *et al.*, 2014]. The autoencoder is a well known example to this type of networks which is distinguished with its architecture and unsupervised learning algorithm.

RBF networks, on the other hand, are three-layered networks known with their fast efficient supervised learning capability from small sized datasets. RBF networks consist of center points arranged in the hidden layer of the network in a parallel form. Each of these points employs a radial function to interpolate the input features before activating the output neurons. The mechanism which was proposed by [Park & Sandberg, 1991] allows locality of response to similar patterns of the data, such that only a fraction of the neurons will respond with a significantly large activation value at a time. The level of activation is a function of the similarity between the received input example and the center points.

We combined the architecture of both normal autoencoders and the RBF

networks to build a new model with parallel layers in the first hidden level of the autoencoder. The aim was to apply the radial activation mechanism to networks learning in unsupervised fashion so that they inherit the RBF networks capability in learning from small sized data. Similar to RBF networks, this activation technique, which is applied on the element wise basis in our network, forces the activation level of the neurons to be relative to the similarity between the input data points and pre-defined center points. There are two basic parameters associated with the center points which are: the centroids μ and the width σ which will be defined using the means and the variance the samples assigned to clusters generated by a clustering algorithm such as the Kmeans algorithm.

The proposed model was tested on two domain application problems: reconstruction and classification. RBFA was able to reconstruct the MNIST images with a comparable visual quality to normal autoencoder, normal autoencoder with dropout and the SDA. The representations generated for the MNIST images were better classified using the random forest classifier than the other prementioned autoencoders' representations.

Finally, we tested the model's ability to learn from small sized high dimensional gene expression datasets. Random forest, SMO, naïve Bayes and logistic regression classifiers show that the representations generated by the RBFA outperformed the SDA's representations in many test cases.

The paper presented in this chapter has also introduced a new PSO-based method for initializing the center and the width (parameters) of the radial center points used in the RBF networks. These optimized center point parameters allowed for learning discriminating representations when used with batch-normalization and regularisation techniques.

RBFA: Radial Basis Function Autoencoders

Maisa Daoud *mtdd1@students.waikato.ac.nz*
Michael Mayo *michael.mayo@waikato.ac.nz*
Sally Jo Cunningham *Sallyjo@waikato.ac.nz*
Department of Computer Science
University of Waikato, Hamilton. NZ

Abstract

We are introducing a new variation of the existing autoencoder called Radial Basis Function Autoencoders (RBFA). This version employs radial symmetric functions, in the first step of encoding, to map the input data vectors into a new form. The transformation which relies on the similarity between the input data examples and predefined center points features forces the neurons in the first hidden level to respond similarly to similar patterns of the data. This basic idea of radial transformation (activation) can be interpreted as regularization on the hidden neurons which are penalized based on the similarity criterion so only a fraction of them will become highly activated at one time. The paper also introduces a new method for defining the Gaussian radial function's parameters using Particle Swarm Optimization (PSO). Results indicate that the accuracy of four different classifiers was significantly affected by the number of layers in the first hidden level of the network. RBFA has shown promising results compared to other state of the art autoencoders using different datasets including high-dimensional small-sized medical data. Optimizing the radial function parameters improved the classification accuracy of the four classifiers in some cases.

Keywords: RBF networks, autoencoders, gene expressions, PSO

6.1 Introduction

Autoencoders, with all of their variations, are classical examples of the unsupervised learning-based feed forward neural networks. They learn the characteristics of the data through a simple encoding-decoding process that runs for a set of iterations to optimize a reconstruction-based objective function. Autoencoders have a hierarchical structure of layers activated by simple yet nonlinear functions to learn representations that amplify aspects of the data which are found in practice to be useful in different applications such as reconstruction and classification. Consequently, the ultimate goal of training the autoencoders is to capture "good" and "robust" representations that are able to distinguish or reconstruct every class of examples. Generally and from a mathematical point of view, the goal of training the autoencoders is to find a function or an approximation to a function that can discover the input-output mappings.

Multi-Layer Perceptrons (MLP), including autoencoders, achieve remarkable results in learning good representations. However, training the deep models usually requires a long running time and under some conditions e.g a limited number of training examples, it is sometimes hard to detect the relationships that were found between the noise but not in the actual real world test data which cause *overfitting* [Srivastava *et al.*, 2014]. Moreover, the decision on the number of the kernel functions (layers) and elements (neurons) per layer is a crucial challenge facing the MLP networks designers [Panchal *et al.*, 2011].

RBF networks are simple feed forward neural networks known for their fast learning capability from small-sized data [Sug, 2009]. The least squared error criterion is the most commonly used objective function in these networks which learn in a supervised manner. RBF networks are well-known for their simple three-layered architecture and providing a comparable performance to deep networks in approximating functions of a finite number of real variables [Clavería González

et al., 2015; Crowther *et al.*, 2004; Park & Sandberg, 1991]. The middle level of the RBF network consists of several nodes, each of which is defined with a centroid and a smoothing factor (or width) vectors which are the well known used parameters for most radial kernal activation functions. Each hidden node interpolates the high dimensional input vector to generate approximation of a continuous function e.g. the probability density in the case of using a Gaussian function.

In this paper, we are introducing a new variation of the existing normal autoencoder called RBFA. This version learns using a similar mechanism to RBF networks but in a complete unsupervised fashion. The first hidden level of the network consists of multiple layers ordered in a parallel form. Each of these layers receives a transformed version of the original input data which controls the neurons level of activation. The transformation is carried out on the element-wise basis using a radial function taking the input data examples and predefined vectors (centeroids and widths) as parameters. The proposed RBFA is similar to RBF networks in that each neuron's level of response is relative to the degree of similarity between the input data and the radial center point features, thus allowing a locality of response to similar patterns of the data.

Despite their simple architectures, RBF based networks, including the RBFA, come with the problem of defining the centers and the widths of the radials. This paper will introduce a new technique for generating optimized center points and width values using the PSO algorithm. The contributions of this paper are: (i) we introduce the RBFA and show that it can learn discriminative data representations; (ii) we introduce a new technique for defining the radial function parameters using the PSO algorithm; (iii) we show that the RBFA is able to reconstruct images with a comparable quality to other state of the art autoencoders; (iv) we experimentally demonstrate how the increase in the number of

hidden layers in the first hidden level of the architecture affect the accuracy of four different classifiers and (v) we experimentally test the RBFA using large-sized and small-sized high-dimensional medical data under different conditions.

This paper is organized as follows: The next section presents the motivation. Section 6.3 introduces a background on basic autoencoders, RBF networks, and presents examples for state of the art versions. Section 6.4 explains the proposed autoencoder. Section 6.5 presents the experimental results and Section 6.6 concludes this work and suggests directions for further research.

6.2 Motivation

MLP networks, including deep autoencoders, have proved proficient in learning good representations from sufficient amount of data [Srivastava *et al.*, 2014] which is not available by all domain-application problems. Radial Basis Function (RBF) networks, on the other hand, proved a comparable proficiency in learning good representations with fast learning capacity even from small-sized data [Sug, 2009]. We were motivated by the idea of combining the capabilities of both networks using a new autoencoder which bridges the gap between both networks.

The suggested RBFA is distinguished from other autoencoders by employing a radially-symmetric function in the first step of encoding to interpolate the input features on an element-wise basis. This idea of employing a radial function such as Gaussian on the element-wise basis was used before by Qian *et al.* [2015] in their Gaussian-Neuron Convolutional Neural Network (GNCNN) which was designed for steganalysis purpose. However, our work differs from that by considering the similarity between the raw input data and the predefined center points feature as an integral parameter to the radial function as opposed to applying the radial activation directly on the raw input data as in [Qian *et al.*, 2015]. The associ-

ation between the input data the center points features in RBF networks was first introduced by [Moody & Darken, 1989] as a new technique for allowing locality response of the network's neurons based on a distance function such that "Only fraction of processing units which centers close to input will respond with activations which differs significantly from zeros" [Moody & Darken, 1989]. In our case, the activation of a neuron ω in the first hidden level of the RBFA will reach its maxima when it receives a signal from a radial point with centroid μ very close to the input x i.e. $(x - \mu)^2 \approx 0$. This technique can be interpreted as regularization on the hidden nodes by forcing neurons to strongly respond based on the similarity criterion.

6.3 Background

This section introduces the basic structure of the autoencoders, RBF networks and their learning procedure.

6.3.1 The Basic Autoencoder

The autoencoder is a fully connected feed forward network learning input data representations by optimizing a reconstruction criterion. Starting with feeding the raw input data instances through the input layer, the activations of the neurons are calculated at every layer and passed through to the layer in the next level of the hierarchy. The middle layer of the autoencoder is commonly used to learn low dimensional representations of the data, hence, it is usually defined with the lowest number of neurons in most autoencoders. The autoencoder maps an input data vector $\vec{x} =< x_1, x_2, ..., x_D >$ with D features to a hidden representation

using the following equation:

$$y_i = \phi(\omega_i y_{i-1} + \vec{b}_i) \tag{6.1}$$

where ϕ is an activation function e.g tanh, sigmoid or ReLU, w is weight matrix for layer i, b is its bias vector and at $i = 0$ $y_0 = x$.

As clarified by Equation 6.1, the representations are mapped at the network layers such that the output of every layer is used as an input to the layer in the next level until a high dimensional output vector is generated at the output layer. The difference between the original input and the output vectors is computed while the coefficients are backpropagated, using a backpropagation algorithm, to update the network's weights for optimal reconstruction. The algorithm iterates over all the training examples for a number of epochs to find an approximation to a function that efficiently generates high quality reconstructed versions.

Improved variations of the basic autoencoder are differentiated with the weight penalties (regularization) added to the reconstruction function, the employment of the drop out technique, and the amount of noise added to the data. Stacked Denoising Autoencoder (SDA) [Vincent *et al.*, 2010b], for example, is trained to denoise corrupted versions of the input. Contractive autoencoders [Rifai *et al.*, 2011] add a penalty on the cost objective function to encourage the intermediate representation to be robust to small changes in the input. K-sparse autoencoders [Makhzani & Frey, 2013] employ a similar technique to the dropout mechanism in principle but it only keeps the highest k activations per layer and sets the rest of the neurons to zeros.

6.3.2 Radial Basis Function Networks

RBF networks are feed forward neural networks commonly known for their simple three-level architecture and fast learning capacity. RBF networks learn in a supervised manner to approximate a target function using a linear combination of radial kernels. The network can be simply formulated using an $N \times D$ dataset with N training examples each of which has D features, represented as $\vec{x} =< x_1, x_2, ..., x_D >$ and labeled with a class label y. The first hidden level of the network is structured using K parallel layers activated using a radially symmetric function. The input-output mapping in a simple RBF network can be calculated according to the following equation:

$$y = \sum_{i=1}^{K} \omega_i \varphi_i(\vec{x}) \tag{6.2}$$

where ω_i is the weight matrix for layer i in the first hidden level and φ_i is a kernel radial function, e.g. the Gaussian kernel, which estimates the probability density for an input signal and can be defined according to the following equation:

$$\varphi_i(\vec{x}) = exp(-\frac{||\vec{x} - \vec{\mu}_i||^2}{2\vec{\sigma}_i^2}) \tag{6.3}$$

such that, $||\vec{x} - \vec{\mu}_i||$ is the Euclidean norm for the distance between input vector $\vec{x}$ and center point $\vec{\mu}_i$. $\vec{\sigma}_i^2$ is the width of the radial function.

RBF network centers are a set of hidden layers (called kernel nodes) with the same cardinality as the input data and usually initialized using a number of arbitrarily chosen training examples or the centroids generated by a clustering mechanism such as the Kmeans algorithm. However, these networks can easily overfit the training examples [Que & Belkin, 2016]. Penalizing the network's coefficients using a regularization mechanism such as the ℓ_2 regualizer is a suggested

technique to overcome this problem [Que & Belkin, 2016].

For smooth interpolation and more meaningful extrpolation, Tao [1993] suggested normalizing the network such that the approximation will depend on a smoothed nearest neighbor function. Different normalization methods have been introduced in literature [Grabec, 2007; Tao, 1993; Xu, 2010] so far. However, we are adopting the method suggested in [Moody & Darken, 1989] and used in [Xu, 2010] which can be formulated, using the Gaussian kernel function, according to the following equation:

$$\varphi_i(\vec{x}) = \frac{exp(-\frac{||\vec{x}-\vec{\mu_i}||^2}{2\vec{\sigma_i^2}})}{\sum_{j=1}^{k} exp(-\frac{||\vec{x}-\vec{\mu_j}||^2}{2\vec{\sigma_j^2}})} \tag{6.4}$$

The width of the radial basis functions σ_i is a critical parameter for determining the value of the maximum distance between the input data and the center points and so it needs to be accurately estimated. In traditional Gaussian probability density estimation functions, this parameter represents the variance of the data examples assigned to a particular cluster, hence, it will be hard to utilize it with the RBF networks unless it is sufficiently large. Lee *et al.* [1999] addressed this problem which makes the Gaussian function in-applicable in cases where datasets have constant or nearly constant values in some intervals making the variance approaches to zeros and [Karayiannis & Randolph-Gips, 2003] suggested an alternative radial function called Cosine based activation (defined by Eq. 6.5) to facilitate the training based on gradient descent:

$$\varphi_i(x) = \frac{a_i}{(||x_i - \mu_i||^2 + a_i^2)^{\frac{1}{2}}} \tag{6.5}$$

such that $a_i \in \mathbb{R} - \{0\}$ is a parameter controlling the effectiveness of the similarity measure between the input vectors and the center points.

6.4 Radial Basis Function Autoencoders

The suggested RBFA basically combines the structure of both: autoencoders and RBF networks in one architecture. Intuitively, the model employs the centers concept from the RBF networks where input features are activated by radially symmetric functions based on their distances from predefined center points. This radial transformation to the input vectors will force neurons' level of responses to be a function of the similarity measure.

To train the RBFA, consider an $N \times D$ dataset with unlabeled examples and a network of three levels consisting of an *input layer*, K *hidden layers* arranged in parallel, and an *output layer* which weights are initialized to small numbers using a proper initialization method such as Xavier method [Glorot & Bengio, 2010b]. Note that a layer in the first hidden level of an RBFA has a **radial center** for the radial transformation purpose and a **weight matrix** ω of initialized neurons for the dimentionality mapping purpose.

The training phase starts with **clustering** the dataset into K groups, each of which will have a centroid denoted with μ_i and a width σ_i representing the centroid and variance of the data features assigned to cluster i respectively. Both μ_i and σ_i will be used as parameters for the radial function applied at the center points. However, there are other methods suggested in literature and can be used for defining these parameters.

The **Encoding** phase is similar to training an RBF network where the input data are fed through every center point i in the first hidden level to apply the radial transformation on the Euclidean distance between input vector x and every μ_i. The transformed inputs are passed through a linear mapping function using weight matrix ω_i after that to generate new representations. The algorithm combines the representations generated by the hidden layers for every input vector x

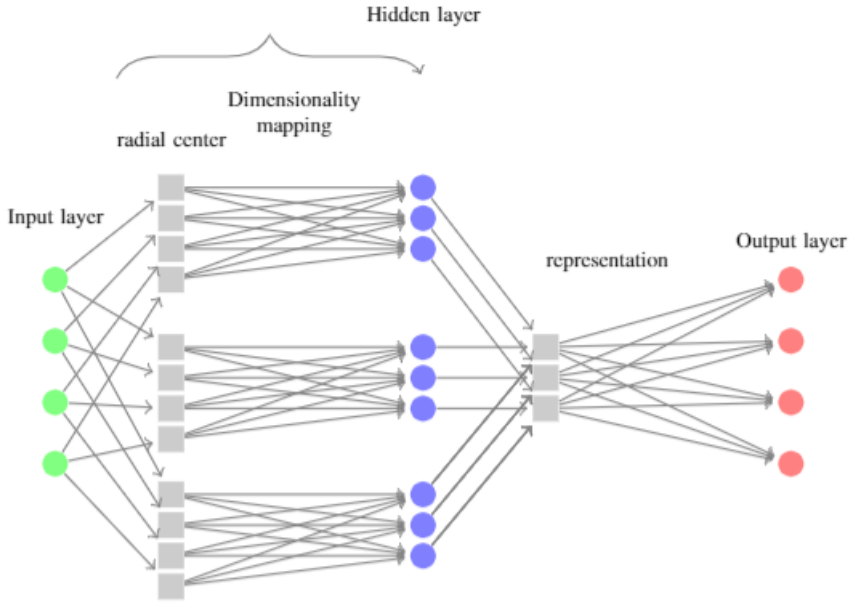


Figure 6.1: Simple RBFA Architecture

according to the following equation:

$$\begin{aligned} y_1(\vec{x}) &= \sum_{i=1}^{K} w_i.\varphi_i(x) \\ \varphi_i(x) &= \frac{\exp(-\frac{(x-\mu_i)^2}{2\sigma_i^2})}{\sum_{j=1}^{K}\exp(-\frac{(x-\mu_j)^2}{2\sigma_j^2})} \end{aligned} \tag{6.6}$$

where $\varphi_i(x)$ is a normalized Gaussian radial function applied on the element-wise basis.

We can mathematically reformulate the calculations done at one layer i from the first hidden level using the input vector $x =< x_1, x_2, .., x_D >$, passing through the radial transformation function to generate $\varphi_i(x)= x' =< x'_1, x'_2, ..., x'_D >$ and to the weight matrix ω_i after that to be mapped into a different dimension M

according to the following equation:

$$\begin{bmatrix} w_{1,1} & w_{1,2} & . & . & w_{1,D} \\ w_{2,1} & . & & & \\ . & & . & & \\ . & & & . & \\ w_{M,1} & & & & w_{M,D} \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \\ . \\ . \\ x'_D \end{bmatrix} \tag{6.7}$$

The **Decoding** phase, however, is as simple as in normal autoencoder where we pass the combined representations through the decoder's levels to calculate the neurons activations using a nonlinear function such as the one described in Equation. 6.1 and to generate a reconstructed version at the output layer. The network is trained to minimize a predefined objective function such as the mean squared error, an optimization e.g. Adam Optimizer Kingma & Ba [2014] and backpropagation algorithms are used to optimize and backpropagate the error derivatives respectively.

Figure 7.2 demonstrates the concepts using a simple RBFA architecture with only one hidden level of three parallel layers $K = 3$, each of which receives an input signal of 4 features $D = 4$, transforms it using the radial functions (square boxes) and maps it using the weigh matrix into a new representation of length $M = 3$ (represented with the circle nodes). The output of the layers are combined and passed to the output layer to generate a reconstructed version of the input with 4 features.

