Decoding Autoencoded Supernovae Spectrum Features

Vongani Cyril Chabalala (vongani@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr. Yabebal Fantaye
AIMS South Africa, University of Stellenbosch

12 October 2018

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Recent observations of Type Ia Supernovae (SNeIa) have divulged the existence of subtypes of SNeIa. One of the challenges I met was to identify spectral features of subclasses of SNeIa by comparing two figures. In this paper we portray how machine learning algorithms make it easier to identify subtypes of Type Ia Supernovae. Deep Learning used was able to perform such identification in a 4 dimensional feature space. Using K-means different groups of spectral of subclasses of SNeIa were produced in search of their main spectral features. 91bg-like events confirmed determine subclasses of SNeIa. Given that there will be a large amount of data in the future, Deep Learning approach will still be usefull to identify subtypes of Type Ia Supernovae.

Keywords: SNIa- Machine learning and data analysis

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Vongani Cyril Chabalala, 12 October 2018

Contents

1. Introduction

Cosmologists in 1998 have shown that the expansion of the universe is accelerating Chakraborty (2011). This acceleration is believed to be powered by an unknown energy source called dark energy. This discovery was made possible by studying Type Ia supernovae (SNeIa) - which are among the most bright objects in the Universe.

Type Ia supernovae show a high degree of uniformity in their spectrum and absolute brightness. This property makes them an ideal standard candle for measuring distant galaxies, which in turn allows cosmologists to infer the expansion rate of the universe.

Classification technique of supernovae (SN) has been developed since 1941, identifying two types of Supernovae(SN): Type I and type II. Type I supernovae do not show hydrogen lines (i.e they are characterized by the absence of hydrogen in its composition) in their spectra but they shows a Si P-Cygni feature at maximum light.

SNeIa are caused by thermonuclear reaction of a binary star systems that consist of a white-dwarf accreting matter from a companion star. The white-dwarf eventually accumulate so much mass that its core reaches a critical density that leads to an uncontrolled fusion of carbon and oxygen, thus releasing a consistent large amount of energy.

Type Ia Supernovae is classified mainly based on its optical spectrum at maximum light (Nordin (2011)). Figure 1.1 shows the optical spectra of SNe Ia containing neural and singly ionized lines of Si, Ca, Mg, S and O at maximum light. Fe II lines which are allowed dominates the spectra two after maximum . Forbidden Fe II, Fe II and Co III omission lines become spectral features a month after peak brightness and some Ca II lines are dominant at late times(Hillebrandt and Niemeyer (2000)).

Although a lot of attention has been give to the study and observation of SNeIa, we still do not have a complete picture about their progenitor systems. All type Ia supernovae show some dispersion in their maximum but their maximum spectra looks similar. Moreover, there is no consensus in the literature whether all SNeIa belong to one homogeneous class or there are sub-classes Foley et al. (2012).

In this essay, we consider Machine Learning (ML) tools to study the presence of SNeIa sub-classes in an unsupervised manner using their spectroscopic features. Our approach involves two steps: first we employ Autoencoders to reduce the spectral data from about 300 dimension to less than 20 dimensions, and then later using the K-mean clustering algorithm to infer evidence for the presence of sub-classes. Finally, we will compare our finding with an already existing human classifications.

1.1 Light curve

A light curve is one of characteristics of any type of supernovae. It is a graph of B-band magnitude - a particular frequency interval - as a function of time(t). Supernovae phases are defined in the light curve with respect to the maximum brightness, which is by definition t=0 - day zero. Figure 1.2 shows that the negative numbers correspond to the days before the peak brightness and t=0 correspond to the time of maximum light in the B-band. At t=0, the light curve turns over and begin to decline in brightness for the next few weeks. Large amount of high-quality optical data shows that SNeIa light curves have distinct variations. In a classic work in 1993, however, Mark Phills carnegiescience demonstrated a correlation between the peak brightness of SNeIa and the rate at which the brightness declines in the

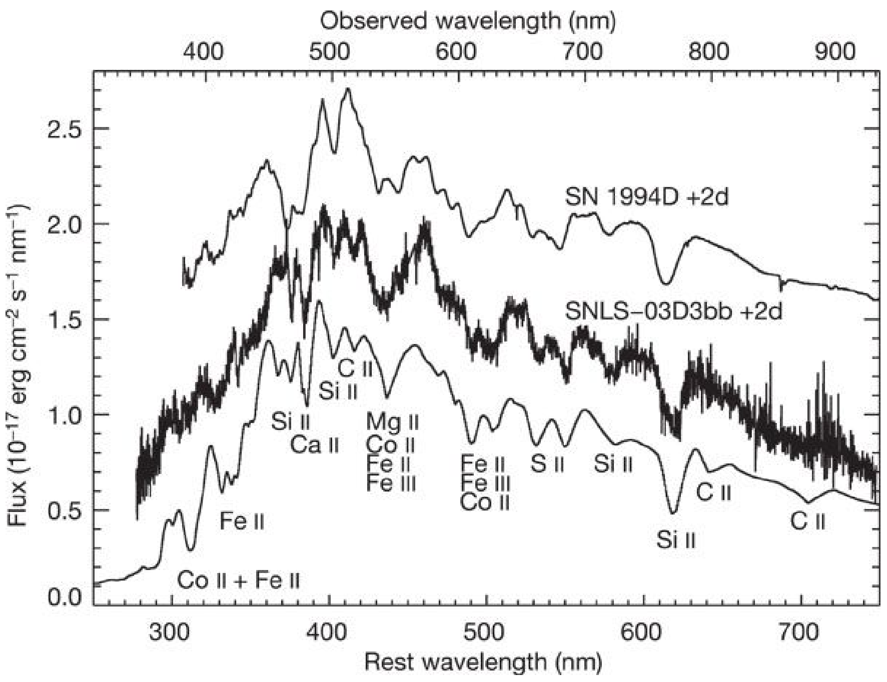


Figure 1.1: optical spectra of SNe Ia containing neural and singly ionized lines of Si, Ca, Mg, S and O at maximum light.(Hillebrandt and Niemeyer (2000))

first 15 days after maximum in the B-band. This correlation, which states brighter supernovae decline more slowly while fainter supernovae decline more rapidly, was crucial for using SNeIA as standard candles. Hillebrandt and Niemeyer (2000).

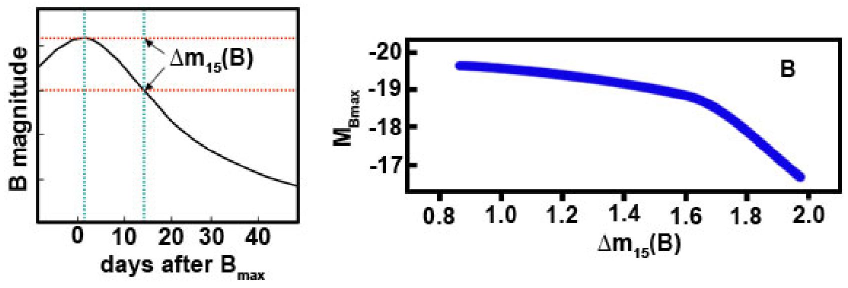


Figure 1.2: Shows the maximum B-band luminosity which illustrate how fast the light curve declines. Fainter objects decline at high speed and brighter objects decline slowly (Light Curves, IITR).

2. Review

2.1 Machine Learning

The word machine learning (ML) was invented by the American computer scientist Arthur Lee Samuel in 1959 (Fürnkranz (2001)). Machine learning is a method used to create advanced models and algorithms that are used in predictive data analyses. In other words machine learning is an area in artificial intelligence that focuses on creating programs that use data to learn themselves. There are four types of machine learning namely: supervised learning, unsupervised learning, semi-supervised machine learning and reinforcement machine learning algorithms. For this study, we will focus mainly on both unsupervised learning and supervised learning to identify SNeIa sub-classes.

2.1.1 Supervised Machine Learning. Supervised learning can be defined as a learning algorithm which generates a function that maps the training data to the desired output data. In supervised learning, the training data are given as labelled data and the output of the result is known. The are two types of Supervised learning namely: classification and regression. In regression learning, prediction of results are in continuous output while the prediction of results are in discrete output in classification learning (Sup).

2.1.2 Unsupervised Machine Learning. Unsupervised machine learning is a sub-field of machine learning techniques which is used to find the patterns and structure of a data set by reducing and clustering the data. Unsupervised learning deals with unlabelled data. The main focus of unsupervised Machine Learning algorithms is to be able to reduce the dimensionality of the data in unsupervised manner and cluster it. There are two commonly algorithms used in unsupervised learning: Clustering and data compression. There are a couple of popular algorithms which are used to reduce dimensionality of the data such as the Principal Component Analysis (PCA) and Autoencoders, we will review both of these algorithms below.

2.1.3 Semi-supervised Learning. This type of machine learning is employed usually when one has a large amount of unlabelled data together with a small amount of labeled data. In this setting, a combination of supervised and unsupervised techniques can be used such that an improved accuracy is achieved when compared to using unsupervised learning alone. (Sem).

