DEMOCRATIC REPUBLIC OF ALGERIA PEOPLE
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY:mathematics and informatics
DEPARTEMENT of computer science
N°: ……

DOMAIN: Mathematics and Informatics
FIELD: computer science
SUB-FIELD: decision and optimization informatics

A Dissertation in Fulfillment
for the Requirement of the Degree of MASTER

By:MESSEGUEM Chahrazed

SUBJECT:

Data classification using deep learning approach

Defended publicly on 23/06/2018
Board of Examiners:

Mr. GEMOGI Abdalsattar	University of M'sila	Chairman
Ms. BENOUIS Mohamed	University of M'sila	Supervisor
Mr. GERNA Abderrahim	University of M'sila	Examiner

Academic year: 2017/2018

Abstract

The work presented in this report is located in the pattern recognition area, and the specific motivation in this work is to study the manner of training , their architectures, techniques, and methods. Our system is based on Restricted Boltzmann Machines and specifically those who belong to the deep learning. We detail the systems based on Deep belief Networks used for data learning, this approach offers a distinctive architecture that is used in the experimental section to train two databases so that our neural network can recognize the data proposed to the network input.

Keywords : Deep Learning, Neural Network, Classification, Restricted Boltzmann Machines, Deep Belief Network

Résumé

Le travail présenté dans ce rapport se situe dans le domaine de reconnaissance de la forme, et la motivation particulière dans celui-ci est d'étudier les systèmes d'apprentissage de données, leurs architectures, techniques, et méthodes. Notre système s'appuie sur les machines de Boltzmann restreinte et précisément ceux qui font partis d'apprentissage profond. Nous détaillons les systèmes basés sur les réseaux du Boltzmann empiles restreints pour l'extraction des caractéristiques, à savoir, les chiffres et les visages. Cette approche propose une architecture particulière qu'on vient d'utiliser dans la partie expérimentale pour entrainer nos bases de données afin que notre réseau de neurone arrive à reconnaitre les données proposées à l'entrée du réseau.

Mots clés : l'apprentissage profond, Réseaux de neurons, classification, Restricted Boltzmann Machines (RBM), Deep Croyance Networks (DBN).

ملخص

يقع العمل المقدم في هذا التقرير في منطقة التعرف على الأنماط ، وتتمثل الدوافع المحددة في هذا العمل في دراسة طريقة التدريب وهندستها وتقنياتها وأساليبها. يعتمد نظامنا على ماكينات بولتزمان المقيدة وعلى وجه التحديد أولئك الذين ينتمون إلى التعلم العميق. نحن نقوم بتفصيل الأنظمة التي تعتمد على شبكات المعتقد العميق المستخدمة في تعلم البيانات ، حيث يقدم هذا المنهج بنية مميزة يتم استخدامها في القسم التجريبي لتدريب قاعدتي بيانات بحيث تتمكن شبكتنا العصبية من التعرف على البيانات المقترحة لمدخلات الشبكة.

الكلمات المفتاحية: التعلم العميق,الشبكات العصبونية, التصنيف, ماكينات بولتزمان المقيدة, وشبكات المعتقدة العميقة.

Table of Contents

CHAPTER 3

Implementation and experimentation

List of Figures

List of Tables

PREFACE

PREFACE

The use of intelligent systems in all areas with the development of information technology in recent years has become inevitable. The concept of artificial intelligence begins with the appeal of the idea of teaching intelligence, the most precious thing possessed, that separates man from other living things, to machines. It has evolved through new techniques introduced in science circles. Machine learning is a subject in computer science, aimed at studying theories, algorithms, and applications of systems that learn like humans .

Neural networks have a long history in machine learning. Early experiments have shown both, their expressional power, but also the difficulty to train them. For the last ten years, neural networks are celebrating a comeback under the label "deep learning".

Deep learning, which has been mentioned a lot in recent times, has been widely used. Deep learning is the general name of techniques developed at the point where traditional artificial neural networks are inadequate to solve the problem. It's named with a "deep" prefix, indicating the number of hidden layers is too high.

Deep Learning algorithms consist of artificial neural network based and energy-based models [1]. The architecture comes in many layers and variants.

Deep Belief Networks are a specialized architecture of deep learning and are particularly successful in classification. In this regard, DBN has been used and used in many academic studies. A Deep Belief Network (DBN) is a powerful machine learning technique from the field of deep learning. DBNs are trained using large collections of diverse images. From these large collections, DBNs can learn rich feature representations for a wide range of dataset(images). An easy way to leverage the power of DBNs, without investing tile and effort into training, is to use a pre-trained DBN as a feature extractor. These deep models have many layers and parameters that must be learnt. When the learning process is so complicated and a huge number of parameters are needed, artificial neural networks are rarely used. The problem of this number of layers is that training is time consuming and training becomes trapped at local minima. Therefore we can't achieve acceptable results. One important tool for dealing with this problem is to use DBNs (Deep Belief

Network) that can create neural networks including many hidden layers (Liu et al., 2011).Deep Belief Networks can be used in classification and feature learning. Data representation is very important in machine learning. Therefore, much work has been done for feature preprocessing, feature extraction and feature learning. In feature learning, we can create a feature extraction system and then use the extracted features in classification and other applications. Using unlabeled data in high level feature extraction (Lee et al., 2008) and also increasing discrimination between extracted features are the benefits of DBN for feature learning (Hinton and Salakhutdinov, 2006). However, to briefly explain, the training data set helps find the optimal parameters of a model, the validation data set helps avoid overfitting of the model, and the test data set helps determine the accuracy of the model.

Our dissertation is organized in three chapters

The first chapter, offers an introduction to Machine Learning and its types, understanding their strengths and weaknesses and the most common Machine Learning algorithms.

In the second chapter, we are focused on the field of Deep Learning and a particular family of algorithms that are essentially solved technologies(Structure and Methodology), especially the Deep Belief Networks and their component element the Restricted Boltzmann Machine.

In the third chapter we will present our experimental results and we will interpret the results of the tests performed on two data sets MNIST and ORL for a DBN.

Finally, we end this manuscript with a conclusion and some perspectives .

MACHINE LEARNING

CHAPTER 1

1. Introduction

Machine Learning is a very interesting and challenging field of study, especially since the recent wave of interest in deep learning. The Continuous improvement of Machine Learning techniques combined with the ever increasing amount of data that are stored suggests endless new applications. Traditional programming are not able to treat and train all this data. Let's first make a Comparison between the traditional programming approach and machine learning approach. As we can see in **Figure 1.1** .In traditional programming approach, we began with data as an input and a program and finally obtain the result as an output. In machine learning approach there are some difference, we began with data and result (input, output), we know we obtained on this dataset before ,then we train a program as an output . The obtained program will be used in input of a traditional programming approach after that. Machine learning algorithms have been applied in order to solve big data classification problems which involves the classification of different types of data and the extraction of useful information from the massive and complex data sets with minimal effort.