In the **testing** phase, an input example can be fed through the network to calculate the activations of the neurons across the levels to generate data representations at the middle layer and reconstructed versions at the output layer.

6.4.1 Optimizing the Parameters Using the PSO

RBFA has a simple architecture but it comes, as other RBF networks, with the problems of specifying the number of layers K in the first hidden level, sitting the centers μ and defining the width σ of the radials. We are introducing here a new PSO-based technique for estimating both μ and σ parameters. The suggested method does not optimize the number of layers K but, instead, it will generate K optimized μ_i that maximizes the radial interpolation function (objective function) across all the data examples i.e. the algorithm searches the space for optimized centroids that are as close as possible to the training examples and guarantees maximizing the radial function.

The PSO [Poli *et al.*, 2007] is an optimization algorithm inspired by the social movement of the bird flocks and ideally used for optimizing continuous nonlinear functions. Due to the high-dimensionality and the limited size of some datasets which makes optimizing both μ and σ parameters at the same time to be very complex. We ran the PSO in two stages (i.e one stage for each parameter). The technique starts with generating optimized μ_i for each center point first and generating optimized σ_i based on the previously found optimized μ_i in the second stage.

We started with clustering the dataset into R clusters where $1 \leq R < N$ and using the resulted $R \times D$ centroids as an initial population to the PSO algorithm in which swarm size equals to R. This technique (which can be referred to with "seeding" [Eiben *et al.*, 2003]) is used to avoid the need for long running times which is required to get good solutions from randomly initialized particles.

The algorithm will search the space for a centroid point $\vec{\mu}_i$ that maximizes

the following interpolation function across the training data examples:

$$fitness = \sum_{j=1}^{N} exp(-\frac{||\vec{x_j} - \vec{\mu}_i||^2}{2\sigma_i}) \tag{6.8}$$

where $\sigma_i = 1$.

In each iteration of the algorithm, the fitness of the particles (potential solutions) is calculated and their positions are updated according to their best position found so far and the best position found by the whole swarm (see [Poli *et al.*, 2007] for the update equations). The PSO runs for a set of iterations and returns the best fitted individual which will be used used as a centriod. We re-run the PSO algorithm for k times to generate K optimized μ_i.

In the second stage, PSO was used to generate optimized σ_i based on the optimized μ_i found in stage 1. Similarly, Eq. (6.8) was used as an objective function such that the algorithm returns the best fitted σ_i vector which maximizes the radial kernel function with the constraint $\sigma_i^2 > 0$.

6.5 Experiments

We carried out a set of experiments on datasets with different characteristics to investigate the applicability of the suggested autoencoder in various domains.

6.5.1 Datasets

In the first group of experiments, we used the standard MNIST dataset to test the performance of the proposed autoencoder in reconstructing and classifying images from a large dataset of 60,000 training and 10,000 testing examples.

Secondly, we were interested in testing the ability of the RBFA to learn from small sized high-dimensional datasets. For this purpose, we used 5 benchmark

genomic medical datasets [1] as they satisfy the required criteria. In addition to the high dimensionality and the small-size characteristics, genomic data are usually imbalanced and have overlapped classes which make them very challenging to work with. Table 6.1 describes the used medical dataset names, reference, number of examples, number of features and the number of classes for each of the selected datasets.

Table 6.1: Biological data description

Dataset	reference	examples	features	classes
GLIOMA	Nutt *et al.* [2003]	50	4434	4
TOX-171	Ding & Peng [2005]	171	5748	4
Lung	Bhattacharjee *et al.* [2001]	203	3312	5
Prostate	Singh *et al.* [2002]	102	5966	2
AllAML	Armstrong *et al.* [2001]	72	7129	2

6.5.2 Testing the Extracted Representations

Starting with the MNIST dataset, we visually compared the performance of the RBFA and three other selected autoencoders: a normal autoencoder, SDA, and a normal autoencoder with the dropout technique (Normal+dropout) in reconstructing a set of test images. We used a three hidden layer architecture with [*input*, 1000, 250, 1000, *output*] neurons for the baseline autoencoders and 3 parallized hidden layers, each of which had 250 nodes, [*input*,[250, 250, 250], *output*], for our autoencoder. Due to the sparsity of MNIST data and the consistency of some features, we used the cosine activation given in Eq. 6.5 for our RBFA. The dataset was normalized and the Adam algorithm was adopted to optimize the mean squared error function with a learning decay equals to 0.001. Sigmoid activation function was employed to calculate the activations at the output layer

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

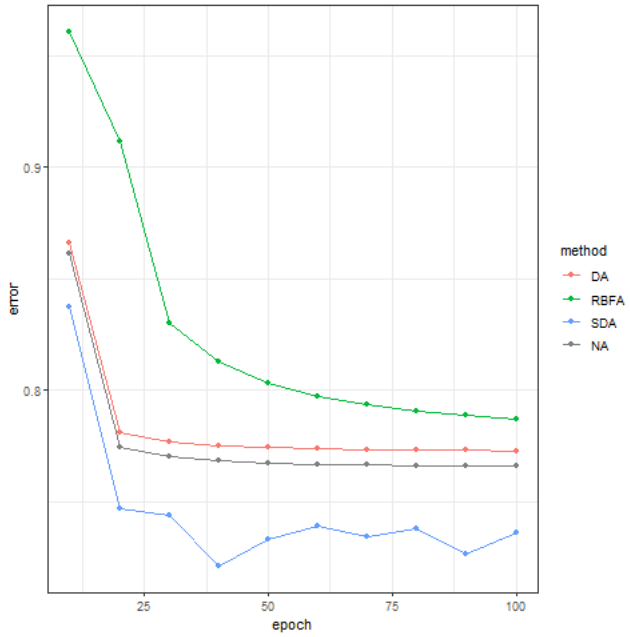
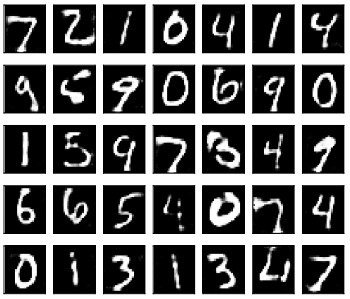


Figure 6.2: Training error for RBFA, Normal+Dropout Autoencoder (DA), Normal Autoencoder (NA), and SDA

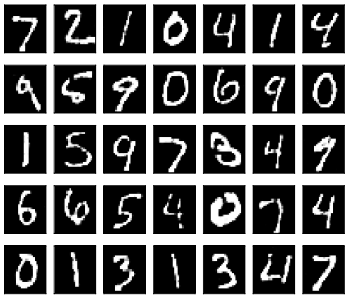
of all autoencoders. The `Tensorflow v.1.8` open source deep learning library was adopted to build a CPU implementation for the RBFA.

The graph in Figure 6.2 describes the relationship between the reconstruction error and the number of epochs for each of the tested autoencoder architectures. As shown by the plot, the cost function was gradually decreasing during the training phase in the RBFA case compared to other autoencoders which had more number of neurons. Nevertheless, the visual comparison in Figure 6.3 shows that the reconstructed images are qualitatively similar, indicating that the RBFA is able to learn features that reconstruct images with a comparable quality to other autoencoder variations.

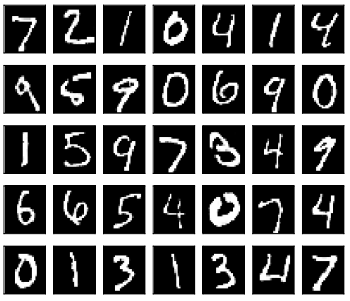
It is a common practice to test the discriminating ability of the autoencoder's learned representations using classification algorithms. There are two different approaches to do this which are: (i) using the learned weights for initializing



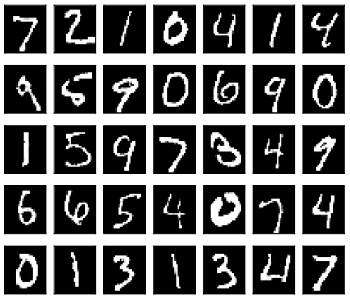
RBFA



Normal+Dropout Autoencoder



Normal Autoencoder



SDA

Figure 6.3: Visual comparison for the performance of RBFA, Normal+Dropaout Autoencoder, Normal Autoencoder, and SDA in reconstructing MNIST images

a deep learning classifier e.g. [Makhzani & Frey, 2013], (ii) extracting compact data representations and testing them using well-known machine learning classification algorithms e.g. [Vincent *et al.*, 2010b; Yousefi-Azar *et al.*, 2017]. To get a better degree of confidence on the quality of the learned representations, we adopted the latter approach which allows testing the representations using different classification algorithms.

Table 6.2 presents the accuracy of the random forest classifier in predicting MNIST representations under two different scenarios. Scenario 1 compares the efficiency of the representations generated by an RBFA ([*input*, [250,250, 250], *output*] with the ones generated by 3 base line autoencoders configured as [*input*, 1000, 250, 1000, *output*]; i.e all experimented networks have the same number of layers = 5. In scenario 2, the RBFA was configured with only 2 hidden layers [*input*, [250, 250], *output*] while the base lines were [input, 250, *output*]; i.e experimented networks have the same depth = 3. Results show that the RBFA representations were better classified using the random forest classifier compared to other representations in both scenarios. A significant improvement in the accuracy was noted with the increase in the number of layers in all tested networks.

Table 6.2: Comparing the accuracy of random forest classifier in discriminating each of the experimented autoencoders representations

Method	scenario1	scenario2
RBFA	**0.953**	**0.9381**
Normal Autoencoder	0.917	0.924
Normal + Dropout	0.935	0.914
SDA	0.922	0.924

6.5.3 The Effect of the Number of Layers in the First Hidden Level on the Accuracy of Different Classifiers

The performance of the RBFA rely on the number of layers in the first hidden level contributing to the final output; such that, a suitable number of layers in this level enables the network to learn interesting features which improve the classification accuracy and generate closer outputs as a result. For this group of experiments we used the MNIST dataset and four classifiers namely random forest, support vector machine trained using the Sequential Minimization Optimization (SMO) Platt [1998] algorithm, naïve Bayes and logistic regression with their default parameters from the `scikit-learn v0.20.2` python package.

With only one hidden layer, RBFA was able to learn some interesting local features but the classification accuracy of the four adopted classifiers was not high compared to other cases. However, a significant increase in the accuracy was noted with the increase in the number of hidden layers to {2,3,8,9,27} as indicated by Figure 6.4. The graph also shows that the SMO and the naïve Bayes had a slight improvement in their accuracy when the number of layers in the first hidden level increased from 8 to 9. Nonetheless, no significant improvement was noted when the number increased into 27 in all cases indicating that $K = 8$ was sufficiently enough to get good classification accuracy for the MNIST dataset using the four adopted classifiers.

6.5.4 Experiments on Medical Data

In these experiments, we were motivated by comparing the performance of the RBFA and the SDA under three different scenarios as follows: (i) normal case where no regularization nor batch normalization is used; (ii) Using ℓ_2 regularization and (iii) Using ℓ_2 regularizer and batch normalization. Results for these

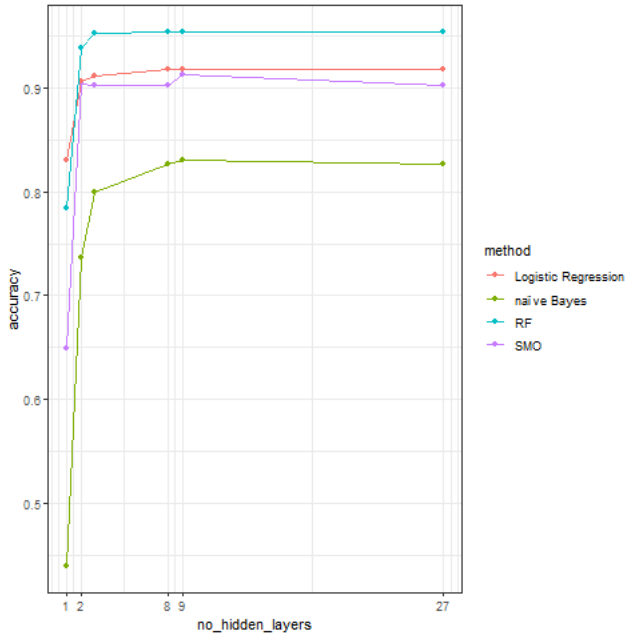


Figure 6.4: The effect of the increase in the number of layers in the first hidden level on the accuracy of random forest (RF), SMO, naïve Bayes and logistic regression classifiers.

three Table 6.3, Table 6.4 and Table 6.5 respectively. For fair comparision, we used the same learning parameters for the two experimented autoencoders as follows: the batch-size was 5, the learning rate 0.001, the number of epochs 500, the optimization algorithm was Adam Optimizer, and the mean squared error was the objective function. We tried two different architectures for the SDA as follows: [*input*,250,*output*] and [*input*,1000,250,1000,*output*]. However, the first architecture has generally outperformed the latter. Hence, we only presented it's results. With the RBFA, we used the Gaussian function for the radial activation, set $K = 2$ and $M = 250$ respectively to build [*input*,[250,250],*output*] network. Due to the limited number of examples in the medical datasets which affected the generalization ability of the baseline autoencoder, we chose to build relatively small-sized networks.

We ran the experiments for 10 times per dataset. In each run, the data was

randomly split into 90% training and 10% testing groups. The autoencoders were trained using the training split for 500 epochs. The representations were generated using the trained encoders for both of the training and the testing parts in the testing phase. We trained the classifiers using the representations generated by the training part of the data and tested them using the testing representations. We used the avg accuracy to compare the performance of the classifiers in detecting the datasets classes.

Generally, the RBFA representations were able to increase the adopted classifiers performance compared to the SDA in many test cases. Table 6.3 indicates that the SMO, naïve Bayes and logistic regression classifiers were respectively able to detect the RBFA representations and consequently achieving higher accuracy scores than the SDA's. However, random forest classifier performed better with the SDA's representations.

Regularization improved the generated representations and consequently increased the accuracy in most cases as indicated by Table 6.4. RBFA representations outperformed the SDA in most test cases and the SMO classifier has shown a notable improvement compared to its performance in the previous experiments where no regularization was applied. The SMO was also able to discriminate the RBFA representations better than the SDA's representations under the same conditions.

Adding batch normalization had generally decreased the accuracy of the classifiers for both autoencoders compared to the previous experiments. However, SMO and logistic regression showed improved performance when used with the RBFA representations and RBFA has generally won the competition in many cases.

Finally, we experimented the suggested optimization method in generating the radial functions parameters and used an open source python library called

`pyswarm` [1] for this purpose. To generate two optimized μ_i , we clustered the dataset into 5 groups and used their centroids to initialize a swarm of 5 particles, the algorithm ran for 20 iterations to search the space for the best candidate which maximize the objective function in Equation 6.8. In the first stage, PSO algorithm was run for 2 times to generate optimized centroids for the two center points in the first hidden level.

To generate the optimized σ_i, we randomly initialized a swarm of 10 potential solutions and ran the PSO for 20 iterations. Again, the PSO was used for two rounds to generate two optimized σ_i. As shown by Tables 6.3, 6.4 and 6.5 optimizing the parameters caused overfitting and did not improve the performance of the classifiers in many cases of the first and the second scenarios. However, the optimization improved the classifiers performance in the third scenario.

6.6 Conclusion and Future works

This paper introduced a new variation of the existing classical autoencoder which proved efficiency in learning good representations for datasets from different application domains. The introduced autoencoder employs a radially symmetric function which imposes a relationship between the neurons level of response and the received input features. Such that, the activation of a neuron in the first level of the hierarchy reaches its maximum if the received input features are very close to the centroid feature associated to it. The response level of the same neuron will be decreased if the input and the centroid features are not close i.e the penalty on the weights is a function of the similarity between the input training example and the radial parameters. This mechanism proved efficiency in learning predictive features, can be interpreted as a regularization on the hidden nodes which allows

[1] `https://pythonhosted.org/pyswarm/`

locality of response to similar patterns of the data.

The paper has also introduced a new technique for defining the Gaussian radial function parameters using the PSO algorithm. The technique showed promising results especially in cases where both regularization and batch normalization were employed. Future works will focus on generating parameters that better improve the performance of the network.

RBF networks are commonly known with their three level architecture which was adopted for our experiments. However, the model can be stacked by adding more layers to the encoder and decoder parts which can be thoroughly investigated in future using large benchmark datasets. Future works can also include testing the combination of this technique with other regularization mechanisms such as drop out or k-sparse to penalize the layers in the deep levels. It would be also interesting to see the principle of clustering and RBF transformation applied to the first layer of other networks.

Table 6.3: Comparing the performance of RBFA and SDA using the F1-measure of of random fores (RF), SMO, naïve Bayes (NB) and logistic regression (LR) classifiers

Dataset name	SDA				RBFA				RBFA-optimized-centers			
	RF	SMO	NB	LR	RF	SMO	NB	LR	RF	SMO	NB	LR
GLIOMA	**0.814**	0.843	0.643	0.814	0.757	**0.886**	**0.757**	**0.914**	0.700	0.500	0.671	0.671
TOX-171	**0.774**	0.863	0.653	0.821	0.747	**0.895**	**0.668**	**0.868**	0.674	0.530	0.632	0.768
lung	**0.882**	**0.932**	0.768	**0.929**	0.868	0.909	**0.936**	0.823	**0.822**	0.791	0.909	0.914
Prostate-GE	0.845	0.836	0.709	0.864	**0.855**	**0.900**	**0.764**	**0.900**	0.818	0.818	0.627	0.855
AllAML	**0.913**	**0.963**	**0.838**	**0.975**	0.900	0.838	0.788	0.863	0.763	0.938	0.825	0.938

Table 6.4: Comparing the performance of RBFA and SDA when both regularized using ℓ_2 norm using the accuracy of the four classifiers

Dataset name	SDA				RBFA				RBFA-optimized-centers			
	RF	SMO	NB	LR	RF	SMO	NB	LR	RF	SMO	NB	LR
GLIOMA	0.771	0.857	0.743	0.843	**0.786**	**0.929**	0.714	**0.857**	**0.786**	0.514	**0.757**	0.514
TOX-171	**0.779**	0.895	0.642	0.858	0.753	**0.911**	**0.663**	**0.868**	0.647	0.636	0.616	0.784
lung	0.814	**0.945**	0.718	**0.945**	**0.859**	0.918	**0.923**	0.918	0.836	0.659	0.900	**0.945**
Prostate-GE	**0.836**	0.882	0.673	0.882	0.827	**0.891**	**0.800**	**0.900**	0.736	0.527	0.591	0.836
AllAML	0.900	**0.975**	0.850	**0.988**	**0.913**	0.913	**0.863**	0.925	0.800	0.525	0.763	0.938

Table 6.5: Comparing the performance of RBFA and SDA when both regularized using ℓ_2 norm and layers were batch normalized using the accuracy of the four classifier

Dataset name	SDA				RBFA				RBFA-optimized-centers			
	RF	SMO	NB	LR	RF	SMO	NB	LR	RF	SMO	NB	LR
GLIOMA	0.771	0.8143	0.686	0.857	**0.786**	**0.900**	**0.757**	**0.886**	0.771	0.843	0.629	0.843
TOX-171	0.747	0.868	0.711	0.837	0.789	0.905	**0.826**	**0.909**	**0.818**	**0.909**	0.727	**0.909**
lung	0.809	0.877	**0.914**	0.882	**0.823**	**0.936**	0.832	**0.936**	0.804	0.900	0.901	0.909
Prostate-GE	**0.809**	0.855	0.627	0.855	0.718	0.755	0.573	0.773	**0.809**	**0.882**	**0.664**	**0.873**
AllAML	**0.929**	0.893	0.750	0.911	0.750	0.911	0.714	0.911	0.850	**0.938**	**0.800**	**0.963**

6.7 Further Analysis and Discussion

As we mentioned earlier, the experiments undertaken in this chapter, which parameters settings are shown in Table 6.6 and Table 6.7, proved that the radial transformation technique improved the autoencoder's ability in learning from small sized datasets.