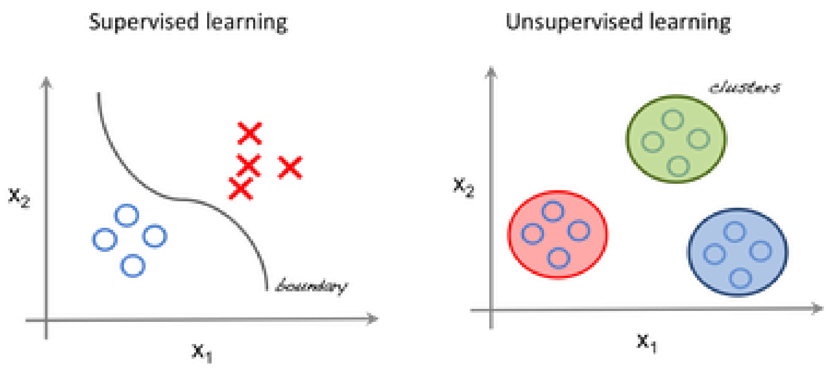


Figure 2.1: Illustration of supervised and unsupervised learning (Uns).

2.1.4 Reinforcement Learning. Reinforcement Learning is a sub-field of machine learning techniques that learn based on environmental interaction. The data is recorded during the interaction with the environment (Rei). Cleaning robot is one of the examples of Reinforcement Learning (Cruz (2017)).

2.1.5 Deep Learning. Deep learning (DL) is one of the machine learning methods which learns from different data types like images, texts and sounds by imitating the workings of human brains. DL has lured a lot of attention because it is good at a type of learning (supervised, unsupervised, semi-supervised and reinforcement learning) that is useful for real world application. A traditional neural networks only consist of only one hidden layer where as Deep neural networks consist of two or more hidden layers.

2.1.6 Neural network. An artificial neural network (ANN) tries to copy the functionalities of a biological neural networks. The artificial neural network consist of the summation function, input layers, output layers and activation function (Sigmoid, Relu, Tanh) Krenker et al. (2011). The first layer of a neural network which is used to provide the input data or features to the network is called the input layer. The Output layer gives out the predictions Thukral (2016).

Multiplication, summation (transfer function) and activation are basic operations of artificial neural network (ANN) Bangal (2009). Figure 2.2 shows a basic unit of ANN (i.e a neuron or node).

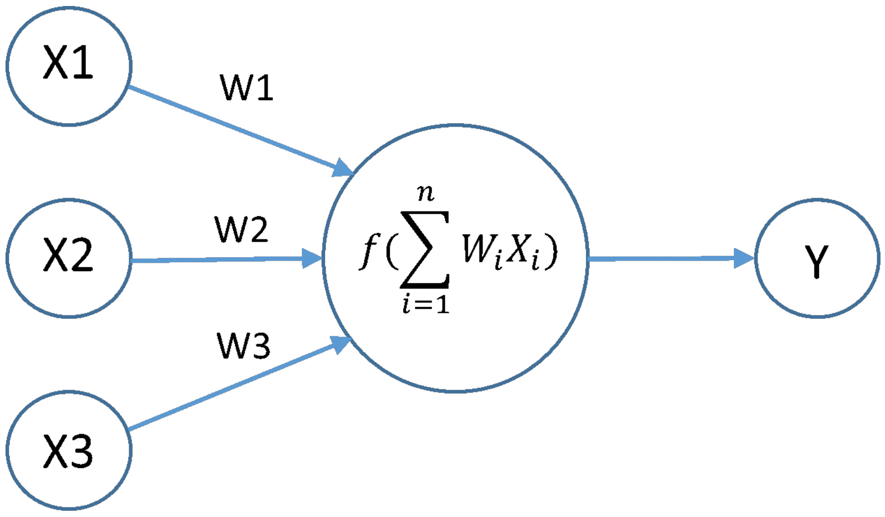


Figure 2.2: A neuron showing the input $(x_1, x_2, x_3, ..., x_n)$, weights $(w_1, w_2, w_3, ...w_n)$ and the activation function to the weighted sum of the inputs (Google).

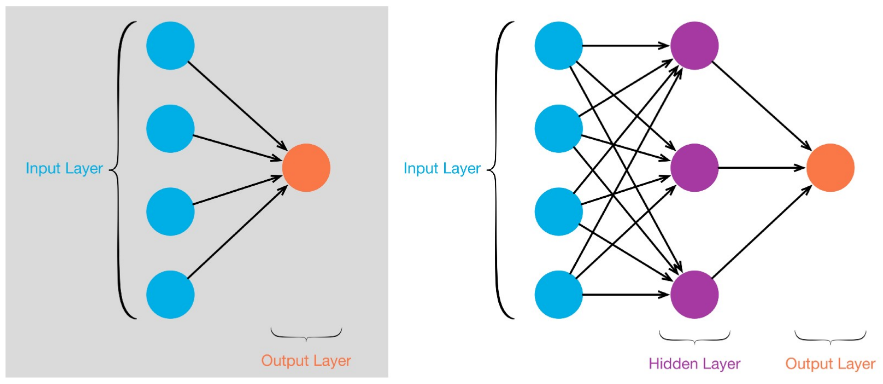


Figure 2.3: Single and Multi-layered (datascience.com)

There are many different structures of ANN namely: radial basis function (RBF) network, feed-forward network, recurrent network, etc. ANNs consist of an input layer, hidden layer(s) and output layer(s).

Commonly, all these networks are trained using a back-propagation (BP) algorithm. Figure 2.3 is an example of a single and multi-layered ANN. Figure 2.4 illustrate feed forward propagation neural network.

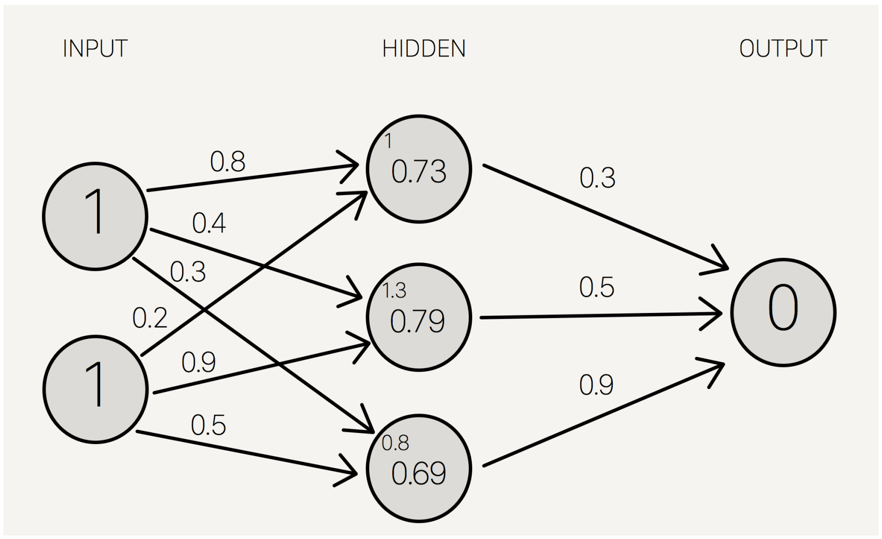


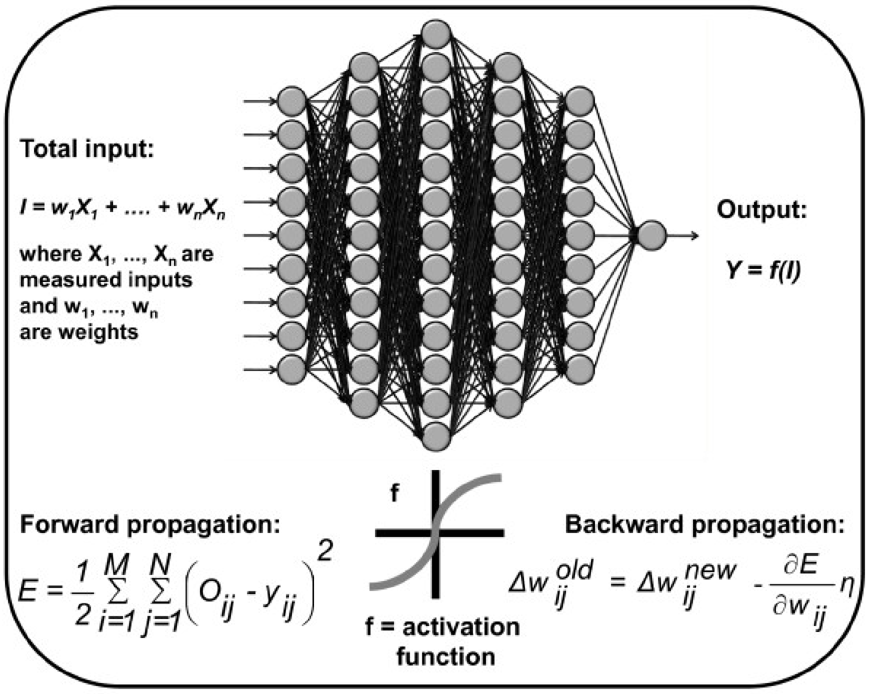
Figure 2.4: Illustration of feed forward propagation neural network (Neural Network).

The error between the desired output and input, which is encoded using different type of loss functions, is minimized by updating the network weights using a backpropagation algorithm - Figure 2.5 shows a backpropagation training procedure.

Gradient of the error minimizing steps:

- Firstly the weights are initialized.
- Perform Forward propagation from the input layer to the output layer.
- Compute the gradient of the error.
- Perform back propagation from the output layer to the input layer to update the weights of each layers.
- Repeat the above steps until the gradient of the error converges to a tolerance ρ.