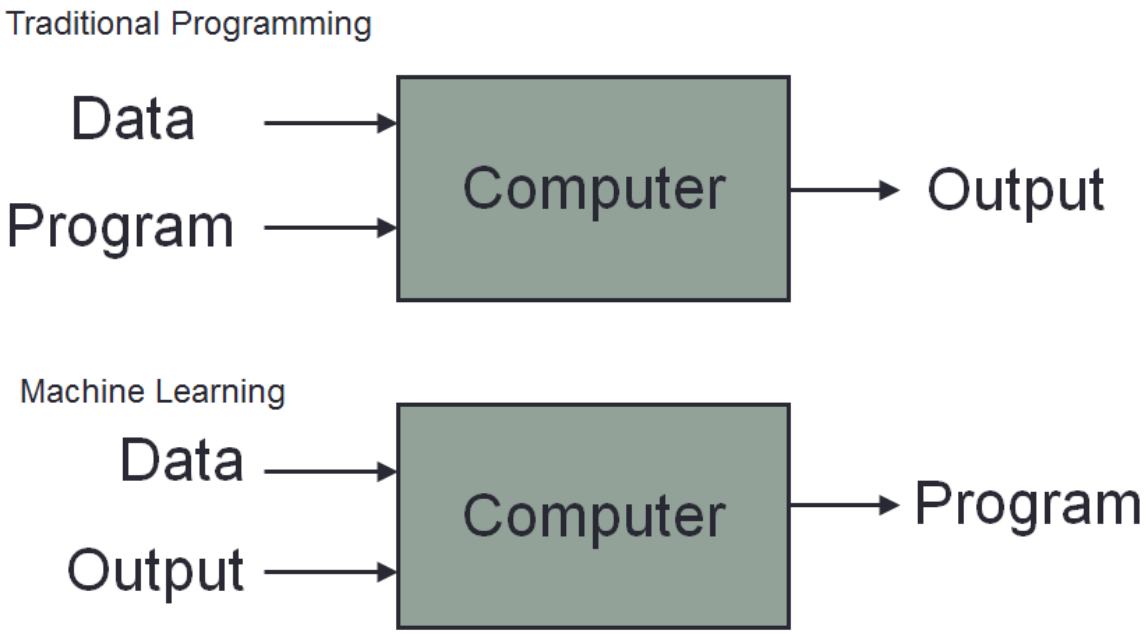


Figure 1.1 Machine learning vs traditional program

2. Machine Learning

Machine learning is becoming important in every discipline. It's also a branch of artificial intelligence. Machine learning can mean using machines(computers and software) to gain meaning from data. It can also mean giving machines the ability to learn from their environment.

2.1 Definition of machine learning

Machine learning is about the exploration and development of mathematical models and algorithms to learn from data such as documents, audio, images, movies, e-commerce, electric power, medicine, and biology. Its paradigm focuses on classifying objectives and consists of modeling an optimal mapping between the data domain and the knowledge set and developing the learning algorithms, which requires a training (labeled) data set, a validation data set, and a test data set. [2]

3. Element of machine learning

3.1 Data: All learning methods are data driven. Sets of data are used to train the system. These sets may be collected by humans and used for training. The sets may be very large. Control systems may collect data from sensors as the systems operate and use that to identify parameters or train the system, so the data we feed our machine learning systems must be mathematical objects, such as vectors, matrices or graphes.

3.2 Models : Models are often used in learning systems. A model provides a mathematical framework for learning. A model is human derived and based on human observations and experiences

3.3 Training : A system that maps an input to an output needs training to do this in a useful way. Just as people need to be trained to perform tasks, machine learning systems need to be trained. Training is accomplished by giving the system an input and the corresponding output and modifying the structure (models or data) in the learning machine so that mapping is learned. In some ways this is like curve fitting or regression. If we have enough training pairs, then the system should be able to produce correct outputs when new inputs are introduced.[3]

4. Types of learning

Depending on the type of available data, machine learning can be categorized into supervised learning, unsupervised learning, and reinforcement learning.

Supervised and unsupervised are mostly used by a lot machine learning engineers and data geeks, Reinforcement learning is really powerful and complex to apply for problems.

4.1. Supervised learning

We give to the computer some pairs of inputs and outputs(Training Data).We first train the model with the lots of training data, then in the future when new inputs are presented you have an intelligent output . And there are two special types(algorithms) as we can see in Figure 1.2 that are interesting to know :

4.1.1 Logistic regression(Classification)

Each learning is associated with a qualitative target value, which corresponds to a class. There can be two classes (binary classification) or more (multiclass classification).

4.1.2 Linear regression(Regression)

Each learning example is associated with a quantative target value. The goal of the model is to estimate the correct output, given a feature vector.[11]

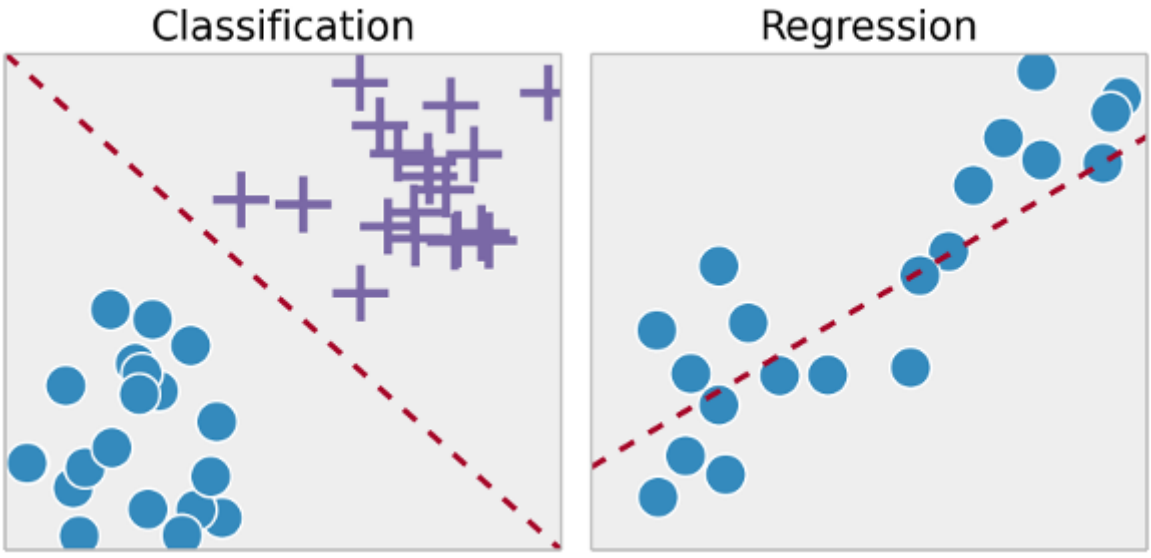


Figure 1.2 classification and regression

4.2 Unsupervised learning

The training data does not include Targets here so we don't tell the system where to go .The training data is not structured (contains noisy data, unknown data , and …etc) and the system has to understand itself from the data we give, automatically extract meaningful for the data . There are also different types for unsupervised learning like Clustering and anomaly detection (clustering is pretty famous)

4.2.1 Clustering

This is a type of problem where we group similar things together. Bit similar to multi class classification but here we don't provide the labels, the system understands from data itself and cluster the data. [4]

4.3 Reinforcement learning

Where the algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback that guides the learning algorithm.[5]

5 Machine learning methods

In machine learning classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class or it may be multi-class too. Here we have the types of classification algorithms in Machine Learning:

5.1 K- Nearst Neighbors (KNN)

k-Nearest-Neighbor classification, or kNN, is a machine learning algorithm that localizes a group of *k* objects in a training case that has the closest proximity to the test object, and then assigns a label derived from the prevalence of a class in the closest proximity. Three important components are needed for this algorithm: a group of labeled objects; a proximity metric; and the number k of nearest neighbors . A popular proximity metric that is used for kNN classification is "Euclidian Distance", explained by formula :

$$D_E(\mathrm{x}, \mathrm{y}) = \sum_{i=1}^{n} (|x_i, y_i|^2)^{1/2} \qquad (1)$$

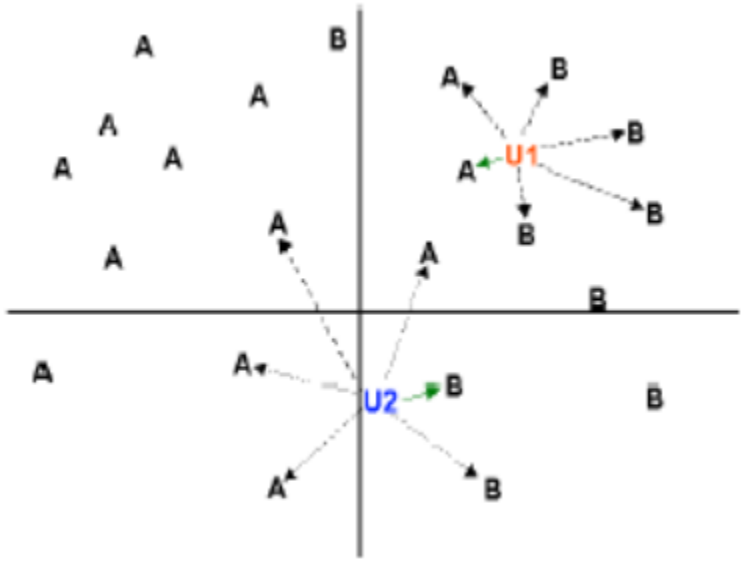


Figure 1.3 Representation of a *k*-Nearest-Neighbor graph.

