Achieving High Mean Accuracy with Semi-Supervised Learning using Small Number of Labeled Observations

Submitted by: Sayed Waleed Qayyumi
School of Computing, Engineering and Mathematics
Western Sydney University
Parramatta, NSW 2150, Australia

August 10, 2019

PhD Candidature
CONFIRMATION REPORT

Supervisors:
Dr Laurence Park
Associate Professor Oliver Obst

School of Computing, Engineering and Mathematics
Western Sydney University
Parramatta, NSW 2150, Australia

Confirmation Report in Fulfilment of Confirmation of PhD Candidature

Contents

1 EXECUTIVE SUMMARY

Classification is a field of machine learning that has to do with grouping data based on a specific criteria. This is also referred to as supervised learning. Technology has made data collection easy and with the abundance of data that is now available for usage, classification methods are becoming more important for decision making. While, we use complex algorithms to classify items, it is something that we humans do naturally. In machine learning, classification algorithms have very wide usage and can be applied to many daily life problems. Image classification, Music identification, Event classification and Speech tagging are some of the important application of classification algorithms. Many simple and complex classification algorithms such as Linear Regression, Perceptron, Naive Bayes Classifier, Decision Trees, Neural Networks and Support Vector Machines are already used extensively in various fields of our daily life. While, these techniques have enriched our ways of work, there are many challenges that are still unsolved. High mean accuracy, availability of labeled data and dimensional distribution of data are some of the main challenges in this field.

From a user point of view, achieving a high mean accuracy is always the first preference. The cost of using classification algorithms can only be justified if we achieve a high mean accuracy. For example, predicting a payment system failure for a bank, with high accuracy could save a bank big amount of money and human resource time but, the challenge in this type of problems is the availability of labeled data. There are many such scenarios where accurate classification or prediction is required but it is not achievable for various reasons. Similarly, it is not always possible to have many labeled data point for training classification model thus we require classification algorithms that can deal with small number of labeled data and returns a high mean accuracy.

Another challenge with classification algorithms is the manifold assumption. Classification algorithms assume that data lies in the low manifold and hence fails to achieve high mean accuracy when used with a manifold distributed data. The objective of our research is to optimise or extend existing methods so that they can achieve high mean of accuracy using small number of labeled observations. We are also intending to introduce new methods that can classify a manifold distributed data with small number of labeled observations and still achieve high mean of accuracy.

Our research will focus mainly on semi-supervised learning. To achieve the overall research objectives, three different categories of classification will be discussed in detail in this report. This will be followed by our research and contribution to the mentioned fields.

1. Supervised learning, which is classification of new observations using full labeled observations.

2. Unsupervised learning or clustering, is classification of observations that has no label information.

3. Semi-supervised learning, which is classification of new observation using limited number of labeled observations.

The intention is to build over the existing state of art and where applicable come up with new ideas and algorithms.

2 INTRODUCTION

Classification is allocating objects in to classes and groups [14]. Classification or supervised learning supposes a relationship R between elements of a set E. Classification remained the work of natural scientist for long and was used by them to classify different organism in to different classes. Libraries used various classification methods to classify books in to various groups. While classification was used by natural scientists in libraries, it got more attraction in 1960s when a Belgian and Polish mathematician [1] published a paper on the subject of classification. This was followed by a book from Birkhoff [2]. The Introduction of Numerical Taxonomy by P. H. Sneath and R. R. Sokal [33] followed suit. Picking up momentum in 1970s, classification emerged as one of the most researched and valuable field of mathematics and statistics. Later, machine learning gave classification a new boost and introduced practical usage of classification outside mathematics and statistics. Now, classification has a very wide usage in our day to day life. Simple tasks as classification of product to product groups or complex tasks like comparing different traits of organism in taxonomy, use classification algorithms. Machine learning also introduced many new usages for these algorithms. For example, assigning an email to spam or not a spam class or diagnosing a patient's disease based on observed data from other patients.

Classification is a form of pattern recognition and is also referred to as supervised learning. An algorithm that can classify is called classifier. While we have come a long way with respect to improving the classification algorithm's performance, we still have many challenges in this field. Mean accuracy of classification, unavailability or partial availability of labeled data and the manifold distribution of data are some of the challenges that we still need to resolve. Normally, a classifier's performance is partially dependent on the accuracy and availability of labeled data. The bigger the size of labeled training dataset, the better the classifier. Unfortunately, we do not have labeled data available in most of the scenarios or it is too expensive to label. Hence, there is a big demand for classifier that can classify using small number of labeled observation with a high mean accuracy. In machine learning, classification or supervised learning is about learning a mapping function between input and output variables. Imagine an input variable X and an output variable Y, we map a function between input and output as per below.

$$Y = \hat{f}(X) \tag{1}$$

The objective in here is approximation of the mapping function f. If the mapping function f is well approximated; an output variable Y can be predicted with high accuracy based on new input X. The process is called supervised learning as the mapping function is supervised to learn using training data. The learning only stops when the performance reaches a specific acceptable threshold. Supervised learning is divided in to two types; Classification and Regression. The difference between these two methods is in the output they predict. Classification is prediction of categorical variables while Regression is prediction of numerical variables.

Contrary to supervised learning, in unsupervised learning there is no output variable Y. Input variable X is used to understand the underlying data structure followed by clustering the same data in to groups. In semi-supervised learning, we have input variable X and very limited instances of output variable Y. Semi-supervised learning sits between supervised learning and unsupervised learning. The input variable X and available label Y is used alongside the unlabeled observations to determine a classifier function. A good example of semi-supervised learning is labeling photos in an archive. In such archives, some photos are labeled and majority are unlabeled. It is quite expensive and time-consuming to label unlabeled observations and hence most of machine learning algorithms fail in this type of classification. This is why semi-supervised learning is considered more challenging than supervised and unsupervised learning. Taking this as a challenge, our research and contribution will be mainly in the area of semi-supervised learning. Our focus will be to optimise existing techniques or introduce new semi-supervised techniques.

3 LITERATURE REVIEW

The literature review will begin with the review of current state of supervised, unsupervised and semi-supervised learning. This will be followed by a detail review of each mentioned method. As our research focus is semi-supervised learning, the semi-supervised section will include our research and contributions as well. We are specifically looking at the graph based approaches to semi-supervised learning. Research on graph based semi-supervised learning has shown promising results. These methods use a concept called random walk over graph. Graph based methods are good with classification of a manifold distributed data as well as classification using very small number of labeled observation due to the way they calculate the distance. The aim of studying graph based methods is to discover the reasons for graph based methods high mean accuracy using small number of labeled observations. Follow on from this study, focus will be mainly on image classification. Image classification is a challenging field of research. While image classification / identification is a very active area of research and convolutional networks have performed very well, there has been no major break throughs in this area as yet.

3.1 Supervised Learning

In supervised learning, model learns from labeled datasets and then apply that learning to label a new data point. Supervised learning [3] is divided in to two types i.e. Classification and Regression. The difference between classification and regression is the type of output. Classification predicts a class label of an unlabeled observation while regression predicts a numerical label. The objective is to come up with a mapping from x to y given a training set that is made up of (x_i, y_i). y_i is the label of x_i. The function that is created using training set can be evaluated through prediction of the same function on a test set. Supervised learning algorithms are generally divided in to two families i.e. generative and discriminative algorithms. The generative models [16, 24] learn the joint probability distribution $p(x, y)$ while discriminative models learn the conditional probability distribution $p(x|y)$ [6].

In discriminative models, to predict label y from a new observation x, we have to evaluate the class using below equation. The equation choses the most likely class for y given x.

$$f(x) = \arg\max_y p(y|x) \tag{2}$$

We know from Bayes rule that $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$. Replacing the $p(x|y)$ in the discriminative model equation with Bayes rule will result in to below equation. As the equation is only interested in arg max, denominator is constant, and so can be disregarded when identifying the argmax.

$$f(x) = \arg\max_{y} p(y|x)p(x) \tag{3}$$

This is the equation used in generative model. In discriminative models, conditional probability distribution $p(x|y)$ is used to model the class boundary while the joint probability distribution models the actual distribution of each class. Given a label y, generative models can generate its respective x and that is why they are called generative models.

3.2 Unsupervised Learning

Unsupervised learning helps in identification of unknown structures in data and does not require data to have labels. There are many different unsupervised methods available but the most common one is clustering. Clustering is to group or segment dataset based on common attributes or behaviours [25, 36]. Unsupervised learning is mainly used to estimate density in a dataset [15]. As mentioned earlier, supervised methods use the information of conditional probability distribution $p(x|y)$ where y is the label and x is the input data; unsupervised learning works based on apriori probability distribution $p(x)$.

3.3 Semi-Supervised Learning

Semi-supervised learning [11] also called (SSL) is a type of learning that lies between supervised and unsupervised learning. Semi-supervised learning uses both labeled and unlabeled data with the intent to form higher accuracy models when compared to those using only labeled data [3, 11]. Mathematically, it can be presented as, $X = (x_i)_{i \in [n]}$ which is divided in to labeled data $X_l = (x_1, ..., x_l)$ that has labels $Y_l = (y_1, ..., y_l)$ and unlabeled data $X_u = (x_l + 1, ..., x_l +_u)$ where no label information is provided. One of the questions that is normally asked regarding semi-supervised learning is, does using the labeled and unlabeled data provide additional accuracy when compared to using the labeled data alone? In short, the answer is yes but it is very strongly correlated to the distribution of labeled dataset and its relevance to the classification problem it is trying to solve [6]. In other words, the $p(x)$ from unlabeled dataset should be useful in inference of $p(y|x)$ from labeled dataset for a semi-supervised method to perform better than the supervised method. If above is not the case, semi-supervised learning may degrade the mean accuracy of classification. Semi-supervised learning algorithms share some common assumptions [32]. The methods can provide high accuracy only when these assumptions hold true. Some of the common assumptions used by semi-supervised learning is listed down below.

1. Smoothness Assumption: If two observations x_1 and x_2 are close to each other then so should be their respective labels y_1 and y_2.

2. Cluster Assumption: If two points x_1 and x_2 share the same cluster then most probably they have same class.

3. Manifold Assumption: High dimensional data lies on a low dimensional manifold.

As mentioned earlier, most of the semi-supervised methods work only if all of the above mentioned assumptions hold true. Graph based methods, explained in section 3.3.2 have the flexibility and capacity to avoid one or more of these assumptions and still achieve a high mean accuracy of classification. This is also our active area of research. Our existing work on this subject which is ready for publication is presented at the end of section 3.3.2.

3.3.1 Self-Learning

The idea of using unlabeled observations along with labeled observations to improve prediction is not new [29, 34]. There was always a need to use the available unlabeled observations to enrich training or in other words improve mean accuracy of classification. This desire has led researchers to use the unlabeled observations in many different ways. Self-learning [35, 19] is also referred to as self-training, self labeling in different texts [6], is a very simple approach to use unlabeled observations. Self-training is a wrapper algorithm that can use different supervised learning methods. Self-training is initiated using a labeled dataset and model is trained using the labeled dataset. The model is then applied to some of the unlabeled observations and label is predicted. The prediction is retained which is then used to train the model again and the process is repeated until all the unlabeled observations are labeled. The accuracy of self-learning is dependent on the supervised algorithm used and criteria adopted. The criteria could be either risk minimization [18] or margin maximization [20]. In margin maximization, the decision boundary is extended due to unlabeled observations. Self-learning does not guarantee an improvement in accuracy of the model and hence contradicts the whole objective of using unlabeled observations. We performed an experiment on Iris dataset to validate the fact that self-learning with margin maximization does not guarantee an improvement in classification mean accuracy. We choose kNN as the supervised method in our experiment. kNN algorithm forms a majority vote between the k most similar observations and an unseen observation. Similarity is defined by a distance metric such as Euclidean distance.

```
  kNN (dataset, sample)
{
   1. Go through each item in my dataset and calculate the distance from that data item to
      my specific sample.
   2. Classify the sample as the majority class between K samples in the dataset having
      minimum distance to the sample.
}
```

If we consider X as the matrix of features for an observation and Y is the class label, kNN looks at k (a positive integer) and estimates the probability of it belonging to class j for a test observation x_0

$$Pr(Y = j|X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{4}$$

Table 1: Impact of unlabeled data points on mean accuracy of classification - Iris dataset

$\#_{TR}$	$\phi_{TR}acc$	$\#_{UD}$	$\#_{ETR}$	$\phi_{ETR}acc$
40	0.963	20	60	0.944
20	0.944	20	40	0.973
25	0.95	25	50	0.95
15	0.958	15	30	0.925
30	0.944	30	60	0.944
40	0.95	30	70	0.975

Self-learning kNN [13] involves predicting class of unlabeled observations. A model is trained using kNN and a training set TR. Unlabeled set UD is then labeled using the predicted class. The unlabeled points that have high confidence of prediction are then added to training set TR along with their predicted class. The extended training set ETR = TR+UD is then used to classify other unlabelled points. See Table 1 for the impact of adding unlabeled points on mean accuracy of prediction using kNN. Mean accuracy of prediction is calculated before and after the addition of unlabeled points. Mean accuracy is calculated by taking the mean of accuracy of random observations of 30 different iterations. It is very obvious from the results that self-learning may or may not improve mean accuracy [27].

3.3.2 Graph Based Methods

Research in graph based methods is highly active area in semi-supervised learning. A graph is constructed where each node in the graph is associated to a data point and edges between nodes exist if the associated pair of points are neighbours (with respect to some distance threshold). The edges of the graph are weighted to indicate the inverse distance (or similarity) between the associated two data points. The distance between any two points in the data is calculated by minimizing the path distance over all possible paths connecting the two points. Graph distance is used as an approximation of geodesic distance between two points on a manifold. Let's assume a graph $G(V, E)$ where V is the set of graph vertices and E is the set of graph edges. Weight is represented by $w(e_{ij})$ for each $e_{ij} \in E$. The edge weight $w(e_{ij})$ represents the local similarity between points i and j. The weight matrix of this graph G is defined by

$$w(e_{ij}) \text{ if } e_{ij} \in E, 0 \text{ otherwise} \tag{5}$$

As described in section 3.3, semi-supervised learning is mostly useful when we have a high proportion of unlabeled observations. At times the cost of labeling observation is too high or requires a lot of effort. Thus it makes sense to use unlabel observations. We need a high proportion of unlabeled observations, to be able to use them to increase accuracy of classification. This is due to the fact that unlabeled observations contain less information as compared to labeled ones. Processing of unlabeled observations is computationally expensive due to high volumes. This in turn requires faster semi-supervised algorithms to process the large number of unlabeled data. Graph based algorithms use graphs to calculate the similarity between two data points. The graph creation process is computationally expensive [4]. It is not feasible to create a graph for each labeled observation to identify the neighbourhood. Thus most of the graph based algorithms available today struggle with the cost of processing large volume of data [26]. This means graph based methods are hard to implement and not efficient in real life problems. As part of solving this problem we have introduced a new method of using graph that is very light on compute cost while still takes advantage of graph based similarity calculation. We have called our method *cgk*NN which is explained in detail in section 4.1.

*cgk*NN works with small number of labeled observations and produces high mean accuracy with lower computational cost. Our method can be classified as a graph based semi-supervised learning method. Graph based algorithms build graphs whose nodes are labeled and unlabeled data points. Labeled data points are used to spread information to unlabeled data points. In these methods, graphs $g = (V, E)$ where V represent a node and E an edges is used to represent the geometry of the data. E represents similarity between two edges. Similarities are shown using a weight matrix W.

3.4 Image Classification

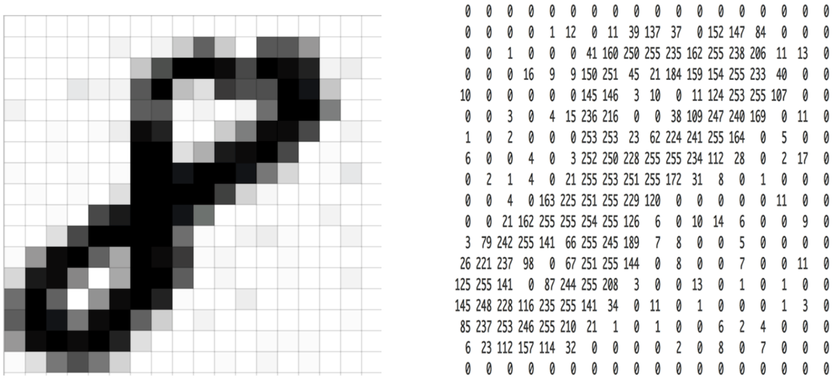
Image classification refers to processes that are used to identify objects within an image or classify image in to a group. Huge amount of data is getting generated daily using image sharing apps and platforms. Companies are interested in getting value out of their huge repositories of digital data to deliver better and smarter services to their customers.

Image classification and recognition [10] is part of computer vision which is a very active area of research and is driving many fields. In automotive industry, Image recognition is now used for developing driverless cars. In gaming industry, image classification and recognition is used to develop the next generation of games. Facebook's famous face recognition algorithm can recognise face with 98% accuracy. In short, image recognition and classification can drive automation in many industries.