We can illustrate Back Propagation by taking a neural network with 3 layers where L is the input layer , L-1 and L-2 are hidden layers and L-3 is the output layer. We will show the derivation of these errors below.Let k be a neuron in layer L, j be a neuron in layer L-1 and i be a neuron in layer L-2 for a neural network with 3 layers.The computed output of he j^{th} neuron is defined as


Total input:
I = w1X1 + ... + wnXn
where X1, ..., Xn are measured inputs and w1, ..., wn are weights
Output:
Y = f(I)
f
Forward propagation:
$E = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N}(O_{ij} - y_{ij})^2$
f = activation function
Backward propagation:
$\Delta w_{ij}^{old} = \Delta w_{ij}^{new} - \frac{\partial E}{\partial w_{ij}}\eta$

!h

Figure 2.5: Illustration of the back propagation neural network architecture (Researchgate).

$$o_j = a(f(x_i, w, b)) \tag{2.1.1}$$

$$= \frac{1}{1 + (\sum_{k=1}^{n} w_k x_{jk} + b_k)} \tag{2.1.2}$$

$$= a(\sum_{k=1}^{n} w_k x_{jk} + b_k) \tag{2.1.3}$$

$$= a(w^l o^{l-1} + b_j^l), \tag{2.1.4}$$

$$\tag{2.1.5}$$

where o^{L-1} is the computed output of the layer L-1 and o^L is the computed output in layer L. The output of the first layers is the input of the second layer to the right. x_j is the output that multiply the weight vector with weights. The expression a() is the activation function, b is the bias and f(x,w,b) is the summation function.

There are different types of loss function. The two common examples are cross entropy and sum of square error(SSE) loss functions. The cross entropy loss function for classification can be defined as

$$C(w, b) = \frac{1}{n}\sum_i \sum_j y_j(x_i) log_2 o_j^L(x_i) + (1 - y_i(x_i)) log_2(1 - o_j^L(x_i)) \tag{2.1.6}$$

The change in weights connects a fixed neuron j with output neuron k. They can be derived as follows

$$\eta\frac{\partial C(w,b)}{\partial w_{jk}^{L}} = \eta\frac{\partial C(w,b)}{\partial o^{L}}\frac{\partial o^{L}}{\partial f(w,x,b)}\frac{\partial f(x,w,b)}{\partial w_{jk}^{L}} \tag{2.1.7}$$

$$= -\mu\frac{1}{n}\sum_{i}(y_j\frac{o_k^{L-1}a^{'}}{a(x_k^L)} - \frac{(1-y_j)o_k^{L-1}a^{'}(x_j^L)}{1-a(x_k^L)}) \tag{2.1.8}$$

$$= -\mu\frac{1}{n}\sum_{i}(\frac{y_j - a(x_k^L)}{a(x_k^L)(1-a(x_k^L))})o_k^{L-1}a^{'}(x_j^L) \tag{2.1.9}$$

$$= -\mu\frac{1}{n}\sum_{i}(o^L - y_j)o_k^{L-1} \tag{2.1.10}$$

$$= -\mu\frac{1}{n}\delta_j^L o_k^{L-1} \tag{2.1.11}$$

where $\delta_j^L = \frac{1}{n}\sum\limits_{i}(o^L - y_j)$,$\eta$ is the learning rate. The change in bias is derived as follows

$$\eta\frac{\partial C(w,b)}{\partial b_{j}^{L}} = \eta\frac{\partial C(w,b)}{\partial o^{L}}\frac{\partial o^{L}}{\partial f(w,x,b)}\frac{\partial f(x,w,b)}{\partial b_{j}^{L}} \tag{2.1.12}$$

$$= -\mu\frac{1}{n}\sum_{i}(y_j\frac{a^{'}(x_j^L)}{a(x_k^L)} - \frac{(1-y_j)a^{'}(x_j^L)}{1-a(x_k^L)}) \tag{2.1.13}$$

$$= -\mu\frac{1}{n}\sum_{i}(\frac{y_j - a(x_k^L)}{a(x_k^L)(1-a(x_k^L))})a^{'}(x_j^L) \tag{2.1.14}$$

$$= -\mu\frac{1}{n}\sum_{i}(o^L - y_j) \tag{2.1.15}$$

$$= -\mu\frac{1}{n}\delta_j^L. \tag{2.1.16}$$

The change in weights which connects a fixed neuron i with the neuron j is derived as follows

$$\eta\frac{\partial C(w,b)}{\partial w_{jk}^{L-1}} = \eta\frac{\partial C(w,b)}{\partial o^{L}}\frac{\partial o^{L-1}}{\partial f(w,x,b)}\frac{\partial f(x,w,b)}{\partial w_{jk}^{L-1}} \tag{2.1.17}$$

$$= \mu\frac{1}{n}\sum_{i}\sum_{k}(a(x_k^L - y_k)w_{ij}\frac{\partial x_k^{L-1}}{\partial w_{ij}^{L-1}} \tag{2.1.18}$$

$$= \mu\frac{1}{n}\sum_{i}a^{'}\frac{\partial x_k^{L-1}}{\partial w_{ij}^{L-1}}\sum_{k}w_{jk}^L o_k^L \tag{2.1.19}$$

$$= \mu\frac{1}{n}\sum_{i}[a(x_j^{L-1})(1-a(x_j^{L-1})]o_j^{L-2}\sum_{k}w_{jk}^L o_k^L \tag{2.1.20}$$

$$= \mu \frac{1}{n} \delta_j^{L-1} o_j^{L-2}. \tag{2.1.21}$$

For a neural network with 3 layers we have

$$\Delta w_{jk} = \mu \frac{1}{n} \delta_j^{L} o_k^{L-1} \tag{2.1.22}$$

$$\Delta w_{ij} = \mu \frac{1}{n} \delta_j^{L-1} o_j^{L-2}. \tag{2.1.23}$$

The updates in bias can be written as follows

$$b_j = b_j - \mu \delta_j. \tag{2.1.24}$$

The updates in weights can be written as follows

$$w_{jk} = w_{jk} - \Delta w_{jk}. \tag{2.1.25}$$

$$w_{ij} = w_{ij} - \Delta w_{ij}. \tag{2.1.26}$$

We can also apply similar steps on a neural network with many hidden layers thus updates in weights can be written as follows

$$\Delta w_{jk} = \mu \delta_j^{L-h} o_j^{L-h-1}, \tag{2.1.27}$$

where h is the number of hidden layers. The general way to write an update for bias and weight that connect nodes j and i together at layer L is written as follows

$$w_{ij} = w_{ij}^{L} - \frac{\partial C(w,b)}{\partial w_{ij}}. \tag{2.1.28}$$

$$b_i^L = b_j^L - \mu \sum_j \delta_j^L. \tag{2.1.29}$$

BP is easy to use, easy to implement and it is applicable to a wide range of problems but it learn slowly and requires expensive resources (AIMS).

2.1.7 Activation Function. An activation function is a function $f : R \to R$ that is differentiable almost everytime and everywhere. In order for neural networks to work effectively, they depend on a non-linearity, to convert the values between each layer. Each neural in a neural network has an activation function which specifies the output of a neuron to give a given input. The three most common activation functions are the sigmoid, Tanh and Rectifier Linear Unit (ReLu) functions. Activation function resides within neurons i.e hidden and output layer neuron posses activation function but input layer neurons do not. Activation functions transform the input data received in order to keep values within a manageable range Wikipedia (a).

Inside the neuron:

- Add weighted sum of input values.
- Apply the activation function on the weighted sum of input values and allow the transformation to take place.
- The output values are found on the output layer.

The sigmoid function produces values which range from 0 and 1. It can be defined by the formula:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.1.30}$$

and its derivative is given by

$$f^{'}(x) = f(x)(1 - f(x)) \tag{2.1.31}$$

The Tahn function produces the values which range from -1 to 1 where zero is the centre of all values.It can be defined by the formula:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.1.32}$$

and its derivative is given by

$$f^{'}(x) = 1 - (f(x))^2 \tag{2.1.33}$$

The Rectifier Linear Unit (ReLu) takes the form :

$$f(x) = max(o, x) \tag{2.1.34}$$

All positive values are transformed to x and all negative values are transformed to zero (Wikipedia (a)).Its derivative is given as

$$f^{'}(x) = 1_{x>0} \tag{2.1.35}$$

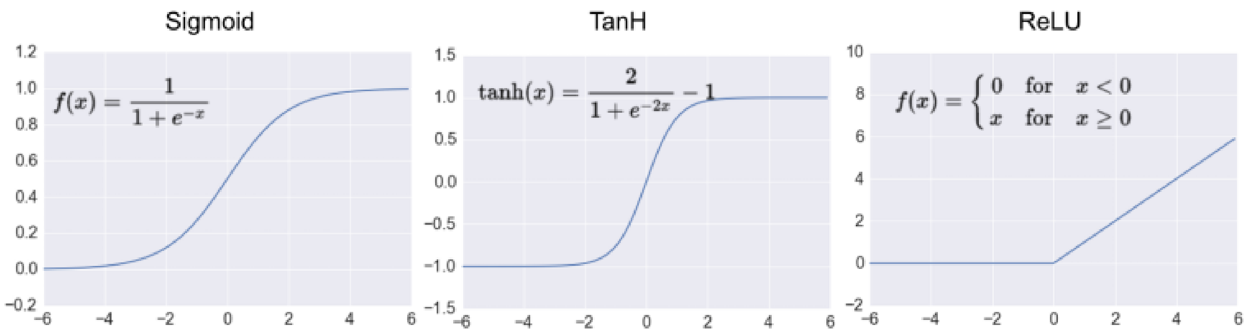


Figure 2.6: Activation functions: Sigmoid, Relu and Tanh functions.