Table 6.6: Parameter settings for the experiments MNIST data

Method	parameter	value
RBFAutoencoder	learning rate	.001
	optimizer	Adam
	activation	sigmoid
	batch size	25
	layers	input, [250,250, 250],output input, [250, 250],output
Autoencoder	learning rate	.001
	optimizer	Adam
	activation	sigmoid
	noising	Gaussian noise
	batch size	25
	layers	input,1000, 250, 1000,output input, 250,output

The architecture proposed in the original paper activates the parallel hidden layer in an element wise base and combines the outputs of the hidden layers at the normalization layer in the middle hidden level of the network by taking their average values. In this section, we are interested in testing the performance of the network using an alternative approach. Specifically, we want to see if concating the parallel hidden level's outputs will affect the performance of the network. To do so, we will interpolate the difference between the input data and the center points and concate the resulted scalar output of each hidden layer at the middle level of the network. Finally, the network will use the output of this layer to activate the output layer.

Table 6.7: Parameter settings for the experiments gene expression data

Method	parameter	value
RBFAutoencoder	learning rate	.001
	optimizer	Adam
	activation	sigmoid
	iterations	100
	batch size	5
	layers	input,[250,250],output
Autoencoder	learning rate	.001
	optimizer	Adam
	activation	sigmoid
	iterations	100
	noising	Gaussian noise
	batch size	5
	layers	input, 250,output

Table 6.8 shows a comparison between the two assembling approaches in generating discriminating representations for the ALLAML dataset. Both networks were defined using the same parameters used before except that we set the number of center points to 2,3,5,7,10,25 to test the effect of changing this parameters' value on the quality of the generated representations.

The results presented in Table 6.8 shows the AUROC mean and standard deviation (obtained in one 10-fold cross validation run) for each of the random forest, SMO, naïve Bayes and logistic regression classifiers tested on the representations generated by RBFA and RBFA-concate methods. These results indicate that the RBFA with averaging generates better quality representations achieving higher AUROC values than the RBFA-concate strategy.

We have also done a set of other experiments to compare the performance of both RBFA and SDA in classifying gene expression exhibiting different characteristics from the ones used in the paper. The datasets used in this section are larger in size, having higher imbalance ratio and binary class outcomes. Table 6.9 shows the average AUROC and standard deviation obtained by running the ex-

Table 6.8: Comparing the quality of the representations generated by the averaging v.s. the concatenation techniques based on the AUROC of the random forest, SMO, naïve Bayes and logistic regression

No. clusters	method	random forest	SMO	naïve Bayes	logistic reg.
2	RBFA-concate	0.700 ± 0.195	**0.550** ± 0.150	0.517 ± 0.050	0.813 ± 0.107
	RBFA	**0.704** ± 0.120	0.500 ± 0.000	**0.754** ± 0.130	**0.842** ± 0.131
3	RBFA-concate	0.657 ± 0.245	0.500 ± 0.000	0.651 ± 0.229	0.767 ± 0.199
	RBFA	**0.740** ± 0.144	0.500 ± 0.000	**0.662** ± 0.119	**0.857** ± 0.105
5	RBFA-concate	**0.747** ± 0.166	0.500 ± 0.000	0.619 ± 0.228	0.738 ± 0.173
	RBFA	0.718 ± 0.164	**0.574** ± 0.159	**0.721** ± 0.135	**0.847** ± 0.142
7	RBFA-concate	**0.572** ± 0.114	0.500 ± 0.000	**0.754** ± 0.089	**0.743** ± 0.123
	RBFA	0.510 ± 0.116	**0.525** ± 0.075	0.553 ± 0.111	0.590 ± 0.207
10	RBFA-concate	**0.638** ± 0.068	0.500 ± 0.000	**0.726** ± 0.172	0.787 ± 0.211
	RBFA	**0.638** ± 0.094	**0.554** ± 0.087	0.644 ± 0.153	**0.809** ± 0.200
25	RBFA-concate	0.569 ± 0.064	0.500 ± 0.000	**0.617** ± 0.202	**0.794** ± 0.120
	RBFA	**0.657** ± 0.162	0.500 ± 0.000	0.529 ± 0.126	0.592 ± 0.120

periments using the 10-fold cross validation approach to generate representations of the data using the networks under study. The results indicate that the proposed activation layer significantly (indicated by bold dot) improved the AUROC value in 6 cases compared to the only two cases in which SDA had significant improvement. On the other hand, the logistic regression classifier was better able to classify the RBFA representations than the SDA's

Table 6.9: AUROC for random forest, SMO. naïve Bayes and logistic regression used wit hthe RBFA and SDA representations. The results present the mean and standard deviation for 10-fold cross validation run

Dataset	model	random forest	SMO	naïve Bayes	logostoc reg.
	RBFA	0.711 ± 0.191	0.500 ± 0.000	0.692 ± 0.196	**0.683** ± 0.191
HNSC	SDA	**0.814** ± 0.185	0.500 ± 0.000	**0.711** ± 0.190	0.600 ± 0.200
	RBFA	0.842 ± 0.127	0.500 ± 0.100	0.567 ± 0.133	**0.761** ± 0.037•
ESCA	SDA	0.842 ± 0.118	0.500 ± 0.000	**0.844** ± 0.116 •	0.645 ± 0.198
	RBFA	0.976 ± 0.021	**0.982** ± 0.011 •	0.914 ± 0.041	**0.977** ± 0.016 •
BRCA	SDA	**0.979** ± 0.031	0.684 ± 0.225	0.914 ± 0.040	0.914 ± 0.053
	RBFA	**0.971** ± 0.034 •	**0.820** ± 0.137 •	0.648 ± 0.060	**0.947** ± 0.038 •
STAD	SDA	0.899 ± 0.083	0.500 ± 0.000	**0.849** ± 0.104 •	0.857 ± 0.100

6.8 Chapter Summary

In this chapter, we have introduced the idea of combining the RBF network's activation mechanism with autoencdoers to allow for inheriting the RBF capabilities in learning from small sized datasets. This layer deterministically transforms the input data into a new form which is a function of the distance between a data example and predefined center points. We showed that the suggested network architecture enhanced the autoencoders' ability in learning discriminating representations from small sized high dimensional datasets. In addition, this transformation technique outperformed other well known random methods of transformation such as noising which is employed by the SDA and the dropout in learning from large and small sized datasets, and otherwise gives comparable results.

Chapter 7

Multi Modal Approach for Training Variational Autoencoders

VAE are probabilistic networks trained to maximize the evidence lower bound (elbo) of the data. The basic idea of training this model is to use the training examples (called observations) to learn the parameters of a distribution model that generates samples of the latent (hidden) representations. These samples are used as inputs to different probabilistic network (generative) that is used to generate new examples of the data. Despite their simplicity and strong theoretical properties, VAE have the problem of singularity which potentially prevents the algorithm from finding several clusters of the data and limits its generative ability to producing only one class of examples or a limited variation of examples belonging to different classes.

The paper presented in this chapter is inspired by previous works in literature [Bousquet *et al.*, 2017] which introduced an alternative optimization function to the commonly used elbo. Moreover, the model proposed here employs the radial

activation technique and the parallel hidden layer architecture, introduced by our previous work, to learn a distribution of multiple components that experimentally guarantees to generate various examples from different clusters of the data.

We experimentally tested the model using both artificial and real world datasets exhibiting different dimensionalities and sizes. The results proved the ability of the model to generate examples from different clusters of both small sized and large sized datasets. Moreover, the representations generated for the gene expression datasets, by this model, are discriminating and increased the classification accuracy of the naïve Bayes classifier.

Multi model Approach for Training Variational Autoencoders Using Small Sized Datasets

Maisa Daoud *mtdd1@students.waikato.ac.nz*
Michael Mayo *michael.mayo@waikato.ac.nz*
Eibe Frank *eibe@waikato.ac.nz*
Department of Computer Science
University of Waikato, Hamilton. NZ

Abstract

Variational autoencoders (VAE) are generative models maximizing the **e**vidence **l**ower **bo**und (elbo) to approximate the marginal likelihood of the data. Generating new examples of the data is one common purpose for training the VAE. However, does maximizing the elbo guarantee the similarity between the distribution of the generated examples and the unknown distribution of the original data? i.e. does maximizing the elbo guarantee finding a model that is able to generate examples from every cluster of the data? This research introduces a new multi-model framework for training the VAE to overcome the problem of singularity encountered by traditional VAE. Results on synthetic and two images datasets proved the ability of the model to generate examples from different clusters of the data. The undertaken experiments have been also applied to high dimensional small sized gene expression datasets which showed the ability of the model to generate separable discriminating representations for this kind of data.

Keywords: Variational autoencoder, mixture models, cancer-data

7.1 Introduction

Generative networks are graphical probabilistic models with a set of parameters trained to approximate the potentially complex unknown distribution of the data. VAE are well-known generative models trained under the assumption that there exists a continuous latent variable $z \in Z$ (i.e. hidden cause) controlling the generation process of every data point $x \in X$ [Dilokthanakul *et al.*, 2016]. The objective of training traditional VAE is to learn the network parameters that approximate the distribution of the data $P(X)$.

VAE-based models, which are commonly trained for density estimation task, maximize the log likelihood of the data or minimize the Kullback-Leibler divergence (elbo) [Theis *et al.*, 2016] as a standard training criteria, while, in fact, maximizing the likelihood function can lead to finding an inappropriate local maxima or encounter the presence of singularities where the learning algorithm can only find one of multiple clusters of the data (see Figure 7.1). VAE have other drawbacks. First, the modeled approximate distribution can differ significantly from the true posterior distribution [Kingma *et al.*, 2016]. Zhao *et al.* [2019] impute the problem to the elbo training objective which tends to overit the distribution of the training data. Secondly, VAEs ignore the potential multi modality fact in the data and use a distribution with a single component to model it.

Recent works on VAE focus on improving the quality of the approximated posterior [Nalisnick *et al.*, 2016] or changing the training optimization objectives [Li & Turner, 2016]. Bousquet *et al.* [2017] studied the generative modelling in terms of the optimal transport problem between the unknown distribution $P(X)$ and the approximated distribution model denoted with $P_G(Y)$ and parameterized with G where Y is generated (through sampling) as a function of Z, and Z is a function of X. They show that the squared Euclidean distance $c(X, Y)$ coincides

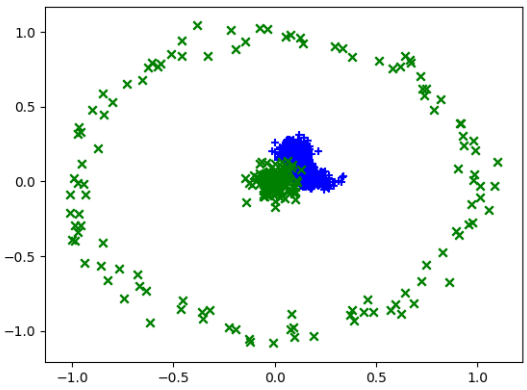


Figure 7.1: The problem of singularity encountered by maximizing the elbo. Green × represent the original data, the blue × represents samples generated from the distribution learned by the decoder.

with the objective of adversarial autoencoder that approximately minimizes the 2-Wasserstein distance $W_2(P(X), P_G(Y))$ and is a good alternative to maximizing the marginal likelihood of the data (which may lead to blurry results when applied to image data).

The model proposed in this paper is distinguished by its multi parallel hidden layer architecture in both parts of the autoencoder. It also employs a new objective function that considers the similarity between the generative network (decoder) parameters and pre-initialized/selected reference point features as an alternative to using the $\log P_G(X)$ or the error (distance) between Y and X. Our experiments on several domain application problems with datasets exhibiting different dimensionalities and sizes prove the ability of the model to generate various high dimensional examples from different regions of the clusters. The results also show that the model is able to generate discriminating representations for small sized high dimentional datasets competing other variational-based methods.

This paper is arranged as follows: The next section provides a basic theoretical background to the VAE. Section 7.3 presents our approach. Section 7.4 discusses

our experiemnetal results and section 7.5 concludes the work.

7.2 Background

This section presents a brief introduction to VAE and other recent related studies. We use the following notations through out this paper. Capital letters (e.g X) denote the variables and lower case letters (e.g, x) denot their values. A probability distribution will be denoted by capital letters (e.g, $P(X)$) and densities will be represented by lower case letters (e.g. $p(x)$). We will also use μ to indicate the mean (average) and $\mu(x_i)$ to indicate the mean parameter learned by the network for the input x_i; similarly, the variance will be denoted by $\sigma(x_i)$.

7.2.1 Variational Autoencoders

The general aim of training generative models such as the VAEs is to find an approximate distribution P_G to the unknown distribution $P(X)$ of the training data (observations) X. Finding $P(X)$ in real world application problems is not an easy task. Variational inference is a well-known solution which introduces Z as latent variables that are assumed to hold information about X and used to generate them. Based on this assumption, the density of the data can be mathematically defined by:

$$p(x) = \int p(x|z)p(z)dz \tag{7.1}$$

Maximizing $p(x)$ requires finding all possible values of Z that are likely to produce x which is intractable and computationally expensive. VAE maximize $p(x)$ by approximating the conditional distribution $P(Z|X)$, defined by Equation

7.2, using $Q(Z)$.

$$P(Z|X) = \frac{P(X|Z)P(Z)}{P(X)} \tag{7.2}$$

As can be seen from above, the posterior distribution $P(Z|X)$ requires $P(X)$. Hence, finding an arbitrary distribution $Q(Z)$ that approximates $P(Z|X)$ will help in solving the problem of approximating $P(X)$. To quantify the similarity between the approximated $Q(Z)$ and the target $P(Z|X)$, VAE use the Kullback-Leibler divergence (KL divergence) which can be defined according to the following equation:

$$D[Q(Z)||P(Z|X)] = E_{Q\sim Z}[\log Q(Z) - \log P(Z|X)] \tag{7.3}$$

By replacing $P(Z|X)$ on the right hand-side of the equation with Equation 7.2 the divergence becomes:

$$D[Q(z)||P(Z|X)] = E_{Q\sim Z}[\log Q(Z) - \log P(X|Z) - \log P(Z)] + \log P(X) \tag{7.4}$$

$P(X)$ was moved out of the expectation as it does not depend on Z. However, rather than using "any arbitrary distribution" for Q it made more sense to construct Q as a distribution that strictly depends on X. Denoting this distribution by $Q(Z|X)$, the divergence can be re-written as:

$$D[Q(Z|X)||P(Z|X)] = E_{Z\sim Q}[\log Q(Z|X) - \log P(X|Z) - \log P(Z)] + \log P(X) \tag{7.5}$$

by re-arranging and negating the equation in Eq.7.5 we get:

$$\log P(X) - D[Q(Z|X)||P(Z|X)] = E_{Z\sim Q}[\log P(X|Z) - KL[Q(Z|X)||P(Z)]] \tag{7.6}$$

The left-hand side of Equation 7.6 maximizes $P(X)$ and minimizes $Q(Z|X)||P(Z|X)$

at the same time but the intractable $P(Z|X)$ is still needed to be found. Variational inference is used here to maximizing the elbo of $P(X)$ according to the following equation:

$$P(X) \geqslant E_{Z\sim Q}[\log P(X|Z) - KL[Q(Z|X)|P(Z)]] \tag{7.7}$$

$P(X|Z)$, on the other hand, is a bit tricky to estimate as it does not only depend on the parameters of P but also the parameters of Q, so we get sample codes from Q that can be reliably decoded using the decoder part of the network. VAE amortizes the cost of inference by learning global parameters instead of learning different parameters for every observation.

Recent works on VAE focus on improving the posterior approximation by changing its form or changing the optimization function. Nalisnick *et al.* [2016] used a Gaussian mixture model (GMM) instead of the commonly used factorized Gaussian, but this model requires running the decoder for K times where K is the number of clusters. Theis *et al.* [2016] investigated different methods for evaluating generative models which are described as integral part in model's performance. Their work indicates that using the log-likelihood of the function given by Equation 7.8 will generate plausible images but poor log-likelihood values:

$$L = \frac{1}{N}\sum_{n} N(Y; X_n, \epsilon^2 I) \tag{7.8}$$

where X_n denotes either the training observations or a number of N observations derived from the training set, and ϵ is a Gaussian noise.

Shu *et al.* [2018] worked on improving the variational inference and reducing its computational cost by replacing the per sample approximation with an amortized distribution learned by an amortized inference model. This approach improved the generalization of the traditional model.