3.4.1 Deep Learning

Deep learning is one of the widely used method for Image classification [17]. While neural networks are covered in section 3.6, we are covering the part that is relevant to image classification. Similar to a brain structure, neural networks are a group of neurons that are connected to each other. Nodes within neural networks are referred to as neurons. Each neuron takes an input, process it and then passes its output to next neuron. The output of previous neuron is an input to the next layer of neurons till the final output is generated. Neural network algorithms are trained using training datasets which is a set of labeled images and then new images are feed to the same algorithm for classification. Computers store images as metrics and hence can compare them easily. Another approach will be to convert images to numeric data for comparison. There are two common methods to convert an image to numerical dataset.

1. Greyscale: Image is converted to greyscale. Greyscale conversion means that the picture will be converted to shades of color from white to black. Computer then assigns a number to each pixel based on how dark the pixel is. Please refer to Figure 1 for the conversion of image data to equivalent numerical data.



(a) An image of number eight

```
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   1  12   0  11  39 137  37   0 152 147  84   0   0   0
  0   0   1   0   0   0  41 160 250 255 235 162 255 238 206  11  13   0
  0   0   0  16   9   9 150 251  45  21 184 159 154 255 233  40   0   0
 10   0   0   0   0   0 145 146   3  10   0  11 124 253 255 107   0   0
  0   0   3   0   4  15 236 216   0   0  38 109 247 240 169   0  11   0
  1   0   2   0   0   0 253 253  23  62 224 241 255 164   0   5   0   0
  6   0   0   4   0   3 252 250 228 255 255 234 112  28   0   2  17   0
  0   2   1   4   0  21 255 253 251 255 172  31   8   0   1   0   0   0
  0   0   4   0 163 225 251 255 229 120   0   0   0   0   0  11   0   0
  0   0  21 162 255 255 254 255 126   6   0  10  14   6   0   0   9   0
  3  79 242 255 141  66 255 245 189   7   8   0   0   5   0   0   0   0
 26 221 237  98   0  67 251 255 144   0   8   0   0   7   0   0  11   0
125 255 141   0  87 244 255 208   3   0   0  13   0   1   0   1   0   0
145 248 228 116 235 255 141  34   0  11   0   1   0   0   0   1   3   0
 85 237 253 246 255 210  21   1   0   1   0   0   6   2   4   0   0   0
  6  23 112 157 114  32   0   0   0   0   2   0   8   0   7   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

(b) Greyscale equivalent

Figure 1: How computers translate images to greyscale

2. RGB: Colors can also be presented as RGB colors. Computers assigns a value to each pixel depending on the RGB value of each pixel.

Deep learning is often compared to human mind. There is an understanding that the field will continue to advance and enter many other domains. This has caused the fear that this may result in unemployment and even to slavery. There is no doubt that deep learning are efficient in performing tasks but they are not mean to solve all problems. There are many limitations and challenges with deep learning that prevents it from competing with other technologies. Gary Markus in [28] lists down the challenges with deep learning. He says "In a world with infinite data, and infinite computational resources, there might be little need for any other technique". We know well that we do not live in such world. We shall never end up with a labeled sample for every problem space in deep learning. Thus, we will have to generalise. Deep learning cannot learn from abstractions and only works best when there are labeled samples available. Deep learning has a heavy reliance on availability of correct and large number of labeled data. Gary also compares deep learning to a black box that learn correlation or patterns by shifting different data points and combinations. It is quite complex to decode the work of a neural network. This limits its usage in the fields where humans might want to know how a system makes a specific decision.

3.5 Metric Learning

Distance metric learning is a process whereby a learned metric is formed before it is feed to a classifier / predictor algorithm. A metric obeys four basic assumptions.

1. Non-negativity: $d(x, x') \geq 0$.
2. Identity: $d(x, x') = 0$.
3. Symmetry: $d(x, x') = d(x', x)$.
4. Triangle Inequality: $d(x, x'') \geq 0$.

Large margin nearest neighbour LMNN [39] can be stated as a convex optimisation problem. This capability enables LMNN to find a global solution efficiently. The objective of the algorithm is to learn a decision rule that can group data in to pre-defined classes. In LMNN, same class neighbours are put together while imposters or data points with different classes are pushed away from each other. Figure 2 shows the metric learning process with LMNN.

$$S = \{(x_i, x_j) : Y_i = y_j \text{ and } x_j \text{ belongs to the k-neighbourhood of } x_i\},$$
$$R = \{(x_i, x_j, x_k) : (x_i, x_j) \in S, y_i \neq y_k\}.$$

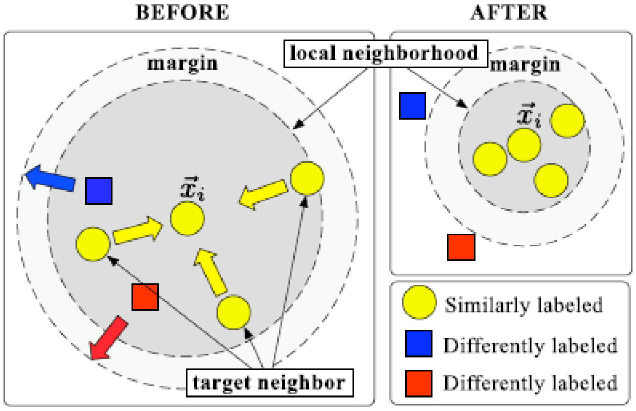


Figure 2: LMNN - Target neighbours and Imposters

3.6 Neural Networks

Neural networks are group of algorithms that are designed based on the structure and functionality of neurons in human brain. Neural networks are mostly used for pattern recognition. Neural networks only accept numerical data only. Images, sound and text data needs to be translated to numerical data before it can be used by a neural network algorithm. Please refer to Figure 1 for the conversion of image data to equivalent numerical data. The main usage of Neural network is in the field of supervised and unsupervised learning but they can also be used to extract features. Neural networks are made up of multiple layers of nodes. A node in different layer will turn on or off depending on the input it receives. Figure 3 shows the structure of a node within a Neural network. A node is made up of an inputs, weights, input function, activation function and an output.

As mentioned earlier, a layer in neural network is a row of nodes. Layer are classified in to three types; an input layer, a hidden layer and an output layer. Each layer has an input and output. Each layer's output is an input for the next layer until a final output is produced. Figure 4 shows layers of a neural network and how they are connected. Neural networks have wide usage in different industries. Speech recognition [9], computer vision [7], Pattern recognition [30] and financial crime detection [31] are some of the areas where neural network helped solved many problems.

Figure 5 shows artificial neural network classified in to different types [8]. There is two main classes of artificial neural networks (Feed Forward NN, Feed backward NN). In FFNN,

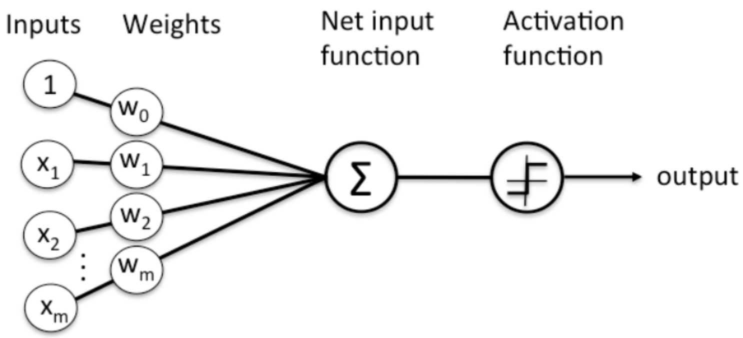


Figure 3: Structure of a node in a Neural network

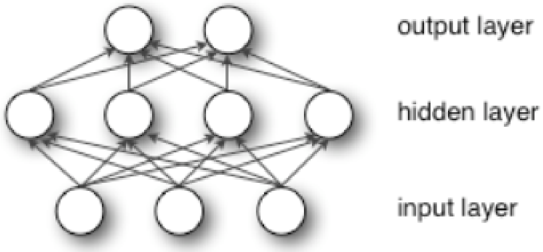


Figure 4: Layers of a Neural network

information transmits in one direction. Information is feed from input node and then passed to hidden node and finally passed on to output node [21]. In feedback NNs, backpropagation between nodes produced a coordinated graph in sequence.

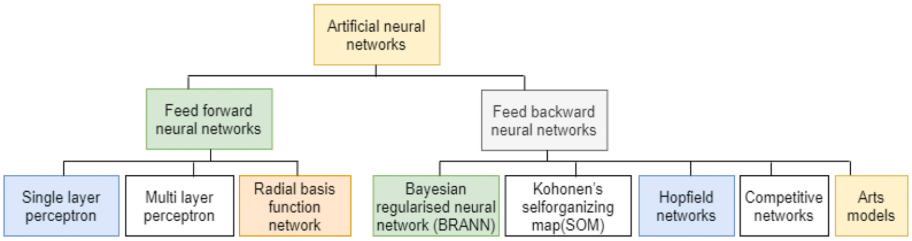

Artificial neural networks
Feed forward neural networks
Feed backward neural networks
Single layer perceptron
Multi layer perceptron
Radial basis function network
Bayesian regularised neural network (BRANN)
Kohonen's selforganizing map(SOM)
Hopfield networks
Competitive networks
Arts models

Figure 5: Classification of Artificial Neural Networks

3.7 Reducing Dimensions

Dimension reduction refers to the techniques of reducing the dimensionallity of data while minimising any changes to the data. With the advancement of technology, we are now able to capture more data than ever. This has helped decision makers to make effective data driven decisions. While, availability of data has helped, it has come with a curse. Unwanted data or variables that do not add much value while analysts are analysing data for a specific problem, can result in unnecessary complication. In other words, analysing many variables is not easy task and hence it is a value add exercise to analyse only a subset of relevant variables with respect to problem in hand. With the increase in data capture activities and the availability of different types of data such as images, videos, apps within smart phones, social media data or sentiments, it is very important we reduce data before analysing it. This will help in understanding data and factors influencing a specific objective. There are many dimension reduction techniques available and this area is also an active research field. Figure 6 shows a three dimensional dataset getting converted to a two dimensional dataset with minimal loss of information. The main objectives of each technique is to reduce dimensions with minimal information loss. All techniques available for dimension reduction are group in to two categories i.e. feature selections or feature reductions techniques [5].

Some of the common methods are listed below.

1. Low Variance: Variance based methods, remove the dimensions that have low variance with respect to other methods.

2. High correlation: Variables that have high correlations also does not add much value. Instead one can use just one of such highly correlated variables. Correlation matrix of all variables can identify all variables with high correlations.

3. Backward Feature Elimination: In this technique sum of square error is calculated after eliminating each variable. The objective is to find the variable with smallest increase in sum of square error.

4. Forward Feature Selection: In this technique, we compare the performance of model by adding one variable at a time. Variables with highest improvements in performance is selected.

5. Factoring: Correlated variables are grouped together to create a single variable from two or more correlated variables. Methods such as Exploratory or Confirmatory factor analysis can be used.

6. Principal Component Analysis: Existing variables are transformed to new variables which are linear combination of original variables. The components are created in such a way that the first component accounts for most of the possible variations of original data. The second component is orthogonal to the first component. This is to capture the variation not captured by the first component.

The field of semi-supervised learning encapsulates many concepts and methods. To review each of these concepts in details, one requires more time. The intention is to extend this literature review in next two years as we solve our research problem.

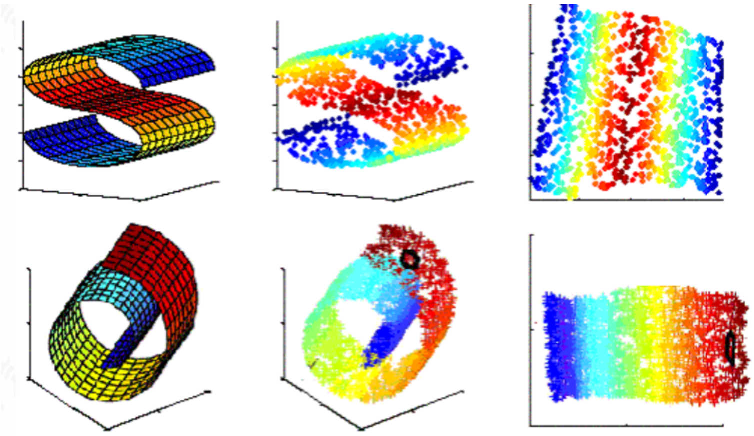


Figure 6: A 3D to 2D conversion - Locally Linear Embedding

4 WORK IN PROGRESS

4.1 Constrained Graph kNN

k Nearest Neighbours kNN has been widely used for classification owing to its simplicity and accuracy. In spite of the wide usage, kNN has performed poorly with a manifold distributed data. Few extensions of kNN are available that deal with the manifold distributed data but the extensions are either costly to compute or improvement in term of mean accuracy of classification is not significant [37]. Classification of a manifold distributed data requires prior knowledge of the shape of data distribution. The shape information is used to assess a distance or similarity function. To help resolve some of these problems, we are introducing a new semi-supervised algorithm which we are referring to as constrained graph kNN (cgkNN). Our method can be used for traditional Gaussian distributed data classification as well as for a non-linear manifold distributed data classification. Inspired by a method called manifold kNN (mkNN) which is one of the best method for classification of a manifold distributed data as well as classification using small number of labeled observations, our method works in similar way but always outperforms mkNN with respect to compute time. Our method also performs as good as, and at times better than mkNN with respect to mean accuracy of classification and classification using small number of labeled observations.

Graph based classification methods are more accurate than traditional methods at classification of a manifold distributed data and have shown higher mean accuracy of classification [37, 36]. kNN [13, 40], k-means [36] and many other algorithms have been extended using graphs to deal with a manifold distributed data. To understand why graph based methods perform better with a manifold distributed data, we need to answer two important questions.