The training phase of kNN classification does not exist, instead all feature values are stored in memory. Unlike other machine learning algorithms like SVMs or decision trees, kNN classifiers are considered slow learners. Creating the model is computationally inexpensive, whereas the classification phase is computationally expensive because each k-nearest neighbor needs a label. This requires calculating the distance metric of the unlabeled instance to all instances in the labeled set, which can be extremely expensive in particular for large datasets. [6]

5.2 Support Vector Machine (SVM)

5.2.1 Definition

Support Vector Machines (SVMs) have become indispensable tools in the Machine Learning domain due to their generalization properties and regularization capability.

The support vector machine can be divided into linear and nonlinear models. It is called linear support vector machine if the data domain can be divided linearly (e.g., straight line or hyperplane) to separate the classes in the original domain. If the data domain cannot be divided linearly, and if it can be transformed to a space called the feature space where the data domain can be divided linearly to separate the classes, then it is called nonlinear support vector machine. [8]

In SVM we should follow some steps for best classification ,first we have define an optimal hyperplane(maximize margin), then Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications and finally Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space , A visualization of this strategy is illustrated in **Figure 1.4 .**

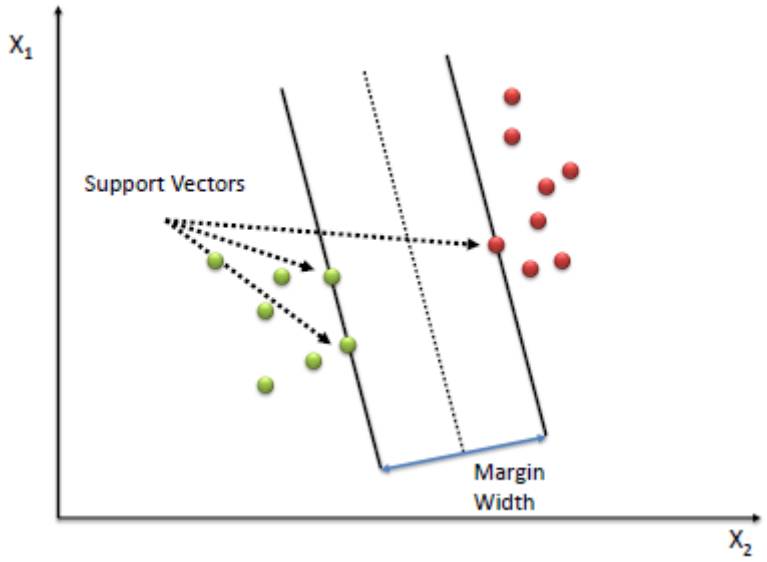


Figure 1.4 Illustration about the hyperplane and the support vectors

.

5.2.2 Types of SVM algorithm

- Linear Hard-Margin SVMs

In the simplest SVM formulation, a training set can be separated by at least one hyperplane, i.e. data are linearly separable and a linear model can be used :

$$Y(\mathbf{x}) = \boldsymbol{w}^{T}\mathbf{x}+b \qquad (2)$$

Since $\boldsymbol{w}^{T}\mathbf{x}+b= 0$ and $c(\boldsymbol{w}^{T}\mathbf{x}+b) = 0$ define the same plane, we have the freedom to choose the normalization of w .Choose normalization such that $\boldsymbol{w}^{T}x_{+}+b=+1$ and $\boldsymbol{w}^{T}x_{-}+b = -1$ for the positive and negative support vectors respectively, as we can see in **Figure1.5.** Then the margin is given by

$$\frac{w}{||w||}(\mathbf{x}_{+}-x_{-}) = \frac{\boldsymbol{w}^{T}(\mathbf{x}_{+}-x_{-})}{||w||} = \frac{2}{||w||} \qquad (3)$$

Learning the SVM can be formulated as an optimization:

Max(w) = $\frac{2}{||w||}$ subject to $\boldsymbol{w^T}x_i$+b $\geq$ 1 if y_i = 1

$\boldsymbol{w^T}x_i$+b $\geq$ -1 if y_i= -1 for i=1…..N

Or equivalently :

Min(w) = $||w||^2$ subject to $\boldsymbol{w^T}x_i$+b $\geq$ 1for i =1…..N (4)

This is a quadratic optimization problem subject to linear constraints and there is a unique minimum.

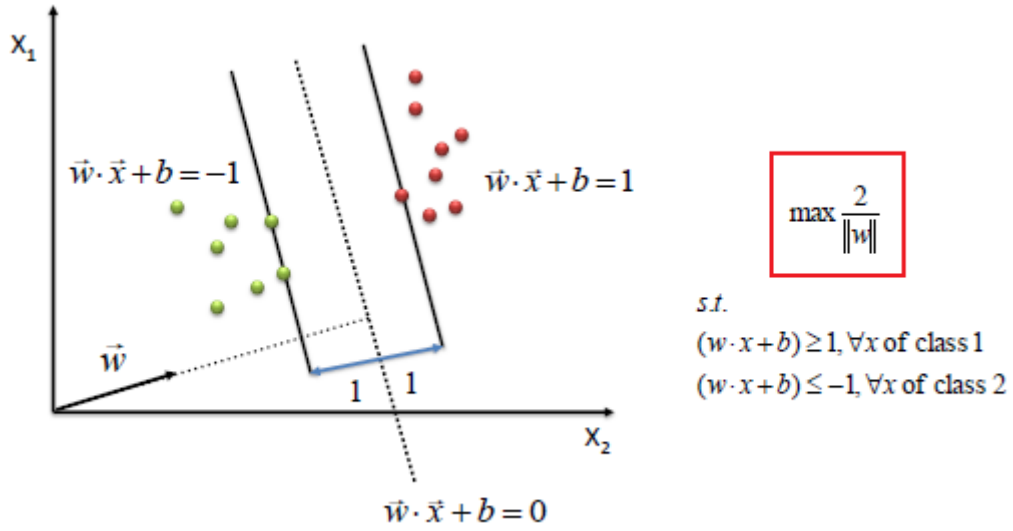


Figure 1.5 linearly separable case

If classes are non separable to an acceptable level or the result in a model doesn't classify correctly and perfect in this case SVM use a new variable ζ is called the Slack variable to finds the hyperplane that maximizes the margin and minimizes the misclassification. The stack variable allow to describes the false positives must be introduced to the optimization problem described in equation(4) .this will become :