Loss Functions are methods of evaluating how well our machine learning algorithms models our datasets. Loss function will only be high if our predictions are totally off. If our predictions are right, the loss function will be low. There are many different types of Loss functions namely: Mean Square Error, Cross Entropy, Likelihood Loss, etc. Mean Square Error and Cross Entropy are commonly used when building machine learning models. Mathematically, Mean Square Error for a given input x and output data z is given by

$$J(x, z) = \| x - z \|^2 . \tag{2.1.36}$$

The idea is to measure how close the reconstructed input z is to the original input x. The cross entropy loss function is a reconstruction function which compute the number of bits of information in the reconstruction compared to the original. Mathematically, the cross entropy function is given by

$$J(x, z) = -\sum_{k}^{d}[x_k log z_k + (1 - x_k) log(1 - z_k)]. \tag{2.1.37}$$

Optimization Algorithms. Learning in backpropagation is performed by computing a gradient of a set of data and updating the network weights in the opposite direction to the gradients to until the local minima is found. This process is implemented by using optimization algorithms.

Optimization algorithms can be divided into two main groups namely: constant learning rate algorithms and adaptive learning rate algorithms. Mathematically, the equation for updating weights can be written as follows

$$w_{i+1} = w_i - \mu(\triangle J(W)) \tag{2.1.38}$$

Here μ is the learning rate, which is a hyper-parameter, that needs to be adjusted. Selecting a learning rate which is small will take too much time towards finding optimal parameter values which minimize loss. When the learning rate is too large, there will be fluctuations around the minimum loss. Gradient descent falls under constant learning algorithms. One of the disadvantages of Gradient descent is that their hyper parameters have to be defined in advance. On the other hand, Adagrad, Adadelta, RMSprop and Adam falls under adaptive learning rate algorithms.

When working on the sparse data-set, adadelta is can be used to make big updates for parameter which do not appear frequently and small updates for parameters which appears frequently. Adam (Adaptive Moment Estimation), on the other hand computes different learning rate. It is fast and it outperforms other techniques in practice (Opt).

2.2 Dimension Reduction

The number of dimensions in a given dataset is related to the number of distinct number of features used to describe a single element of data i.e. input variables. Since dealing with large number of features has many computational draw backs, which collectively are denoted by the *curse of dimensionality*, a low-dimension representation with minimal loss of the information is sought after. The technique to obtain a low-dimension representation from a large-dimensional data is called dimension reduction.

Dimension reduction reduces the time and storage space required, improves the performance of the machine learning model by removing multi-collinearity and makes it easier to visualize the data when reduced to very low dimensions such as 2D or 3D (Wikipedia (b)). Moreover, as we will show later, dimensional reduction is required for clustering algorithms such as the K-nearest neighbor algorithms (k-means) can be applied.

There are many algorithms used for doing dimension reduction. Some examples are: Principal Component Analysis (PCA), Metric Multidimensional Scaling (MDS), Isomap, and Autoencoders.

In the next sections we will review PCAs and Autoencoders in more detail.

2.3 Principal Component Analysis(PCA)

Principal Component Analysis is a technique that uses advanced underlying mathematical principles to transforms a number of possibly correlated variables into principal components (PCs) i.e a small number of variables. Its origins lie in multivariate data analysis, however, it has a wide range of other applications such as facial recognition, de-noising signals, blind source separation, and data compression. The main aim of principal component analysis is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while keeping as much as possible the variation present in the data set. In general terms, Principal Component Analysis utilizes a vector space transform to minimize the dimensionality of our given large data sets. Mathematically, one can interpret the original data set which may have involved many variables in just few variables. After minimizing the dimensionality of the data set it is important to spot trends, patterns and outliers of the data.

Let a data set be represented in terms of an $n \times m$ matrix, X, where the number of columns are features (observations) and the m rows are variables. We want to transform this matrix linearly into another $m \times n$ matrix, Y, so that matrix, P is of dimension $m \times m$,

$$Y = PX \tag{2.3.1}$$

Considering the rows of P to be the vectors $p_1, p_2, p_3, ..., p_{n-1}, p_n$ and the columns of X to be the column vectors $x_1, x_2, x_3, ..., x_{n-1}, x_n$ the above equation can be written as

$$Y = PX \tag{2.3.2}$$

$$= \begin{bmatrix} p_1x_1 & p_1x_2 & p_1x_3 & \cdots & p_1x_n \\ p_2x_1 & p_2x_2 & p_2x_3 & \cdots & p_3x_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_mx_1 & p_mx_2 & p_mx_3 & \cdots & p_mx_n \end{bmatrix} \tag{2.3.3}$$

$$= (Px_1, Px_2, Px_3, ..., Px_N). \tag{2.3.4}$$

Where p_ix_j is the standard Euclidean dot product and $p_ix_j \in R^m$. The rows of P, $(p_1, p_2, p_3, ..., p_m)$ are the new basis for representing the columns of X. Principal component directions will later be rows of P. Principal components analysis defines the independence between principal components in the new basis by looking at the variance of the data in the original basis. It seeks to reduce the co-relation of the original data by finding the directions in which the variance is maximized and then the new basis is defined by using these directions.

The variance of a random variable z with mean μ is given by

$$\sigma_z^2 = E[(z-u)^2]. \tag{2.3.5}$$

For example, if we have a vector of n measurements $\vec{v} = (\vec{v_1}, \vec{v_2}, \vec{v_3}, ..., \vec{v_n})$, with the mean μ_v and subtract this mean from these measurements, we obtain a set of measurements $v = (v_1, v_2, v_n, ..., v_n)$ that has zero mean. The variance is given by

$$\sigma_v^2 = \frac{1}{n} v v^T. \tag{2.3.6}$$

where v^T is the transpose of v. Co-variance is a measure of how much two variables change together . Suppose we have a second vector of n measurements, $m = (m_1, m_2, m_3, ...m_n)$ with zero mean. We can obtain the variance of v and m. When two variables are identical, the variance becomes a special case of co-variance.The co-variance of the measurements is a given by the relation:

$$\sigma_v^2 = \frac{1}{1-n} vm^T. \tag{2.3.7}$$

Knowing that m is the number of variables and n is the number of features, we can now generalize the above idea of co-variance and variance and consider our mxn data matrix, X. The matrix, X in terms of m row vectors, each of length n is given by the relation :

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & 1x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \dots & x_{m,n} \end{bmatrix}. \tag{2.3.8}$$

$$= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \in R^{mxn}, x_i^T \in R^n \tag{2.3.9}$$

Where x_i is a vector of the n sample (feature) for the i^{th} variable. The co-variance of X is given as

$$C_X = \frac{1}{1-n} \begin{bmatrix} x_1x_1^T & x_1x_2^T & x_1x_3^T & \dots & x_1x_m^T \\ x_2x_1^T & x_2x_2^T & x_2x_3^T & \dots & x_2x_m^T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_mx_1^T & x_mx_2^T & x_mx_3^T & \dots & x_mx_m^T \end{bmatrix}, \in R^{mxm}. \tag{2.3.10}$$

$$= \frac{1}{1-n} XX^T \tag{2.3.11}$$

The matrix X consist of the diagonal entries which are referred to as the variances and the off-diagonal entries are the co-variances thus the matrix is known as the co-variance matrix. The original problem is to transform the original matrix using the relation $Y = PX$ for matrix P. We have to determine some features that we would like the transformed matrix Y to display and relate this to the feature of the corresponding co-variance matrix C_Y. Large variances are taken into account because they correspond to interesting dynamics in the system. Maximizing the signal measure by variance (i.e make the diagonal entries small) and minimizing the co-variances between variable (i.e reduce the off-diagonal entries) are two requirements for constructing the co-variance matrix C_Y. The main objective is to come up with a diagonal matrix C_Y since zero is the possible minimum co-variance. The co-variance matrix, C_Y in terms of X and P is given by the relation:

$$\begin{aligned} C_Y &= \frac{1}{1-n} YY^T && (2.3.12) \\ &= \frac{1}{1-n}(PX)(PX)^T && (2.3.13) \\ &= \frac{1}{1-n}(PX)(X^TP^T) && (2.3.14) \\ &= \frac{1}{1-n}P(XX^T)P^T && (2.3.15) \\ &= \frac{1}{1-n}PSP^T && (2.3.16) \\ . & && (2.3.17) \end{aligned}$$

Where $S = XX^T = (XX^T)^T = (X^T)^T(X)^T$ is an nxn symmetric matrix. In theory a square symmetric matrix is orthogonally diagonalisable. S can be given by

$$S = EDE^T. \tag{2.3.18}$$

where E is an mxm orthogonal matrix and D is a diagonal matrix. D has the eigenvalues of S as its diagonal entries and E has columns which are orthonormal eigenvectors of S. The number of orthonormal eigenvectors that S has is its rank r. The rows of P are the eigenvectors of S selected to make sure that $E^T = P$ and $E = P^T$. Substituting $E^T = P$ into our derived expression for the co-variance matrix C_Y we get

$$\begin{aligned} C_Y =& \frac{1}{1-n}PSP^T && (2.3.19) \\ &= \frac{1}{1-n}E^T(EDE^T)E && (2.3.20) \\ . & && (2.3.21) \end{aligned}$$

We have $E^TE = I$, where I is the mxm identity matrix because E is an mxm orthogonal matrix. The co-variance matrix C_Y can be written as

$$C_Y = \frac{1}{1-n}D \tag{2.3.22}$$

The above equation 2.3.22 allows us to conclude that the variances provide information about the relative importance of each Principal Components (PC). The first PC correspond to the largest variance, the second PC correspond to the second largest variance and so on. After obtaining the eigenvalues and the eigenvectors, eigenvalues are sorted in descending order and placed in this order on the diagonal of D. Eigenvectors are placed in the same order to form the columns of orthonormal matrix E (i.e the eigenvector that corresponds to the first column is placed first, the eigenvector that correspond to the second largest eigenvalue in the second column and so on). Our objective of diagonalising the co-variance matrix of the transformed data was achieved. PCs are eigenvectors of the co-variance matrix(Richardson (2009)).