1. What is a manifold distributed data?

2. What is random walk?

Definition: A data set is considered to be a manifold distributed if its intrinsic dimension is less than its data space dimension [37]. A manifold is very different from dimensionality.

Dimensionality refers to the number of variables or coordinates used to represent the data. The shape in Figure 7 shows a two dimensional manifold represented in three dimensional space. Any point on the manifold can be represented with two coordinates, but without knowledge of the manifold, they must be represented using three coordinates. In short, we define a manifold as a continuous geometrical structure that has fixed number of dimensions. The number of intrinsic dimension of a manifold distributed data is less than its data dimension.

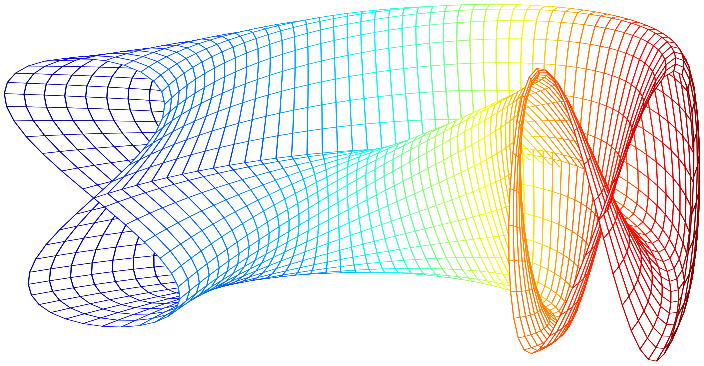


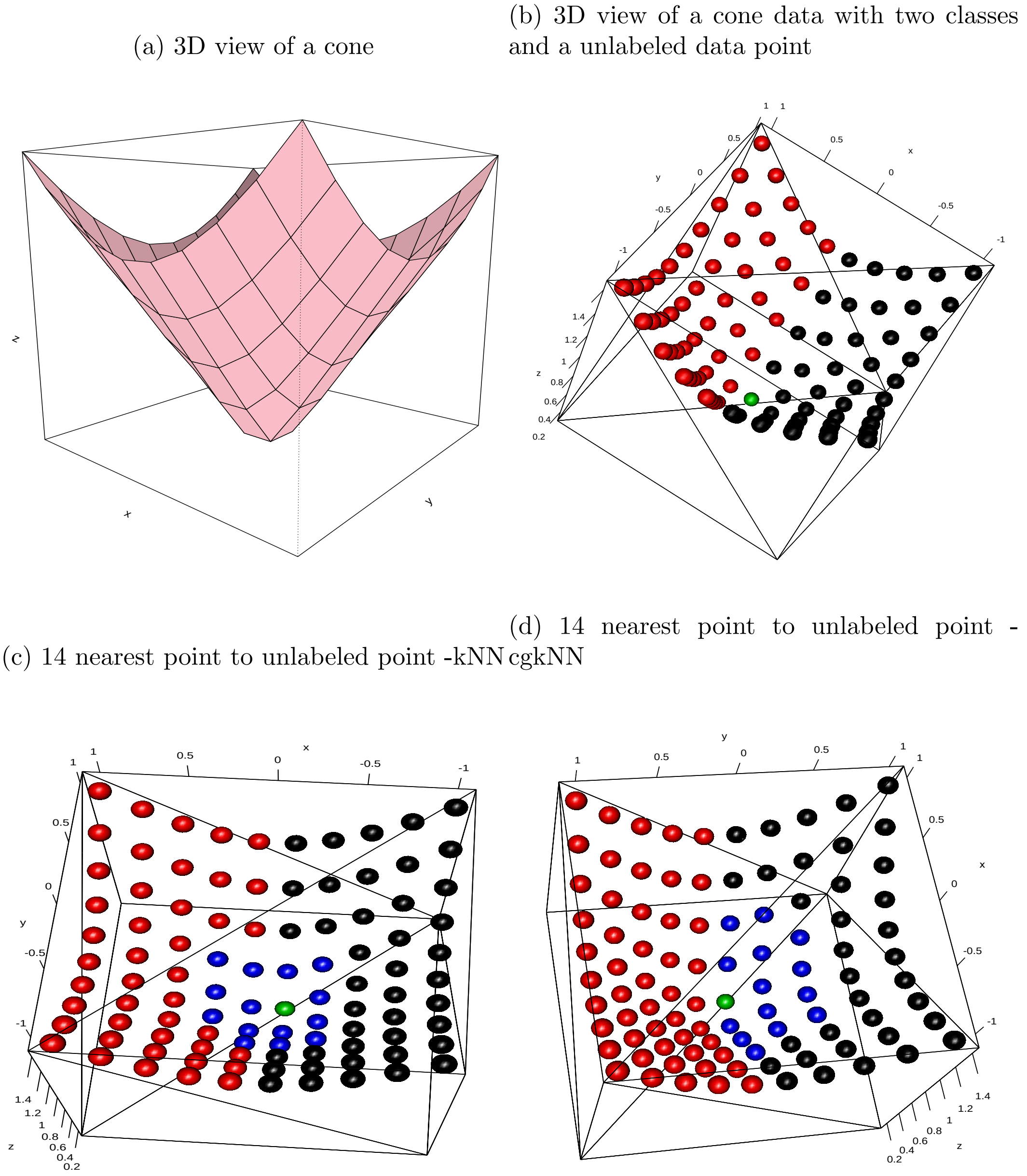
Figure 7: A 2D manifold with 3D data points

As mentioned earlier, traditional classification algorithms struggle to classify a manifold distributed data [41]. kNN, which is an effective and simple classification algorithm to classify normal distributed data, also fails to classify a manifold distributed data with high mean accuracy. The main reason for this limitation is the distance function used. Distance functions cannot calculate the true distance between various points of a manifold distributed dataset and thus results in high error rate. Distance metrics for the three dimensional data will calculate distances within the three dimensional space, regardless of the manifold space; ideally distances should be calculated along the manifold. Graph based methods can calculate the distance between two points on a manifold distributed dataset quite accurately [42, 22] and they can also use other available information such as class labels of the labeled data points. To demonstrated the limitation of kNN and strength of graph based classification, we have provided a synthetic dataset of a cone shape with class boundary of $x = 0$. The cone shaped synthetic dataset can be generated as per below equations.

$$Z = \sqrt{X^2 + Y^2} \tag{6}$$

The dataset contains three variable X, Y, Z. Z is a function of X, Y. Figure 8a shows the graphical representation of the dataset. We have divided the dataset in to two classes (black and red). An unlabeled point is shown in green in Figure 8b. This unlabeled data point is part of the black class but sits very close to red class. Figure 8c shows classification by kNN of this unlabeled data point with $k = 14$. A $k = 14$ is chosen for clarity only. All blue coloured data points are the nearest neighbours calculated by Euclidean distance used by kNN. It is obvious from the Figure 8c that kNN choses the nearest data point with respect to Euclidean distance and hence chose any point within the data space. kNN classification in this scenario is inaccurate. Figure 8d shows the same dataset classified with *cgk*NN. It is obvious from the nearest neighbours (blue points), that *cgk*NN picks up the relevant data space by considering the class information and by using graph for random walk as it identifies the local neighbourhood. The walk on this graph is what we refer to as random walk.

Figure 8: How cgkNN works on a manifold distributed data



(a) 3D view of a cone

(b) 3D view of a cone data with two classes and a unlabeled data point

(c) 14 nearest point to unlabeled point -kNN

(d) 14 nearest point to unlabeled point - cgkNN

Table 2: k closest neighbours by weight

Closest k	1	2	3	4	5
Obsrv No	12	10	2	19	11
weight	7.006982e-4	7.002936e-4	6.998980e-4	6.990813e-4	6.988739e-4

Class	0	1
Sum of weight	2.1e-3	1.4e-3

Constrained Graph kNN (cgkNN) is an extension of kNN whereby we use tired random walk to identify nearest neighbours. The method can be applied to labeled as well as unlabeled datasets. While our method outperforms many known methods for labeled datasets it is among the best methods for classification of datasets with many unlabeled and few labeled observations. The experimental results shown compares our method with the best available. We use an approach called tired random walk also called constrained random walk to measure the distance [37]. We have used the class labels of the labeled data for distribution as well as constrained information. The constrained information is used to modify the weight of graph edges between labeled samples. While our algorithm is explained in detail later with an example, a step by step procedure is outlined below.

4.1.1 The algorithm

1. Input $\chi = \chi_T \cup \chi_U, y$ and k, where χ_T is labeled data, χ_U is unlabeled dataset, y is a class label, k is number of nearest neighbours, σ is the Gaussian smoothing, α is the strength reduction rate.

2. Create a graph adjacency matrix W of dataset χ as per below.

   (a) $W_{ij} = 1$ if $x_i, x_j \in \chi_T$ and have same class label.

   (b) $W_{ij} = 0$ if $x_i, x_j \in \chi_T$ and have different class labels.

   (c) $W_{ij} = exp(-\|x_i - x_j\|^2/2\sigma^2)$ if at least one of x_i, x_j is unlabeled.

3. Calculate transition matrix P as per below. D is a diagonal matrix.

$$P = D^{-1}W \tag{7}$$

4. Compute the P_{TRW} (tired random walk transition matrix) using equation

$$P_{TRW} = \sum_{t=0}^{\infty}(\alpha P)^t = (I - \alpha P)^{-1} \tag{8}$$

5. Evaluate sample's similarity using equation

$$\bar{w_{ij}} = w(x_i, x_j) = \frac{(P_{TRW})_{ij} + (P_{TRW})_{ji}}{2} \tag{9}$$

6. Find k nearest neighbours of an unlabeled sample using equation

$$x_i = \underset{x_j \in \chi_T}{\arg\max}\, w(x, xj) \tag{10}$$

7. Determine the class label using equation. C is the class label and x is a sample observation.

$$y = \underset{c=1,2,\ldots,C}{\arg\max} \sum_{i=1}^{k} w(x, x_i) I(y_i = c) \tag{11}$$

In tired random walk, transition probability of a walker reduces with a fixed proportion α. In our algorithm we have fixed this ratio to 0.01 and thus the transition probability becomes smaller after a fixed number of steps. Please see [37] for more details on why tired random walk performs better than traditional random walk. α forces tired random walk to identify local neighbourhood. Another common approach for a manifold distributed data classification is the use of strengthening trees. Strengthening trees are used to reinforce strong relationship and reduce the weaker ones. We have avoided using any strengthening mechanism. We have conducted many experiments to evaluate the cost and benefit of using such strengthening mechanism and have found that such mechanisms do not add much value but have unnecessary overhead in term of additional computational cost. This is one of the reason why our method works much faster than the one mentioned in [37]. Please refer to Figure 14 for difference in compute cost between mkNN and cgkNN. The main cost of using strengthen trees is the compute time. Algorithms that use strengthen trees as a strengthening mechanism build these trees for each labeled observation in training dataset. Thus, their processing cost increases as the number of labeled observations increases. This makes the approach presented in [37] called mkNN only applicable to datasets with few labeled and many unlabeled observations. This also makes this approach undesirable for a dataset that has many labeled observations. In other words, the method will not take advantage of many labeled observations and restrict itself to k observations per class. As strengthen trees are required for each labeled observation, mkNN restrict itself to very small number of labeled observation. The algorithm struggles to compute classification of an average size dataset if decent number of observations were selected per class and not k labeled observations.

Most of the methods that can classify a manifold distributed data use random walks over a graph to identify nearest neighbours. Random walk and creating trees to identify nearest or furthest neighbours is a costly activity. It is not worth using these methods unless the gain outweighs the additional cost. Algorithms that use trees ultimately reduces processing cost by compromising other cost intensive processing activities. Like in [36, 37] the cost is minimized by selecting only k labeled points per class for training. This is the same k in K nearest neighbour classification. Obviously, the number is limited to k to keep the cost of creating trees to minimal. Similar approaches are adopted by different algorithms whereby creation of trees is kept to minimal to avoid costly processing.

In short, traditional kNN struggles with classification of a manifold distributed data. Graph based approaches can classify a manifold distributed data but have very high compute cost. Our method can be described as best of both i.e. high mean accuracy and less compute cost for classification using graph.

4.1.2 Example

The dataset used in this example is a subset of banknotes dataset. The dataset contains twenty labeled observations, ten from each class and one unlabeled observation. We have intentionally selected a subset that will result in incorrect classification by kNN and correct classification by cgkNN. The objective is to show how traditional kNN can misclassify a manifold distributed data based on Euclidean distance and how the same observation is classified correctly using cgkNN. Banknotes dataset has two classes i.e. 1 or 0 and hence our example also has two classes. Figure 9 shows the dataset from two different angles. It also shows the nearest neighbours as classified by Euclidian distance of kNN and also by cgkNN. The yellow point is the unlabeled data point while the aqua colour points represent the nearest neighbours. Blue colour represents class 0 while pink color represents class 1. The nearest neighbours are either class $0, 1$. You can see that observations $2, 9 : 12$ are calculated as the closest five observations to the unlabeled observation using Euclidian distance. The same dataset returns different nearest neighbours with cgkNN. The nearest neighbours as per cgkNN weightage are observations $2, 10 : 12, 19$. As the data points are in three dimensional space, we have shown two different views of the same for each method. The correct calculation of one nearest neighbour results in correct classification of the unlabeled point. kNN classify this unlabeled point as class 1 while cgkNN classify this unlabeled point as class 0 which is correct classification.

Figure 10 shows different views of how cgkNN classify this unlabeled points. It not only calculates the nearest points accurately; it uses the class information to drive a better outcome. The Final weight matrix based on tired random walk of the mentioned dataset is shown in Table 2. Using this matrix cgkNN calculates the sum of weights and thus assigns the unlabeled data point to the class with highest sum of weights.

As mentioned in the algorithm, the nearest points are calculated using similarity matrix $\bar{w}_{ij} = w(x_i, x_j) = \frac{(P_{TRW})_{ij}+(P_{TRW})_{ji}}{2}$. Once a similarity weight matrix is calculated, nearest k labeled points are identified along with their classes. This is followed by sum of weight operations by class. The class with highest sum of weights wins and unlabeled point is labeled with the same class.

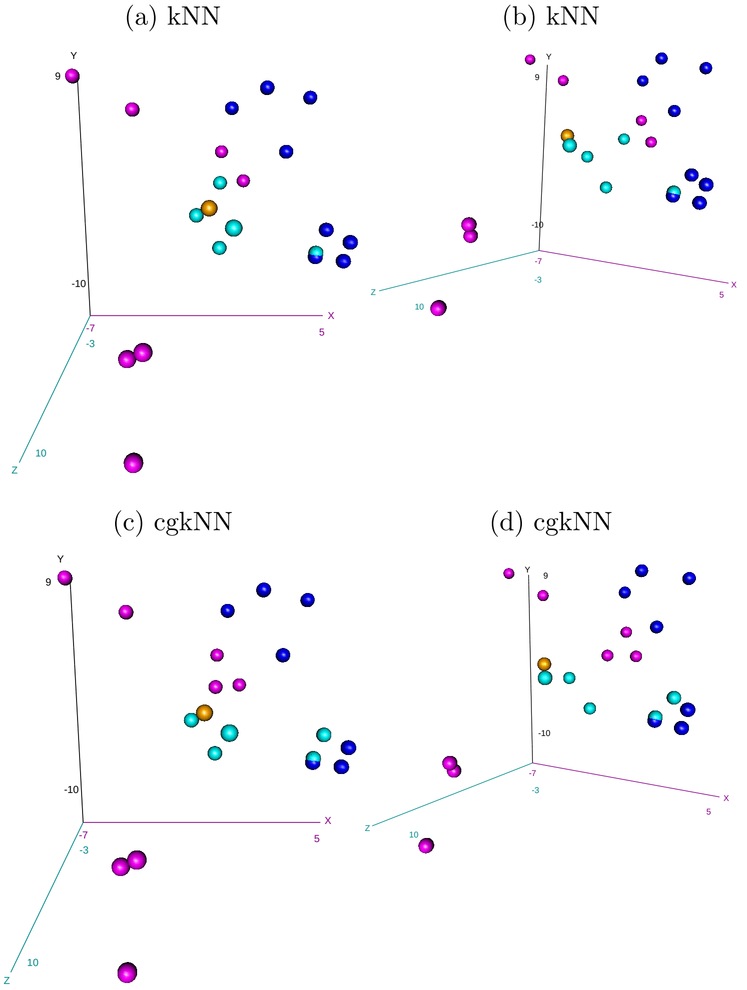
Table 2 also shows sum of weights by class for this specific example. The sum of weight for class 0 is higher than the sum of weight for class 1. Thus the algorithm classifies the unlabeled observation to class 0.

4.1.3 Experimental results

We have performed experiments on six publically available datasets. These are the same datasets used by [37]. The objective was to compare computational cost and mean accuracy of prediction for cgkNN and mkNN. Mean accuracy is calculated using result from ten different iterations with random samples in each iteration. During experiments we noticed that the mean accuracy of predictions using traditional kNN declines as the number of k increases. This is quite an obvious pattern in all of the mentioned datasets.

Please refer to Figure 12 to see the comparison of mean accuracy by mkNN and cgkNN for various datasets. Banknotes dataset is compared twice but with different number of labeled observations. It is very clear from Figure 12 that mean accuracy of cgkNN is always

Figure 9: Location of unlabeled point and five nearest neighbours - kNN and cgkNN



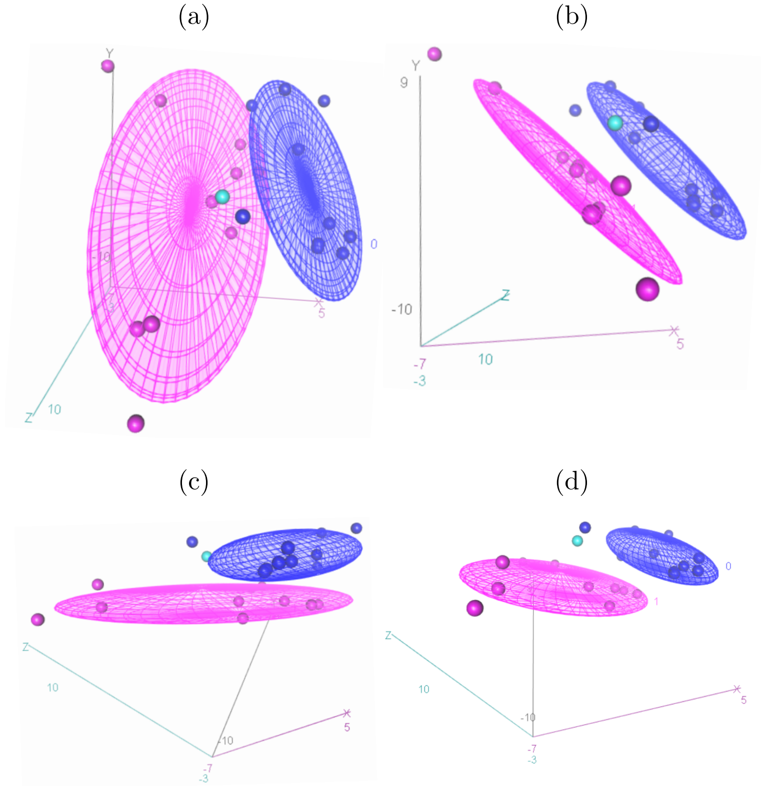
(a) kNN

(b) kNN

(c) cgkNN

(d) cgkNN

Figure 10: Different views of how cgkNN cluster various classes and then classify

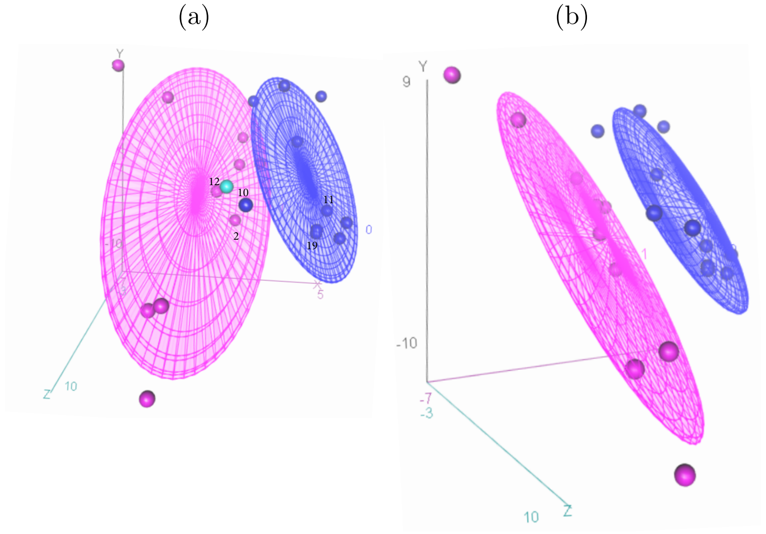

(a)

(b)