Min(w) = $||w||^2$+ ε(ζ) (5) subject to : s(wx'+yi)+ζ ≥ I

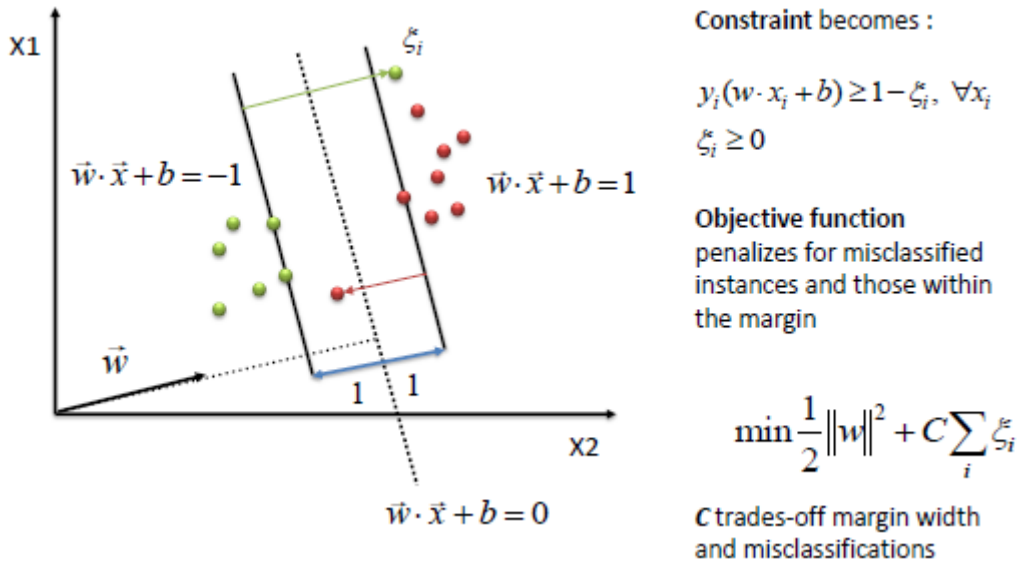


Figure 1.6 Illustration about the Slack variable

- The Nonlinear SVM with Kernels

An extension to nonlinear decision surfaces is necessary since real-life classification problems are hard to be solved by a linear method. While the Soft Margin SVM can give a good performance with non-linearly separable classes, it can be further improved by working in the so-called feature space, The use of kernel functions delivers a powerful method to obtain generalized optimal separating hyperplane in an appropriate feature space , Cover's theorem states that if the transformation is nonlinear and the dimensionality of the feature space is high enough, then input space may be transformed into a new feature space where the patterns are linearly separable with high probability. This nonlinear transformation is performed in an implicit way through the so-called kernel functions. [9]

Kernel function: is a function that for all **u**, **v** from a space χ (which need not to be a vector space) satisfies

$$\mathbf{k}(\mathbf{u},\mathbf{v}) = (\Phi(\mathbf{u}),\Phi(\mathbf{v}))\ . \qquad (6)$$

where Φ is a mapping from the space χ to a Hilbert space *F* that is used called the feature space

$$\Phi : \mathbf{u} \in \chi \rightarrow \Phi(\mathbf{u}) \in F. \qquad (7)$$

the kernel function transform the data into a higher dimensional feature space to make it possible to perform the linear separation, as we can see in **Figure1.7** [12]

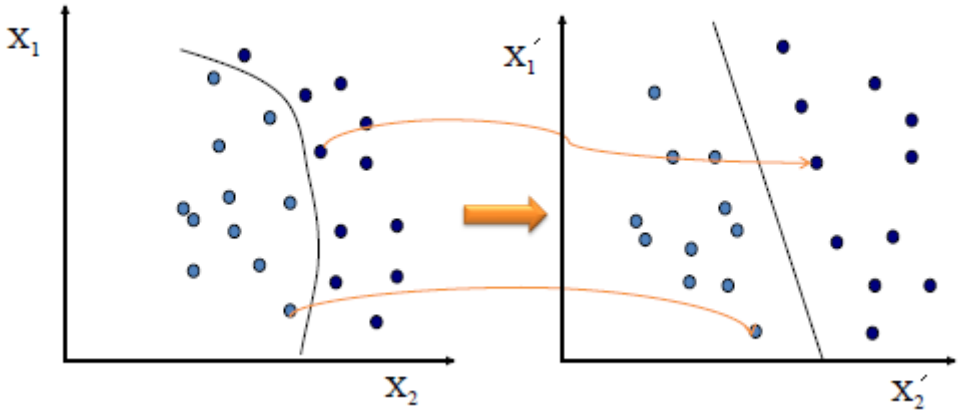


Figure 1.7 A non-linearly separable case.

The use of SVMs provides many advantages. The algorithm is based on an established theory, desires only tens of training specimens, and is unresponsive to the number of dimensions . However, the learning technique has relatively complex training and categorization algorithms, and high memory and time utilization during training and classification phases .

5.3 Artificial Neural Network :

An Artificial Neural Network is a branch of Artificial Intelligence, by inspiration from the human brain. The simple neuron model is made from studies of the human brain neurons. A neuron in the brain receives its chemical input from other neurons through its dendrites. If the input exceeds a certain threshold, the neuron fires its own impulse on to the neurons it is connected to by its axon. Below is a very simplified figure as each of the neurons of the brain is connected to about 10000 other neurons. [13]. Artificial Neural Network Artificial Neural Network (ANN) is a mathematical model that is inspired by the structure and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. The block diagram of **Figure 1.8** shows the mathematical model of a neuron, which

forms the basis for designing ANNs.

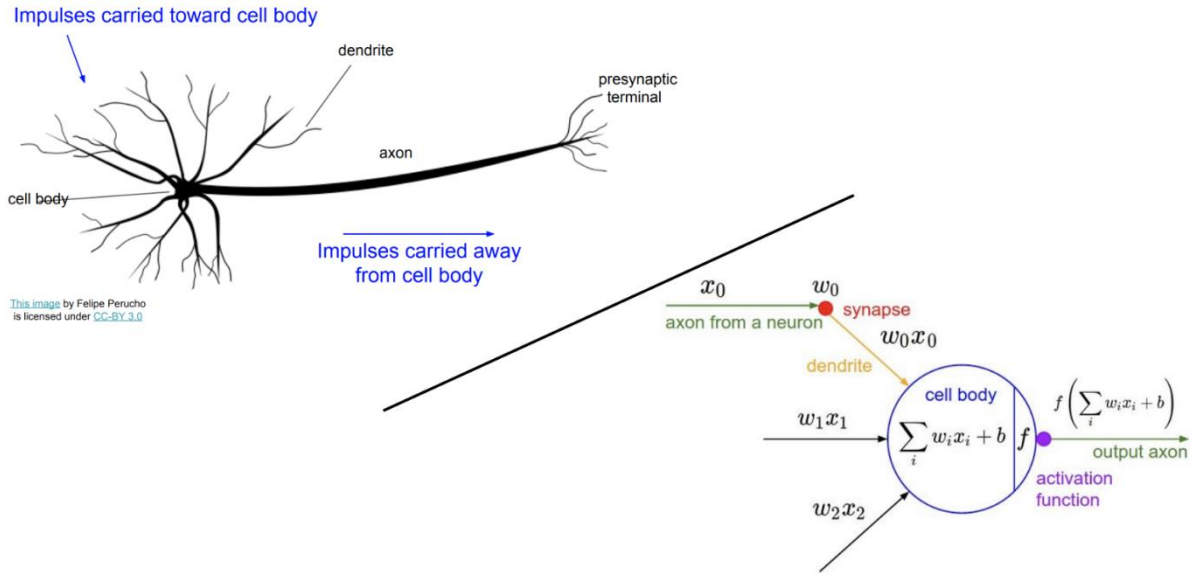


Figure 1.8 illustrations of biological and artificial neurons

Here we identify three basic elements of the neuronal model:

- A set of Synapses or connecting Links, each of which is characterized by a Weight or Strength of its own. Specifically, a signal x_i at the input of synapse i connected to neuron j is multiplied by the synaptic weight W_{ij}.
- .An Adder for summing the input signals, weighted by the respective synapses of the neuron.
- An Activation Function for limiting the amplitude of the output of a neuron. The activation function is also referred to as a Squashing Function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value. [15]

5.3.1 Activation functions

The activation function translates the input signals to output signals. There are four types of activation functions are commonly used, sigmoid, piecewise linear, unit step (threshold), and Gaussian.

5.3.1.1 Sigmoid

The sigmoid function consists of 2 functions, logistic and tangential. The values of logistic function range from 0 and 1 and -1 to +1 for tangential function , defined by the following formula

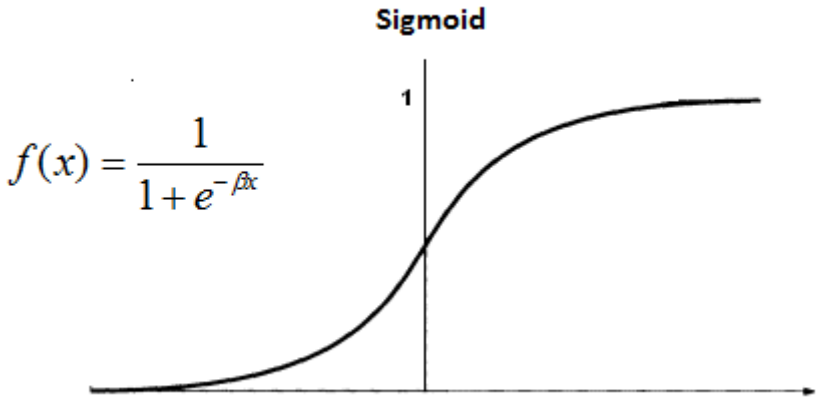


Figure 1.9 Sigmoid function

5.3.1.2 Piecewise Linear

The output is proportional to the total weighted output.