As mentioned earlier, Bousquet *et al.* [2017] studied the generative modelling in terms of optimal transport problem between the true unknown distribution $P(X)$ and the latent variable model distribution $P_G(Y)$ where Y is a function of Z i.e. $P(G(Z))$. Starting with the joint distribution $P(X, Y)$ where $P(X \sim P_X, Y \sim P_G)$ they show that the squared Euclidean distance coincides with the objective of adversarial autoencoder which approximately minimizes the 2-Wasserstein distance $W_2(P(X), P_G(Y))$ (Equation 7.9) and is a good alternative to maximizing the marginal-likelihood.

$$W_c(P_X, P_G) = inf_{P \in P_{X,Y}(X,Y) \sim P}[c(X, Y)] \tag{7.9}$$

where $c(X, Y)$ is any measurable cost function

7.3 Multi Modal Framework for VAE

In this model, we will assum that the observed data has been drawn from a multi modal distribution of K components each of which have the μ_k and σ_k parameters. Initial values for these parameters can be obtained by random selection to K data points, random assignment to K number of components or clustering the training examples into K clusters and assigning the centroid and the variance of the generated clusters into μ_k and σ_k^2 respectively. A network model (see Figure 7.2) is build to allow refining the K components simultaneously, this model can be represented by the following equations:

$$Z \sim N(0, I) \tag{7.10a}$$

$$Y|Z \sim \prod_{k=1}^{K} N(\mu_k(Z; \beta), diag(\sigma_k^2(Z; \beta))) \tag{7.10b}$$

where $\mu_k(Z;\beta)$ and $\sigma_k(Z;\beta)$ are the aggregated μ and σ parameters of the decoder given (learned) by the network and parametarized by β. However, the method does not require an accurate assumption around the distribution of the data: it relies on the proper selection to the μ_k and σ_k values which will allow for learning as close $\mu_k(Z;\beta)$ and $\sigma_k(Z;\beta)$ to the assigned μ_k and σ_k value as possible. Similarly, we here defined the generative network (decoder) as a network learning a mixture of Gaussians while, in fact, we can use a more relaxed application-dependant distribution for this network and deal with it as multiple batches of normal distributions $Y|Z \sim N(\mu_k(Z;\beta), diag(\sigma_k^2(Z;\beta)))$ instead. The main idea is to learn proper $\mu_k(Z;\beta)$ and $\sigma_k(Z;\beta)$ parameters of the decoder rather than the probability density of data.

Our network model, depicted by Figure 7.2, consists of an input layer, K parallel hidden layers and K output layers arranged in a parallel form and activated by a non-linear activation function (e.g, sigmoid function). We used the radial activation mechanism introduced in our s work [Daoud *et al.*, 2019] for modifying the input features to improve the neuron response to similar patterns of the data as a consequence. The model is trained using the same learning algorithm used for traditional VAE and uses Stochastic Gradient Descent (SGD) for back propagating the error derivatives. The input data is encoded by multiple hidden layers on the encoder side of the network. The encoder's outputs are used to generate the input codes for the decoder using the VAE re-parametrization trick, which generates at least one single code Z for every single data point i of X according to the following

$$\epsilon \in (0,1) \tag{7.11a}$$

$$z_i = \mu_i(x_i) + \sigma_i(x_i) * \epsilon \tag{7.11b}$$

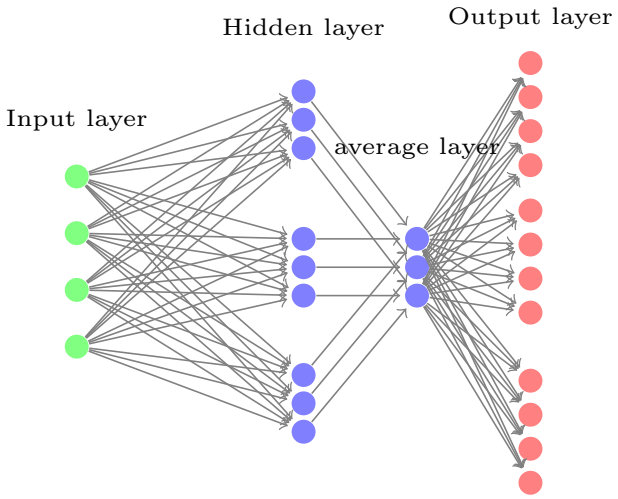


Figure 7.2: Mult imodel VAE consisting of an Input layer (4-D), 3 hidden layers (3-D), 1 average layer (3-D) and three output layers (4-D).

where μ_i and σ_i are functions learned for every data point i using the encoder layers. The encoder's samples activate the parallel layers of the decoder where the aggregated $\mu_k(Z)$ and $\sigma_k(Z)$ form the decoder's parameters.

We use the 2-Wasserstein distance function which is commonly used for measuring the distance between two probability distributions in optimal transport problem and defined by [Delon & Desolneux, 2019] according to Equation 7.12 to compute the difference between the reference point features and the decoders' parameters:

$$L = \frac{1}{K} \sum_{k}^{K} ||\mu_k(Z) - \mu_k||^2 + ||\sigma_k(Z) - \sigma_k|| \tag{7.12}$$

This objective function is used as an alternative to the Euclidean distance used to measure the error $c(P(X), P_G(Y))$ between the samples X and Y in [Bousquet *et al.*, 2017] so we can avoid making accurate assumption around the distribution learned by the model to generate $Y \sim X$.

Finally, to reduce the model's overfitting, a regularization technique was employed using the traditional VAE regularizer [Kingma & Welling, 2013] and [Shu

et al., 2018] $KL[Q(Z|X)|P(Z)]$ to make our objective function:

$$\mathcal{L} = L + KL[Q(Z|X)|P(Z)] \tag{7.13}$$

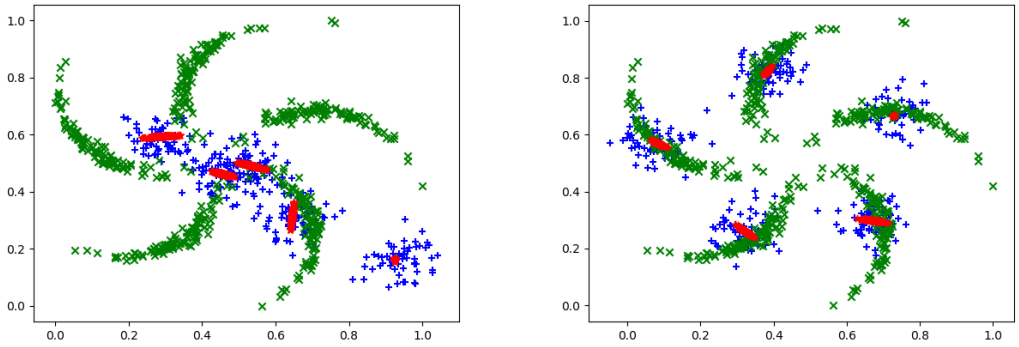
To generate new examples of the data at the testing phase, we can use the parameters learned by the decoder to build multi-models of normal distributions or a mixture of Gaussians with uniformally distributed components that we can sample from to generate $Y \sim X$ from random Z.

7.4 Experimental Results

This section presents our experimental results on three different categories of data. The aim was to test the applicability of the model on various types of domains including low dimensional small sized synthetic data, the MNIST data, relatively, low dimensional large sized dataset and high dimensional small sized cancer datasets.

7.4.1 Results on the Synthetic Datasets

We first tested the ability of our model in generating examples for three different synthetic two dimensional datasets. 1) The pinwheel dataset [Johnson *et al.*, 2016] with: N=500 and dimensionality D=2. 2) A dataset sampled from a mixture of three components defined with the following parameters: [[4.0,4.0],[5.0,5.0], [6.5,5]] for μ_k, [[[0.25, 0.21],[0.21,0.25]],[[0.25, -0.21],[-0.21,0.25]],[[0.25, .21],[.21,.25]]] for the covariance matrices and [.2,.5,.3] are the mixing coefficients. 3) The circle dataset generated using the scikit-learn python package with the following parameters: N=300, scale factor=0.15 between the circles, random-state=1 and noise=0.05.



(a) (b)

Figure 7.3: Results on pinwheel dataset: (a) original data (green), the samples (blue) and the $\mu_k(Z)$ (red) at initial state (b) results after 100 epochs

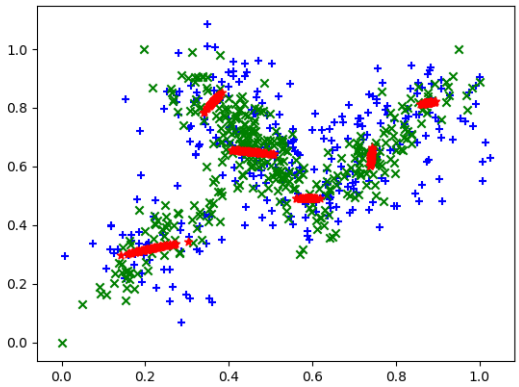
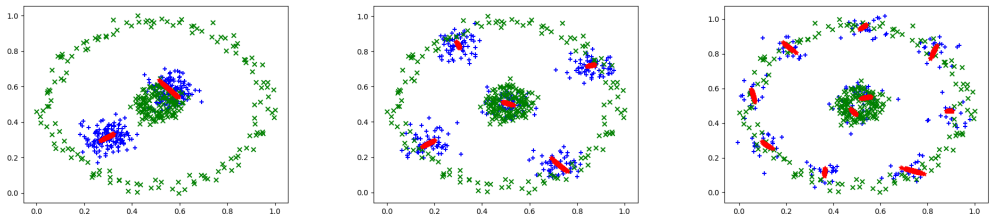


Figure 7.4: Results using synthetic dataset 2



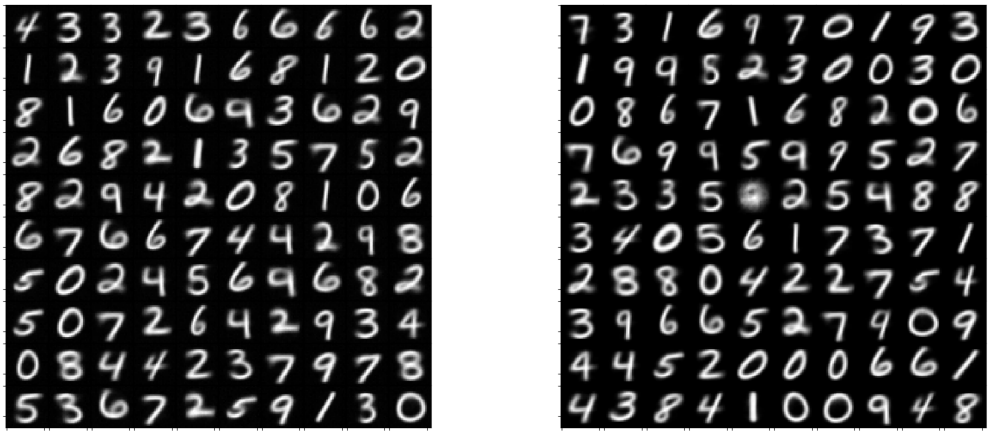
(a) (b) (c)

Figure 7.5: Results on circle dataset: (a) $K = 2$ (b) $K = 5$ (c) $K = 10$ after 100 epochs

Starting with the pinwheel dataset, we trained a model with 5 clusters to generate new synthetic 2-D examples from sampled codes (generated using the encoder). The codes were used as inputs to the decoder to sample new examples of the data. Figure 7.3a and 7.3b, respectively, represent the generated examples at the initial state (epoch=0) and after 100-epochs of training. Each of these figures depicts a plot for the original training examples (green), $\mu_k(Z)$ (red) and the sampled examples Y (blue). As can be seen by Figure 7.3b the multi modal learning technique allowed the decoder to learn and generate examples from different clusters of the data using the randomly initialized layers.

We repeated the experiment with the second synthetic dataset which consists of three separate classes. We used a model of $K = 7$ and the same parameters used in the previous experiments. Figure 7.4 shows that the method was able to generate new examples spread all over the clusters for this dataset.

The circle dataset was more interesting and challenging to try as the observations were nonlinearly separated into two clusters. We ran different experiments with different K values to test the ability of the model in generating examples from different regions of the original 2 clusters. As can be seen in Figure 7.5, the method was not able to generate perfect examples spread across the original clusters of the data when $K = 2$, however, better performance of the algorithm



(a) (b)

Figure 7.6: Results on MNIST dataset : (a) Model's samples from random codes (b) Digits learned by every μ_k layer of the decoder

was noted with the increase in K to 5 and 10 respectively.

7.4.2 Results on MNIST Dataset

The model has also been evaluated in terms of its ability to generate visually appealing digits from the MNIST dataset. We used a network of $K = 100$ with 500 neurons per layer and an average layer of 100 neurons. Figure 7.6a and 7.6b show the samples generated from 100-D random codes and the digits learned by each of the decoders μ_k respectively.

While the network was able to generate examples from a few clusters K, we noticed that with the increase in K, the network was able to learn and generate different styles of the same digit, so we presented the results generated by the model that was build with $K = 100$. Digit 0 for example, was learned by 14 μ_k(Z), each of which had a unique style as can be noted in Figure 7.6b.

7.4.3 Results on high dimensional cancer data

Finally, we were interested in testing the performance of the model in generating discriminating representations for high dimensional data. We chose gene expression datasets for their distinguishing characteristics which include: a limited number of examples, high dimensionality and imbalanced distribution of the examples among the classes.

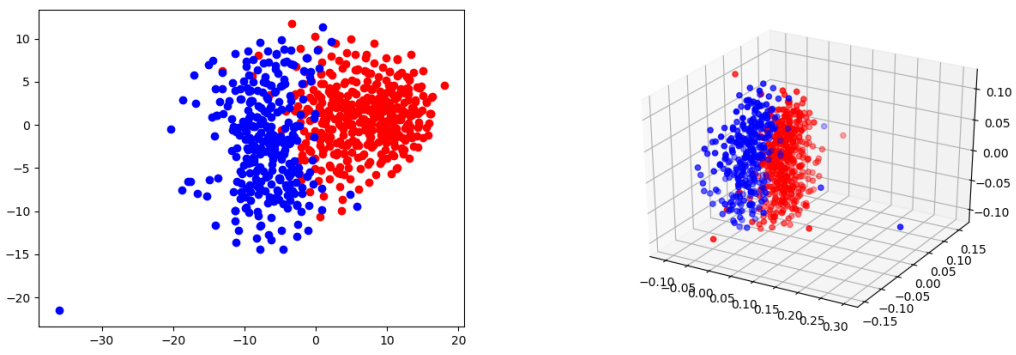
The datasets described by Table 7.1 were used separately, after normalization, to train the models. This table presents the datsets names, the number of features, the number of examples, number of classes and their references. Our architecture is represented by (500-250-500) to indicate the number of neurons in the layers arranged in the first, second and third hidden levels of the network. We set the other parameters according to the following: $K = 5$, learning rate $=$ 0.001 and dropout rate $= 0.3$. We also used the SGD back propagation algorithm and the sigmoid function for activating the layers.

The traditional VAE and the Gaussian Mixture Variational Autoencoder (GMVAE) [Dilokthanakul *et al.*, 2016] were adopted as baseline methods for comparison purpose and we used the same parameters, applied to our model, to train both of these networks except that we set K to the number of classes for GMVAE models the same way it was set in the original paper.

The ALLAML dataset has 71 examples distributed unequally between its classes with a majority:minority class ratio of 1.84. We trained our model and used the encoder $Q(Z|X)$ to sample 50 representations from every training example. Figure 7.7a and 7.7b show the distribution of the generated representations after reducing their dimensionality, by the t-SNE dimensionality reduction method, into 2-D and 3-D space respectively. The figures show that the method is able to generate representations with separable classes when they are reduced with t-SNE. However, we were interested in evaluating the generated representa-

Table 7.1: Experimented datasets, the columns indicate the dataset name, the number of features, the number of examples, the numbers of classes and citation to the dataset source.

Dataset	features	examples	classes	source
Leukemia	7070	72	2	[Golub *et al.*, 1999]
ALLAML	7129	72	2	[Armstrong *et al.*, 2001]
Prostate-GE	5966	102	2	[Singh *et al.*, 2002]
Lung	3312	203	5	[Bhattacharjee *et al.*, 2001]
ESCA	4442	194	2	[Tomczak *et al.*, 2015]
HNSC	3463	553	2	[Tomczak *et al.*, 2015]
LUAD	3509	1207	2	[Tomczak *et al.*, 2015]



(a) (b)

Figure 7.7: Two dimentionsl (a) and three dimensional (b) visualization for the ALLAML representations generated by t-SNE method.

tions using a simple probabilistic classifier. Table 7.2 presents the classification accuracy for the representations generated using the encoder part of our multi modal VAE, traditional VAE and the GMVAE respectively. Each of these experiments was run for 10 rounds by randomly splitting the dataset into 75% training and 25% testing examples. The results, which represent the average accuracy of the 10 runs, indicate that the naïve Bayes classifier was able to better detect our methods representations than the representations of other methods.

Table 7.2: Classification accuracy of naïve Bays used with the representations generated by the multi modal VAE, traditional VAE and the GMVAE

Dataset	multi modal	VAE	GMVAE
Leukemia	**.714**	.571	.571
ALLAML	**.700**	.577	.571
Prostate-GE	.600	**.730**	.709
Lung	**.667**	.619	.619
ESCA	**.900**	**.900**	**.900**
HNSC	**.911**	**.911**	**.911**
LUAD	**.911**	.901	.901

7.5 Conclusion

The multi modal framework for training the VAE introduced in this paper learns a set of parameters that are as close to set of predefined reference data points as possible. The training objective of this model considers the aggregated means learned by the layers located at the generative network rather than the density of the data. This allows for making relaxed application-dependent assumptions around the distribution learned by the decoder. In other words, this model learns a set of aggregated parameters ($\mu_{k=1..K}$,$\sigma_{k=1..K}$) that can be used to build a multi modal distribution such as a mixture of Gaussians, with uniformally distributed components, or batches of multi-variate distributions that we can use for generating new examples of the data.

Nevertheless, this model requires a proper selection to the number of components/clusters of the data to be able to generate examples from different regions of different clusters and, at the same time, to avoid the over-fitting problem. This problem of making decision on a proper number of clusters will be discussed in our future work.

7.6 Further Analysis

In addition to the experiments undertaken for the original paper, we wre interested here to justify the decision made on the number of center points and to present other empirical and comparative results to the performance of the network proposed in this chapter and its base line competing networks.

In all of our medical experiments we set $K = 5$ as we found that a network with 5 center points was able to give representations that are qualitatively similar to the representations generated by networks with larger number of center points. Figure 7.8 depicts a graph showing the effect of the increase in K on the quality of the representations generated for the ALLAML dataset. The quality was

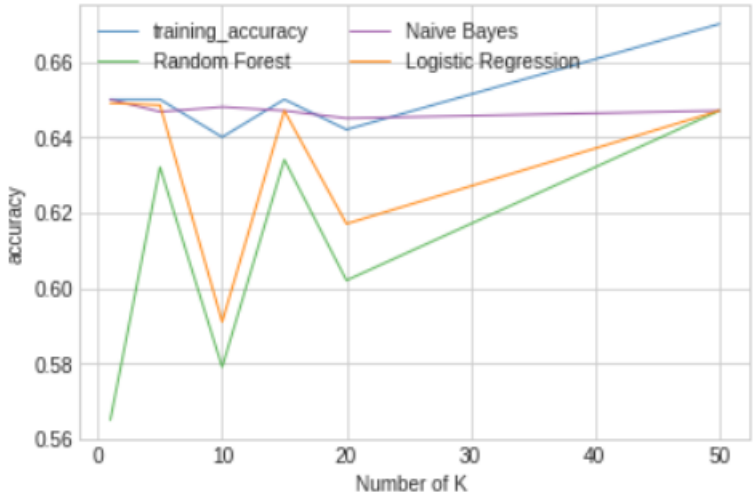


Figure 7.8: The effect of increasing K on the classification accuracy of random forest, naïve Bayes, SMO and logistic regression classifiers

measured by the accuracy of the random forest, SMO, naïve Bayes and logistic regression classifiers. The graph clearly shows that $K = 5$ does not only achieve the highest classification accuracy for all classifiers but also reduces the overfitting, this can be noted by comparing the classification accuracy of the training data, classified using the logistic regression classifier, represented by the blue line

and the classification accuracy of the testing data, classified by the same classifier, represented by the orange line.