(c)

(d)

Figure 11: k closest observations to unlabeled observation


(a)

(b)

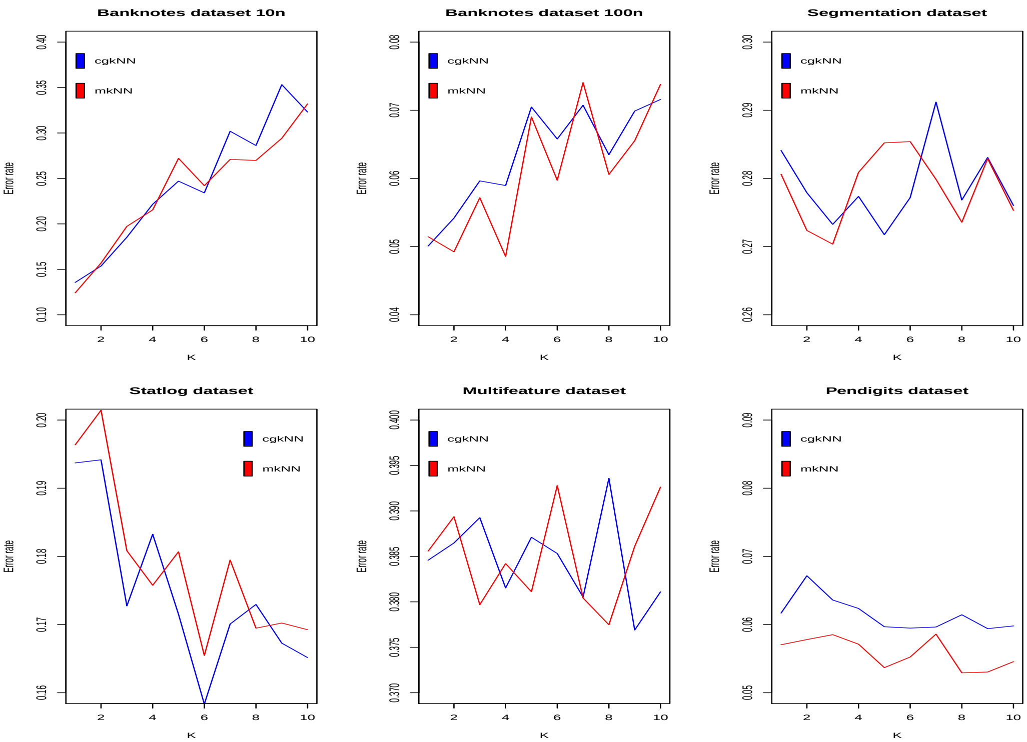


Figure 12: Mean error rate comparision of cgkNN and mkNN

in line with mkNN irrespective of number of labeled observations available. Table 3 shows mean accuracy for various datasets with different size of training. It should be noted that while cgkNN performs as good as or better than mkNN with respect to mean accuracy of prediction, it is is much faster than mkkNN. Table 3 shows the processing time for various datasets. It is obvious from the results that cgkNN can produce same results as mkNN but with almost half the time required for computation. It should be noted that these calculations are based on same datasets and all environmental variables such as network, processing power were keep the same to get a comparable result. You would notice that the mean accuracy of classification of all these datasets increases if we consider a good size training dataset. This may not be achievable in scenarios where we have limited labeled observations.

Figure 13 shows the mean error rate of various datasets for k of 1 to 10 using mkNN and cgkNN. Figure 13 also compares the result for different number of training observations from the same dataset. It is obvious from Figure 13 that the performance of both methods are in line with each other irrespective of number of training observations selected. On the other hand, Table 3 shows a comparison of our method to mkNN with respect to mean accuracy and processing time. The results are based on selection of various number of training observations per class. You can see from results that while mean accuracy remains the same for both the methods, our method is much efficient on processing time. Processing time of mkNN gets worse as the number training observations increases. The main reason for this is the usage of strengthen trees in mkNN which are required for all the labeled training observations. Thus the processing time increases as we increase number of observations.

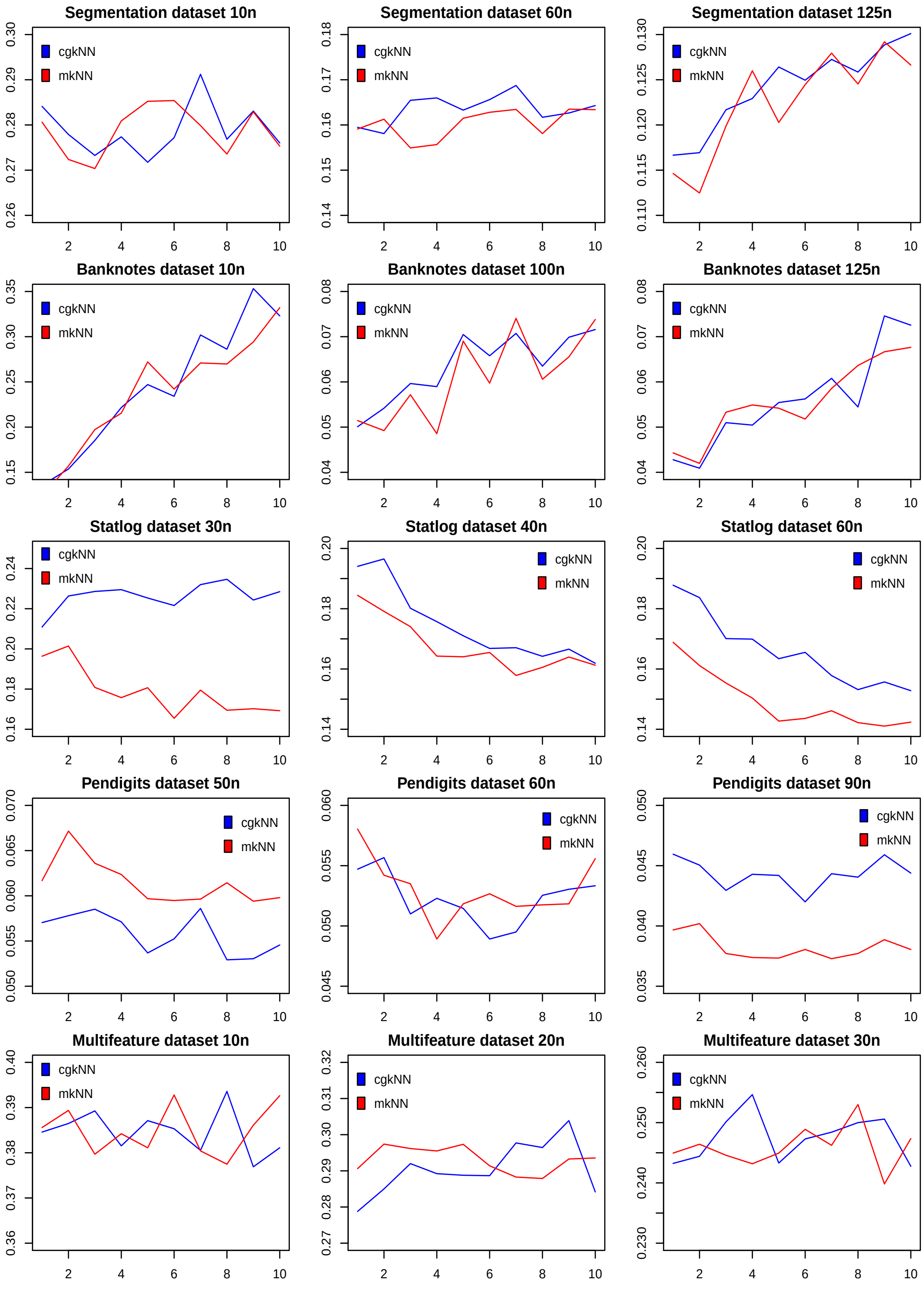


Figure 13: Mean error rate of five real-world data sets with different number of training sets. Mean error rate on the ordinate and k on the abscissa.

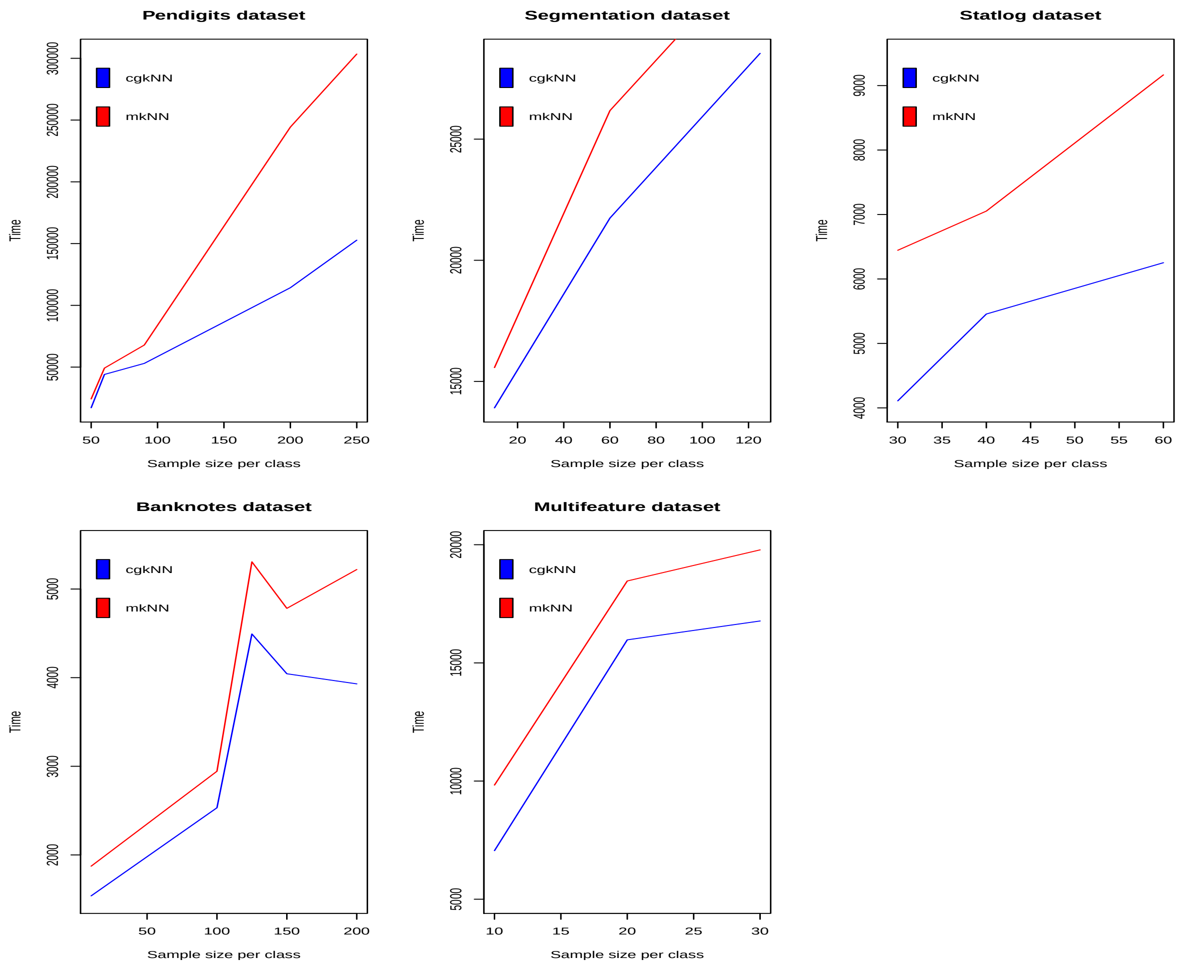


Figure 14: Compute time of five real-world data sets with different number of training sets. Compute time on the ordinate and Sample size on the abscissa.

		ϕ Error		Processing time		Difference	
Dataset	#TR	$cgkNN$	$mkNN$	$cgkNN$	$mkNN$	ϕ Error	Time
Banknotes	20	24.4	23.8	1,538	1,873	0.6	-335
Banknotes	200	6.4	6.1	2,532	2,944	0.3	-412
Banknotes	250	5.6	5.6	4,493	5,307	0	-814
Banknotes	300	5.1	5.2	4,044	4,782	-0.1	-738
Banknotes	400	4.3	4.2	3,930	5,220	0.1	-1,290
Multifeature	100	38.5	38.5	7,055	9,829	0	-2,774
Multifeature	200	29.1	29.3	15,975	18,466	-0.2	-2,491
Multifeature	300	24.8	24.6	16,773	19,782	0.2	-3,009
Pendigits	500	5.6	6.1	17,084	24,234	-0.5	-7,150
Pendigits	600	5.2	5.3	44,068	49,194	-0.1	-5,126
Pendigits	900	4.4	3.8	53,000	67,857	0.6	-14,857
Segmentation	70	27.9	27.9	13,908	15,570	0	-1,662
Segmentation	420	16.4	16	21,742	26,178	0.4	-4,436
Segmentation	875	12.4	12.3	28,541	32,940	0.1	-4,399
Statlog	180	22.6	17.9	4,109	6,445	4.7	-2,336
Statlog	240	17.4	16.8	5,456	7,053	0.6	-1,597
Statlog	360	16.6	14.9	6,252	9,166	1.7	-2,914

Table 3: Mean accuracy of prediction and processing time mkNN vs cgkNN

Figure 14 shows a comparison of compute time for various samples of the five datasets for *mk*NN and *cgk*NN. It is visible that *cgk*NN performs much better with respect to compute time when compared to *mk*NN.

5 RESEARCH AIM AND OBJECTIVES

5.1 Overall Research Aim

This research will examine different methods of semi-supervised learning with specific emphasis on a manifold distributed data. The research will also examine best approaches to semi-supervised learning with limited number of labeled observations. New methods to semi-supervised learning and optimisation / extension of existing methods are also in scope.

5.2 Research Objectives

The objectives of this research are as follows:

1. Part 1: Below areas will be covered.

    (a) Examine and explore graph based methods for classification and clustering. Explore and explains why random walk works in these methods.

(b) Examine and explore the application of multi-dimensional scaling in hierarchical and a manifold clustering.

(c) Examine and explore semi-supervised learning for image classification, especially work done by Facebook.

2. Part 2: Below areas will be covered.

    (a) Examine and explore distance metric learning and classification. This will be a thorough review of work done by Kilian Weinberger including Large Margin nearest neighbour LMNN.

    (b) Examine and explore self-supervised learning in context of the work done by google research.

3. Part 3: Contributions

    (a) Development of a graph based classification algorithm for a manifold distributed data and unlabeled data. Submission of the same to a relevant journal.

    (b) An article on why random walk or tired random walk works. Submission of the same to a relevant conference.

    (c) Development of new algorithm using graph based methods for clustering. Submission to a relevant journal.

    (d) Extension of existing graph based algorithms and potential paper submissions to a relevant conference.

    (e) Development / extension / optimisation of an image classifier. Submission of the same to a journal.

5.3 Research Questions

The following research questions will be applied to this area of study:

1. What is the most effective way of classifying a manifold distributed data?

2. What is the most effective way of labeling data with small number of labeled and large number of unlabeled observations?

Most of the next two years of research will revolve around answering these two questions.

6 PROPOSED RESEARCH SCHEDULE

A Gantt timeline chart was designed to establish rough timeline of all activities for next three years. This is to ensure all activities are completed within the agreed time frame. Please refer to Appendix C for Gantt Chart and Schedule. To date, timeframes allocated for various tasks have been met, and we are progressing without any difficulty.

Paper	Title/Content	Respective Journal
1	Constrained Graph Nearest Neighbour Classification	Elsevier
2	Probabilistic Nearest Neighbours Classification for Unlabeled Data	AI2019
3	Clustering using constrained Graph Nearest Neighbours	Elsevier
4	Semi-supervised learning for breast cancer image classification	TBA
5	Semi-supervised learning with metric learning	TBA

Table 4: Outline of Proposed Papers for Publication

7 CONTRIBUTION OF PROPOSED RESEARCH

7.1 Benefits of Research

This research will examine how different types of semi-supervised learning behave with small number of labeled observations. The focus will be to optimise algorithms to work with small number of labeled observations and produce high mean accuracy of classification. The research will also be focusing on classification of a manifold distributed data and data with high number of dimensions while, concurrently, investigating the semi-supervised algorithms for breast cancer image classification. This is in order to address the gaps in classification accuracy of medical images. This research will also examine work done by Facebook and google in the field of semi-supervised learning and image classification with a view to improve and build on existing work.

7.2 Communication of Results

Results will be communicated mainly via publications in journals. Plans for five papers for submission to relevant journals are outlined below in Table 4.

8 CONCLUSION

Achieving higher mean accuracy of classification can add tremendous value to tasks that use various classification algorithms. The cost of labeling unlabeled observations is increasing with the increase in cost of human resources. Therefore, it is important that effective semi-supervised learning algorithms are developed that can label unlabeled observations and can achieve high mean accuracy with minimal cost of human resource.

This report has highlighted the potential for further research in this area. To develop new or optimise the existing algorithms for classification that can achieve high mean accuracy of classification irrespective of the type of data used, is the basic objective of this project.

References

[1] L. Apostel. Le problème formel des classifications empiriques. *Centre National de Recherche de*, 1963.

[2] G. Birkhoff. Lattice theory, vol. 25 of american mathematical society colloquium publications. *American Mathematical Society, Providence, RI*, 1967.

[3] Z. Bodo and Z. Minier. On supervised and semi-supervised k-nearest neighbor algorithms. In *Proceedings of the 6th Joint Conference on Mathematics and Computer Science*, page 1, 2008.

[4] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[5] M. A. Carreira-Perpinán. A review of dimension reduction techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, 9:1–69, 1997.

[6] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[7] D. Cho, Y.-W. Tai, and I. S. Kweon. Deep convolutional neural network for natural image matting using initial alpha mattes. *IEEE Transactions on Image Processing*, 28(3):1054–1067, 2018.

[8] S.-B. Cho and J. H. Kim. Combining multiple neural networks by fuzzy integral for robust classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(2):380–384, 1995.

[9] L. S. Choon, A. Samsudin, and R. Budiarto. Lightweight and cost-effective mpeg video encryption. In *Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications, 2004.*, pages 525–526. IEEE, 2004.

[10] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[11] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, and T. S. Huang. Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1553–1566, 2004.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms second edition. *The Knuth-Morris-Pratt Algorithm, year*, 2001.

[13] T. M. Cover, P. E. Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[14] I. Dahlberg. Wissensorganisation: Entwicklung, aufgabe, anwendung, zukunft. *Zagadnienia Informacji Naukowej*, 53(2 (106)), 2015.

[15] P. Dayan, M. Sahani, and G. Deback. Unsupervised learning. *The MIT encyclopedia of the cognitive sciences*, 1999.

[16] A. Fujino, N. Ueda, and K. Saito. Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):424–437, 2008.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[18] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[19] L. Grimaudo, M. Mellia, E. Baralis, and R. Keralapura. Select: Self-learning classifier for internet traffic. *IEEE Transactions on Network and Service Management*, 11(2):144–157, 2014.

[20] A. Grønlund, K. G. Larsen, and A. Mathiasen. Optimal minimal margin maximization with boosting. *arXiv preprint arXiv:1901.10789*, 2019.

[21] M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[22] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[23] J. He, J. G. Carbonell, and Y. Liu. Graph-based semi-supervised learning as a generative model. 2007.

[24] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[25] D. T. Larose and C. D. Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley Sons, 2014.

[26] R. Liu, J. Zhou, and M. Liu. Graph-based semi-supervised learning algorithm for web page classification. In *Sixth International Conference on Intelligent Systems Design and Applications*, volume 2, pages 856–860. IEEE, 2006.

[27] B. Longstaff, S. Reddy, and D. Estrin. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–7. IEEE, 2010.

[28] G. Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

[29] D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics, 2006.

[30] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27, 1990.

[31] A. E. Omolara, A. Jantan, O. I. Abiodun, M. M. Singh, M. Anbar, and D. Kemi. State-of-the-art in big data application techniques to financial crime: a survey. *Int. J. Comput. Sci. Network Secur.*, 18(7):6–16, 2018.

[32] M. Peikari, S. Salama, S. Nofech-Mozes, and A. L. Martel. A cluster-then-label semi-supervised learning approach for pathology image classification. *Scientific reports*, 8(1):7193, 2018.

[33] P. H. Sneath, R. R. Sokal, et al. *Numerical taxonomy. The principles and practice of numerical classification.* 1973.

[34] J. Tanha, M. van Someren, and H. Afsarmanesh. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(1):355–370, 2017.

[35] P. A. Thoits. Self-labeling processes in mental illness: The role of emotional deviance. *American journal of Sociology*, 91(2):221–249, 1985.