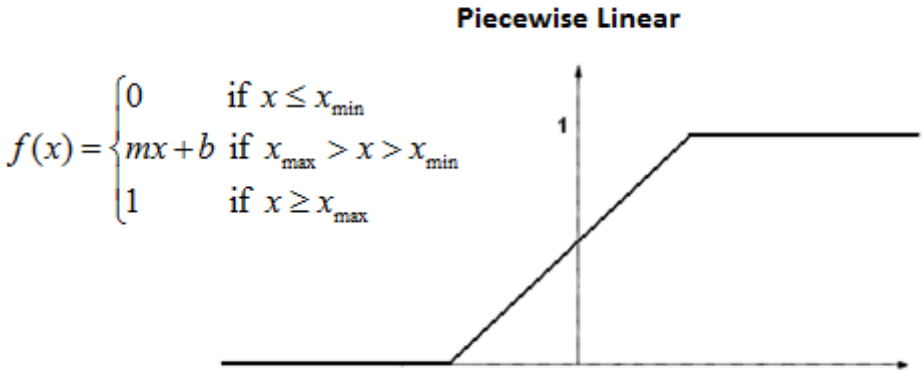


Figure 1.10 Piecewise Linear function

5.3.1.3 Unit step (threshold)

The output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

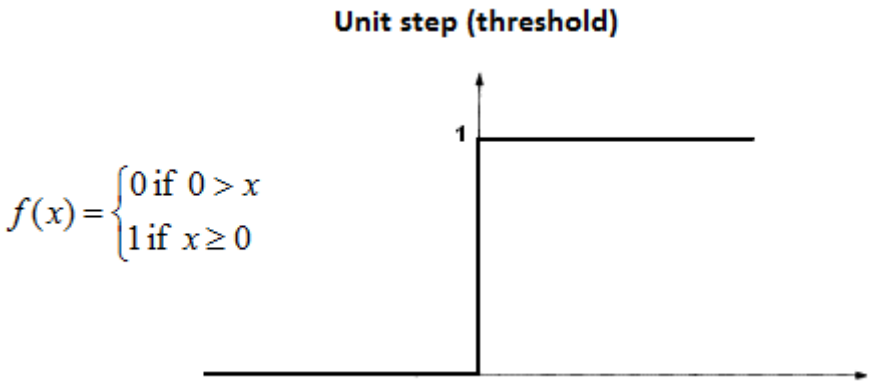


Figure 1.11 Threshold function

5.3.1.4 Gaussian

Gaussian functions are bell-shaped curves that are continuous. The node output (high/low) is interpreted in terms of class membership (1/0), depending on how close the net input is to a chosen value of average.

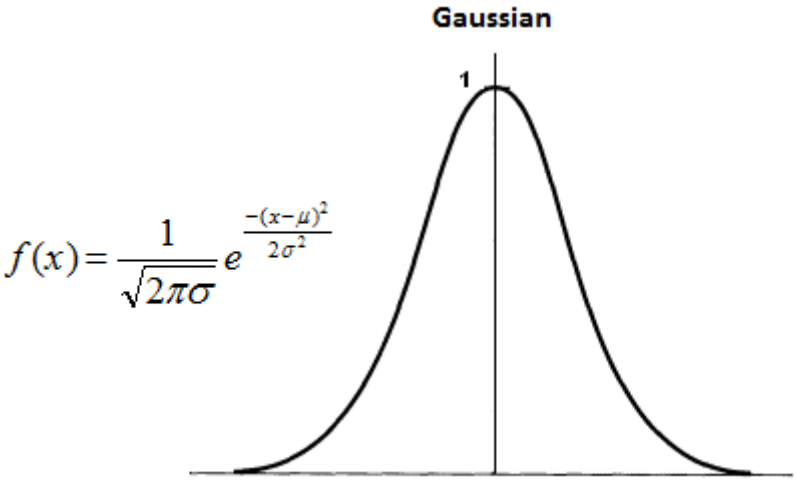


Figure 1.12 Gaussian function

The neural network will be formed by those artificial neurons , where those neurons are organized in layers and we have three layers. The first layer consists of input neurons. Those neurons send data on to the second layer, which in turn sends the output neurons to the third layer. And every input connect to every hidden and every hidden connect to every output .The single artificial neuron will do a dot product between w and x, then add a bias, the result is passed to an activation function that will add some non-linearity.

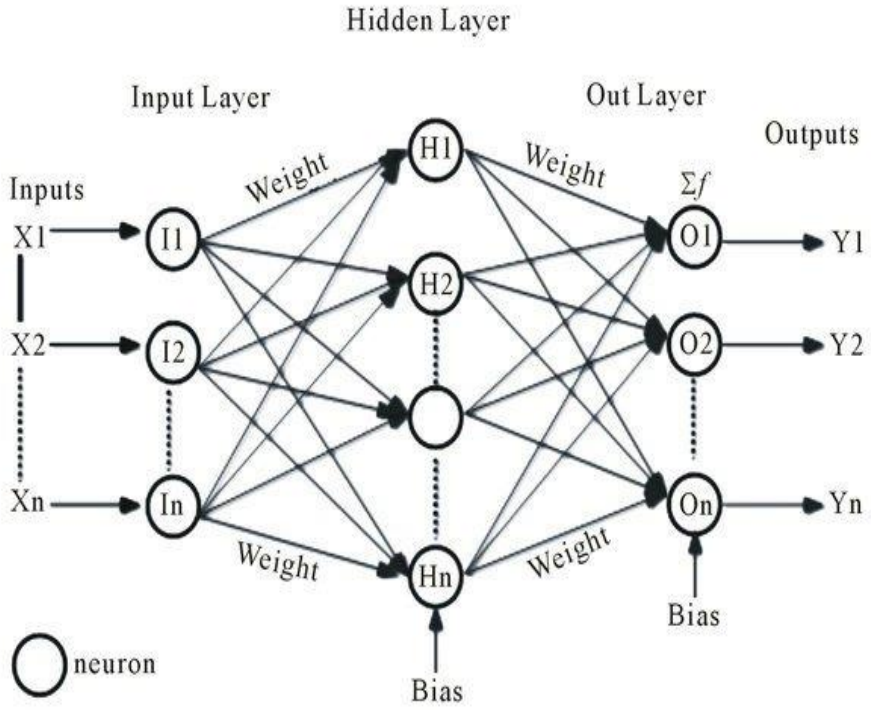


Figure 1.13 An example neural network consisting of one hidden layer

5.3.2 Network topologies

There are different types of neural networks, but they are generally classified into feed-forward and feed-back networks.

5.3.2.1A feed-forward network

is a non-recurrent network which contains inputs, outputs, and hidden layers; the signals can only travel in one direction. Input data is passed onto a layer of processing elements where it performs calculations. Each processing element makes its computation based upon a weighted sum of its inputs. The new calculated values then become the new input values that feed the next layer. This process continues until it has gone through all the layers and determines the output. A threshold transfer function is sometimes used to quantify the

output of a neuron in the output layer.

5.3.2.2 A feed-back network

has feed-back paths meaning they can have signals traveling in both directions using loops. All possible connections between neurons are allowed. Since loops are present in this type of network, it becomes a non-linear dynamic system which changes continuously until it reaches a state of equilibrium. Feed-back networks are often used in associative memories and optimization problems where the network looks for the best arrangement of interconnected factors.

5.3.3 The back-propagation algorithm

The back-propagation algorithm is used to update the neural network weights when they are Not able to make the correct prediction. Hence, we should train the neural network before applying back-propagation .