Table 7.3 presents the results of other supporting experiments which shows how classifiers other than the naïve Bayes, which is used in the paper, perform with the representations generated by the networks used in this study. The results show that the representations generated by the multi modal VAE were better classified achieving statistically significant better AUROC values than the others, or otherwise similar. Indicating that this method is able to generate better separable representations than the base line methods in many test cases.

Table 7.3: The AUROC of random forest, SMO. naïve Bayes and logistic regression used with the representations generated by multi modal VAE, VAE and GMVAE. Results present the mean AUROC and standard deviation for one 10-fold cross validation run

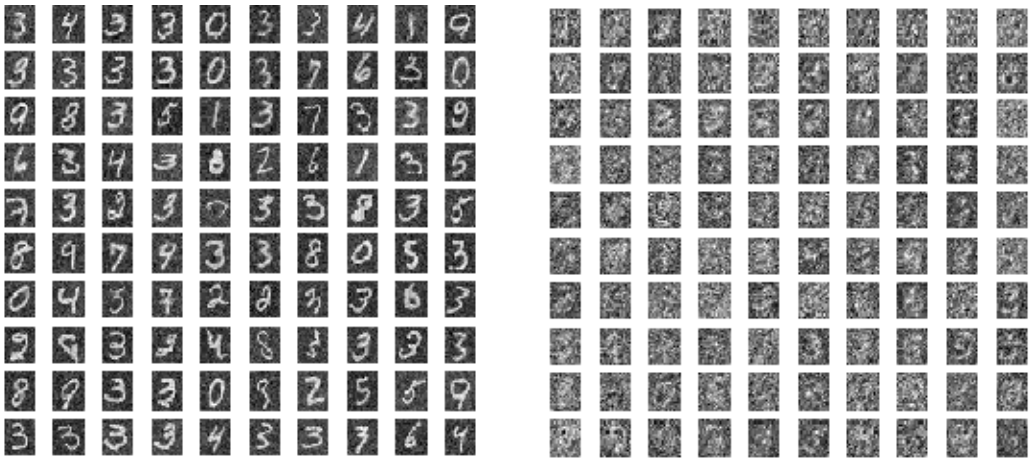
Dataset	model	random forest	SMO	naïve Bayes	logistic reg.
Leukemia	multi modal	0.646 ±0.105	0.646 ±0.105	**0.712**±0.083•	0.591 ±0.09
	VAE	0.646 ±0.105	0.646 ±0.105	0.604±0.11	0.561 ±0.127
	GMVAE	0.646 ±0.105	0.646 ±0.105	0.605±0.042	0.591±0.069
ALLAML	multi modal	0.648 ±0.117	0.649 ±0.116	**0.690** ±0.114•	0.634 ±0.176
	VAE	0.648 ±0.117	0.649 ±0.116	0.663 ±0.110	**0.647** ±0.101
	GMVAE	0.648 ±0.117	0.649 ±0.116	0.578 ±0.057	0.612 ±0.09
Lung	multi modal	0.684 ±0.048	0.684 ±0.048	0.684±0.048	**0.704** ±0.018
	VAE	0.684 ±0.048	0.684 ±0.048	0.684 ±0.048	0.684±0.048
	GMVAE	0.684 ±0.048	0.684 ±0.048	0.684 ±0.048	0.684 ±0.048
Prostate-GE	multi modal	0.481 ±0.046	**0.569** ±0.051•	**0.587** ±0.105	**0.578** ±0.046•
	VAE	**0.500** ±0.046	0.480 ±0.070	0.441 ±0.096	0.470 ±0.149
	GMVAE	**0.500** ±0.051	0.528 ±0.015	0.421 ±0.077	0.558 ±0.123

The network introduced in this chapter is a generative network which can be used for generating new examples of the data. For that reason, we will show

how it compares to the method proposed in chapter 5 for solving the imbalanced classification problem. We used the multi modal VAE for generating synthetic minority class examples for the datasets shown in Table 7.4 which also shows that our deterministic method proposed in that chapter significantly beaten the stochastic multi modal VAE in nearly all cases.

Table 7.4: Comparing the performance of Decoder/PSO and multi modal VAE of random forest, SMO, naïve Bayes and logistic regression

Dataset	method	random forest	SMO	naïve Bayes	logistic reg.
Leukemia	decoderPSO	**0.863** $\pm$ 0.186 •	0.500 $\pm$ 0.000	0.500 $\pm$ 0.000	**0.858** $\pm$ 0.135 •
	multi modal	0.677 $\pm$ 0.140	**0.668** $\pm$ 0.166 •	**0.661** $\pm$ 0.147 •	0.689 $\pm$ 0.126
GLI	decoderPSO	**0.799** $\pm$ 0.158 •	**0.500** $\pm$ 0.000	**0.544** $\pm$ 0.158•	**0.769** $\pm$ 0.162 •
	multi modal	0.608 $\pm$ 0.111	0.688 $\pm$ 0.123 •	0.363 $\pm$ 0.167	0.595 $\pm$ 0.168
Prostate-GE	decoderPSO	**0.818** $\pm$ 0.101 •	0.554 $\pm$ 0.099	0.554 $\pm$ 0.000	**0.803** $\pm$ 0.120 •
	multi modal	0.597 $\pm$ 0.0936	**0.606** $\pm$ 0.109	**0.597** $\pm$ 0.106	0.655 $\pm$ 0.135
SMK-CAN	decoderPSO	**0.693** $\pm$ 0.111	0.601 $\pm$ 0.082	0.531 $\pm$ 0.047	**0.733** $\pm$ 0.107 •
	multi modal	0.619 $\pm$ 0.088	**0.689** $\pm$ 0.104	**0.568** $\pm$ 0.158	0.629 $\pm$ 0.087
ALLAML	decoderPSO	**0.846** $\pm$ 0.104•	0.767 $\pm$ 0.079	**0.788** $\pm$ 0.132	**0.879** $\pm$ 0.093•
	multi modal	0.730 $\pm$ 0.121	**0.802** $\pm$ 0.096	0.673 $\pm$ 0.183	0.716 $\pm$ 0.213



(a) (b)

Figure 7.9: MNIST images generated from random codes using the multi modal VAE v.s. the VAE, both networks were trained using the subset (200 imbalanced) MNIST data

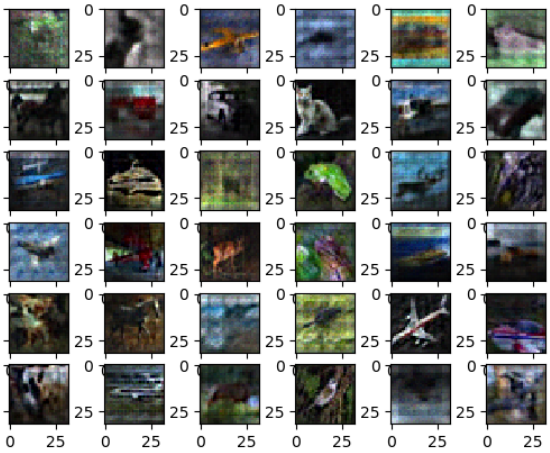


Figure 7.10: Random cifar10 images generated by the multi-decoder approach from random codes

One of our objectives was to introduce an inference model that is able to learn from small sized imbalanced datasets. For this reason, we repeated the MNIST experiments using a subset of randomly chosen 200 images consisting of 100 images from digit "3" and 100 images from the rest of the classes. Figure 7.9 depicts a qualitative comparison between 100 images generated from random codes using the VAE and our multi modal VAE networks.

Finally, we trained both the VAE and the multi modal VAE using 1000 randomly selected cifar10 images to compare their performance in generating examples from random codes sampled from $P(Z)$. For both models we used $CONV_{64}^{3\times3}, CONV_{32}^{3\times3}, CONV_{16}^{3\times3}, flatten_{100}$ layers for the encoder and a reflection of it for the decoder/s where $CONV_{xx}^{yy}$ represents a convolutional layer with xx filters and a kernel size of yy and $flatten_{100}$ is a fully connected flat layer which be used to learn the parameters of the distribution that generates the latent codes. Figure 7.10 depicts some images sampled from the $P(X|Z)$ learned by the generative network. Instead of clustering the data to get the reference points we randomly selected a number of 36 images from the training data and initialized the σ_k to small random numbers. However, the VAE model was not

able to generate comparable samples to the multi modal VAE.

7.7 Chapter Summary

In this chapter, we have empirically demonstrated the problem of singularity encountered by variational autoencoders and introduced a new variation of the network which proved successful in learning from small sized datasets. The experiments have shown that the network is also able to learn from large sized datasets and generate examples from different clusters of the data depending on μ_i. A good selection to the center points will not only allow for generating examples from different clusters of the data but will also allow for generating variations of the same cluster.

Chapter 8

Conclusions

This thesis introduced autoencoder-based techniques to improve the classification accuracy of high dimensional small sized datasets. Each of the proposed methods answered questions related to either improving the generated representations, generating (over-sampling) more representations or learning discriminating representations for the target dataset. The models described in Chapter 3 and Chapter 5 introduced techniques that can be applied to previously trained layers by triggering their neurons using optimized vectors. This technique allows for generating better outputs (Chapter 3) and various representations for a given input example (Chapter 5) respectively.

In contrast, we integrated the RBF activation mechanism with the autoencoder to improve the neurons response to similar patterns of the data in networks learning using the unsupervised learning mechanism. The method, presented in Chapter 6) demonstrated success in learning from the target gene expression datasets compared to the base-line SDA. Finally, chapter 7 introduced a novel framework for training probabilistic graph based autoencoder which also generates discriminating representations from different classes of the data competing the representations generated by other state-of-the-art probabilistic autoencoding models.

This chapter summarizes the main contributions achieved in this thesis and it is organized according to the following: Section 8.1 provides the summary of results, Section 8.2 concludes the contributions and Section 8.3 presents a list of future works.

8.1 Summary of results

This section summarizes the main findings and results achieved in this thesis which are stated as below:

- The PSO-autoencoder combination of methods (Chapter 3) is a new technique that activates the decoder part of a trained autoencoder to generate variations of a given example. The resulted versions were more visually appealing compared to the ones generated by the trained encoders in some cases. This method showed promising results particularly in cases where the autoencoder can not efficiently regenerate an input example due to poor training conditions e.g un-proper selection to the networks parameters.

- The survey presented in Chapter 4 categorized the most recent neural network based prediction model into: (i) filtering approaches (ii) predicting approaches and (iii) clustering approaches based on the functionality that the neural network applies. The paper introduced the pre-processing practices, the networks parameters and the settings adopted in each of the considered models and their consequent accuracy. The summary provided by the published paper can be used for selecting future cancer prediction models parameters and choosing the base line methods.

- Chapter 5 introduced a new technique for generating synthetic minority class examples for the gene expression data. The technique outperformed

the most commonly used oversampling techniques SMOTE as well as a newer version DBSMOTE in many cases.

- The RBF activation technique applied in the RBF networks is used for activating a set of parallel layers in the first hidden level of a proposed autoencoding network. The intention was to improve the autoencoder's learning ability by allowing the neurons to respond similarly to similar patterns of the data. The technique was empirically tested on different datasets with different dimensionalities and proved able to generate discriminating representations for the gene expression datasets.

- We applied the RBF activation trick to the VAE to test its applicability to stochastic graphical models. We also introduced a novel architectural model and a new loss function that guarantees generating examples from every cluster of the data to overcome the problem of singularities encountered by traditional VAE.

8.2 Contributions

The main contributions of this thesis were submitted to peer reviewed journals and presented in international conferences. These contribution are listed below:

- Introducing a new method combining both a pre trained autencoder and PSO algorithm to generate new variations of an input example.

- Categorizing the sate-of-the-art neural network based cancer methods into three categories and providing an organized summary of models parameters that can be used for future works.

- Proposing a new synthetic over sampling technique for high dimensional imbalanced small sized datasets.

- Introducing a new RBF-based autoencoder and a PSO-based method for initializing the center points parameters in RBF-based networks instead of the conventional Kmeans centroids which leads to overfitting when the input-space is not linearly separable .
- Introducing a new multi modal approach for training VAE.

8.3 Future works

This section presents a group of directions that were opened by the results of the published works, the conferences and the presentations that we have been to.

8.3.1 The interpretability of the RBF autoencoders

The RBF activations introduced in Chapter 6 showed promising results in learning predictable representations from different types of datasets. This thesis aimed and focused at introducing methods that improve the classification accuracy of the high dimensional small sized cancer data. However, an interesting future work can be done to get dynamic interpretation for the networks employing the RBF activations. A quick review to the literature on available interpretation techniques revealed that the images datasets are the commonly type of data used for this purpose; so we left this topic for the near future but will introduce a brief discussion about it here.

Our plan for the next work is to focus on expressing the computation of an arbitrary neuron activity in response to an input example and its effect on the internal computations of other neurons in the chain. This study will help in understanding the mechanism of the network and the contribution of the input training patterns in activating the feature maps. The plan is to use a visualization technique such as the ones provided in [Donahue *et al.*, 2014; Zeiler & Fergus, 2014] to identify: First: the patches within a dataset that are responsible for

strong activations at higher layers of the models. Secondly, the structures within each patch that stimulate a particular feature map.

8.3.2 Determining the number of K

The performance of the RBF-based networks with the parallel-hidden layer architecture relies on the number of radial kernels K (center points features) which determines the computational power of every training data [Que & Belkin, 2016]. Determining optimim number of K that retains the performance of the network and increase its generalization ability is not an easy task and has received limited attention by the research community which mostly focuses on determining the width of the kernels e.g. [Mohseni *et al.*, 2015; Que & Belkin, 2016].

Different approaches can be used for solving the sensitivity to the number of kernels problem. Hierarchical clustering techniques which discover a suitable number of clusters within the datast is a good alternative solution to using a fixed pre-determined number of clusters. A potentially better alternative is to build growing RBF networks that grow by one kernel at a time to gradually improve the performance of the network. This method has been previously applied to the supervised RBF clissification networks [Alsalamah *et al.*, 2018; Lu *et al.*, 2016] and we are interested in investigating its applicability to our models which learn in a complete unsupervised fashion.

Appendix A

Co-Authorship Forms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Co-Authorship Form

Postgraduate Studies Office
Student and Academic Services Division
Wahanga Ratonga Matauranga Akonga
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
Phone +64 7 838 4439
Website: http://www.waikato.ac.nz/sasd/postgraduate/

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in your appendices for all the copies of your thesis submitted for examination and library deposit (including digital deposit).

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Using Swarm Optimization To Enhance Autoencoder's Images

Nature of contribution by PhD candidate	Discussing the methodology, designing and running the experiments and writing the paper
Extent of contribution by PhD candidate (%)	80%

CO-AUTHORS

Name	Nature of Contribution
Michael Mayo	Discussing the experiments, discussing the difficulties and the results. Revising the paper and giving feedback

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and

Name	Signature	Date
M. Mayo		20.11.19

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Co-Authorship Form

Postgraduate Studies Office
Student and Academic Services Division
Wahanga Ratonga Matauranga Akonga
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
Phone +64 7 838 4439
Website: http://www.waikato.ac.nz/sasd/postgraduate/

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in your appendices for all the copies of your thesis submitted for examination and library deposit (including digital deposit).

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

A survey of neural network-based cancer prediction models from microarray data

Nature of contribution by PhD candidate	Reading the literature and ordering the papers in categories, writing up the paper
Extent of contribution by PhD candidate (%)	80%

CO-AUTHORS

Name	Nature of Contribution
Michael Mayo	Discussing the paper, revising the paper and giving feedback

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and

Name	Signature	Date
M. Mayo	M Mayo	20·11·19

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Co-Authorship Form

Postgraduate Studies Office
Student and Academic Services Division
Wahanga Ratonga Matauranga Akonga
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
Phone +64 7 838 4439
Website: http://www.waikato.ac.nz/sasd/postgraduate/

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in your appendices for all the copies of your thesis submitted for examination and library deposit (including digital deposit).

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

A Novel Synthetic Over-Sampling Technique for Imbalanced Classification of Gene Expressions Using Autoencoders and Swarm Optimization

Nature of contribution by PhD candidate	Discussing the methodology, design the experiments, writing up the paper
Extent of contribution by PhD candidate (%)	80%

CO-AUTHORS

Name	Nature of Contribution
Michael Mayo	Discussing the paper, designing the experiments, revising the paper and giving feedback

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and

Name	Signature	Date
M. Mayo	M Mayo	20.11.19

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Co-Authorship Form

Postgraduate Studies Office
Student and Academic Services Division
Wahanga Ratonga Matauranga Akonga
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
Phone +64 7 838 4439
Website: http://www.waikato.ac.nz/sasd/postgraduate/

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in your appendices for all the copies of your thesis submitted for examination and library deposit (including digital deposit).

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

RBFA: Radial Basis Function Autoencoders

Nature of contribution by PhD candidate	Discussing the methodology, design the experiments, writing up the paper
Extent of contribution by PhD candidate (%)	80%

CO-AUTHORS

Name	Nature of Contribution
Michael Mayo	Discussion during the weekly meeting, revising and editing the paper and giving feedback
Sally Jo Cunningham	Proof reading and giving feedback

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and

Name	Signature	Date
M. Mayo	M Mayo	20·11·19

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Co-Authorship Form

Postgraduate Studies Office
Student and Academic Services Division
Wahanga Ratonga Matauranga Akonga
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
Phone +64 7 838 4439
Website: http://www.waikato.ac.nz/sasd/postgraduate/

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in your appendices for all the copies of your thesis submitted for examination and library deposit (including digital deposit).