[36] E. Tu, L. Cao, J. Yang, and N. Kasabov. A novel graph-based k-means for nonlinear manifold clustering and representative selection. *Neurocomputing*, 143:109–122, 2014.

[37] E. Tu, Y. Zhang, L. Zhu, J. Yang, and N. Kasabov. A graph-based semi-supervised k nearest-neighbor method for nonlinear manifold distributed data classification. *Information Sciences*, 367:673–688, 2016.

[38] P. Turaga and R. Chellappa. Nearest-neighbor search algorithms on non-euclidean manifolds for computer vision applications. In *Proceedings of the seventh Indian conference on computer vision, graphics and image processing*, pages 282–289. ACM, 2010.

[39] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[40] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[41] G. Zhong, L.-N. Wang, X. Ling, and J. Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265–278, 2016.

[42] A. Zinovyev and E. Mirkes. Data complexity measured by principal graphs. *Computers & Mathematics with Applications*, 65(10):1471–1482, 2013.

A Appendix - Copy of Journal Article Ready for Submission

Constrained Graph Nearest Neighbour Classification

Sayed Waleed Qayyumi[a,1], Laurence Park[b], Oliver Obst[c]

Western Sydney University, Victoria Rd, Rydalmere NSW 2116, Australia

Western Sydney University[a,b,c]

[a] *Western Sydney University,School of Computing, Engineering and Mathematics, South Campus, Rydalmere NSW 2116, Australia*
[b] *Western Sydney University,School of Computing, Engineering and Mathematics, South Campus, Rydalmere NSW 2116, Australia*
[c] *Western Sydney University,School of Computing, Engineering and Mathematics, South Campus, Rydalmere NSW 2116, Australia*

Abstract

k Nearest Neighbours kNN has been widely used for classification owing to its simplicity and accuracy. In spite of the wide usage, kNN has performed poorly with manifold distributed data. Few extensions of kNN are available that deal with manifold distributed data but the extensions are either costly to compute or improvement in term of mean accuracy of classification is not significant. Classification of manifold distributed data requires prior knowledge of the shape of data distribution. The shape information is used to assess a distance or similarity function.

In this paper, we are introducing a new semi-supervised algorithm which we are referring to as constrained graph kNN (cgkNN). Our method can be used for traditional Gaussian distributed data classification as well as for non-linear manifold distributed data classification. Inspired by a method called manifold kNN (mkNN) which is one of the best method for classification of manifold distributed data as well as classification using small number of labeled observations, our method works in similar way but always outperforms mkNN with respect to compute time. Our method also performs as good as, and at times better than mkNN with respect to mean accuracy of classification and classification using small number of labeled observations.

Keywords: Graph based Semi-Supervised Learning, k Nearest Neighbours, Manifold Distributed Data, Tired Random Walk.

1. Introduction

Classification models are important tools for data analysis allowing class labels to be predicted for a given observation. Classification models require training data, which is a set of example observations with manually assigned labels for each observation. Often we find that unlabeled observation data can be easily obtained but experts are not always available to manually label data for model training. Semi-supervised learning uses the class label information from manually labeled training data along with distribution of unlabeled data in an attempt to provide more suitable classification models for a given data space. Recent research has shown that high mean accuracy of classification can be obtained using graph based semi-supervised k Nearest Neighbours, when only a small proportion of the data is manually classified. These graph based algorithms take advantage of the data manifold structure when computing class labels. More often the complexity of these graph based algorithms lead to long compute times when performing classification. Traditional classification algorithms not only fail to classify manifold distributed data with high mean accuracy but also struggle to classify using small number of labeled observations.

*Corresponding author
Email address: S.Qayyumi@westernsydney.edu.au (Western Sydney University)
URL: https://www.westernsydney.edu.au (Western Sydney University)

In this paper, we introduce a new graph based semi-supervised classification algorithm that provides high classification accuracy and due to its reduced complexity, requires shorter compute times. The algorithm can also classify with high mean accuracy using small number of labeled observations.

The contributions of this work are:

1. An algorithm for classification of manifold distributed data. The algorithm has the ability to classify with high mean accuracy using small number of labeled observations.
2. Experimental results, showing our algoritms performance with respect to mean accuracy of classification and compute time with other similar algorithms.

The article will proceed as follows: Section 2 describes related work. Section 3 presents constrained graph algorithm. Section 4 presents experimental results and comparison of *cgk*NN and *mk*NN. This is followed by conclusions in Section 5.

2. Related work

The idea of using unlabeled observations along with labeled observations to improve prediction is not new. There was always a need to use the available unlabeled observations to enrich training or in other words improve mean accuracy of classification. This desire has led researchers to use the unlabeled observations in many different ways. Self-supervised learning also called self-learning is a very simple approach to use unlabeled observations. While self-learning is an easier approach for using unlabeled available observations, it does not guarantee improvement in mean accuracy of classification. Self-supervised learning is one of the most widely used semi-supervised method [1, 2, 3] which can be applied to any classification algorithm. For example, self-training *k*NN involves predicting class of unlabeled observations using *k*NN [4] and labeled observations as input. *k*NN is trained using a training set TR and unlabeled set UD is then labeled using the predicted class [5, 6]. The unlabeled points that have high confidence of prediction are then added to training set TR along with their predicted class. The extended training set ETR = TR+UD is then used to classify other unlabelled points. See Table 2 for the impact of adding unlabeled points on mean accuracy of prediction using *k*NN. Mean accuracy of prediction is calculated before and after the addition of unlabeled points. Mean accuracy is calculated by taking the mean of accuracy of random observations of 30 different iterations. It is very obvious from the results that self-learning may or may not improve mean accuracy. The fact that self-learning does not guarantee an improvement in mean accuracy is another reason we have introduced our method *cgk*NN for classification. *cgk*NN works with small number of labeled observations and produces high mean accuracy with much less compute cost. Our method can be classified as a graph based semi-supervised learning method. There are many different approaches available for semi-supervised learning [5, 7]. Semi-supervised learning is a class of machine learning that falls between supervised learning and unsupervised learning. In Semi-supervised learning, labeled as well as unlabeled data is used to train an algorithm. Supervised learning is based on training set that is fully labeled while unsupervised learning uses unlabeled data for training. Semi-supervised learning is a complex area of classification where most of the commonly used classification algorithms fail due to lack of / limited number of labeled observations. Supervised learning algorithms can be used as semi-supervised learning algorithms via a process called self learning [1, 2, 3]. In this process, classifier is trained using limited number of available observations. Observations with predicted classes that have high confidence are refeed to the algorithm to improve its performance. Semi-supervised methods are either generative [8, 9] or graph based methods [10]. Generative methods estimate the distribution of data points belonging to different classes. Generative methods, model how the data is generated [11] and learns a function $f(x, y)$ that can be used to score the configuration determined by x and y together. This way one can find the y for a new x by finding a y for which the score of $f(x, y)$ is maximum. Naive Bayes and Hidden Markov Models are some of the example of generative models. Graph based algorithms build graphs whose nodes are labeled and unlabeled data points. Labeled data points are used to spread information to unlabeled data points. In these methods, graphs $g = (V, E)$ where V represent a node and E an edges is used to represent the geomatry of the data. E represents similarity between two edges. Similarities are shown using a weight matrix W. Graph based methods are very useful when only small amount of labeled observations is available and the cost of labeling large unlabeled observations is high. While many methods exist for classification; accuracy is not always driven by the method itself, underlying data also plays a major role in classification accuracy. Most of the methods available today are developed with the assumption that data lies in low manifold. This assumption results in low performance by these methods when data used is not in low manifold. Graph based classification methods are more accurate at classification of manifold distributed data and have shown higher mean accuracy of classification [12, 1]. kNN [4, 13], k-means [1] and many other algorithms have been extended using graphs to deal with manifold distributed data. To understand why graph based methods perform better with manifold distributed data, we need to answer two important questions.

1. What is manifold distributed data?
2. What is random walk?

Definition: A data set is considered to be manifold distributed if its intrinsic dimension is less than its data space dimension [12]. A manifold is very different from dimensionality. Dimensionality simply refers to n in m x n dataset where m is the number of rows and n is the number of columns. The shape in Figure 1 shows a two dimensional manifold with three dimensional data points. In short, we define manifold as a continuous geometrical structure that has fixed number of dimensions. The number of intrinsic dimension of manifold distributed data is less than its data dimension.

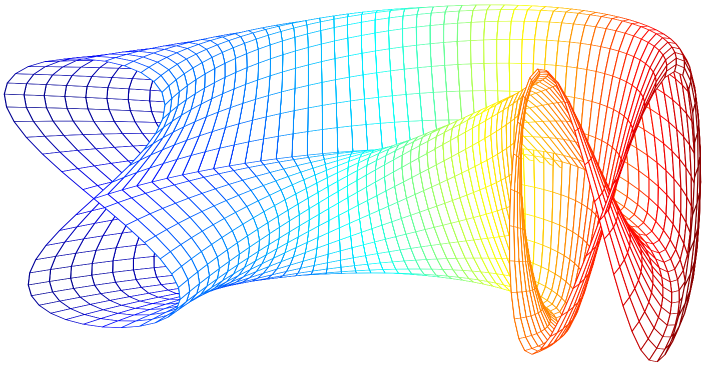


Figure 1: A 2D manifold with 3D data points

As mentioned earlier, traditional classification algorithms struggle to classify manifold distributed data. kNN, which is an effective and simple classification algorithm to classify normal distributed data, also fails to classify manifold distributed data with high mean accuracy. The main reason for this limitation is distance function used by kNN. Distance functions cannot calculate the true distance between various points of a manifold distributed dataset and thus results in high error rate. On the contrary, graph based methods can calculate the distance between two points on manifold distributed dataset quite accurately and they can also use other available information such as class labels of the labeled data points. To demonstrated the limitation of kNN and strength of graph based classification, we have chosen a synthetic dataset of a cone shape. The cone shaped synthetic dataset can be generated as per below equations.

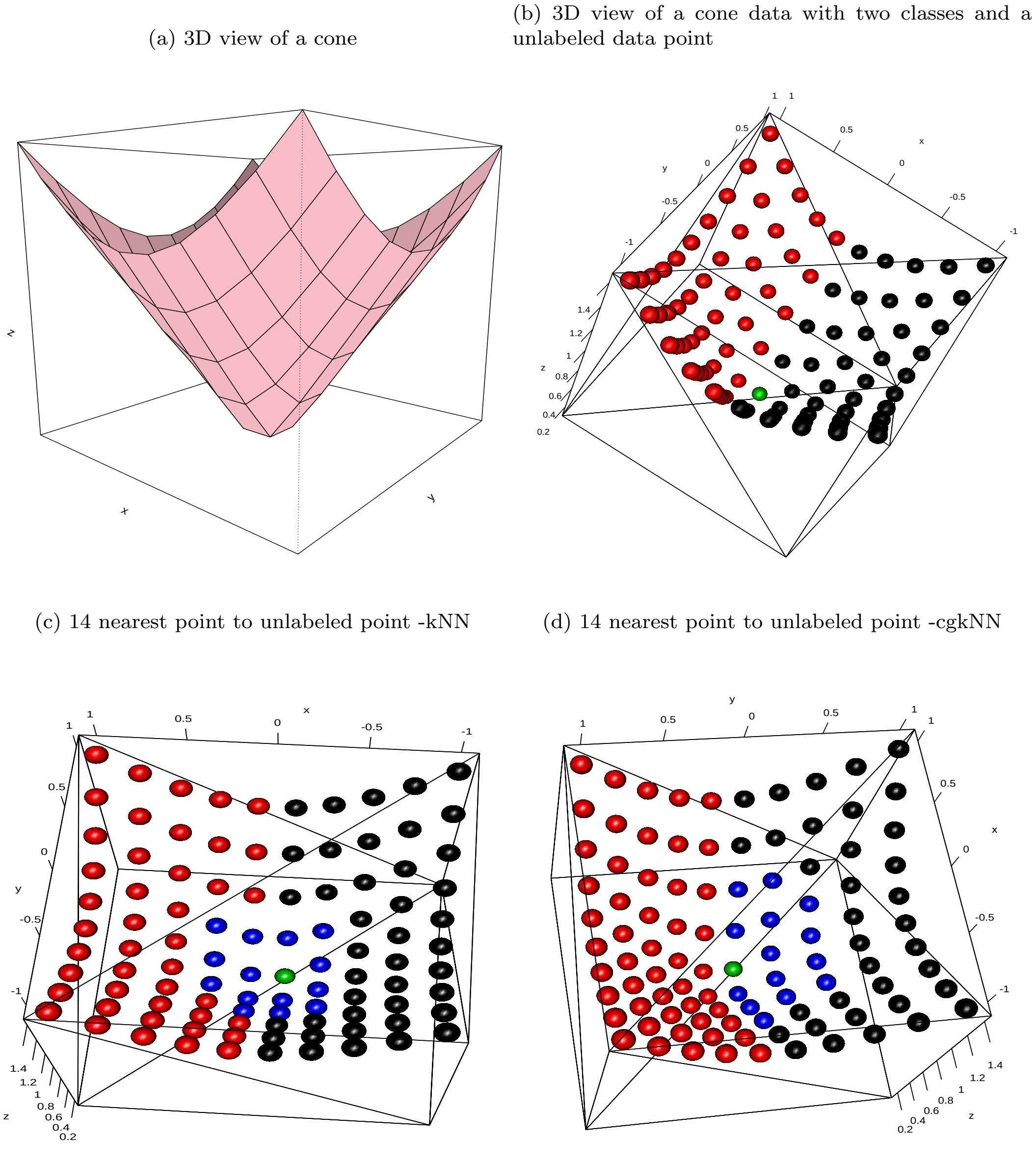
$$X = Y = seq(-1, 1, len = 10) \tag{1}$$

$$Z = \sqrt{X^2 + Y^2} \tag{2}$$

The dataset contains three variable X, Y, Z. Z is a function of X, Y. Figure 2a shows the graphical representation of the dataset. We have divided the dataset in to two classes i.e. black and red classes for demonstration purpose. An imaginary unlabeled point is also selected which can be seen in green colour in