2.3.1 Autoencoders. A neural network that is trained to try to copy its input to its output is called a Autoencoder (AE). A simple Autoencoder network consists of an encoder and decoder function. The number of hidden units (z) should be less then the dimensions (d) of the input data set. Two main interesting practical applications of Autoencoders are data denoising and dimensionality reduction for data visualization. Autoencoders can learn data projections that are more interesting than PCA or other basic techniques, with appropriate dimensionality and sparsity constraints. Autoencoders are also trained to make the new representation have various nice properties and they are also trained to preserve as much information as possible when an input is run through the encoder and then the decoder. Different types of Autoencoders are meant to achieve different types of properties.

Autoencoder model learn the most important attributes of the input data set and find a way to reconstruct the original input from an encoded state. Autoencoders are used for dimensionality reductions and feature learning. There are four types of Autoencoders namely: Denoising Autoencoder, Sparse Autoencoder, Variational autoencoder (VAE) and Contractive autoencoder (CAE). The assumption behind Autoencoders is that the transformation from input to hidden layers to output will help us learn important properties of the data-set (Towards Data Science).

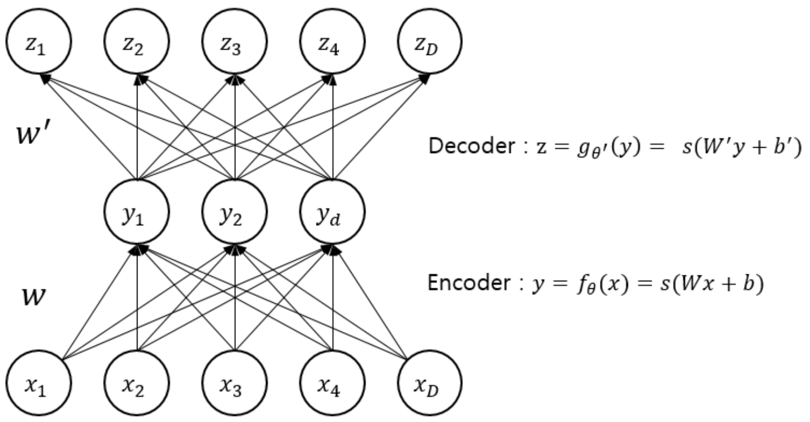

z_1
z_2
z_3
z_4
z_D
w'
Decoder : $z = g_{\theta'}(y) = s(W'y + b')$
y_1
y_2
y_d
w
Encoder : $y = f_\theta(x) = s(Wx + b)$
x_1
x_2
x_3
x_4
x_D

Figure 2.7: A simple Auto-encoder network where $x_i = (x_1, x_2, ..., x_d)$ is our input data,$y_i = (y_1, x_2, ..., y_d)$ is the reduced data and $z_i = (z_1, z_2, ..., z_d)$ is the reconstructed data. The input x is reproduced in the output layer z. The middle layer y " compresses" the input x, effectively reducing the dimensionality of the given data.The Autoencoder is devided into two parts namely: encoder and decoder. The encoder part produces abstract representations of the input data, while the decoder part unfolds the abstract representation so as to reproduce the input data (Sim)

A simple and deep auto-encoder architectures are shown in figure. 2.7 and figure 2.8. An Autoencoder is trained to reduce the dimensionality of our input data . $x \in$R^n is the input vector which is found in the first layer and n is the number of dimensions. During training a new vector $y \in R^m$ is computed where $m < n$. A new vector z in the last layer is produced by by mapping the internal representation back to its original dimension (n). The weight matrices w and $w^{'}$ connecting nodes from one layer to the other are called weight parameters of the Auto-encoder. Weight are kept constant and optimized during the training phase. Here we assume that weight matrices contain both weights and bias. To minimize the error that calculate the distance d between the input x and output z , we adjust the weights when training the network. The general equation of the transformation from layer to layer can be written as follows

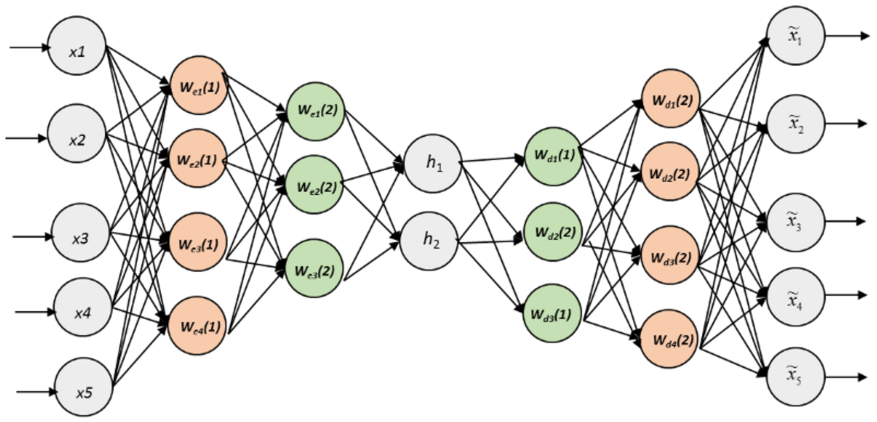


Figure 2.8: Deep Auto-encoder architecture consisting of hidden layers, where hidden layers provide abstract representations at an increasing rate of the input signal. Layer 4 stores there most abstract representations of the input signals (Dee)

$$y_i = f_i(x) = \sigma(w_i^t x), \tag{2.3.23}$$

Where w_i is the weight vector which contain the bias term, x is the input vector, y_i is the intermediate node and σ is the activation function that vary in nature. In this essay both ReLU and sigmoid functions are used. The new obtained vector y can be decoded into a new vector z that reconstruct x. Mathematically

$$y_j = g_j(x) = \sigma(w_j^t x). \tag{2.3.24}$$

We can extend a simple Autoencoder to a deep Autoencoder as illustrated in figure 2.8. The only difference between the two is the number of hidden layers. Mathematically, Autoencoders with a large number of hidden layers are able to learn complex underlying patterns in the data, thus providing a summary representations of the input data Sasdelli et al. (2016).

2.4 Data clustering

Clustering is one of machine learning techniques that involves grouping of data points. If we have a set of data points, we can use clustering algorithms to classify each data points into a specific group. Similarities of properties of the data points pushes them to be in the same group Steinbach et al. (2000). Clustering is a method of unsupervised learning which is used in many fields. There are many clustering algorithms namely: K-means, Agglomerative Hierarchical Clustering, Mean-Shift Clustering, etc. In this essay we will focus only on K- means algorithm to classify sub-classes of SNeIa. In the next section we will review K-means algorithm.

2.4.1 K-means. K-means is one of the unsupervised machine learning algorithms which produces easy and understandable solutions by solving clustering problems. K-means has been used in many applications such as image processing, web page clustering, recommendation system, medical treatment, text classification, etc. When performing this algorithm we begin by choosing k vectors of the same dimensions of the data randomly. These randomly selected k vectors will act as centres for potential clusters. The distance between centre candidate and all vectors in the data set is calculated. The next step is to assign each data point to the cluster represented by its closest centre candidate. After defining the first set of clusters, the centre candidates are updated to the centroids which are defined by all the members of each particular cluster. The process can be repeated until number of centroids are unchanged. K-means is fast, robust easy to understand, relatively efficient and it gives best results when the data sets are separated well from from each other. K- means requires a random selection of k-value c (centres) which is highly difficult, unable to work well with clusters of different size and different density, fails to work for categorical data (i.e K-means work only when the mean of the data is defined), unable to handle noisy data and outliers and it fails for non-linear data set Manikandan et al. as shown in Figure 4.1

K-means algorithm : Assume that we have inputs $x_i = (x_1, x_2, ..., x_n)$ and value k

- Select k points randomly as clusters called centroids.
- After calculating the distances, assign the data point to the cluster center whose distance from the cluster center is minimum of all the selected cluster centers.
- Compute again the new cluster center using the equation below:

$$v_i = \frac{1}{c_i} \sum_{j=1}^{c_i} x_j \tag{2.4.1}$$

  where c_i is the number of cluster center.
- Repeat step 3 to obtained new cluster centers.
- stop if there is no data point which was reassigned , otherwise repeat from step 3. Figure 2.9 illustrate all steps of k-means. (Rajurkar et al. (2015)).