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Multi-modal Approach for Training Variational Autoencoders

Nature of contribution by PhD candidate	Discussing the methodology, reading the literature, design the experiments, writing up the paper
Extent of contribution by PhD candidate (%)	80%

CO-AUTHORS

Name	Nature of Contribution
Michael Mayo	Discussion during the weekly meeting, revising and editing the paper and giving feedback
Eibe Frank	Discussion, revising and editing the paper and giving feedback

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and

Name	Signature	Date
M. Mayo	M Mayo	20.11.19
E. Frank	Eibe Frank	20.11.19

References

(1984). *SIGCOMM Comput. Commun. Rev.*, **13-14**.

(2008). *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, ACM, New York, NY, USA, general Chair-Czerwinski, Mary and General Chair-Lund, Arnie and Program Chair-Tan, Desney.

(2015). *Using the amsthm Package*. American Mathematical Society, `http://www.ctan.org/pkg/amsthm`.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. 87

Ablamowicz, R. & Fauser, B. (2007). Clifford: a maple 11 package for clifford algebra computations, version 11.

Abril, P.S. & Plant, R. (2007). The patent holder's dilemma: Buy, sell, or troll? *Communications of the ACM*, **50**, 36–44.

Adya, A., Bahl, P., Padhye, J., A.Wolman & Zhou, L. (2004). A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proceedings of the IEEE 1st International Conference on Broadnets Networks (BroadNets'04)*, 210–217, IEEE, Los Alamitos, CA.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. & Cayirci, E. (2002). Wireless sensor networks: A survey. *Comm. ACM*, **38**, 393–422.

Akyildiz, I.F., Melodia, T. & Chowdhury, K.R. (2007). A survey on wireless multimedia sensor networks. *Computer Netw.*, **51**, 921–960.

Alhamdoosh, M. & Wang, D. (2014). Fast decorrelated neural network ensembles with random weights. *Information Sciences*, **264**, 104–117. 69

Allender, E. (1996). Circuit complexity before the dawn of the new millennium. vol. 18, Springer Berlin Heidelberg.

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D. & Levine, A.J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, **96**, 6745–6750. 72

Alsalamah, M., Amin, S. & Palade, V. (2018). Heart disease diagnosis based on deep radial basis function kernel machine. *EAI Endorsed Transactions on Ambient Systems*, **5**. 155

Alshamlan, H.M., Badr, G.H. & Alohali, Y.A. (2014). The performance of bio-inspired evolutionary gene selection methods for cancer classification

using microarray dataset. *International Journal of Bioscience, Biochemistry and Bioinformatics*, **4**, 166.

Andler, S. (1979). Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages*, POPL '79, 226–236, ACM Press, New York, NY.

Angermueller, C., Pärnamaa, T., Parts, L. & Stegle, O. (2016). Deep learning for computational biology. *Molecular Systems Biology*, **12**, 878. 44

Anisi, D.A. (2003). *Optimal Motion Control of a Ground Vehicle*. Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.

Archer, Jr., J.E., Conway, R. & Schneider, F.B. (1984). User recovery and reversal in interactive systems. *ACM Trans. Program. Lang. Syst.*, **6**, 1–19.

Armstrong, S.A., Staunton, J.E., Silverman, L.B., Pieters, R., den Boer, M.L., Minden, M.D., Sallan, S.E., Lander, E.S., Golub, T.R. & Korsmeyer, S.J. (2001). Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, **30**, 41. 114, 144

Armstrong, S.A., Staunton, J.E., Silverman, L.B., Pieters, R., den Boer, M.L., Minden, M.D., Sallan, S.E., Lander, E.S., Golub, T.R. & Korsmeyer, S.J. (2002). Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, **30**, 41. 72

Babenko, A. & Lempitsky, V. (2015). Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, 1269–1277. 25

Babenko, A., Slesarev, A., Chigorin, A. & Lempitsky, V. (2014). Neural codes for image retrieval. In *European Conference on Computer Vision*, 584–599, Springer. 25

Bahl, P., Chancre, R. & Dungeon, J. (2004). SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceeding of the 10th International Conference on Mobile Computing and Networking (MobiCom'04)*, 112–117, ACM, New York, NY.

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, 37–49. 1

Barua, S., Islam, M.M., Yao, X. & Murase, K. (2014). MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, **26**, 405–425. 82

Bashiri, A., Ghazisaeedi, M., Safdari, R., Shahmoradi, L. & Ehtesham, H. (2017). Improving the prediction of survival in cancer patients by using machine learning techniques: Experience of gene expression data: A narrative review. *Iranian Journal of Public Health*, **46**, 165. 46

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007a). Greedy layer-wise training of deep networks. In *Advances in Nural Information Processing Systems*, 153–160. 24, 66

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007b). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, **19**.

Bengio, Y., Courville, A. & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, **35**, 1798–1828. 2, 24

Bhat, R.R., Viswanath, V. & Li, X. (2017). Deepcancer: Detecting cancer through gene expressions via deep generative learning. *IEEE 2017 International Conference on Big Data Intelligence and Computing*. 49, 58, 59, 66

Bhattacharjee, A., Richards, W.G., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M. *et al.* (2001). Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences*, **98**, 13790–13795. 72, 114, 144

Bishop, C.M. (1994). Mixture density networks.

Blagus, R. & Lusa, L. (2012). Evaluation of smote for high-dimensional class-imbalanced microarray data. In *Machine learning and applications (icmla), 2012 11th international conference on*, vol. 2, 89–94, IEEE. 83

Borkowska, E.M., Kruk, A., Jedrzejczyk, A., Rozniecki, M., Jablonowski, Z., Traczyk, M., Constantinou, M., Banaszkiewicz, M., Pietrusinski, M., Sosnowski, M. *et al.* (2014). Molecular subtyping of bladder cancer using k ohonen self-organizing maps. *Cancer Medicine*, **3**, 1225–1234. 44, 49, 63

Bosch, A., Andrew, Z. & Muñoz, X. (2006). Scene classification via pLSA. In *European Conference on Computer Vision*, 517–530, Springer, Berlin Heidelberg. 25

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, 177–186, Springer. 88

Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C.J. & Schoelkopf, B. (2017). From optimal transport to generative modeling: the vegan cookbook. *arXiv preprint arXiv:1705.07642*. 128, 131, 136, 138

Bowman, M., Debray, S.K. & Peterson, L.L. (1993). Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, **15**, 795–825.

Braams, J. (1991). Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, **12**, 291–301.

Brachman, R.J. & Schmolze, J.G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, **9**, 171–216.

Braga-Neto, U.M. & Dougherty, E.R. (2004). Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, **20**, 374–380. 69

Bunkhumpornpat, C., Sinapiromsaran, K. & Lursinsap, C. (2012). Db-smote: density-based synthetic minority over-sampling technique. *Applied Intelligence*, **36**, 664–684. 19, 76

Buss, J.F., Rosenberg, A.L. & Knott, J.D. (1987a). Vertex types in book-embeddings. Tech. rep., Amherst, MA, USA.

Buss, J.F., Rosenberg, A.L. & Knott, J.D. (1987b). Vertex types in book-embeddings. Tech. rep., Amherst, MA, USA.

Chandrashekar, G. & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, **40**, 16–28. 43

Chaudhary, K., Poirion, O.B., Lu, L. & Garmire, L. (2017). Deep learning based multi-omics integration robustly predicts survival in liver cancer. *bioRxiv*, 114892. 44, 48, 50, 54, 55, 56, 66

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, **16**, 321–357. 18, 82, 89

Chen, H. & Alan, M.F. (2003). Continuous restricted boltzmann machine with an implementable training algorithm. In *IEE Proceedings-Vision,Image and Signal Processing*, vol. 150.

Chen, H., Yu, Z., Han, G., You, J. & Li, L. (2012). Ng 2 ce: Double neural gas based cluster ensemble framework. In *Computer Science & Education (ICCSE), 2012 7th International Conference on*, 26–31, IEEE. 65

Chen, M., Weinberger, K.Q., Xu, Z.E. & Sha, F. (2015a). Marginalizing stacked linear denoising autoencoders. *Journal of Machine Learning Research*, **16**, 3849–3875.

Chen, S., Cowan, C.F. & Grant, P.M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks*, **2**, 302–309. 10, 12, 13

Chen, Y.C., Ke, W.C. & Chiu, H.W. (2014). Risk classification of cancer survival using ann with gene expression data from multiple laboratories. *Computers in Biology and Medicine*, **48**, 1–7. 18, 49, 50, 58

Chen, Y.C., Chang, Y.C., Ke, W.C. & Chiu, H.W. (2015b). Cancer adjuvant chemotherapy strategic classification by artificial neural network with gene expression data: An example for non-small cell lung cancer. *Journal of Biomedical Informatics*, **56**, 1–7. 18, 49, 50, 52, 58, 66, 67

Cheng, R. & Jin, Y. (2014). A competitive swarm optimizer for large scale optimization. *IEEE transactions on cybernetics*, **45**, 191–204. 15, 16

Cheng, R. & Jin, Y. (2015). A competitive swarm optimizer for large scale optimization. *IEEE Transactions on Cybernetics*, 191–204. 28, 31

Cheok, M.H., Yang, W., Pui, C.H., Downing, J.R., Cheng, C., Naeve, C.W., Relling, M.V. & Evans, W.E. (2003). Treatment-specific changes in gene expression discriminate in vivo drug response in human leukemia cells. *Nature Genetics*, **34**, 85. 72

Ching, T., Himmelstein, D.S., Beaulieu-Jones, B.K., Kalinin, A.A., Do, B.T., Way, G.P., Ferrero, E., Agapow, P.M., Zietz, M., Hoffman, M.M. *et al.* (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, **15**, 20170387. 4

Chow, C.K. & Liu, C.N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, **IT-14**, 462–467.

Chowdary, D., Lathrop, J., Skelton, J., Curtin, K., Briggs, T., Zhang, Y., Yu, J., Wang, Y. & Mazumder, A. (2006). Prognostic gene expression signatures can be measured in tissues collected in rnalater preservative. *The Journal of Molecular Diagnostics*, **8**, 31–39. 72

Christopher, M.R.P., Chopra, S. & LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. *In Advances in neural information processing systems*.

Chuang, L.Y., Ke, C.H. & Yang, C.H. (2016). A hybrid both filter and wrapper feature selection method for microarray classification. *arXiv preprint arXiv:1612.08669*.

Ciancio, C., Ambrogio, G., Gagliardi, F. & Musmanno, R. (2016).

Heuristic techniques to optimize neural network architecture in manufacturing applications. *Neural Computing and Applications*, **27**, 2001–2015. 26

Ciatti, S., Gennaro, G., Mungai, V. & Nannini, G. (2015). Should volumetric breast density be included in breast cancer prediction models? proposal of an integrated quantitative and reproducible approach. 47

Clark, M. (1991). Post congress tristesse. In *TeX90 Conference Proceedings*, 84–89, TeX Users Group.

Clarkson, K.L. (1985a). *Algorithms for Closest-Point Problems (Computational Geometry)*. Ph.D. thesis, Stanford University, Palo Alto, CA, uMI Order Number: AAT 8506171.

Clarkson, K.L. (1985b). *Algorithms for Closest-Point Problems (Computational Geometry)*. Ph.D. thesis, Stanford University, Stanford, CA, USA, aAT 8506171.

Clavería González, Ó., Monte Moreno, E. & Torra Porras, S. (2015). Regional forecasting with support vector regressions: The case of spain. *AQR–Working Papers, 2015, AQR15/06*. 102

Cohen (1996). Special issue: Digital libraries.

Cohen, S., Nutt, W. & Sagic, Y. (2007). Deciding equivalances among conjunctive aggregate queries. *J. ACM*, **54**.

Colaprico, A., Silva, T.C., Olsen, C., Garofano, L., Cava, C., Garolini, D., Sabedot, T., Malta, T.M., Pagnotta, S.M., Castiglioni, I., Ceccarelli, M., Bontempi, G. & Noushmehr, H. (2015). Tcgabiolinks: An r/bioconductor package for integrative analysis of TCGA data. *Nucleic Acids Research*. 87

Connell, J.P. & Wellborn, J.G. (1991). Competence, autonomy, and relatedness: A motivational analysis of self-system processes.

Conti, M., Di Pietro, R., Mancini, L.V. & Mei, A. (2009a). (new) distributed data source verification in wireless sensor networks. *Inf. Fusion*, **10**, 342–353.

Conti, M., Di Pietro, R., Mancini, L.V. & Mei, A. (2009b). (old) distributed data source verification in wireless sensor networks. *Inf. Fusion*, **10**, 342–353.

Coronado, E., Villalobos, J., Bruno, B. & Mastrogiovanni, F. (2017). Gesture-based Robot Control: Design Challenges and Evaluation with Humans.

CROSSBOW (2008). XBOW sensor motes specifications. Http://www.xbow.com.

Crowther, P., Cox, R. & Sharma, D. (2004). A study of the radial basis function neural network classifiers using known data of varying accuracy and complexity. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 210–216, Springer. 103

Culler, D., Estrin, D. & Srivastava, M. (2004). Overview of sensor networks. *IEEE Comput.*, **37**, 41–49.

Curtis, C., Shah, S.P., Chin, S.F., Turashvili, G., Rueda, O.M., Dunning, M.J., Speed, D., Lynch, A.G., Samarajiwa, S., Yuan, Y. *et al.* (2012). The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, **486**, 346–352. 72

Danaee, P., Ghaeini, R. & Hendrix, D.A. (2016). A deep learning approach for cancer detection and relevant gene identification. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 22, 219, NIH Public Access. 44, 49, 50, 51, 54, 55, 56, 66, 69

Daoud, M. & Mayo, M. (2018). A novel synthetic over-sampling technique for imbalanced classification of gene expressions using autoencoders and swarm optimization. In *Australasian Joint Conference on Artificial Intelligence*, 603–615, Springer.

Daoud, M. & Mayo, M. (2019). A survey of neural network-based cancer prediction models from microarray data. *Artificial intelligence in medicine*. 18

Daoud, M., Mayo, M. & Cunningham, S.J. (2019). RBFA: Radial basis function autoencoders. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2966–2973, IEEE. 137

de Souto, M.C., Costa, I.G., de Araujo, D.S., Ludermir, T.B. & Schliep, A. (2008). Clustering cancer gene expression data: a comparative study. *BMC Bioinformatics*, **9**, 497. 72

Delon, J. & Desolneux, A. (2019). A wasserstein-type distance in the space of gaussian mixture models. *arXiv preprint arXiv:1907.05254*. 138

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, **7**, 1–30. 91

Dheeru, D. & Karra Taniskidou, E. (2017). UCI machine learning repository. 49

Dijkstra, E. (1979). Go to statement considered harmful. In *Classics in soft-*

ware engineering (incoll), 27–33, Yourdon Press, Upper Saddle River, NJ, USA.

Dilokthanakul, N., Mediano, P.A., Garnelo, M., Lee, M.C., Salimbeni, H., Arulkumaran, K. & Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*. 131, 143

Ding, C. & Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, **3**, 185–205. 114

Doaud, M. & Mayo, M. (2017). Using swarm optimization to enhance autoencoder's images. In *MDAI 2017*, 118–131.

Doersch, C., Gupta, A. & Efros, A.A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 1422–1430. 17, 53

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. & Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, 647–655. 154

Dong, Y., Du, B., Zhang, L. & Zhang, L. (2017). Dimensionality reduction and classification of hyperspectral images using ensemble discriminative local metric learning. *IEEE Transactions on Geoscience and Remote Sensing*, **55**, 2509–2524. 78

Dosovitskiy, A. & Brox, T. (2016). Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4829–4837. 25

Dosovitskiy, A., Springenberg, J.T., Riedmiller, M. & Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, 766–774. 17, 53

Douglass, B.P., Harel, D. & Trakhtenbrot, M.B. (1998). Statecarts in use: structured analysis and object-orientation. In G. Rozenberg & F.W. Vaandrager, eds., *Lectures on Embedded Systems*, vol. 1494 of *Lecture Notes in Computer Science*, 368–394, Springer-Verlag, London.

Dunlop, D.D. & Basili, V.R. (1985). Generalizing specifications for uniformly implemented loops. *ACM Trans. Program. Lang. Syst.*, **7**, 137–158.

Dwivedi, A.K. (2018). Artificial neural network model for effective cancer classification using microarray gene expression data. *Neural Computing and Applications*, **29**, 1545–1554. 58, 60

Dyrskjøt, L., Thykjaer, T., Kruhøffer, M., Jensen, J.L., Marcussen, N., Hamilton-Dutoit, S., Wolf, H. & Ørntoft, T.F. (2003). Identifying distinct classes of bladder carcinoma using microarrays. *Nature Genetics*, **33**, 90. 72

Eberhart, R. & Kennedy, J. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, vol. 4, 1942–1948, Citeseer. 14

Editor, I., ed. (2007). *The title of book one*, vol. 9 of *The name of the series one*. University of Chicago Press, Chicago, 1st edn.

Editor, I., ed. (2008). *The title of book two*, chap. 100. The name of the series two, University of Chicago Press, Chicago, 2nd edn.

Eibe Frank, B.P.P.R.I.H.W., Geoffrey Holmes (????). Weka data mining software. 90

Eiben, A.E., Smith, J.E. *et al.* (2003). *Introduction to evolutionary computing*, vol. 53. Springer. 112

El-Bakry, H.M. (2010). An efficient algorithm for pattern detection using combined classifiers and data fusion. *Information Fusion*, **11**, 133–148. 43

Ester, M., Kriegel, H.P., Sander, J., Xu, X. *et al.* (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, vol. 96, 226–231. 19

Fakoor, R., Ladhak, F., Nazi, A. & Huber, M. (2013). Using deep learning to enhance cancer diagnosis and classification. In *Proceedings of the International Conference on Machine Learning*. 43, 51, 54, 55, 56

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, **27**, 861–874. 47

Fawcett, T. & Provost, F. (1997). Adaptive fraud detection. *Data mining and knowledge discovery*, **1**, 291–316. 78

Fear, S. (2005). *Publication quality tables in LaTeX*. `http://www.ctan.org/pkg/booktabs`.

Figueredo, A.J. & Wolf, P.S.A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, **20**, 317–330.

Frank, H.G.P.B.R.P.W.I.H., E. (????). Weka data mining software.

Fujiwara, T., Hiramatsu, M., Isagawa, T., Ninomiya, H., Inamura, K., Ishikawa, S., Ushijima, M., Matsuura, M., Jones, M.H., Shimane,

M. *et al.* (2012). Ascl1-coexpression profiling but not single gene expression profiling defines lung adenocarcinomas of neuroendocrine nature with poor prognosis. *Lung Cancer*, **75**, 119–125. 72

Galbally, J., Arun, R., Marta, G.B., Julian, F. & Ortega-Garcia, J. (2013). Iris image reconstruction from binary templates: An efficient probabilistic approach based on genetic algorithms. *Computer Vision and Image Understanding*, **117**.