Figure 2b. This unlabeled data point is part of the black class but sits very close to red class. Figure 2c shows classification by kNN of this unlabeled data point with $k = 14$. All blue coloured data points are the nearest neighbours calculated by Euclidean distance used by kNN. It is obvious from the Figure 2c that kNN choses the nearest data point with respect to Euclidean distance and hence chose any point within the data space. kNN classification in this scenario is inaccurate. Figure 2d shows the same dataset classified with cgkNN. It is obvious from the nearest neighbours (blue points), that cgkNN picks up the relevant data space by considering the class information and by using graph for random walk as it identifies the local neighbourhood. The walk on this graph is what we refer to as random walk. The approach is explained in detail in 3.2.

Figure 2: How cgkNN works on manifold distributed data



(a) 3D view of a cone

(b) 3D view of a cone data with two classes and a unlabeled data point

(c) 14 nearest point to unlabeled point -kNN

(d) 14 nearest point to unlabeled point -cgkNN

Table 1: k closest neighbours by weight

Closest k	1	2	3	4	5
Obsrv No	12	10	2	19	11
weight	7.006982e-4	7.002936e-4	6.998980e-4	6.990813e-4	6.988739e-4

Class	0	1
Sum of weight	2.1e-3	1.4e-3

3. Constrained Graph *k*NN

*k*NN is one the simplest and most used method for classification. If we consider X as the matrix of features and Y as the class label, *k*NN looks at k which will always be a positive integer, and estimates the probability of it belonging to class j for a test observation x_0.

$$Pr(Y = j|X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{3}$$

As described earlier *k*NN is widely used for classification problems. The ease of use and good performance in term of mean accuracy of classification has made *k*NN an algorithm of choice for researchers and general public working on classification problems. *k*NN can easily and accurately classify any type of data that lies in low manifold but with high manifold distributed data the accuracy of classification drops drastically. This problem with *k*NN lead us to come up with a new approach for classification that works not only on normal distributed data but also on manifold distributed data.

Constrained Graph *k*NN (*cgk*NN) is an extension of *k*NN whereby we use tired random walk to identify nearest neighbours. The method can be applied to labeled as well as unlabeled datasets. While our method outperforms many known methods for labeled datasets it is among the best methods for classification of datasets with many unlabeled and few labeled observations. We use an approach called tired random walk also called constraint random walk to measure the distance [12]. Basically, we use the class labels of the labeled data for distribution as well as constraint information. The constraint information is used to modify the weight of graph edges between labeled samples. While our algorithm is explained in detail later with an example, a step by step procedure is outlined below.

3.1. The algorithm

1. Input $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_U, y$ and k, where $\mathcal{X}_T$ is labeled data, $\mathcal{X}_U$ is unlabeled dataset, y is class label, k is number of nearest neighbours.
2. Create an graph adjacency matrix W of dataset $\mathcal{X}$ as per below.
   (a) $W_{ij} = 1$ if $x_i, x_j \in \mathcal{X}_T$ and have same class label.
   (b) $W_{ij} = 0$ if $x_i, x_j \in \mathcal{X}_T$ and have different class labels.
   (c) $W_{ij} = exp(-\|x_i - x_j\|^2/2\sigma^2)$ if at least one of x_i, x_j is unlabeled.
3. Calculate transition matrix P as per below.

$$P = D^{-1}W \tag{4}$$

4. Compute the P_{TRW} (accumulated transition probability matrix) using equation

$$P_{TRW} = \sum_{t=0}^{\infty} (\alpha P)^t = (I - \alpha P)^{-1} \tag{5}$$

5. Evaluate sample's similarity using equation

$$\bar{w}_{ij} = w(x_i, x_j) = \frac{(P_{TRW})_{ij} + (P_{TRW})_{ji}}{2} \tag{6}$$

6. Find nearest neighbours of an unlabeled sample using equation

$$x_i = \arg\max_{x_j \in \mathcal{X}_T} w(x, xj) \tag{7}$$

7. Determine the class label using equation

$$y = \arg\max_{c=1,2,\ldots,C} \sum_{i=1}^{k} w(x, x_i) I(y_i = c) \tag{8}$$

As seen above, *cgk*NN uses random walk over a graph, it is important to understand how $cgkNN$ uses tired random walk to calculate neighbourhood.

3.2. Tired Random Walk

Tired random walk is a form of simple random walk. To understand tired random walk, it is important we understand simple random walk first. In simple words, random walk is a sequence of fixed length steps in random direction. The random movement can be one, two or n dimensional. In a d-dimensional vector, a simple random walk of n steps is denoted by S_n and is defined by

$$S_n = x + \sum_{i=1}^{n} X_j \tag{9}$$

where x represents the position on vector at time $n = 0$ and X_j represents the movement from time j to $j+1$. Later in this text, you will note that our approach uses tired random walk instead of classical random walk. The main difference between a tired random walk and classical random walk lies in transition probability of walker.

$$x_1 = Tx_0 \tag{10}$$

$$x_2 = Tx_1 \tag{11}$$

$$x_2 = T(Tx_0) \tag{12}$$

$$x_2 = T^2 x_0 \tag{13}$$

$$x_n = T^n x_0 \tag{14}$$

In tired random walk, transition probability of a walker reduces with a fixed ratio. In our algorithm we have fixed this ratio to 0.01 and thus the transition probability becomes zero after a fixed number of steps. Please see [12] for more details on why tired random walk is better than traditional random walk. Another common approach in manifold distributed data classification is the use of strengthen trees. Strengthen trees are used as a strengthening mechanism to validate and enforce strong relationship and reduce the weaker ones. We have avoided using any strengthening mechanism. We have conducted many experiments to evaluate the cost and benefit of using such strengthening mechanism and have found that such mechanisms do not add much value but have unnecessary overhead in term of additional compute cost. This is one of the reason why our method works much faster than the one mentioned in [12]. Please refer to Figure 8 for difference in compute cost between *mk*NN and *cgk*NN. The main cost of using strengthen trees is the compute time. Algorithms that use strengthen trees as a strengthening mechanism build these trees for each labeled observation in training dataset. Thus their processing cost increases as the number of labeled observations increases. This makes the approach presented in [12] called *mk*NN only applicable to datasets with few labeled and many unlabeled observations. This also makes this approach undesirable for a dataset that has many labeled observations. In other words, the method will not take advantage of many labeled observations and restrict itself to k observations per class. As strengthen trees are required for each labeled observation, *mk*NN restrict itself to very small number of labeled observation. The algorithm struggles to

Table 2: Impact of unlabeled data points on mean accuracy of classification - Iris dataset

$\#_{TR}$	$\phi_{TR}acc$	$\#_{UD}$	$\#_{ETR}$	$\phi_{ETR}acc$
40	0.963	20	60	0.944
20	0.944	20	40	0.973
25	0.95	25	50	0.95
15	0.958	15	30	0.925
30	0.944	30	60	0.944
40	0.95	30	70	0.975

compute classification of an average size dataset if decent number of observations were selected per class and not k labeled observations.

Most of the methods that can classify manifold distributed data use random walks over a graph to identify nearest neighbours. Random walk and creating trees to identify nearest or furthest neighbours is a costly activity. It is not worth using these methods unless the gain outweighs the additional cost. Algorithms that use trees ultimately reduces processing cost by compromising other cost intensive processing activities. Like in [1, 12] the cost is minimized by selecting only k labeled points per class for training. This is the same k in K nearest neighbour classification. Obviously, the number is limited to k to keep the cost of creating trees to minimal. Similar approaches are adopted by different algorithms whereby creation of trees is kept to minimal to avoid costly processing.

In short, traditional kNN struggles with classification of manifold distributed data. Graph based approaches can classify manifold distributed data but have very high compute cost. Our method can be described as best of both i.e. high mean accuracy and less compute cost for classification using graph.

3.3. Example

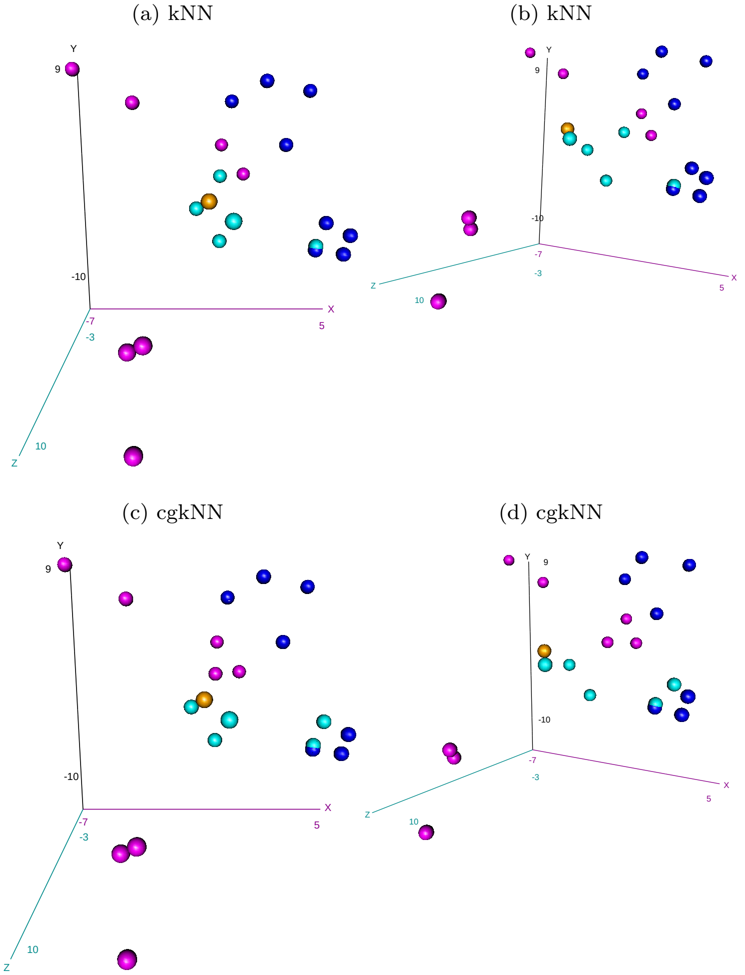
The dataset used in this example is a subset of banknotes dataset. The dataset contains twenty labeled observations, ten from each class and one unlabeled observation. We have intentionally selected a subset that will result in incorrect classification by kNN and correct classification by cgkNN. The objective is to show how traditional kNN can misclassify manifold distributed data based on Euclidean distance and how the same observation is classified correctly using cgkNN. Banknotes dataset has two classes i.e. 1 or 0 and hence our example also has two classes. Figure 3 shows the dataset from two different angles. It also shows the nearest neighbours as classified by Euclidian distance of kNN and also by cgkNN. The yellow point is the unlabeled data point while the aqua colour points represent the nearest neighbours. Blue colour represents class 0 while pink color represents class 1. The nearest neighbours are either class $0, 1$. You can see that observations $2, 9 : 12$ are calculated as the closest five observation to the unlabeled observation using Euclidian distance. The same dataset returns different nearest neighbours with cgkNN. The nearest neighbours as per cgkNN weightage are observations $2, 10 : 12, 19$. As the data points are in three dimensional space, we have shown two different views of the same for each method. The correct calculation of one nearest neighbour results in correct classification of the unlabeled point. kNN classify this unlabeled point as class 1 while cgkNN classify this unlabeled point as class 0 which is correct classification.

Figure 4 shows different views of how cgkNN classify this unlabeled points. It not only calculates the nearest points accurately, it uses the class information to drive a better outcome. The Final weight matrix based on tired random walk of the mentioned dataset is shown in Table 1. Using this matrix cgkNN calculates the sum of weights and thus assigns the unlabeled data point to the class with highest sum of weights.

As mentioned in the algorithm, the nearest points are calculated using similarity matrix $\bar{w}_{ij} = w(x_i, x_j) = \frac{(P_{TRW})_{ij}+(P_{TRW})_{ji}}{2}$. Once a similarity weight matrix is calculated, nearest k labeled points are identified along with their classes. This is followed by sum of weight operations by class. The class with highest sum of weights wins and unlabeled point is labeled with the same class.

Table 1 also shows sum of weights by class for this specific example. The sum of weight for class 0 is higher than the sum of weight for class 1. Thus the algorithm classifies the unlabeled observation to class 0.

Figure 3: Location of unlabeled point and five nearest neighbours - kNN and cgkNN



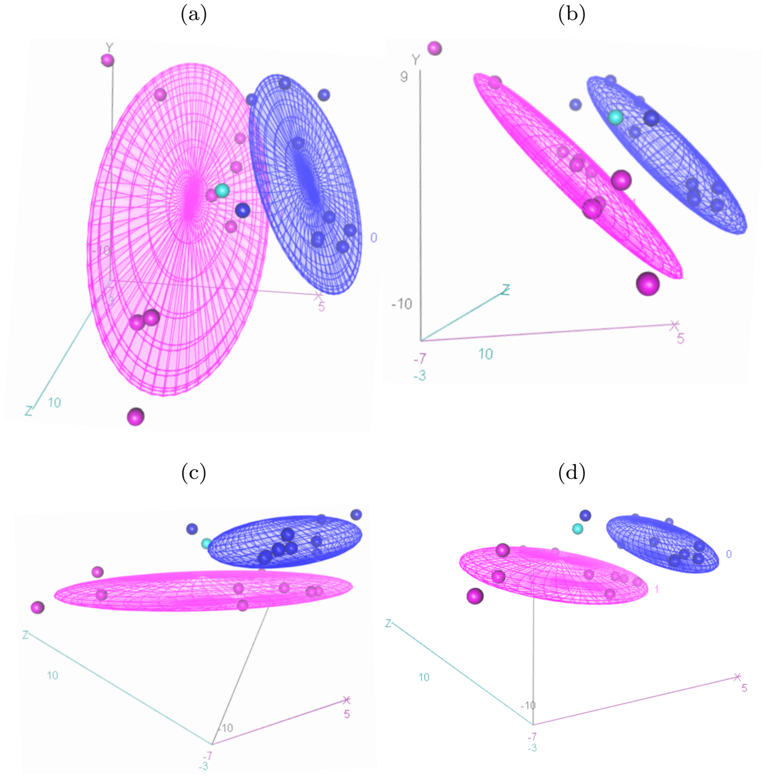
(a) kNN

(b) kNN

(c) cgkNN

(d) cgkNN

Figure 4: Different views of how cgkNN cluster various classes and then classify



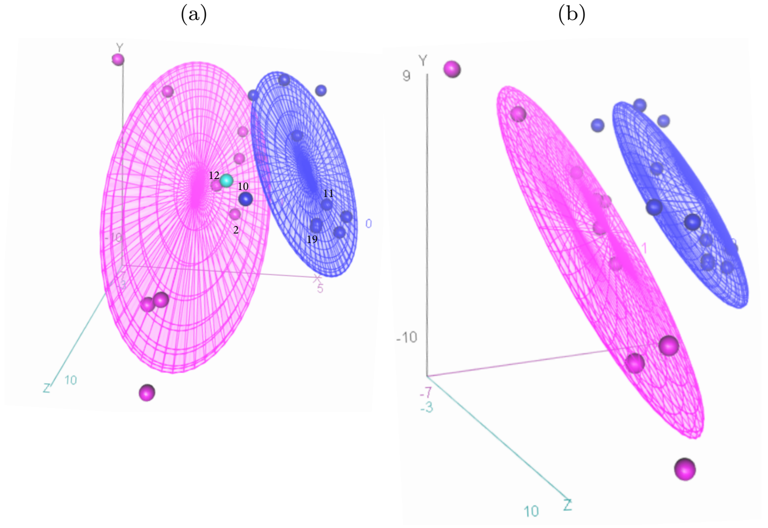
(a)

(b)

(c)

(d)

Figure 5: k closest observations to unlabeled observation



(a)

(b)

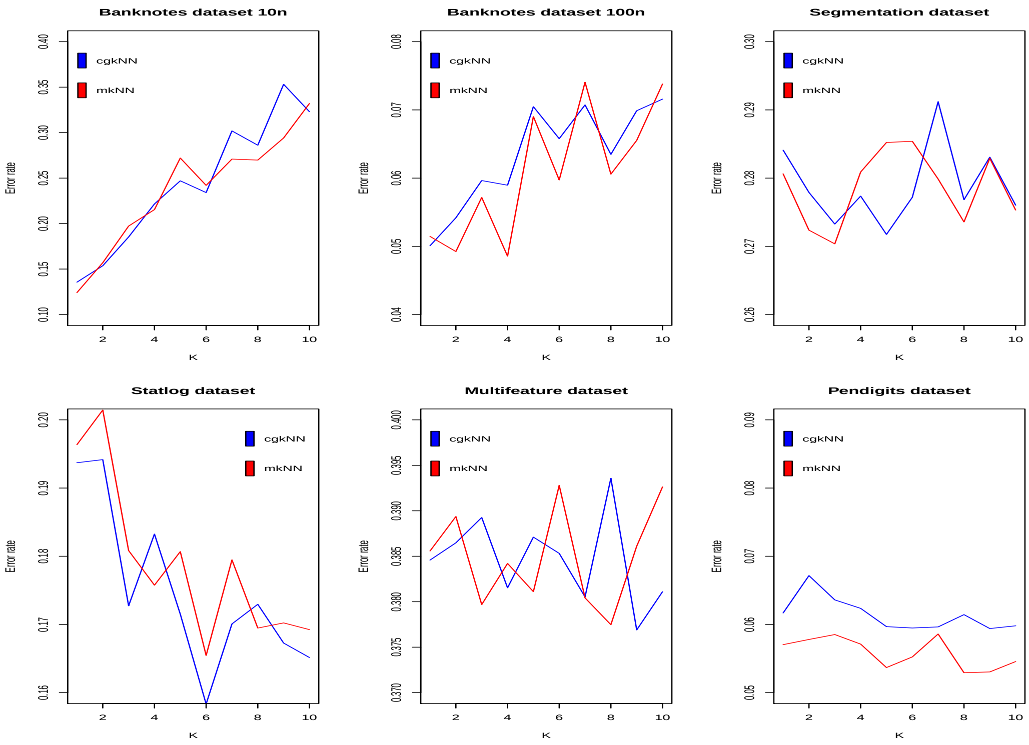


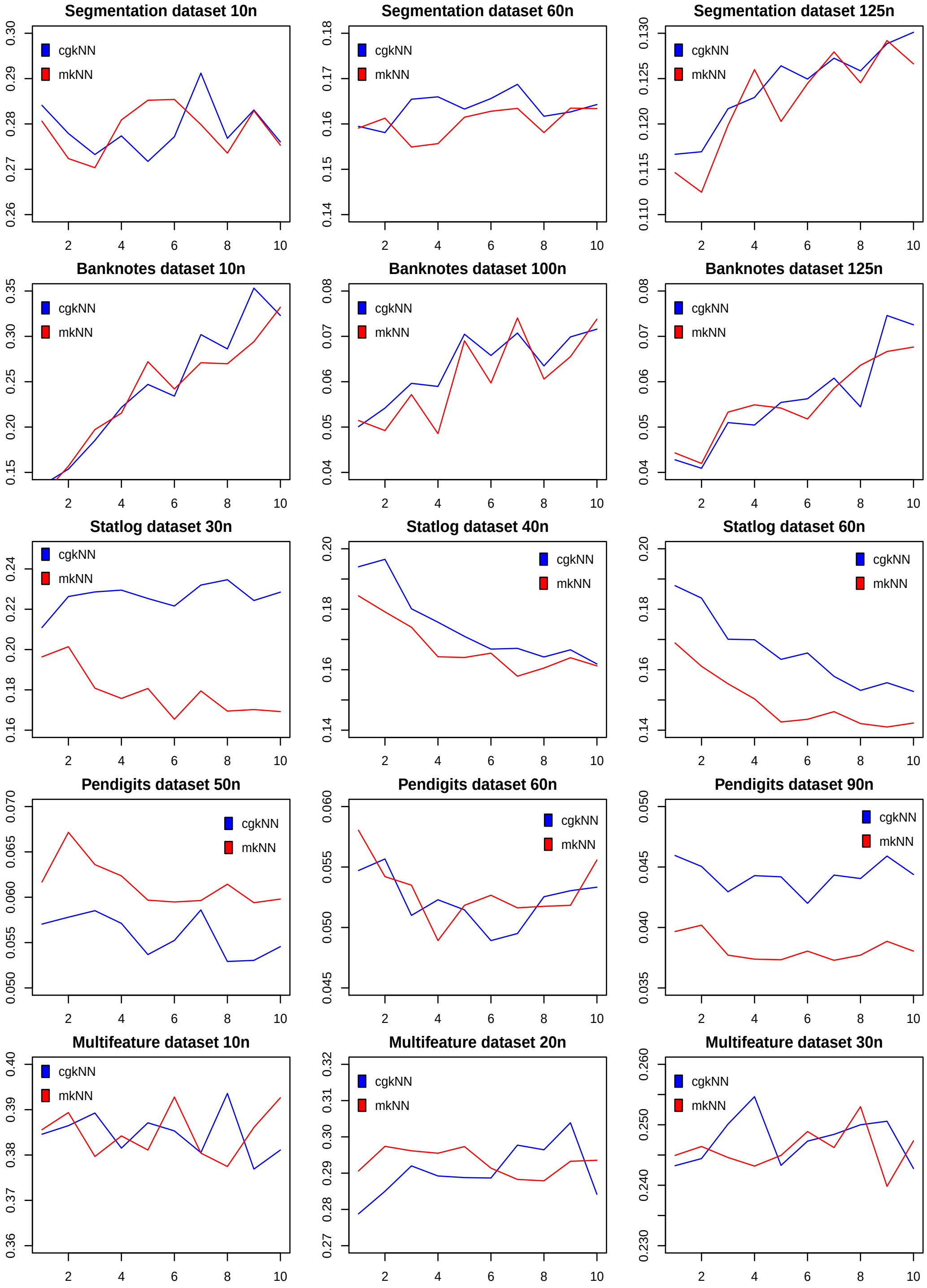
Figure 6: Mean error rate comparision of cgkNN and mkNN

4. Experimental results

We have performed experiments on six publically available datasets. These are the same datasets used by [12]. The objective was to compare computational cost and mean accuracy of prediction for *cgk*NN and *mk*NN. Mean accuracy is calculated using result from ten different iterations with random samples in each iteration. During experiments we noticed that the mean accuracy of predictions using traditional *k*NN declines as the number of k increases. This is quite an obvious pattern in all of the mentioned datasets.

Please refer to Figure 6 to see the comparison of mean accuracy by *mk*NN and *cgk*NN for various datasets. Banknotes dataset is compared twice but with different number of labeled observations. It is very clear from Figure 6 that mean accuracy of *cgk*NN is always inline with *mk*NN irrespective of number of labeled observations available. Table 3 shows mean accuracy for various datasets with different size of training. It should be noted that while *cgk*NN performs as good as or better than *mk*NN with respect to mean accuracy of prediction, it is is much faster than *mkk*NN. Table 3 shows the processing time for various datasets. It is obvious from the results that *cgk*NN can produce same results as *mk*NN but with almost half the time required for computation. It should be noted that these calculations are based on same datasets and all environmental variables such as network, processing power were keep the same to get a comparable result. You would notice that the mean accuracy of classification of all these datasets increases if we consider a good size training dataset. This may not be achievable in scenarios where we have limited labeled observations.