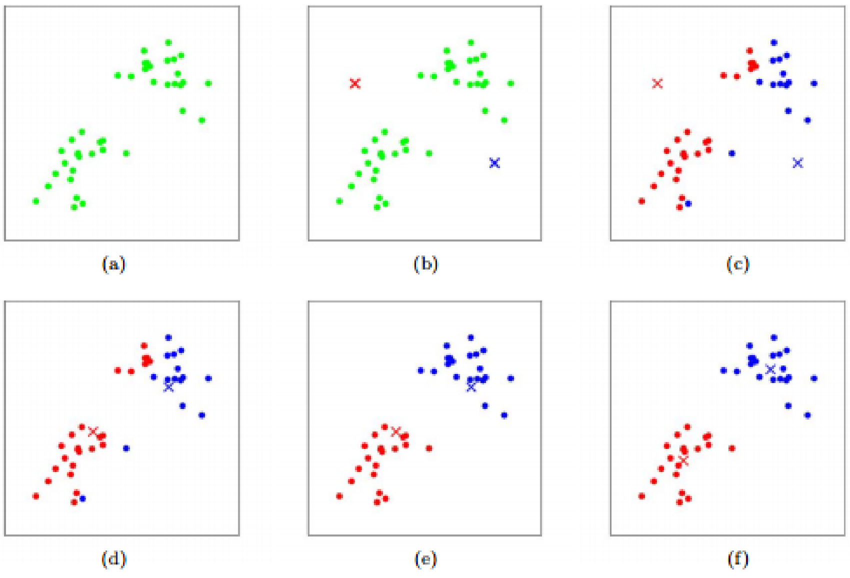


Figure 2.9: K-means algorithm. Data points are shown as dots, and cluster centers are shown as crosses. (a) show the Original data-set. (b) Shows randomly selected initial cluster centers(Martus). (c-f) shows step 2,3,4,5 described above.

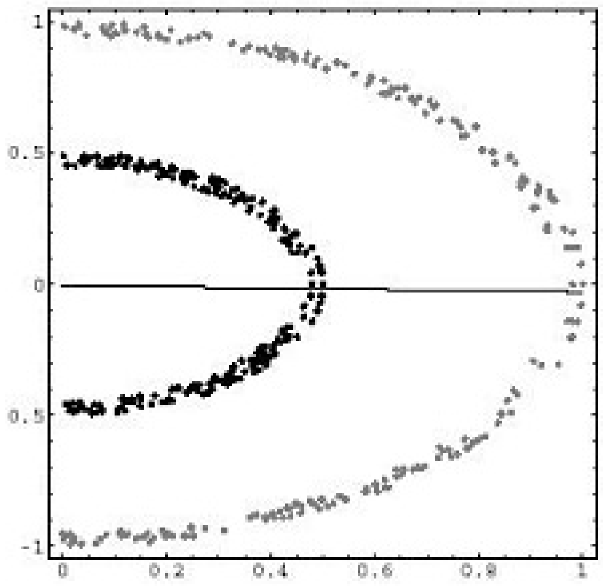


Figure 2.10: Shows k-means algorithm failing to work on non-linear data set (Data Clustering Algorithms).

2.4.2 Similarity measures and Distance measures. k-Means can be used for classification but before that requires similarity or distance measure (Cosine Similarity). Similarities and distances are measured between the data point during training. Distance measures are used to find the distance between two set of points while similarity measures are used to find how similar or different set of objects are to each other. There are different distance measures such as Euclidean distance, Minkowski distance, Hamming distance, Manhattan distance, Chebychev distance, Mahalanobis distance. In this work we will look at Euclidean distance. Similarity measures such as Jaccard similarity and Cosine similarity will also be considered.

Let x,y be points in a set X. The distance between the two is denoted as $D(x, y)$.

Euclidean distance is one of the distance measure we considered for SNeIa classification. Euclidean distance compute the distance between two points in an euclidean space(Flach (2012)).

$$D_e = (\sum_{i=1}^{n}(x_i - y_i))^{\frac{1}{2}} = \| x_i - y_i \| = ((x-y)^T \sqrt{(x-y)}). \tag{2.4.2}$$

Jaccard similarity works well for binary data (i.e data file which consist of only zeroes and ones), sets or multi-sets and real values. Let X and Y be finite sets. It is denoted as $J(X, Y)$ and it can be given as

$$J(X,Y) = \frac{\mid X \cap Y}{\mid X \cup Y \mid} = \frac{\mid X \cap Y}{\mid + \mid Y \mid - \mid X \cap Y \mid}, \tag{2.4.3}$$

where $X \cap Y$ represent the intersection and $X \cup Y$ represent the union of the two sets Leskovec et al. (2014).

Cosine similarity measures the similarity between two non-zero vectors by calculating the angle between them. Data points that are of the same direction have high similarities than data points which are perpendicular to each other. Mathematically, cosine similarity can be given by

$$cos(\Theta) = \frac{\vec{x}.\vec{y}}{\| x \|_2 \| y \|_2} = \frac{\sum_{i=1}^{c} x_i y_i}{\sqrt{\sum_{j=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}, \tag{2.4.4}$$

where x_i is the component of vector x and y_i is the component vector y (Leskovec et al. (2014)).

3. Data preparation and method

In this essay, we use the data compiled and processed in the paper Sasdelli et al. (2016). In the following we describe how the data was prepared by the authors of that paper.

A set of SNeIa spectra was compiled from different well known sources. Before applying machine learning tools to extract information on the spectra, one has to take into account the high impact of random noise originated during the process of observations. Noise coming from detectors, calibration, and photon statistics affect the spectra. The aim is to extract features by filtering the noise without degrading the signal. Savitzky–Golay filter is used to solve this problem. Compared to other smoothing methods, the SG filter, with an appropriate choice of parameters, is more successful in preserving the shape of the peaks and valleys, even for weak spectral features. The data matrix was built by using derivative spectroscopy which remove all the systematics and Savitzky–Golay filter.

The matrix set consists of 3677 derivative spectra (rows) and 296 columns from which 456 spectra correspond to observations of different SNeIa at maximum light. The data matrix consist of all SNeIa spectra of all known and unknown epoch of observation.

In Sasdelli et al. (2016), K means algorithm applied on PCA and Autoencoder reduced features were demostrated to check if unsupervised learninc can pick main spectral features underlying the classification suggested by Wang et al. (2009): normal, high velocity, 91T -like and 91bg-like SNe. The authors in that paper argued the k-means algorithm seems to justify those sub-classes.

Our aim in this chapter is to study further the effect of choosing a shallow vs deep layers for the Autoencoder architecture.

3.1 Strategy

Similar to Sasdelli et al. (2016), our method consist of two steps: dimensionality reduction via Autoencoders, and clustering using K-means. The first step is to reduce the dimensionality of the wavelength grid from 296 bins to between 4-11 bins. We then apply K-means clustering on this low-dimensional representations.

The Autoencoder consist of two parts namely: encoder and decoder. The output vector is obtained by encoding the original data through several layers of non-linear transformations.

For the k-means clustering, first k vectors are randomly randomly selected. Then by distance between centre candidate and all vectors in the data set is calculated. The next step is to assign each data point to the cluster represented by its closest centre candidate.

4. Results

The results below were found by running a 4 dimensional feature space through the k means algorithm. Setting k means algorithms to search for 4 clusters allows us to compare them with those classified by Wang et al. (2009). Colours in figure 4.1 correspond to clusters obtained by K-means. Figure 4.2 shows classification suggested by Wang et al. (2009). Peculier objects were not shown on this graphs in this figures. The two figures do not look alike but they share some similarities. The scatter plots in the feature spaced formed by feature 1 and 1 and feature 3 and 3 look similar to each other in both figures. The blue colour is the one which is dominant, thus we can conclude by saying that group 3 correspond to 1919bg-like SNe. Figure 4.3 and figure 4.4 do not look alike and they do not show any similarities due to some error made in my code.