Gashaw, I., Grümmer, R., Klein-Hitpass, L., Dushaj, O., Bergmann, M., Brehm, R., Grobholz, R., Kliesch, S., Neuvians, T., Schmid, K. *et al.* (2005). Gene signatures of testicular seminoma with emphasis on expression of ets variant gene 4. *Cellular and Molecular Life Sciences CMLS*, **62**, 2359–2368. 72

Geer LY, G.R.H.L.H.J.H.S.L.C.S.W.B.S., Marchler-Bauer A (2010). The ncbi biosystems database. *Science translational medicine*.

Geiger, D. & Meek, C. (2005). Structured variational inference procedures and their realizations (as incol). In *Proceedings of Tenth International Workshop on Artificial Intelligence and Statistics,* The Barbados, The Society for Artificial Intelligence and Statistics.

Geman, S., Bienenstock, E. & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, **4**, 1–58. 13

Gerndt, M. (1989). *Automatic Parallelization for Distributed-Memory Multiprocessing Systems*. Ph.D. thesis, University of Bonn, Bonn, Germany.

Gibbons, J.D. & Chakraborti, S. (2011). Nonparametric statistical inference. In *International encyclopedia of statistical science*, 977–979, Springer. 91

Girosi, F. (1992). Some extensions of radial basis functions and their applications in artificial intelligence. *Computers & Mathematics with Applications*, **24**, 61–80.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587. 53

Glorot, X. & Bengio, Y. (2010a). Understanding the difficulty of training deep feedforward neural networks. *Aistats*, **9**, 249–256. 34

Glorot, X. & Bengio, Y. (2010b). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256. 109

Glorot, X., Bordes, A. & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323.

Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A. *et al.* (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, **286**, 531–537. 60, 72, 144

Gong, Y.J., Zhang, J., Chung, H.S.H., Chen, W.N., Zhan, Z.H., Li, Y. & Shi, Y.H. (2012). An efficient resource allocation scheme using particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, **16**, 801–816. 15

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Genera-

tive adversarial nets. In *Advances in neural information processing systems*, 2672–2680. 2, 10

Goodman, S.N., Fanelli, D. & Ioannidis, J.P. (2016). What does research reproducibility mean? *Science translational medicine*, **8**, 341ps12–341ps12.

Goossens, M., Rahtz, S.P., Moore, R. & Sutor, R.S. (1999). *The Latex Web Companion: Integrating TEX, HTML, and XML*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edn.

Gordon, G.J., Jensen, R.V., Hsiao, L.L., Gullans, S.R., Blumenstock, J.E., Ramaswamy, S., Richards, W.G., Sugarbaker, D.J. & Bueno, R. (2002). Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, **62**, 4963–4967. 72

Grabec, I. (2007). The normalized radial basis function neural network and its relation to the perceptron. *arXiv preprint physics/0703229*. 108

Gregor, K., Ivo, D., Graves, A., Rezende, D.J. & Wierstra, D. (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv*, **1502**, 04623.

Gu, F., Khoshelham, K., Valaee, S., Shang, J. & Zhang, R. (2018). Locomotion activity recognition using stacked denoising autoencoders. *IEEE Internet of Things Journal*, **5**, 2085–2093. 12

Gundy, M.V., Balzarotti, D. & Vigna, G. (2007). Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies*, WOOT '07, USENIX Association, Berkley, CA.

Gundy, M.V., Balzarotti, D. & Vigna, G. (2008). Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies*, WOOT '08, 99–100, USENIX Association, Berkley, CA.

Gundy, M.V., Balzarotti, D. & Vigna, G. (2009). Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies*, WOOT '09, 90–100, USENIX Association, Berkley, CA.

Gupta, A., Wang, H. & Ganapathiraju, M. (2015). Learning structure in gene expression data using deep architectures, with an application to gene clustering. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, 1328–1335, IEEE.

Hamosh, A., Scott, A.F., Amberger, J.S., Bocchini, C.A. & McKusick, V.A. (2005). Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, **33**, D514–D517. 50

Han, H., Wang, W.Y. & Mao, B.H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. *Advances in intelligent computing*, 878–887. 19, 82

Harel, D. (1978). Logics of programs: Axiomatics and descriptive power. MIT Research Lab Technical Report TR-200, Massachusetts Institute of Technology, Cambridge, MA.

Harel, D. (1979). *First-Order Dynamic Logic*, vol. 68 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, NY.

Harvard CodeBlue (2008). CodeBlue: Sensor networks for medical care. Http://www.eecs.harvard.edu/mdw/ proj/codeblue/.

Heering, J. & Klint, P. (1985). Towards monolingual programming environments. *ACM Trans. Program. Lang. Syst.*, **7**, 183–213.

Herlihy, M. (1993). A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, **15**, 745–770.

Hervé, J., Douze, M., Schmid, C. & Pérez, P. (2006). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR)*, 3304–3311, IEEE Conference. 25

Hinton, G. & Ruslan, S.R. (2006). Reducing the dimensionality of data with neural networks. *Science*, **313**, 504–507. 25, 26, 29

Hoare, C.A.R. (1972). Chapter ii: Notes on data structuring. In O.J. Dahl, E.W. Dijkstra & C.A.R. Hoare, eds., *Structured programming (incoll)*, 83–174, Academic Press Ltd., London, UK, UK.

Hollis, B.S. (1999). *Visual Basic 6: Design, Specification, and Objects with Other*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn.

Hörmander, L. (1985a). *The analysis of linear partial differential operators. III*, vol. 275 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, Germany, pseudodifferential operators.

Hörmander, L. (1985b). *The analysis of linear partial differential operators. IV*, vol. 275 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, Germany, fourier integral operators.

Hoshida, Y., Brunet, J.P., Tamayo, P., Golub, T.R. & Mesirov, J.P. (2007). Subclass mapping: identifying common subtypes in independent disease data sets. *PloS One*, **2**, e1195. 72

Hou, Q., Bing, Z.T., Hu, C., Li, M.Y., Yang, K.H., Mo, Z., Xie, X.W., Liao, J.L., Lu, Y., Horie, S. *et al.* (2018). Rankprod combined with genetic algorithm optimized artificial neural network establishes a diagnostic and prognostic prediction model that revealed c1qtnf3 as a biomarker for prostate cancer. *EBioMedicine*. 49, 58, 60

Huang, M.W., Chen, C.W., Lin, W.C., Ke, S.W. & Tsai, C.F. (2017). Svm and svm ensembles in breast cancer prediction. *PloS one*, **12**, e0161501. 47

Huson, I.J. (2017). Mrmr ba: A hybrid gene selection algorithm for cancer classification. *Journal of Theoretical and Applied Information Technology*, **95**.

IEEE (2004). Ieee tcsc executive committee. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '04, 21–22, IEEE Computer Society, Washington, DC, USA.

Inan, O., Uzer, M.S. & Yilmaz, N. (2013). A new hybrid feature selection method based on association rules and pca for detection of breast cancer. *International Journal of Innovative Computing, Information and Control*, **9**, 727–729.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*. 81

Japkowicz, N. & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, **6**, 429–449.

Johnson, M., Duvenaud, D.K., Wiltschko, A., Adams, R.P. & Datta, S.R. (2016). Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, 2946–2954. 139

Karabulut, E.M. & Ibrikci, T. (2017). Discriminative deep belief networks for microarray based cancer classification. *Biomedical Research*, **28**.

Karayiannis, N.B. & Mi, G.W. (1997). Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural networks*, **8**, 1492–1506.

Karayiannis, N.B. & Randolph-Gips, M.M. (2003). On the construction and training of reformulated radial basis function neural networks. *IEEE Transactions on Neural Networks*, **14**, 835–846. 108

Kelley, D.R., Snoek, J. & Rinn, J.L. (2016). Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, **26**, 990–999.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *IEEE International Conference on Neural Networks*, 504–507. 28

Kennedy, J. & Mendes, R. (2002). Population structure and particle swarm performance. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on 2002*, vol. 2, 1931—-1938, IEEE. 28

Kentridge (2017). Kentridge biomedical repository. 49

Kingma, D.P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 111

Kingma, D.P. & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. 2, 10, 138

Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I. & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, 4743–4751. 131

Kirschmer, M. & Voight, J. (2010). Algorithmic enumeration of ideal classes for quaternion orders. *SIAM J. Comput.*, **39**, 1714–1747.

Klein, H.U., Ruckert, C., Kohlmann, A., Bullinger, L., Thiede, C., Haferlach, T. & Dugas, M. (2009). Quantitative comparison of microarray experiments with published leukemia related gene expression signatures. *BMC Bioinformatics*, **10**, 422. 72

Knuth, D.E. (1981a). *Seminumerical Algorithms*. Addison-Wesley.

Knuth, D.E. (1981b). *Seminumerical Algorithms*, vol. 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 2nd edn.

Knuth, D.E. (1984). *The TEXbook*. Addison-Wesley, Reading, MA.

Knuth, D.E. (1997). *The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.)*. Addison Wesley Longman Publishing Co., Inc.

Knuth, D.E. (1998). *The Art of Computer Programming*, vol. 1 of *Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., 3rd edn., (book).

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, **21**, 1–6. 62

Kong, W.C. (2001a). *E-commerce and cultural values*, name of chapter: The implementation of electronic commerce in SMEs in Singapore (Inbook-w-chap-w-type), 51–74. IGI Publishing, Hershey, PA, USA.

Kong, W.C. (2001b). The implementation of electronic commerce in smes in singapore (as incoll). In *E-commerce and cultural values*, 51–74, IGI Publishing, Hershey, PA, USA.

Kong, W.C. (2002). Chapter 9. In T. Thanasankit, ed., *E-commerce and cultural values (Incoll-w-text (chap 9) 'title')*, 51–74, IGI Publishing, Hershey, PA, USA.

Kong, W.C. (2003). The implementation of electronic commerce in smes in singapore (incoll). In T. Thanasankit, ed., *E-commerce and cultural values*, 51–74, IGI Publishing, Hershey, PA, USA.

Kong, W.C. (2004). *E-commerce and cultural values - (InBook-num-in-chap)*, chap. 9, 51–74. IGI Publishing, Hershey, PA, USA.

Kong, W.C. (2005). *E-commerce and cultural values (Inbook-text-in-chap)*, chapter: The implementation of electronic commerce in SMEs in Singapore, 51–74. IGI Publishing, Hershey, PA, USA.

Kong, W.C. (2006). *E-commerce and cultural values (Inbook-num chap)*, chapter (in type field) 22, 51–74. IGI Publishing, Hershey, PA, USA.

Korach, E., Rotem, D. & Santoro, N. (1984). Distributed algorithms for finding centers and medians in networks. *ACM Trans. Program. Lang. Syst.*, **6**, 380–401.

Kornerup, J. (1994). *Mapping Powerlists onto Hypercubes*. Master's thesis, The University of Texas at Austin, (In preparation).

Kosiur, D. (2001). *Understanding Policy-Based Networking*. Wiley, New York, NY, 2nd edn.

Krishnaiah, V., Narsimha, D.G. & Chandra, D.N.S. (2013). Diagnosis of lung cancer prediction system using data mining classification techniques. *International Journal of Computer Science and Information Technologies*, **4**, 39–45. 66

Kubat, M., Matwin, S. *et al.* (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, vol. 97, 179–186, Nashville, USA. 81

Kumar, C.A. & Ramakrishnan, S. (2014). Binary classification of cancer microarray gene expression data using extreme learning machines. In *Computational Intelligence and Computing Research (ICCIC), 2014 IEEE International Conference on*, 1–4, IEEE. 18, 49, 58, 60, 66

Kumar, V. & Kumar, D. (2018). Gene expression data clustering using variance-based harmony search algorithm. *IETE Journal of Research*, 1–12. 50

Laiho, P., Kokko, A., Vanharanta, S., Salovaara, R., Sammalkorpi, H., Järvinen, H., Mecklin, J., Karttunen, T., Tuppurainen, K., Davalos, V. *et al.* (2007). Serrated carcinomas form a subclass of colorectal cancer with distinct molecular basis. *Oncogene*, **26**, 312. 72

Lam, H.K., Ekong, U., Liu, H., Xiao, B., Araujo, H., Ling, S.H. & Chan, K.Y. (2014). A study of neural-network-based classifiers for material classification. *Neurocomputing*, **144**, 367–377. 45

Lamport, L. (1986). *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, MA.

Law, M.H., Figueiredo, M.A. & Jain, A.K. (2004). Simultaneous feature

selection and clustering using mixture models. *IEEE transactions on pattern analysis and machine intelligence*, **26**, 1154–1166.

Lawrence, S., Giles, C.L., Tsoi, A.C. & Back, A.D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, **8**, 98–113. 45

Lee, C.C., Chung, P.C., Tsai, J.R. & Chang, C.I. (1999). Robust radial basis function neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **29**, 674–685. 108

Lee, J. (1981). Transcript of question and answer session. In R.L. Wexelblat, ed., *History of programming languages I (incoll)*, 68–71, ACM, New York, NY, USA.

Lee, N. (2005). Interview with bill kinder: January 13, 2005. *Comput. Entertain.*, **3**.

Leisch, F. & Dimitriadou, E. (2010). *mlbench: Machine Learning Benchmark Problems*. R package version 2.1-1.

Leung, M.K., Delong, A., Alipanahi, B. & Frey, B.J. (2016). Machine learning in genomic medicine: a review of computational problems and data sets. *Proceedings of the IEEE*, **104**, 176–197. 44

Leung, T. & Jitendra, M. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, **43**. 25

Levesque, H.J. (1984a). Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, **23**, 155–212.

Levesque, H.J. (1984b). A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, 198–202, American Association for Artificial Intelligence, Austin, Texas.

Li, C.L., Buyuktur, A.G., Hutchful, D.K., Sant, N.B. & Nainwal, S.K. (2008). Portalis: using competitive online interactions to support aid initiatives for the homeless. In *CHI '08 extended abstracts on Human factors in computing systems*, 3873–3878, ACM, New York, NY, USA.

Li, L.J., Su, H., Lim, Y. & Fei-Fei., L. (2014). Object bank: An object-level image representation for high-level visual recognition. *International Journal of Computer Vision*, **107**. 25

Li, Y. & Turner, R.E. (2016). Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, 1073–1081. 131

Lichman, M. (2013). UCI machine learning repository.

Lin, C., Jain, S., Kim, H. & Bar-Joseph, Z. (2017). Using neural networks for reducing the dimensions of single-cell rna-seq data. *Nucleic Acids Research*, **45**, e156–e156.

Lin, W.J. & Chen, J.J. (2012). Class-imbalanced classifiers for high-dimensional data. *Briefings in bioinformatics*, **14**, 13–26. 78, 82

Liu, J., Wang, X., Cheng, Y. & Zhang, L. (2017). Tumor gene expression data classification via sample expansion-based deep learning. *Oncotarget*, **8**, 109646. 51, 60

Lu, C., Han, H., Qiao, J. & Yang, C. (2016). Design of a self-organizing recurrent RBF neural network based on spiking mechanism. In *2016 35th Chinese Control Conference (CCC)*, 3624–3629, IEEE. 155

Macías-García, L., Luna-Romera, J.M., García-Gutiérrez, J., del Mar Martínez-Ballesteros, M., Riquelme-Santos, J.C. & González-Cámpora, R. (2017). A study of the suitability of autoencoders for preprocessing data in breast cancer experimentation. *Journal of Biomedical Informatics*. 44, 50, 54, 55, 66

Mahendran, A. & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5188–5196.

Makhzani, A. & Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*. 106, 117

Mandal, S. & Banerjee, I. (2015). Cancer classification using neural network. *International Journal*, **172**. 18, 49, 50, 58, 59

Mandal, S., Saha, G. & Pal, R. (2013). Reconstruction of dominant gene regulatory network from microarray data using rough set and bayesian approach. *Journal of Computer Science & Systems Biology*, **6**, 262–270. 59

Masci, J., Meier, U., Cireşan, D. & Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, 52–59, Springer. 2

McCracken, D.D. & Golden, D.G. (1990). *Simplified Structured COBOL with Microsoft/MicroFocus COBOL*. John Wiley & Sons, Inc., New York, NY, USA.

Mills, K.I., Kohlmann, A., Williams, P.M., Wieczorek, L., Liu, W.m., Li, R., Wei, W., Bowen, D.T., Loeffler, H., Hernandez, J.M. *et al.* (2009). Microarray-based classifiers and prognosis models identify subgroups

with distinct clinical outcomes and high risk of aml transformation of myelodysplastic syndrome. *Blood*, **114**, 1063–1072. 72

Mishra, S., Shaw, K. & Mishra, D. (2012). A new meta-heuristic bat inspired classification approach for microarray data. *Procedia Technology*, **4**, 802–806.

Mohseni, M., Ghaderi, R., Asvadi, A. & Ezoji, M. (2015). Selection of rbf neural network parameters based on normalized cut clustering. *Journal of Soft Computing and Information Technology*, **3**. 155

Monti, S., Tamayo, P., Mesirov, J. & Golub, T. (2003). Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, **52**, 91–118. 72

Moody, J. & Darken, C.J. (1989). Fast learning in networks of locally-tuned processing units. *Neural computation*, **1**, 281–294. 105, 108

Moosa, J.M., Shakur, R., Kaykobad, M. & Rahman, M.S. (2016). Gene selection for cancer classification with the help of bees. *BMC medical genomics*, **9**, 47.

Mullender, S., ed. (1993). *Distributed systems (2nd Ed.)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Mumford, E. (1987). Managerial expert systems and organizational change: some critical research issues. In *Critical issues in information systems research (incoll)*, 135–155, John Wiley & Sons, Inc., New York, NY, USA.

Nalisnick, E., Hertel, L. & Smyth, P. (2016). Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, vol. 2. 131, 135

Natarajan, A., Motani, M., de Silva, B., Yap, K. & Chua, K.C. (2007). Investigating network architectures for body sensor networks. In G. Whitcomb & P. Neece, eds., *Network Architectures*, 322–328, Keleuven Press, Dayton, OH.

NCBI repository (????). 49

NCI repository (????).

Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, **12**, 271–315.

Newman, D., Hettich, S., Blake, C. & Merz, C. (1998). Uci repository of machine learning databases.

Ng, H.W., Nguyen, V.D., Vonikakis, V. & Winkler, S. (2015). Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*, 443–449, ACM.

Nielson, F. (1985). Program transformations in a denotational setting. *ACM Trans. Program. Lang. Syst.*, **7**, 359–379.

Niyogi, P. & Girosi, F. (1996). On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Computation*, **8**, 819–842.

Novak, D. (2003). Solder man. In *ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part I - Vol. 145 (July 27–27, 2003)*, 4, ACM Press, New York, NY.