Learning networks is typically achieved through a supervised manner. It can be assumed to be Available a learning environment that contains both the learning models and models of desired output corresponding to input (this is known as "target models"). As we will see, learning is typically based on the minimization of measurement errors between network outputs and desired outputs. This implies a back propagation through a network similar to that which is learned. For this reason algorithm learning is called back-propagation.

The idea is to find the minimum error function $E(w)$ in relation to the connections weights.

Steps to train the network :

- Prepare the activation function input (sum of products between inputs and weights)
- Apply activation function
- Prediction error

After getting the predicted outputs, next is to measure the prediction error of the network. We can use the squared error function defined as follows:

$\boldsymbol{E} = (\boldsymbol{desired} - \boldsymbol{predicted})$

Almost time there is a prediction error and it should be minimized until reaching an acceptable error, so we must be change the weights in order to minimize the error. We can use the weights update equation

$$W_{new} = w_{old} + \eta(d - y)x \qquad (9)$$

$\mathbf{w_{old}}$:current weights

$\mathbf{W_{new}}$: new update weights

$\boldsymbol{\eta}$: network learning rate

$\boldsymbol{d}$: desired output

$\mathbf{y}$: predicted output

$\boldsymbol{X}$: current input at which the network made false prediction

Continue these operations until prediction error reaches an acceptable value.

- ✓ Updating weights.
- ✓ Retraining network.
- ✓ Calculating prediction error.

The back-propagation algorithm is used to understand effect of each weight over the prediction error.

Conclusion

After the exploration of the Machine Learning making an overview of its important sides, we can say that the task for machine learning is to optimize the network's classification performance, this is achieved by training datasets with popular algorithms and techniques capable of accomplishing this task. But this traditional algorithms has limitations for example the performance is solely dependent on the training set in addition to that the calculation complexity due the usage of all training samples for classification .

DEEP LEARNING

CHAPTER 2

1. Introduction

The simple machine learning algorithms work very well on a wide variety of important problems. However, they have not succeeded in solving the central problems in AI, such as pattern recognizing speech or objects.

The development of deep learning was motivated in part by the failure of traditional algorithms to generalize well on such AI tasks. The challenge of generalizing to new examples becomes exponentially more difficult when working with high-dimensional data, and how the mechanisms used to achieve generalization in traditional machine learning are insufficient to learn complicated functions in high-dimensional spaces. Such spaces also often impose high computational costs. Deep learning was designed to overcome these and other obstacles. Because Deep Belief Networks (DBNs) are based on Restricted Boltzmann Machines (RBMs), which are particular energy-based models, we introduce here the main mathematical concepts helpful to understand them, including Contrastive Divergence (CD) and some sampling's methods.

2. Deep Learning

2.1 History of Deep Learning " state-of-the-art"

Historically, the concept of deep learning was originated from artificial neural network research. Feed-forward neural networks or MLPs with many hidden layers, which are often referred to as deep neural networks (DNNs), are good examples of the models with a deep architecture. Back-propagation (BP), popularized in 1980's, has been a well-known algorithm for learning the parameters of these networks. Although back-propagation alone did not work well in practice then for learning networks with more than a small number of hidden layers (Bengio, 2009) [17]. The pervasive presence of local optima in the non-convex objective function of the deep networks is the main source of difficulties in the learning. Back-propagation is based on local gradient descent, and starts usually at some random initial points. It often gets trapped in poor local optima when the batch-mode BP algorithm is used, and the severity increases significantly as the depth of the networks increases. This difficulty is partially responsible for steering away most of the machine learning and signal processing research from neural networks to shallow models that have convex loss functions (e.g., SVMs, CRFs, and MaxEnt models), for which global optimum can be efficiently obtained at the cost of less modeling power. The optimization difficulty associated with

the deep models was empirically alleviated using three techniques: a larger number of hidden units, better learning algorithms, and better parameter initialization techniques.

Using hidden layers with many neurons in a DNN significantly improves the modeling power of the DNN and creates many closely optimal configurations. Even if parameter learning is trapped into a local optimum, the resulting DNN can still perform quite well since the chance of having a poor local optimum is lower than when a small number of neurons are used in the network. Using deep and wide neural networks, however, would cast great demand to the computational power during the training process and this is one of the reasons why it is not until recent years that researchers have started exploring both deep and wide neural networks in a serious manner.

Better learning algorithms also contributed to the success of DNNs. For example, stochastic BP algorithms are in place of the batch-mode BP algorithms for training DNNs nowadays. This is partly because the stochastic gradient descend (SGD) algorithm is the most efficient algorithm when training is carried out on a single machine and the training set is large (Bottou and LeCun, 2004) [10]. But more importantly the SGD algorithm can often jump out of the local optimum due to the noisy gradients estimated from a single or a small batch of samples. Other learning algorithms such as Hessian free (Martens 2010) [20] or Krylov subspace methods (Vinyals and Povey 2011) have shown a similar ability.[21]

The DNN parameter initialization technique that attracted the most attention is the unsupervised pretraining technique proposed in (Hinton et al. 2006) [22]. In these work a class of deep Bayesian probabilistic generative models, called deep belief network (DBN), was introduced. To learn the parameters in the DBN, a greedy, layer-by-layer learning algorithm was developed by treating each pair of layers in the DBN as a Restricted Boltzmann Machine (RBM) (which we will discuss later). This allows for optimizing DBN parameters with computational complexity linear in the depth of the network. It was later found out that the DBN parameters can be directly used as the initial parameters of an MLP or DNN and result in a better MLP or DNN than those randomly initialized after the supervised BP training when the training set is small. As such, DNNs learned with unsupervised DBN pre-training followed by back-propagation fine-tuning is sometimes also called DBNs in the literature (e.g., Dahl et al., 2011; Mohamed et al., 2010, 2012). More recently, researchers have been more careful in distinguishing DNNs from DBNs (Dahl et al., 2012; Hinton et al., 2012) [29], and when

DBN is used to initialize the parameters of a DNN, the resulting network is called DBN-DNN (Hinton et al., 2012). [30]

As visualized in Figure 2.2, the history of neural networks can be plotted on a hype curve with open questions and resulting uncertainty of the future of neural networks.

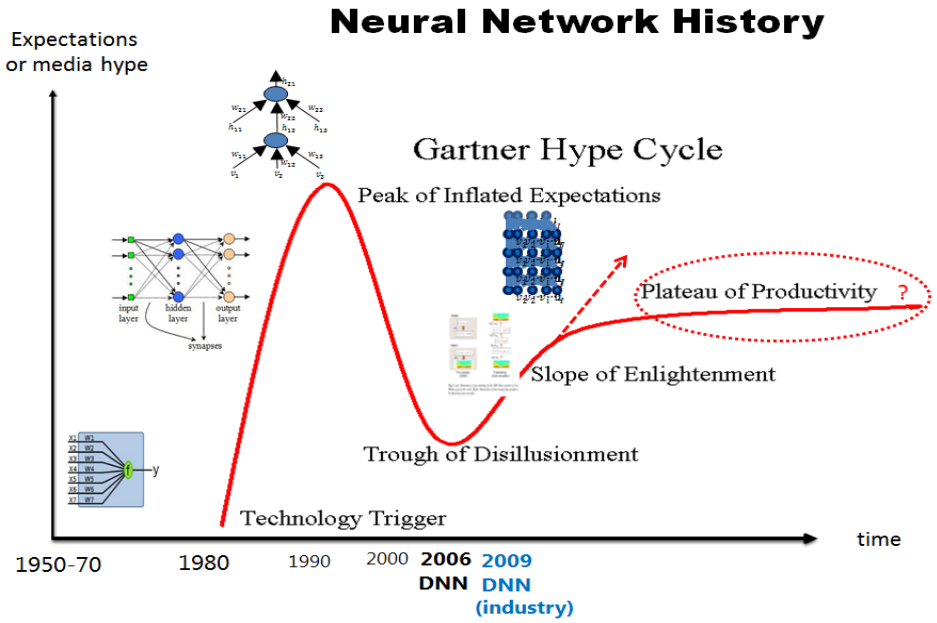


Figure 2.1 History of neural networks

2.2 Applications and Commercial activities

Deep learning is presented as an important step towards realizing strong AI and for this reason, many organizations have become interested in its use for many and particular applications. In recent days deep neural networks and deep learning realize exceptional performance on many important problems in computer vision, speech recognition, and natural language processing. They're now deployed on a large scale by companies such as Google, Microsoft, and Facebook.