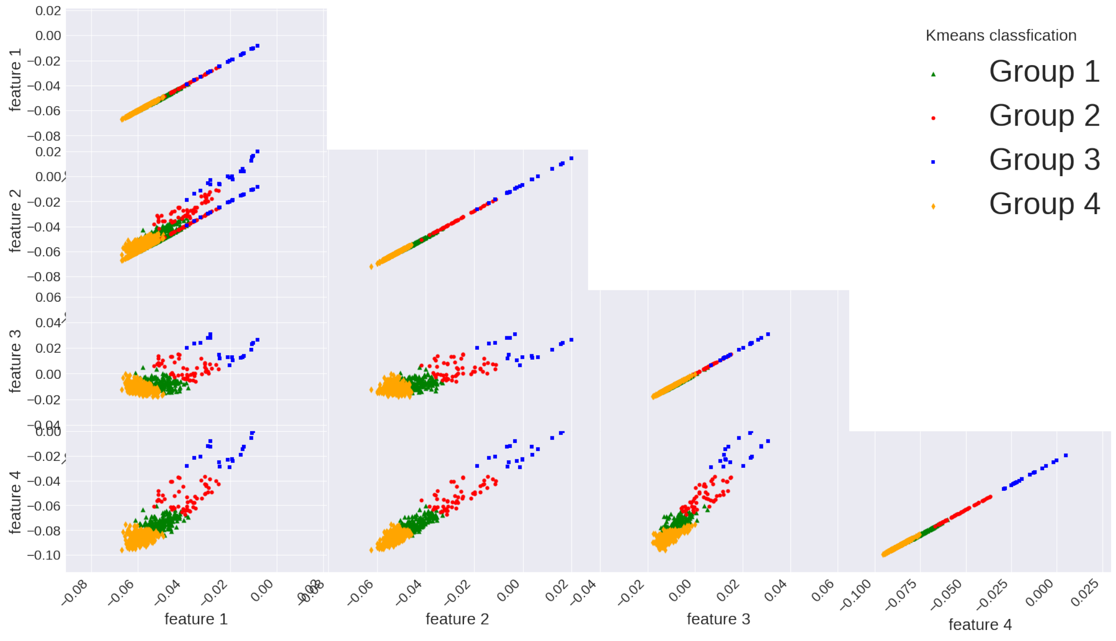


Figure 4.1: Shows a 4 dimensional feature space resulting from Deep Learning (7 layered Autoencoder model). 4 Colours represent the 4 groups found by the K-Means algorithm.

I can use other methods like PCA to reduce the dimensionality of space but resent paper have shown that Autoencoders are more accurate when it comes to abstracting the representations of the original data during reduction. I can improve below by using other algorithms like isomap which is also responsible for low space dimensional reduction.

Figure 4.5 shows the comparison between 7 layered Autoencoder and 10 layered Autoencoder in their capacity to reconstruct the original data. On the horizontal axis we have the number of features and on the vertical axis we have the deviation between the original data and the reconstruction. Both models are competitive but we ended up choosing 7 layered model to classify spectra features of subclasses of SNeIa due to the fact that we can see some similarities between 4.1 and 4.2.

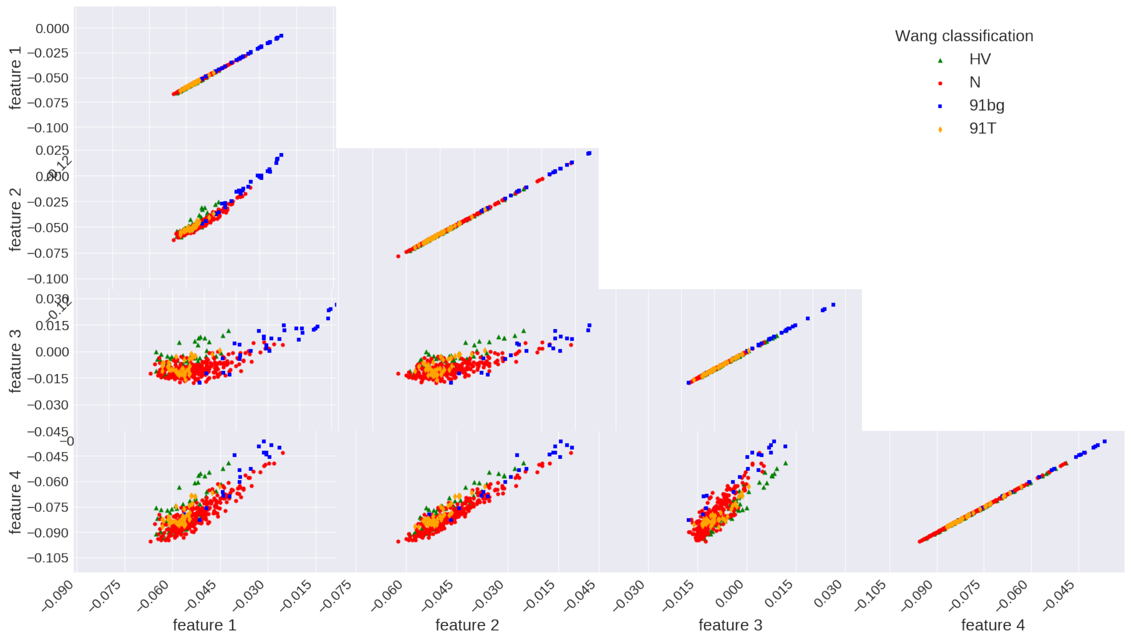


Figure 4.2: Shows classification of sub-types of SNeIa by Wang et al. (2009)

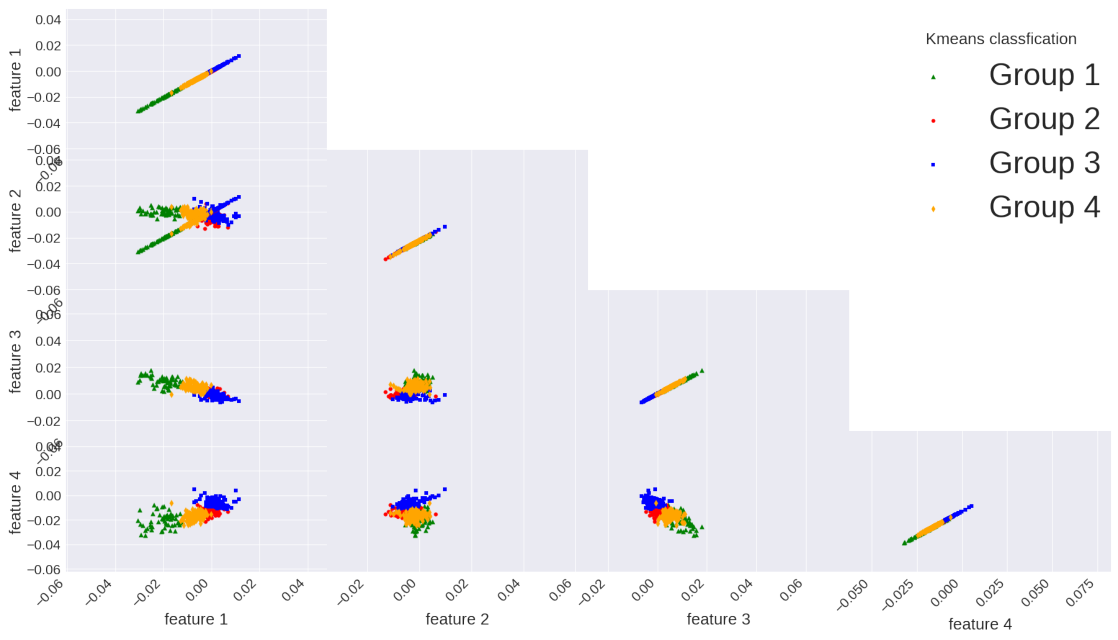


Figure 4.3: Shows a 4 dimensional feature space resulting from DL (10 layered Autoencoder model). 4 Colours represent the 4 groups found by the K-Means algorithm .

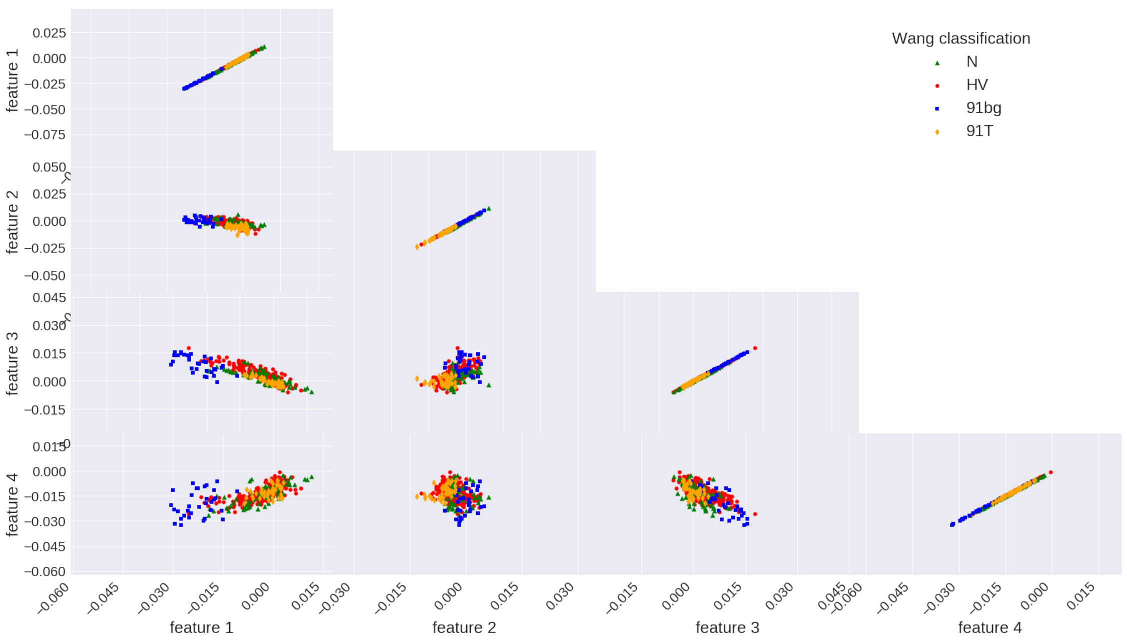


Figure 4.4: Shows classification of sub-types of SNeIa by Wang et al. (2009)