Nutt, C.L., Mani, D., Betensky, R.A., Tamayo, P., Cairncross, J.G., Ladd, C., Pohl, U., Hartmann, C., McLaughlin, M.E., Batchelor,

T.T. *et al.* (2003). Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Research*, **63**, 1602–1607. 114

Obama, B. (2008). A more perfect union. Video.

Okun, O. & Priisalu, H. (2007). Random forest for gene expression based cancer classification: overlooked issues. In *Iberian Conference on Pattern Recognition and Image Analysis*, 483–490, Springer. 76

OMIM repository (2005). Online mendelian inheritance in man. Https://www.omim.org/. 18

Oyelade, J., Isewon, I., Oladipupo, F., Aromolaran, O., Uwoghiren, E., Ameh, F., Achas, M. & Adebiyi, E. (2016). Clustering algorithms: Their application to gene expression data. *Bioinformatics and Biology insights*, **10**, 237.

Panchal, G., Ganatra, A., Kosta, Y. & Panchal, D. (2011). Behaviour analysis of multilayer perceptronswith multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, **3**, 332. 102

Park, J. & Sandberg, I.W. (1991). Universal approximation using radial-basis-function networks. *Neural computation*, **3**, 246–257. 99, 103

Parsaie, A. & Haghiabi, A.H. (2015). Predicting the longitudinal dispersion coefficient by radial basis function neural network. *Modeling Earth Systems and Environment*, **1**, 34.

Parsopoulos, K.E. & Vrahatis, M.N. (2002). Recent approaches to global

optimization problems through particle swarm optimization. *Natural computing*, **1**, 235–306. 15

Patil, S., Naik, G., Pai, R. & Gad, R. (2018). Stacked autoencoder for classification of glioma grade iii and grade iv. *Biomedical Signal Processing and Control*, **46**, 67–75.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, San Mateo, CA.

Perronnin, F. & Christopher, D. (2007). Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition*, 1–8, IEEE Conference. 25

Petricoin, E.F., Ardekani, A.M., Hitt, B.A., Levine, P.J., Fusaro, V.A., Steinberg, S.M., Mills, G.B., Simone, C., Fishman, D.A., Kohn, E.C. *et al.* (2002). Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, **359**, 572–577. 72

Petrie, C.J. (1986a). New algorithms for dependency-directed backtracking (master's thesis). Tech. rep., Austin, TX, USA.

Petrie, C.J. (1986b). *New Algorithms for Dependency-Directed Backtracking (Master's thesis)*. Master's thesis, University of Texas at Austin, Austin, TX, USA.

Philbin, J., Ondrej, C., Isard, M., Sivic, J. & Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE. 25

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training

support vector machines. *machines Technical Report MSR-TR-98-14, Microsoft Research*. 118

Poker-Edge.Com (2006). Stats and analysis.

Poli, R., Kennedy, J. & Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, **1**, 33–57. 15, 80, 81, 112, 113

Pomeroy, S.L., Tamayo, P., Gaasenbeek, M., Sturla, L.M., Angelo, M., McLaughlin, M.E., Kim, J.Y., Goumnerova, L.C., Black, P.M., Lau, C. *et al.* (2002). Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, **415**, 436–442. 72

Pratiwi, M., Harefa, J., Nanda, S. *et al.* (2015). Mammograms classification using gray-level co-occurrence matrix and radial basis function neural network. *Procedia Computer Science*, **59**, 83–91.

Prieto, A., Prieto, B., Ortigosa, E.M., Ros, E., Pelayo, F., Ortega, J. & Rojas, I. (2016). Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, **214**, 242–268. 45

Qian, Y., Dong, J., Wang, W. & Tan, T. (2015). Deep learning for steganalysis via convolutional neural networks. In *Media Watermarking, Security, and Forensics 2015*, vol. 9409, 94090J, International Society for Optics and Photonics. 104

Que, Q. & Belkin, M. (2016). Back to the future: Radial basis function networks revisited. In *AISTATS*, 1375–1383. 13, 107, 108, 155

Radford, A., Metz, L. & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. 46

Ravi, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B. & Yang, G.Z. (2017). Deep learning for health informatics. *IEEE Journal of Biomedical and Health Informatics*, **21**, 4–21. 44

Re, A., Mauro, C. & Leonardo, V. (2016). Population structure and particle swarm performance. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on 2002*, 163–185, Springer International Publishing.

Reid, B.K. (1980). A high-level approach to computer document formatting. In *Proceedings of the 7th Annual Symposium on Principles of Programming Languages*, 24–31, ACM, New York.

Reyes-Nava, A., Sánchez, J.S., Alejo, R., Flores-Fuentes, A. & Rendón-Lara, E. (2018). Performance analysis of deep neural networks for classification of gene-expression microarrays. In *Mexican Conference on Pattern Recognition*, 105–115, Springer.

Rifai, S., Vincent, P., Muller, X., Glorot, X. & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 833–840. 106

Rimé, B. & Schiaratura, L. (1991). Gesture and speech in fundamentals of nonverbal behavior.

Risinger, J.I., Maxwell, G.L., Chandramouli, G., Jazaeri, A., Aprelikova, O., Patterson, T., Berchuck, A. & Barrett, J.C. (2003). Microarray analysis reveals distinct gene expression profiles among different histologic types of endometrial cancer. *Cancer Research*, **63**, 6–11. 72

Rosenwald, A., Alizadeh, A.A., Widhopf, G., Simon, R., Davis, R.E., Yu, X., Yang, L., Pickeral, O.K., Rassenti, L.Z., Powell, J. *et al.*

(2001). Relation of gene expression phenotype to immunoglobulin mutation genotype in b cell chronic lymphocytic leukemia. *Journal of Experimental Medicine*, **194**, 1639–1648.

Rous, B. (2008). The enabling of digital libraries. *Digital Libraries*, **12**, to appear.

Roweis, S. (2012). sam roweis: data.[www page]. *URL http://www. cs. nyu. edu/˜ roweis/data. html*. 32

Ruslan, S., Mnih, A. & Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on machine learning*, 791–798, ACM. 27

Saeedi, M., Zamani, M.S. & Sedighi, M. (2010a). A library-based synthesis methodology for reversible logic. *Microelectron. J.*, **41**, 185–194.

Saeedi, M., Zamani, M.S., Sedighi, M. & Sasanian, Z. (2010b). Synthesis of reversible circuit using cycle-based approach. *J. Emerg. Technol. Comput. Syst.*, **6**.

Salas, S. & Hille, E. (1978). *Calculus: One and Several Variable*. John Wiley and Sons, New York.

Sanger, T.D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, **2**, 459–473.

Sathishkumar, E., Thangavel, K. & Chandrasekhar, T. (2013). A novel approach for single gene selection using clustering and dimensionality reduction. *arXiv preprint arXiv:1306.2118*.

Scheithe, J., Licandro, R., Rota, P., Reiter, M., Diem, M. & Kampel, M. (2019). Monitoring acute lymphoblastic leukemia therapy with stacked

denoising autoencoders. In *Computer Aided Intervention and Diagnostics in Clinical and Medical Images*, 189–197, Springer. 12

Schilling, R.J., Carroll, J.J. & Al-Ajlouni, A.F. (2001). Approximation of nonlinear systems with radial basis function neural networks. *IEEE Transactions on neural networks*, **12**, 1–15.

Scientist, J. (2009). The fountain of youth. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.

Sharif Razavian, A., Azizpour, H., Sullivan, J. & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 806–813. 25

Sharma, A. & Singh, S. (2016). Neural network for diagnosis of ovarian cancer based on proteomic patterns in serum. *Journal of Scientific and Technical Advancements*, **2**, 25–27.

Shi, Y. *et al.* (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, vol. 1, 81–86, IEEE. 15

Shin, H.C., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D. & Summers, R.M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, **35**, 1285–1298. 4

Shiokawa, Y., Date, Y. & Kikuchi, J. (2018). Application of kernel principal component analysis and computational machine learning to exploration of metabolites strongly associated with diet. *Scientific reports*, **8**, 3426. 2

Shu, R., Bui, H.H., Zhao, S., Kochenderfer, M.J. & Ermon, S. (2018). Amortized inference regularization. In *Advances in Neural Information Processing Systems*, 4393–4402. 135, 138

Shu, Y., Zhu, M., He, K., Hopcroft, J. & Zhou, P. (2017). Understanding deep representations through random weights. *arXiv preprint arXiv:1704.00330*.

Sims, K. (1991). Artificial evolution for computer graphics. *ACM*, **25**, 319–328.

Singh, D., Febbo, P.G., Ross, K., Jackson, D.G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A.A., D'Amico, A.V., Richie, J.P. *et al.* (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, **1**, 203–209. 72, 114, 144

Siriseriwan, W. (2016). smotefamily: A collection of oversampling techniques for class imbalance problem based on smote. 89

Smith, S.W. (2010). An experiment in bibliographic mark-up: Parsing metadata for xml export. In R.N. Smythe & A. Noble, eds., *Proceedings of the 3rd. annual workshop on Librarians and Computers*, vol. 3 of *LAC '10*, 422–431, Paparazzi Press, Milan Italy.

Smolik, M. & Skala, V. (2018). Large scattered data interpolation with radial basis functions and space subdivision. *Integrated Computer-Aided Engineering*, **25**, 49–62. 14

Spector, A.Z. (1990). Achieving application requirements. In S. Mullender, ed., *Distributed Systems*, 19–33, ACM Press, New York, NY, 2nd edn.

Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural net-

works from overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958. 68, 99, 102, 104

Steck, H., Krishnapuram, B., Dehing-oberije, C., Lambin, P. & Raykar, V.C. (2008). On ranking in survival analysis: Bounds on the concordance index. In *Advances in Neural Information Processing Systems*, 1209–1216. 48

Steinbach, M., Ertöz, L. & Kumar, V. (2004). The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, 273–309, Springer. 63

Sug, H. (2009). Performance comparison of RBF networks and mlps for classification. In *Proceedings of the 9th WSEAS International Conference on Applied Informatics And Communications (AIC '09)*. 102, 104

Suganthan, P.N. (1999). Particle swarm optimiser with neighbourhood operator. In *CEC 99. Proceedings of IEEE Congress on Evolutionary Computation*, 1931—-1938, IEEE. 15, 28

Svensson, P. (2016). Machine learning techniques for binary classification of microarray data with correlation-based gene selection. 68

Szymanski, L., McCane, B. & Atkinson, C. (2019). Switched linear projections and inactive state sensitivity for deep neural network interpretability. *arXiv preprint arXiv:1909.11275*.

Tan, C.C. & Eswaran, C. (2010). Reconstruction and recognition of face and digit images using autoencoders. *Neural Computing and Applications*, **19**, 1069–1079. 2

Tan, J., Ung, M., Cheng, C. & Greene, C.S. (2015). Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 20, 132, NIH Public Access. 43, 51, 54, 56

Tang, J., Alelyani, S. & Liu, H. (2014). Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, 37. 47

Tang, Y. & Salakhutdinov, R.R. (2013). Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems*, 530–538.

Tao, K.M. (1993). A closer look at the radial basis function (RBF) networks. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, 401–405, IEEE. 108

Teng, P. (2018). Machine-learning quantum mechanics: Solving quantum mechanics problems using radial basis function networks. *Physical Review E*, **98**, 033305. 14

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, **8**, 257–277.

Theis, L., Oord, A.v.d. & Bethge, M. (2016). A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR 2016)*, 1–10. 131, 135

Thornburg, H. (2001). Introduction to bayesian statistics.

Titus, A.J., Bobak, C.A. & Christensen, B.C. (2018). A new dimension of breast cancer epigenetics. 49, 50, 54, 55, 56

Tomczak, K., Czerwińska, P. & Wiznerowicz, M. (2015). The cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemporary oncology*, **19**, A68. 49, 70, 144

TUG (2017). Institutional members of the TEX users group.

Tzamaloukas, A. & Garcia-Luna-Aceves, J.J. (2000). Channel-hopping multiple access. Tech. Rep. I-CA2301, Department of Computer Science, University of California, Berkeley, CA.

Van den Bergh, F. (2001). An analysis of particle swarm optimizers [ph. d. thesis]. *Pretoria: Natural and Agricultural Science Department, University of Pretoria*. 95

Van den Bergh, F. & Engelbrecht, A.P. (2006). A study of particle swarm optimization particle trajectories. *Information sciences*, **176**, 937–971. 14

Vedaldi, A.M.A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5188–5196.

Verhaak, R.G., Wouters, B.J., Erpelinck, C.A., Abbas, S., Beverloo, H.B., Lugthart, S., Löwenberg, B., Delwel, R. & Valk, P.J. (2009). Prediction of molecular subtypes in acute myeloid leukemia based on gene expression profiling. *Haematologica*, **94**, 131–134. 72

Veytsman, B. (????). acmart—Class for typesetting publications of ACM.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzago, P.A. (2010a). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, **11**, 3371–3408. 24, 25

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.A. (2010b). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, **11**, 3371–3408. 2, 10, 11, 12, 21, 80, 106, 117

Walczak, S. & Cerpa, N. (1999). Heuristic principles for the design of artificial neural networks. *Information and software technology*, **41**, 107–117. 26

Wang, X. & Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802. 53

Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, A.M., Look, M.P., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M.E., Yu, J. *et al.* (2005). Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *The Lancet*, **365**, 671–679. 72

Way, G.P. & Greene, C.S. (2017). Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *bioRxiv*, 174474. 50, 54, 55

Weiss, G.M. (2004). Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, **6**, 7–19. 78

Wenzel, E.M. (1992). Three-dimensional virtual acoustic displays. In *Multimedia interface design (incoll)*, 257–288, ACM, New York, NY, USA.

Werneck, R., Setubal, J.a. & da Conceicão, A. (2000a). (new) finding minimum congestion spanning trees. *J. Exp. Algorithmics*, **5**.

Werneck, R., Setubal, J.a. & da Conceicão, A. (2000b). (old) finding minimum congestion spanning trees. *J. Exp. Algorithmics*, **5**, 11.

Woodward, W.A., Krishnamurthy, S., Yamauchi, H., El-Zein, R., Ogura, D., Kitadai, E., Niwa, S.i., Cristofanilli, M., Vermeulen, P., Dirix, L. *et al.* (2013). Genomic and expression analysis of microdissected inflammatory breast cancer. *Breast cancer research and treatment*, **138**, 761–772. 72

Woolley, B.G. & Stanley, K.O. (2011). A comparison between representations for evolving images. 957–964, ACM.

Wu, Q., Boueiz, A., Bozkurt, A., Masoomi, A., Wang, A., DeMeo, D.L., Weiss, S.T. & Qiu, W. (2018). Deep learning methods for predicting disease status using genomic data. *Journal of biometrics & biostatistics*, **9**.

Xiao, Y., Wu, J., Lin, Z. & Zhao, X. (2018). A deep learning-based multi-model ensemble method for cancer prediction. *Computer Methods and Programs in Biomedicine*, **153**, 1–9. 50, 51, 61

Xu, L. (2010). Learning algorithms for RBF functions and subspace based functions. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 60–94, IGI Global. 108

Xue, B., Zhang, M., Browne, W.N. & Yao, X. (2016). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, **20**, 606–626. 43

Yagi, T., Morimoto, A., Eguchi, M., Hibi, S., Sako, M., Ishii, E., Mizutani, S., Imashuku, S., Ohki, M. & Ichikawa, H. (2003). Identification of a gene expression signature associated with pediatric aml prognosis. *Blood*, **102**, 1849–1856. 72

Yeoh, E.J., Ross, M.E., Shurtleff, S.A., Williams, W.K., Patel, D., Mahfouz, R., Behm, F.G., Raimondi, S.C., Relling, M.V., Patel, A. *et al.* (2002). Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, **1**, 133–143. 72

Yousefi, S., Song, C., Nauata, N. & Cooper, L. (2016). Learning genomic representations to predict clinical outcomes in cancer. *arXiv preprint arXiv:1609.08663*. 17, 66

Yousefi-Azar, M., Varadharajan, V., Hamey, L. & Tupakula, U. (2017). Autoencoder-based feature learning for cyber security applications. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, 3854–3861, IEEE. 117

Yu, X., Yu, G. & Wang, J. (2017). Clustering cancer gene expression data by projective clustering ensemble. *PloS One*, **12**, e0171429. 44, 62, 65, 67

Yu, Z., Chen, H., You, J., Li, L. & Han, G. (2012). Som 2 ce: Double self-organizing map based cluster ensemble framework and its application in cancer gene expression profiles. In *IEA/AIE*, 351–360, Springer. 65

Yu, Z., Chen, H., You, J., Liu, J., Wong, H.S., Han, G. & Li, L. (2015). Adaptive fuzzy consensus clustering framework for clustering analysis of cancer data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, **12**, 887–901. 44, 62, 64, 65

Yuan, Y., Shi, Y., Li, C., Kim, J., Cai, W., Han, Z. & Feng, D.D. (2016). Deepgene: an advanced cancer type classifier based on deep learning and somatic point mutations. *BMC Bioinformatics*, **17**, 476. 50, 52, 58, 59, 67

Zeiler, M.D. & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833, Springer. 154

Zhang, D., Zou, L., Zhou, X. & He, F. (2018). Integrating feature selection and feature extraction methods with deep learning to predict clinical outcome of breast cancer. *IEEE Access*. 49, 50, 51, 55, 56

Zhang, G.P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **30**, 451–462.

Zhao, S., Song, J. & Ermon, S. (2019). Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 5885–5892. 131

Zheng, Z., Wu, X. & Srihari, R. (2004). Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter*, **6**, 80–89. 78

Zhou, G., Lu, J., Wan, C.Y., Yarvis, M.D. & Stankovic, J.A. (2008). *Body Sensor Networks*. MIT Press, Cambridge, MA.

Zhou, G., Wu, Y., Yan, T., He, T., Huang, C., Stankovic, J.A. & Abdelzaher, T.F. (2010). A multifrequency mac specially designed for wireless sensor network applications. *ACM Trans. Embed. Comput. Syst.*, **9**, 39:1–39:41.

Zhu, X., Huang, Z., Yang, Y., Shen, H.T., Xu, C. & Luo, J. (2013). Self-taught dimensionality reduction on the high-dimensional small-sized data. *Pattern Recognition*, **46**, 215–229.

Zhu, Z., Zhou, J., Ji, Z. & Shi, Y.H. (2011). Dna sequence compression using adaptive particle swarm optimization-based memetic algorithm. *IEEE Transactions on Evolutionary Computation*, **15**, 643–658. 15

Zhu, Z., Wang, X., Bai, S., Yao, C. & Bai, X. (2016). Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, **204**, 41–50.

Zhuotun, Z., Wang, X., Bai, S., Yao, C. & Bai, X. (2016). Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, **204**.