Figure 7 shows the mean error rate of various datasets for k of 1 to 10 using *mk*NN and *cgk*NN. Figure 7 also compares the result for different number of training observations from the same dataset. It is obvious from Figure 7 that the performance of both methods are in line with each other irrespective of number of training observations selected. On the other hand, Table 3 shows a comparison of our method to *mk*NN with respect to mean accuracy and processing time. The results are based on selection of various number of training observations per class. You can see from results that while mean accuracy remains the same for both the methods, our method is much efficient on processing time. Processing time of *mk*NN gets worse as the number training observations increases. The main reason for this is the usage of strengthen trees in *mk*NN which are required for all the labeled training observations. Thus the processing time increases as we



Segmentation dataset 10n

Segmentation dataset 60n

Segmentation dataset 125n

Banknotes dataset 10n

Banknotes dataset 100n

Banknotes dataset 125n

Statlog dataset 30n

Statlog dataset 40n

Statlog dataset 60n

Pendigits dataset 50n

Pendigits dataset 60n

Pendigits dataset 90n

Multifeature dataset 10n

Multifeature dataset 20n

Multifeature dataset 30n

Figure 7: Mean error rate of five real-world data sets with different number of training sets. Mean error rate on the ordinate and k on the abscissa.

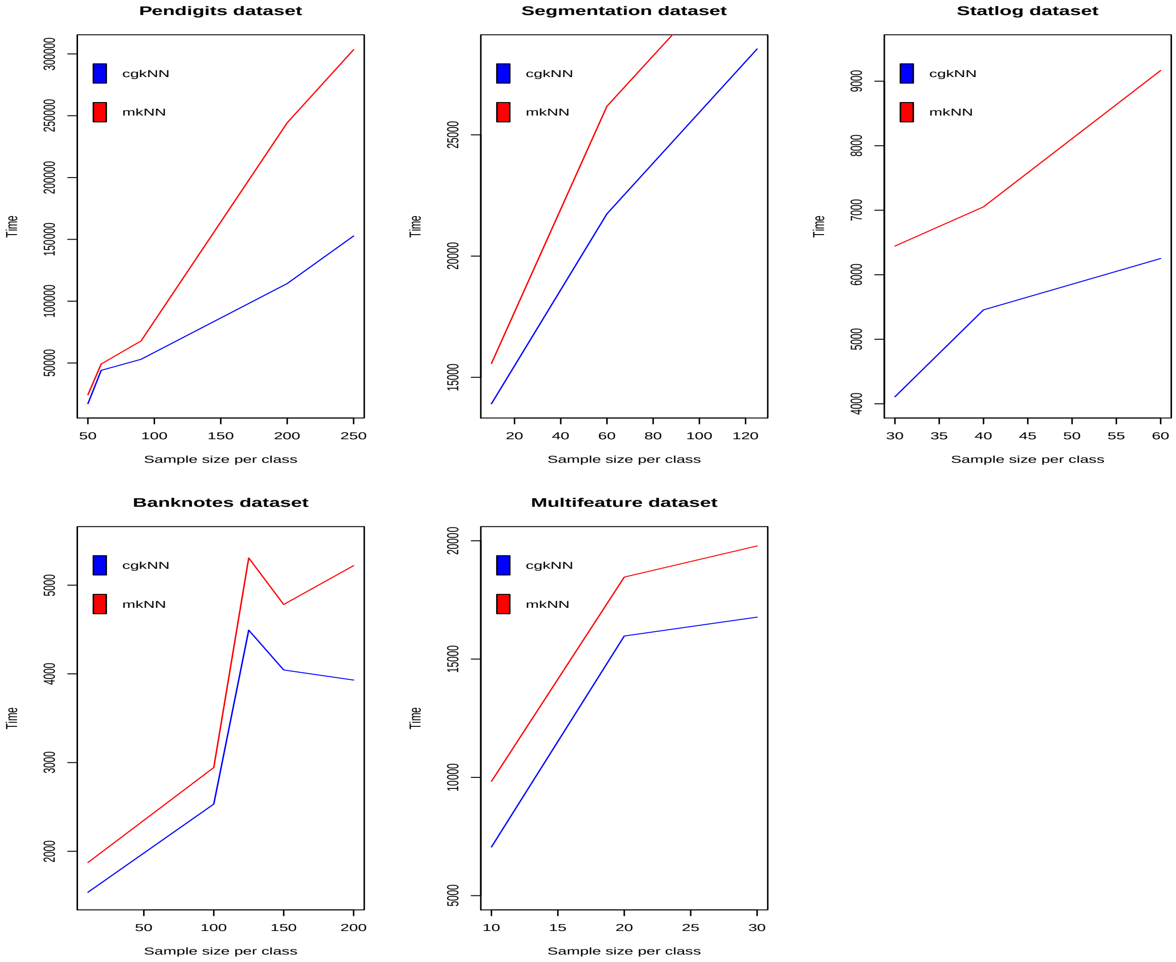


Figure 8: Compute time of five real-world data sets with different number of training sets. Compute time on the ordinate and Sample size on the abscissa.

		ϕ Error		Processing time		Difference	
Dataset	#TR	$cgkNN$	$mkNN$	$cgkNN$	$mkNN$	ϕ Error	Time
Banknotes	20	24.4	23.8	1,538	1,873	0.6	-335
Banknotes	200	6.4	6.1	2,532	2,944	0.3	-412
Banknotes	250	5.6	5.6	4,493	5,307	0	-814
Banknotes	300	5.1	5.2	4,044	4,782	-0.1	-738
Banknotes	400	4.3	4.2	3,930	5,220	0.1	-1,290
Multifeature	100	38.5	38.5	7,055	9,829	0	-2,774
Multifeature	200	29.1	29.3	15,975	18,466	-0.2	-2,491
Multifeature	300	24.8	24.6	16,773	19,782	0.2	-3,009
Pendigits	500	5.6	6.1	17,084	24,234	-0.5	-7,150
Pendigits	600	5.2	5.3	44,068	49,194	-0.1	-5,126
Pendigits	900	4.4	3.8	53,000	67,857	0.6	-14,857
Segmentation	70	27.9	27.9	13,908	15,570	0	-1,662
Segmentation	420	16.4	16	21,742	26,178	0.4	-4,436
Segmentation	875	12.4	12.3	28,541	32,940	0.1	-4,399
Statlog	180	22.6	17.9	4,109	6,445	4.7	-2,336
Statlog	240	17.4	16.8	5,456	7,053	0.6	-1,597
Statlog	360	16.6	14.9	6,252	9,166	1.7	-2,914

Table 3: Mean accuracy of prediction and processing time mkNN vs cgkNN

increase number of observations. Figure 8 shows a comparison of compute time for various samples of the five datasets for mkNN and cgkNN. It is visible that cgkNN performs much better with respect to compute time when compared to mkNN.

5. Conclusions

In this paper we proposed a new extension to k nearest neighbour algorithm to classify non-linear manifold distributed data as well as traditional Gaussian distributed data. We have called this method cgkNN. cgkNN can classify manifold distributed data with high mean accuracy and unlabeled observations given a small number of labeled samples. We have compared our results with one of the best available approach mkNN and have found our method to perform much faster than mkNN and many other approaches available. Our method also performs as good as and at instances better than mkNN with respect to mean accuracy of classification. Our experiments also show that high mean accuracy can be achieved with small number of labeled observations. Method presented in this paper shows that unlabeled observations can add value in term of improvement in accuracy of classification with small compute cost. Our approach, outperforms the best available methods for unlabeled data classification is still very light on compute time.

References

[1] E. Tu, L. Cao, J. Yang, N. Kasabov, A novel graph-based k-means for nonlinear manifold clustering and representative selection, Neurocomputing 143 (2014) 109–122.

[2] D. McClosky, E. Charniak, M. Johnson, Effective self-training for parsing, in: Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics, Association for Computational Linguistics, 2006, pp. 152–159.

[3] J. Tanha, M. van Someren, H. Afsarmanesh, Semi-supervised self-training for decision tree classifiers, International Journal of Machine Learning and Cybernetics 8 (1) (2017) 355–370.

[4] T. M. Cover, P. E. Hart, et al., Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.

[5] Z. Bodo, Z. Minier, On supervised and semi-supervised k-nearest neighbor algorithms, in: Proceedings of the 6th Joint Conference on Mathematics and Computer Science, 2008, p. 1.

[6] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, T. S. Huang, Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (12) (2004) 1553–1566.

[7] M. Peikari, S. Salama, S. Nofech-Mozes, A. L. Martel, A cluster-then-label semi-supervised learning approach for pathology image classification, Scientific reports 8 (1) (2018) 7193.

[8] A. Fujino, N. Ueda, K. Saito, Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (3) (2008) 424–437.

[9] D. P. Kingma, S. Mohamed, D. J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Advances in neural information processing systems, 2014, pp. 3581–3589.

[10] J. He, J. G. Carbonell, Y. Liu, Graph-based semi-supervised learning as a generative model (2007).

[11] O. Chapelle, B. Scholkopf, A. Zien, Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews], IEEE Transactions on Neural Networks 20 (3) (2009) 542–542.

[12] E. Tu, Y. Zhang, L. Zhu, J. Yang, N. Kasabov, A graph-based semi-supervised k nearest-neighbor method for nonlinear manifold distributed data classification, Information Sciences 367 (2016) 673–688.

[13] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al., Top 10 algorithms in data mining, Knowledge and information systems 14 (1) (2008) 1–37.

B Appendix - Copy of Journal Article Sent for Peer Review

Probabilistic Nearest Neighbours for Unlabelled Data Classification

1st Sayed W Qayyumi
School of Computing, Engineering and Mathematics
Western Sydney University
Sydney, Australia
S.Qayyumi@westernsydney.edu.au

2nd Dr Laurence Park
School of Computing, Engineering and Mathematics
Western Sydney University
Sydney, Australia
L.Park@westernsydney.edu.au

***Abstract*—Unlabelled data points contains valuable information that can be used to train classification models. Most of the classification models are dependent on availability of class information for training. Thus unlabelled data points are often ignored while training a classification model. Unlabelled data points can increase the accuracy of prediction in scenarios where models are trained using limited number of observations. In this paper, we have investigated the possible approaches to using unlabelled data points to enrich training datasets. We have also introduced a probabilistic nearest neighbour approach to prediction and using the unlabelled data points. Finally, we have compared our approach to self-trained kNN for unlabelled datasets.**

***Index Terms*—kNN, Classification, unlabelled data points, Probabilistic nearest neighbour**

I. Introduction

Classification and prediction models are trained using historical data with class labels. A trained model's accuracy of prediction is largely dependent on the training dataset. At times, we have available observations that can enrich the training dataset but due to unavailability of class labels they are ignored. This could be due to the fact that appending class label is an expensive and at times, a time consuming exercise. This paper discusses the possibility of using available unlabelled observations to enrich a training set. We are interested to know if unlabelled i.e. data points that have no class information can add value in training a model. We are especially interested in reviewing performance of a model before and after the unlabelled points are added. A new probability nearest neighbour pNN approach is introduced that can use unlabelled data points. pNN labels the unlabelled data points. These labelled points are then added to existing labelled data points to enrich training dataset. The mentioned approach uses distance measure along with probability to predict class labels. Finally, we have compared our method to self-trained kNN. kNN is one of the famous available methods for classification. We have found that our method is able to improve classification and average accuracy of prediction when compared to self-training kNN.

There are three scenarios that we want to discuss and validate here;

1) Can we use unlabelled observations in training models without a degradation in average accuracy of prediction?
2) What is the best technique of using available unlabelled points to train a model?
3) Can we identify unlabelled points that result in improvement of average prediction accuracy?

II. Enrichment of Training Set with Unlabelled Data Points

Training datasets can be enriched by adding more observations but adding unlabelled observations to a training dataset may or may not enrich the training datasets. Class labels which are important attributes in enriching a training datasets are not available in unlabelled observations. We have used two datasets i.e. Iris and observations disease datasets to validate our approach of enriching training dataset. We have run at least 30 iterations of random training and test data sets from each of the mentioned data sources. In each iteration we have divide the data into three datasets;

1) TR dataset: This is the training dataset with a labelled class.
2) UD dataset: This is a dataset for unlabelled data points.
3) TS dataset: This is test dataset which is used to test accuracy of prediction in all scenarios.

We are using a random set of *n* labelled observation as our training dataset TR. Similarly, we have m random observation as new unlabelled data points UD. We have used these datasets to train kNN and our model. Same test dataset TS is used to calculate average accuracy of prediction using both methods. kNN [1] with self-training is compared to our Probabilistic nearest neighbour approach for prediction accuracy.

$$ETR = TR \cup UD \tag{1}$$

$$Predict(TR, TS) \approx Predict(ETR, TS) \tag{2}$$

TABLE I
IMPACT OF UNLABELLED DATA POINTS ON ACCURACY

TR	Accuracy	UD	ETR	Accuracy
40	0.963	20	60	0.944
20	0.944	20	40	0.973
25	0.95	25	50	0.95
15	0.958	15	30	0.925
30	0.944	30	60	0.944
40	0.95	30	70	0.975

III. RELATED WORK

There are many different approaches available for semi-supervised learning [2]. Self-training is one of the basic and most widely used semi-supervised method [3] [4] [5]. These methods are based on some common assumption like smoothness assumption (SA), cluster assumption (CA) and manifold assumption (MA) [1]. A smoothness assumption states that data in high density regions should share the same label while cluster assumption assumes that the points in same cluster will have same label. Manifold assumption is assuming that the data lies in low manifold. Many generative [6] and graph based semi-supervised learning methods [7] are also available. Self-training is about using labelled data to train a model. The trained model is then used to predict the class of unlabelled observations and thus the predictions with high confidence is then added to training dataset along with predicted class label. A new model is then trained on extended training dataset. In this paper we are comparing the self-training kNN with our probabilistic nearest neighbour pNN model.

IV. SELF-TRAINING kNN

kNN is one the simplest method for classification. If we consider X as the matrix of features for an observation and Y is the class label, kNN looks at k (a positive integer) and estimates the probability of it belonging to class j for a test observation x_0.

$$Pr(Y = j|X = x_0) = \frac{1}{k}\sum_{i\in\mathcal{N}_0} I(y_i = j) \qquad (3)$$