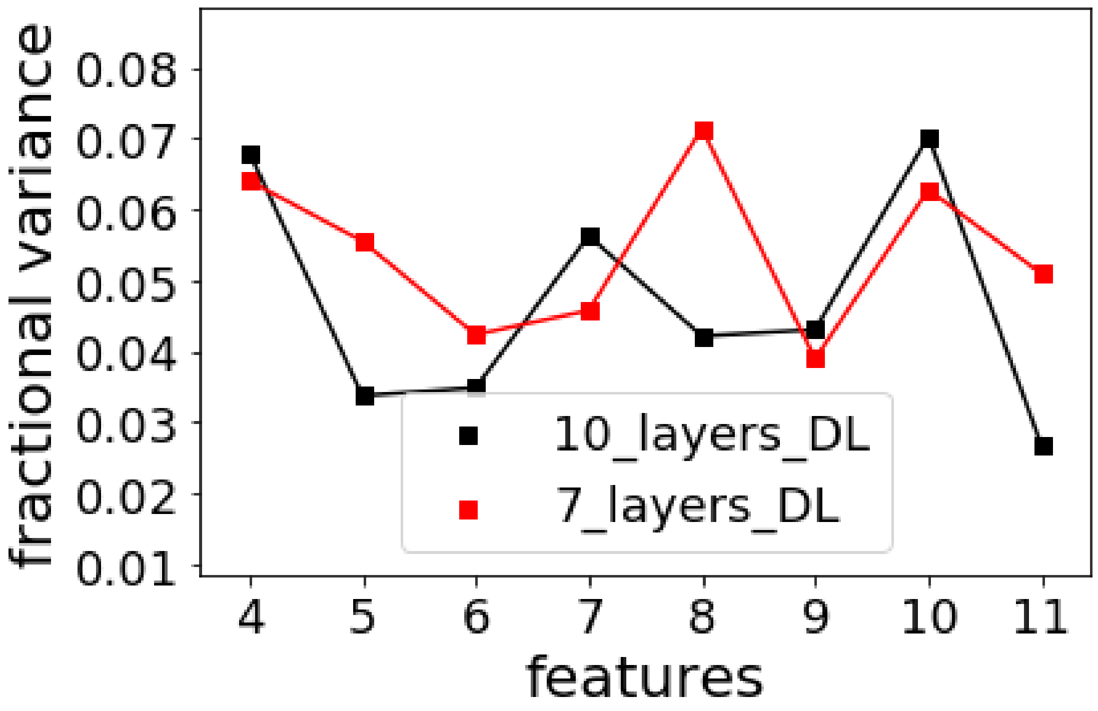


Figure 4.5: Comparison between 7 layered Autoencoder and 10 layered Autoencoder in their capacity to reconstruct the original data. On the horizontal axis, we have the number of features and on the vertical axis, we have the variation between the original data and the reconstructed data.

5. Conclusion and Summary

We combine two Machine Learning techniques to be able to identify subclasses of SNeIa within a set of spectra. Deep Learning for dimension reduction is introduced on the obtained SNIa spectra. We come up with two Deep Learning models and compare them so that better results can be produced. We prove 7 layered Autoencoder effectiveness in spectroscopy data analysis and show that it outperforms 10 layered Autoencoder in the reconstruction of measured spectra and reducing the dimensionality of the feature space from 296 to 4. I introduce unsupervised learning techniques to see if is possible to identify spectroscopic features in subclasses of SNIa spectra at maximum brightness. 91bg-like events confirmed determine subclasses of SNeIa. We used the reduced data as an input for the K-means algorithm.Due to the fact that SNeIa are uniform, I could not find the strong evidences of different types because i only used two machine learning tools. Due to time constraint, I could not investigate the effect of using different loss function, activation function, optimization method and important choices, which I will consider in the future.

Acknowledgements

Firstly I would like thank God for his blessings and for giving me the opportunity to belong to AIMS family. I would secondly like to express gratitude towards my supervisor Dr. Yabebal Fantaye for his guidance and for sharing his scientific knowledge during the elaboration of my essay.

References

AIMS. Structured masters research projects. archive.aims.ac.za, https://docs.google.com/viewer?a=v&pid=sites&srcid=YWltcy5hYy56YXxhcmNoaXZlfGd4OjZjOGE2YWNlMTdhN2M2YjI, Accessed August, 2018.

Bangal, B. C. Automatic generation control of interconnected power systems using artificial neural network techniques. 2009.

carnegiescience. Mark phillips. carnegiescience.edu, https://carnegiescience.edu/scientist/mark-phillips, Accessed September, 2018.

Chakraborty, W. Accelerating expansion of the universe. *arXiv preprint arXiv:1105.1087*, 2011.

Cosine Similarity. Terra incognita. blog.christianperone.com, http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/, Accessed September, 2018.

Cruz, F. Teaching robots with interactive reinforcement learning. 2017.

Data Clustering Algorithms. Mark phillips. google.com, https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm, Accessed September, 2018.

datascience.com. Multi-layer neural networks. towardsdatascience.com, https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f, Accessed September, 2018.

Flach, P. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

Foley, R. J., Kromer, M., Marion, G. H., Pignata, G., Stritzinger, M. D., Taubenberger, S., Challis, P., Filippenko, A. V., Folatelli, G., Hillebrandt, W., et al. The first maximum-light ultraviolet through near-infrared spectrum of a type ia supernova. *The Astrophysical Journal Letters*, 753(1):L5, 2012.

Fürnkranz, J. Machine learning in games: A survey. *Machines that learn to play games*, pages 11–59, 2001.

Google. neural network pictures. www.google.com, https://stackoverflow.com/questions/40537503/deep-neural-networks-precision-for-image-recognition-float-or-double, Accessed August, 2018.

Hillebrandt, W. and Niemeyer, J. C. Type ia supernova explosion models. *Annual Review of Astronomy and Astrophysics*, 38(1):191–230, 2000.

Kaur, P. and Sapra, R. Ontology based classification and clustering of research proposals and external research reviewers. *International Journal of Computers & Technology*, 5(1):49–53, 2013.

Krenker, A., Bešter, J., and Kos, A. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pages 1–18, 2011.

Leskovec, J., Rajaraman, A., and Ullman, J. D. *Mining of massive datasets*. Cambridge university press, 2014.

Light Curves, IITR. Type ia supernova light curves. COSMOS, http://astronomy.swin.edu.au/cosmos/T/Type+Ia+Supernova+Light+Curves, Accessed August, 2018.

Manikandan, K., Visalakshi, R., and Ponnusamy, R. Literature survey of data mining clustering algorithms.

Martus, J. Introduction to machine learning.

Neural Network. Steven miller,2018. github, https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/, Accessed August, 2018.

Nordin, J. *Spectral Properties of Type Ia Supernovae and Implications for Cosmology*. PhD thesis, Department of Physics, Stockholm University, 2011.

Rajurkar, P. P., Bhor, A. G., Rahane, K. K., Pathak, N. S., and Chaudhari, A. N. Efficient information retrieval through comparison of dimensionality reduction techniques with clustering approach. *International Journal of Computer Applications*, 129(4), 2015.

Researchgate. Illustration of the architecture of the back propagation neural network. Researchgate, https://www.researchgate.net/figure/Illustration-of-the-architecture-of-the-back-propagation-neural-network-BPNN-The-BPNN_fig3_51200358, Accessed August, 2018.

Richardson, M. Principal component analysis. *URL: http://people. maths. ox. ac. uk/richardsonm/SignalProcPCA. pdf (last access: 3.5. 2013). Aleš Hladnik Dr., Ass. Prof., Chair of Information and Graphic Arts Technology, Faculty of Natural Sciences and Engineering, University of Ljubljana, Slovenia ales. hladnik@ ntf. uni-lj. si*, 6:16, 2009.

Sasdelli, M., Ishida, E., Vilalta, R., Aguena, M., Busti, V., Camacho, H., Trindade, A., Gieseke, F., de Souza, R., Fantaye, Y., et al. Exploring the spectroscopic diversity of type ia supernovae with dracula: a machine learning approach. *Monthly Notices of the Royal Astronomical Society*, 461(2): 2044–2059, 2016.

Steinbach, M., Karypis, G., Kumar, V., et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

Thukral, E. S. Artificial neuron learning technique in soft computing: A review. *International Journal of Advanced Research in Computer Science*, 7(6), 2016.

Towards Data Science. Autoencoders. Towards Data Science, https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f, Accessed August, 2018.

Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.-L. A., Elkhatib, Y., Hussain, A., and Al-Fuqaha, A. Unsupervised machine learning for networking: Techniques, applications and research challenges. *arXiv preprint arXiv:1709.06599*, 2017.

Wang, X., Li, W., Filippenko, A. V., Foley, R., Kirshner, R., Modjaz, M., Bloom, J., Brown, P., Carter, D., Friedman, A., et al. The golden standard type ia supernova 2005cf: observations from the ultraviolet to the near-infrared wavebands. *The Astrophysical Journal*, 697(1):380, 2009.

Wikipedia. Activation function. Wikipedia, https://en.wikipedia.org/wiki/Activation_function, Accessed August, 2018a.

Wikipedia. Dimensionality reduction. Wikipedia, https://en.wikipedia.org/wiki/Dimensionality_reduction, Accessed September, 2018b.

Wikipedia contributors. Unsupervised learning — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=862068532. [Online; accessed 2-October-2018].