In December 2013, Facebook recruited Yann lecun to head its new artificial intelligence lab that was to have operations in California, London, and New York. The AI lab developed deep learning techniques to help Facebook do tasks such as automatically tagging uploaded pictures with the names of the people in them In March 2013, Google hired Geoffrey Hinton and two of his students. Their work was to concentrate on both improving existing machine learning product At Google and help deal with the growing amount of data Google. Google also bought Hinton's company, DNN research.

In 2014, Google bought also deep mind Technologies, this is a British start-up that developed a system capable of learning how to play Atari video games by using just raw pixels as data input.

In 2015 Alpha Go system was demonstrated which realized one of the long-standing "grand challenges" of AI by learning the game of Go, it’s enough to beat a human professional Go player.

3. A Three-Category Classification in DL

As described earlier, deep learning refers to a rather wide class of machine learning techniques and architectures, with the hallmark of using many layers of non-linear information processing that are hierarchical in nature. Depending on how the architectures and techniques are intended for use, e.g., synthesis/generation or recognition/classification, one can broadly categorize most of the work in this area into three classes:

3.1 Generative deep architectures, which are intended to capture high-order correlation of the observed or visible data for pattern analysis or synthesis purposes, and/or characterize the joint statistical distributions of the visible data and their associated classes. In the latter case, the use of Bayes rule can turn this type of architecture into a discriminative one.

3.2 Discriminative deep architectures, which are intended to directly provide discriminative power for pattern classification purposes, often by characterizing the posterior distributions of classes conditioned on the visible data and hidden data.

3.3 Hybrid deep architectures, where the goal is discrimination which is assisted (often in a significant way) with the outcomes of generative architectures via better optimization or/and regularization, or where discriminative criteria are used to learn the parameters in any of the deep generative models in category 1) above.

4. Deep Learning models

4.1 Convolutional Neural Networks (CNNs)

A convolutional neural networks (CNNs) are a powerful machine learning technique from the field of deep learning, that are the most frequently models used for image processing and computer vision. A convolutional neural network takes an image expressed as an array of numbers, applies a series of operations to that array and, at the end, returns the probability that an object in the image belongs to a particular class of objects. For instance, a convolutional neural network can let you know the probability that a photo you took contains a building or a car or what have you. It might be used to distinguish between very similar instances of something.[23] Convolutional neural network contain sequence of layers(convolution layer, ReLU (rectified linear units) layer, pooling layer ,fully connected layer, loss layer (during the training process))to build its architecture which is depicted in the following **Figure2.2**.

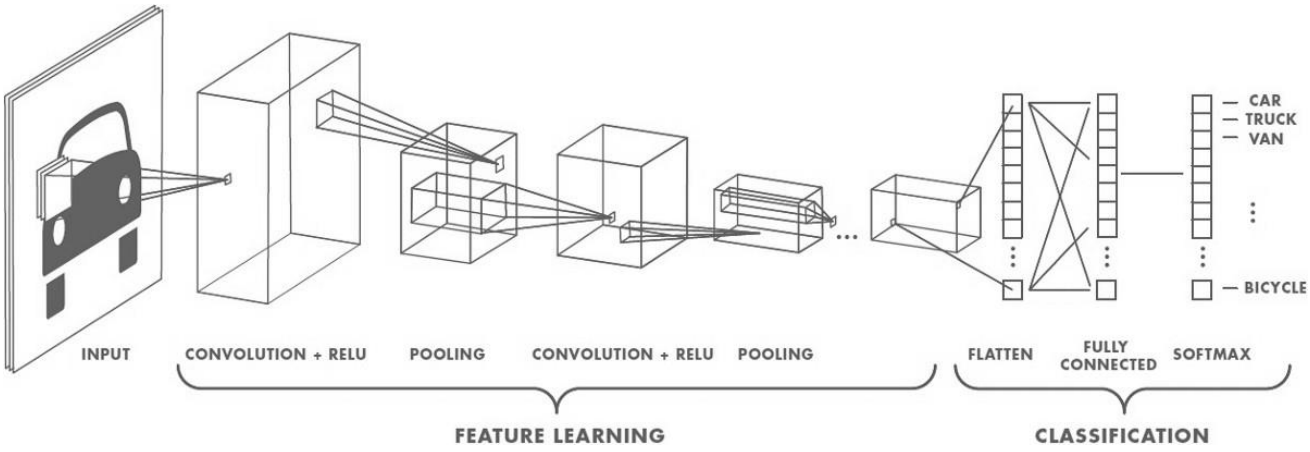


Figure 2.2 Convolutional Neural Network architecture

4.2 Stacked Auto-Encoders

Stacked Auto-Encoder (SAE) is one of the most important types of deep network. It is a neural network consisting of multiple layers of sparse auto-encoders stacked one over another. The SAE owes its excellent computational ability to this stacked collection of the auto-encoders . The figure below shows the structure of the SDA, consisting of three layers, named as input layer, code layer and reconstruction layer. The original input X enters at the input layer, and undergoes encoding through forward-propagation to return Y. Next, the encoded input Y is decoded back to give X'. X' is a reconstruction of the

original input X and has the same dimensionality as that of X.[33]

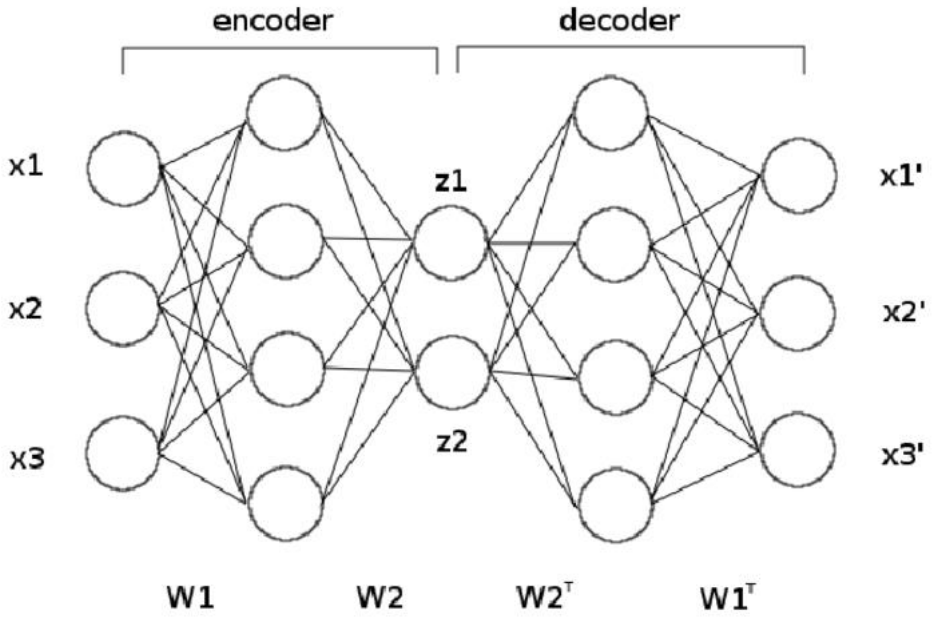


Figure 2.3 Auto-Encoder structure

4.3 Restricted Boltzmann Machines (RBMs)

A restricted Boltzmann machine is an energy-based generative model defined over a visible layer **v** (sometimes called input) and a hidden layer **h** (sometimes called hidden factors or representation). RBM is a stochastic neural network consisting of two layers, layer of i binary visible units , **v**= [v_1,v_2, . . . ,v_i] where $v_i \in \{0,1\}$, and a layer of j binary hidden units , **h** = [h_1,h_2, . . . ,h_j] where $h_j \in \{0,1\}$.[24]