The self-training kNN involves predicting class of unlabelled observations using kNN [8]. kNN is trained using training set TR and unlabelled points are then labelled using the predicted class [1] [9]. The unlabelled points along with their respected predicted class are then added to training set TR. We call this new dataset as Extended Training set ETR. See Table I for the impact adding of unlabelled point on accuracy of prediction. Average accuracy of prediction is calculated before and after the addition of unlabelled points. While, the number of observation remains the same in TR and UD dataset, different random observations are picked in each iteration. See Figure 1. In this figure each accuracy number is for the same size of TR and UD but different random observations.

A distribution of 30 such iteration is used to calculate the significance of results. Paired t-test and p-value for these distributions are given in Table I. These values show that there is no significant difference in accuracy of prediction before and after labelled points were added. What this signifies is the fact that while the approach is good for using unlabelled points, there is no improvement in the average accuracy of prediction.

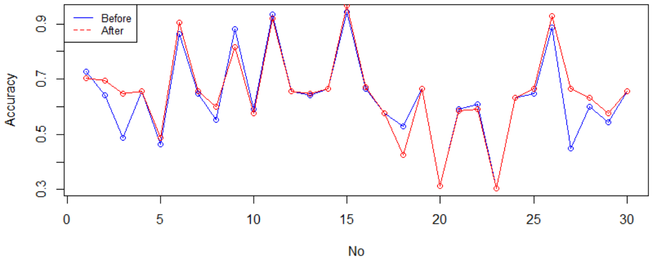


Fig. 1. Accuracy before and after self-training kNN - Iris Dataset

$$p - value = 0.7837 \qquad (4)$$

$$t = -0.27707 \qquad (5)$$

V. PROBABILISTIC NEAREST NEIGHBORS pNN

This is a new approach that we are introducing in this paper. The approach is similar to kNN but a class label is predicted using a probabilistic approach i.e. probability of an observation belonging to a specific class. The probability is calculated by calculating probability of an item belonging to a specific class with respect to the K closest points. See Table V which shows the nearest five points to a point in UD dataset from Iris dataset. A class is assigned to the unlabelled point based on the highest sum of probability see table IV. The probabilistic approach assigns probability to all points. For labelled points it assigns 1 to the labelled class while for unlabelled points it assigns probability by using the k nearest neighbours from TR dataset. The idea is to predict the class of unlabelled points from UD dataset and then add them to the main training set TR. The new extended training set ETR which is *n* + *m* observations, is used to predict class of a test set TS. Probabilistic approach is followed to predict the class of test set. We expect that the accuracy of prediction based on extended training set ETR to be the same or higher than the accuracy based on initial training set TR with *n* observations. You can see from table V that the nearest points for the unlabelled point is either of class *Versicolor* or *Virginica*. The probability of this point belonging to each class is given in table IV. As the probability of this unlabelled point for class *Versicolor* is the highest, the unlabelled point is classified as class *Versicolor*. We classify all unallocated points in the same way. The UD dataset is then added to the training set TR along with the newly predicted class. This method also calculates the optimum value of k based on the maximum value of the difference in means of the average accuracy before and after the unlabelled observations are added.

TABLE II
OPTIMUM K SELECTION

K	Accuracy Before	Accuracy After	Difference
19	0.5261808	0.5234818	-0.0026991
18	0.5238866	0.5213225	-0.0025641
17	0.5214575	0.5202429	-0.0012146
16	0.5186235	0.5191633	0.0005398
15	0.5180837	0.5194332	0.0013495
14	0.5167341	0.5168691	0.000135
13	0.5161943	0.5160594	-0.000135
12	0.5164642	0.51417	-0.0022942
11	0.5109312	0.51417	0.0032389
10	0.5068826	0.5063428	-0.0005398
9	0.5040486	0.5024291	-0.0016194
8	0.5051282	0.497031	-0.0080972
7	0.494197	0.4867746	-0.0074224
6	0.4824561	0.4808367	-0.0016194
5	0.468691	0.4659919	-0.0026991

$$Pr(Y = j|X = x_0) = \frac{1}{k}\max(\sum_{i \in \mathcal{N}_0} Pr(y_i = j)) \quad (6)$$

A. Opimal K

The method calculates optimal k by finding the maximum value of the difference in means of the prediction accuracy before and after unlabelled data points are added. If AB was accuracy before and AA was accuracy after adding the unlabelled points, the probabilistic approach will calculate k as per below.

$$k = \max(\overline{AB} - \overline{AA}) \quad (7)$$

The difference in mean of accuracy before and after the addition of unlabelled points is also significant. The self-training kNN approach shows almost no improvement in difference of means, while the Probabilistic approach shows a considerable improvement. The results of t-test and p-values based on accuracy from 30 random training set TR and ETR dataset is given in table I.

Table II shows optimal k value selection from a snapshot of heart disease dataset. The algorithm converges by selecting k with the highest difference in accuracy. Algorithm starts with K= 3 and increases by 1 in each iteration till it reaches k=30. Optimal k is selected from the list of 27 available values.

$$p - value = 0.03607 \quad (8)$$

$$t = -2.1982 \quad (9)$$

This method performs much better than the kNN approach with respect to prediction on TR dataset and also on the ETR dataset. The average accuracy of this approach is higher than the average accuracy of the kNN approach. Figure 2 shows a distribution of 30 different iterations of random TR and UD datasets, each of them with the same number of observations. The Paired t-test and p-value using probabilistic approach is also given. You can see that the results are significant. This signifies that the method not only let you add the unlabelled points to enrich your training set but also uses the data points to improve average accuracy of prediction.

TABLE III
COMPARISON OF SELF-TRAINED kNN AND PROBABILISTIC APPROACH -IRIS DATASET

No	Self-training kNN		Probabilistic Approach	
	P-Value	t-test	p-Value	t-test

TD = 15
UD = 20
	0.7837	-0.27707	0.03607	-2.1982

TD = 10
UD = 15
	0.1469	-1.4904	0.0000009	-6.1904

TD = 25
UD = 25
	0.4385	-0.78551	0.1094	-1.6515

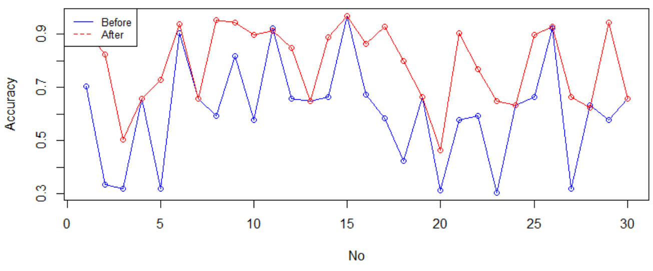


Fig. 2. Accuracy before and after - Probabilistic Approach, Iris dataset

Tables III shows a comparision of significance of results of average accuracy of prediction by self-trained kNN and Probabilistic nearest neighbours model.

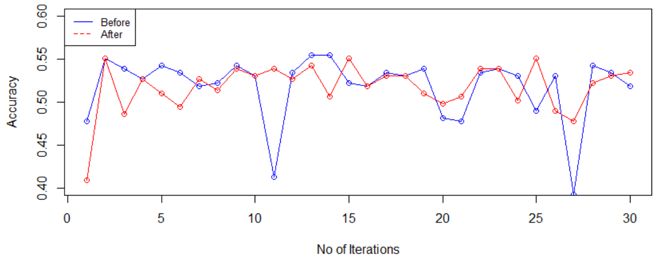


Fig. 3. Accuracy before and after - self-training kNN, Heart Disease dataset

VI. HEART DISEASE DATASET

To validate and compare our approach to labelling unlabelled data and prediction to kNN self-training, we have tested our approach on heart disease dataset also known as cleveland dataset. The data was collected by V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. The dataset has 14 attributes. The task is to detect the presence of heart disease in the patient. It is integer valued from 0 showing no presence to 3 showing presence of heart disease. The dataset is divided

TABLE IV
FIRST UNLABELLED DATA POINT

Sepal Length	Sepal Width	Petal Length	Petal Width	Prb Versicolor	Prb Sethosa	Prb Verginica
7	3.2	4.7	1.4	0.6	0	0.4

TABLE V
FIVE NEAREST POINT TO THE FIRST UNLABELLED POINT

Sepal Length	Sepal Width	Petal Length	Petal Width	Class	Prb Versicolor	Prb Sethosa	Prb Verginica	Distance
6.8	2.8	4.8	1.4	versicolor	1	0	0	0.4583
6.3	2.7	4.9	1.8	virginica	0	0	1	0.9695
6.5	2.8	4.6	1.5	versicolor	1	0	0	0.6557
6.1	3	4.6	1.4	versicolor	1	0	0	0.9274
6.9	3.1	5.4	2.1	virginica	0	0	1	1

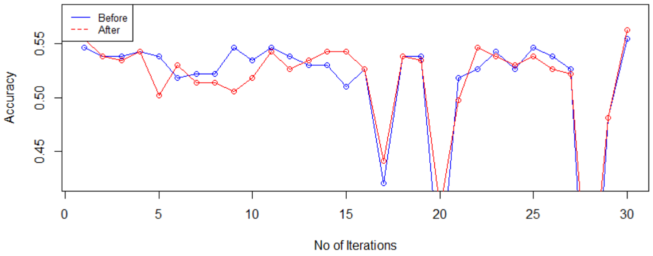


Fig. 4. Accuracy before and after - pNN, Heart Disease dataset

in to three datasets i.e. TD, UD and TS. The results of kNN self-training and Probabilistic approach are shown in the Fiqure 3 and Figure 4 respectively. Difference of means, p-value and paired t-test results are also shown in below. You can see from the results that Probabilistic return better and improved results. We have not restricted our self to only this two datasets but have applied this approach to other datasets, while some datasets shows major improvement in training dataset enrichment others have noticeable improvements. In all experiments we conducted, we have witnessed improvement of probabilistic approach on self-training kNN.

REFERENCES

[1] Bodo, Zalan and Minier, Zsolt , On supervised and semi-supervised k-nearest neighbor algorithms, Proceedings of the 6th Joint Conference on Mathematics and Computer Science 2008 , pp. 1

[2] Peikari, Mohammad and Salama, Sherine and Nofech-Mozes, Sharon and Martel, Anne L, A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification, Scientific reports 2018, volume=8, pp. 7193

[3] Rosenberg, Chuck and Hebert, Martial and Schneiderman, Henry, Semi-Supervised Self-Training of Object Detection Models, WACV/MOTION 2005 pp. 29–36

[4] McClosky, David and Charniak, Eugene and Johnson, Mark, Effective self-training for parsing, Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics 2006 pp. 152–159

[5] Tanha, Jafar and van Someren, Maarten and Afsarmanesh, Hamideh, Semi-supervised self-training for decision tree classifiers, International Journal of Machine Learning and Cybernetics, 2017 volume 8, pp. 355–370

[6] Fujino, Akinori and Ueda, Naonori and Saito, Kazumi,Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle, IEEE Transactions on Pattern Analysis and Machine Intelligence 2008, pp. 424–437

[7] He, Jingrui and Carbonell, Jaime G and Liu, Yan, Graph-Based Semi-Supervised Learning as a Generative Model 2007, pp. 2492–2497

[8] Cover, Thomas and Hart, Peter, Nearest neighbor pattern classification, IEEE transactions on information theory 1967, pp. 21–27

[9] Cohen, Ira and Cozman, Fabio G and Sebe, Nicu and Cirelo, Marcelo C and Huang, Thomas S, Semi-supervised Learning of Classifiers: Theory, Algorithms for Bayesian Network Classifiers and Application to Human-Computer Interaction, 2003

C Appendix - Proposed Research Schedule

	Month / Year																																									
	2018						2019												2020												2021											
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
Literature Review - General	🟩	🟩	🟩	🟩	🟩	🟩																																				
Formulate Research Questions					🟩	🟩	🟩	🟩																																		
Review of existing research (Graph based Methods)							🟩	🟩																																		
Identification of gaps and weaknesses								🟩	🟩																																	
Experimenting to addresss gaps and weaknesses									🟩	🟩																																
Introduction of new / extension of existing algorithms										🟩	🟩																															
Application and experimenting with new datasets											🟩	🟩																														
Report findings												🟩																														
Collate all Information, Begin writing article												🟩	🟩																													
Submit to journal													🟨																													
Prepare for Confirmation													🟨																													
Literature Review - Image classification													🟨	🟨	🟨	🟨	🟨	🟨																								
Review of existing research (Facebook research)																	🟨	🟨																								
Identification of gaps and weaknesses																		🟨	🟨																							
Experimenting to addresss gaps and weaknesses																			🟨	🟨																						
Introduction of new / extension of existing algorithms																				🟨	🟨																					
Application and experimenting with new datasets																					🟨	🟨																				
Report findings																						🟨																				
Collate all Information, Begin writing article																						🟨	🟨																			
Submit to journal / conference																							🟨																			
Prepare for Confirmation																							🟨																			
Literature Review - Metric Learning and Self Learning																					🟨	🟨	🟨	🟨	🟨	🟨																
Review of existing research - Google Research / LMNN Kilian Weinberger																									🟨	🟨																
Identification of gaps and weaknesses																										🟨	🟨															
Experimenting to addresss gaps and weaknesses																											🟨	🟨														
Introduction of new / extension of existing algorithms																												🟨	🟨													
Application and experimenting with new datasets																													🟨	🟨												
Report findings																														🟨												
Collate all Information, Begin writing article																														🟨	🟨											
Submit to journal / conference																															🟨											
Prepare for Confirmation																																🟨										

Completed 🟩
Scheduled 🟨

Figure 15: Proposed Research Schedule

List of Tables

List of Figures