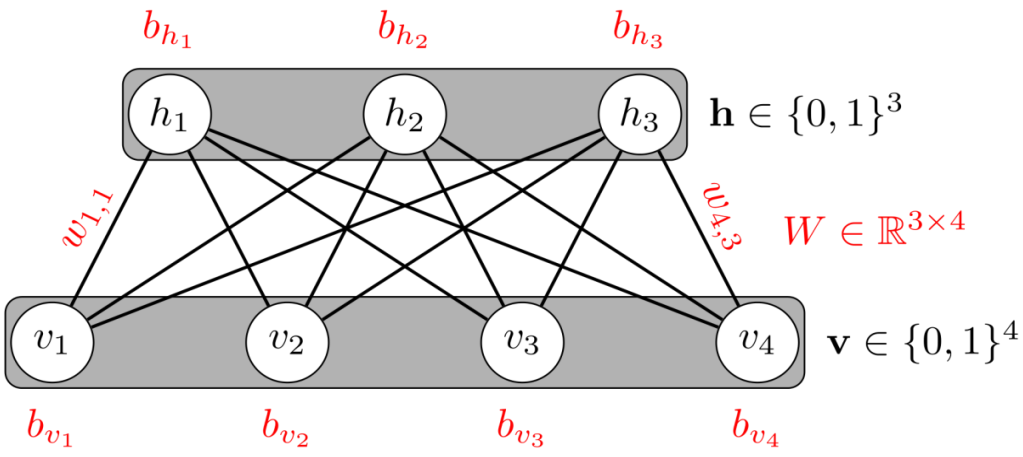


Figure 2.4 Restricted Boltzmann machine's architecture

The energy of the joint configuration (v,h) in Boltzmann machine is given as follows:

E(v, h) = -h^TWv-c^{Tt}v-b^Th

$$= -\sum_{i=1}^{i} c_i v_i - \sum_{j=1}^{j} b_i h_i - \sum_{i=1}^{i}\sum_{j=1}^{j} W_{ij} c_i h_j \quad (2.1)$$

Where:

W represents the symmetric interaction term between visible unit i and hidden unit j, while bi and cj are bias terms for hidden units and visible units respectively. The network assigns a probability distribution value with energy function to each state in visible and hidden units(v, h) , and we can be defined as :

$$P(v, h) = \frac{1}{Z} e^{-E(v,h)} \quad (2.2)$$

Where Z as partition function or normalization constant, is obtained by summing over all possible pairs of visible and hidden vectors.

$$Z = \sum_{v,h} e^{-E(v,h)} \quad (2.3)$$

Since there are no connections between any two units within the same layer, given a particular random input configuration, **v**, all the hidden units are independent of each other and the probability of **h** given **v** becomes:

$$P(h \mid v) = \prod_j P(h_j = 1|v) \quad (2.4)$$

where :

$$P(h_j = 1|v) = \alpha\,(b_j + \sum_{i=1}^{i} W_{ji}\, v_i)$$

Similarly given a specific hidden state, **h**, the probability of **v** given **h** is obtained By :

$$P(h \mid v) = \prod_i P(v_i = 1|h) \quad (2.5)$$

where :

$$P(v_i = 1|h) = \alpha\,(c_i + \sum_{j=1}^{j} W_{ji}\, h_j)$$

Hence, given a specific training vector **v** its probability can be raised by adjusting the weights and the biases of the network in order to lower the energy of that particular vector while raising the energy of all the others. To this end, we can perform astochastic gradient ascent on the log-likelihood manifold obtained from the training data vectors1, by computing the derivative of the log probability with respect to the network parameters $\theta \in b_j$, c_i ,W_{ji}:

$$\frac{\partial \log p(v)}{\partial \theta} = \underbrace{-\sum_h p(h \mid v)\, \partial \frac{E(v,h)}{\partial \theta}} + \underbrace{\sum_{v,h} p(v,h)\, \partial \frac{E(v,h)}{\partial \theta}} \quad (2.6)$$

Positive phase negative phase

As in the maximum likelihood learning procedure, we aim at finding the set of network parameters for which the probability of the (observed) training dataset is maximized. Computing $\partial \frac{E(v,h)}{\partial \theta}$is straightforward. Thus, in order to obtain anunbiased stochastic estimator of the log-likelihood gradient, we need a procedure to sample from $p(\mathbf{h} \mid \mathbf{v})$ and another to sample from $p(\mathbf{v},\mathbf{h})$ [17]. In the so-called positive phase, **v** is clamped to the observed input vector, **x**, and **h** is sampled from **v**, while in the negative phase, both **v** and **h** are sampled ideally from the model .Sampling can be accomplished by setting up a Markov Chain Monte Carlo(MCMC) using alternating Gibbs sampling. Each iteration of Gibbs sampling consists of updating all of the hidden units in parallel using (2.4) followed by updating all of the visible units in parallel using (2.5) [27]. This process is represented in **Figure 2.5**

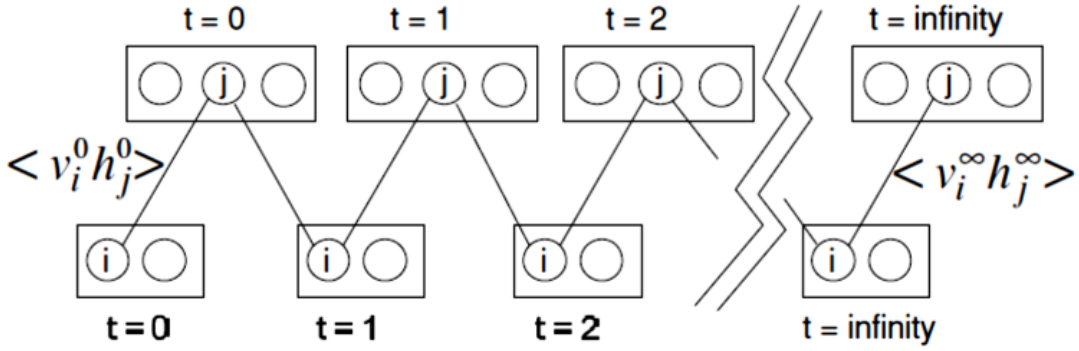


Figure 2.5 Gibbs sampling. Each Gibbs sampling step means updating of all hidden units according to equation (2.4) and then updating all visible units according to equation (2.5). The chain is initialized by setting the binary states of the visible units to be the same as a data vector.

- **Contrastive Divergence (CD) :**

Since Gibbs sampling method is slow, Contrastive Divergence (CD) algorithm is used [28]. In this method visible units are initialized using training data. Then binary hidden units are computed according to equation *(2.4)*. After determining binary hidden unit states, vi values are recomputed according to equation *(2.5)*. Finally, probability of hidden unit activations is computed and using these values of hidden units and visible units, negative phase is computed.

Although CD method is not a perfect gradient computation method, but its results are acceptable [28]. By repeating Gibbs sampling steps, CD-k method is achieved. The k parameter is the number of repetitions of Gibbs sampling steps. This method has a higher performance and can compute gradient more exactly [17]

RBMupdate(**v**,**η**,W,**b**,**c**)

This is the RBM update procedure for binomial units. It can easily adapted to other types of units.

V is a sample from the training distribution for the RBM

η is a learning rate for the stochastic gradient descent in Contrastive Divergence

W is the RBM weight matrix, of dimension (number of hidden units, number of inputs)

b is the RBM offset vector for input units

c is the RBM offset vector for hidden units

Notation: P($\boldsymbol{h_2}\cdot = 1|\boldsymbol{v_1}$) is the vector with elements P($\boldsymbol{h_{2i}} = 1|\boldsymbol{v_1}$)

for all hidden units i **do**

• compute P($\boldsymbol{h_{1i}}$=1|**v**) (for binomial units, sigm($\boldsymbol{c_1} + \boldsymbol{\eta_j W_{ij} v_j}$))

• sample $\boldsymbol{h_{2i}} \in \{0,1\}$ from P($\boldsymbol{h_{1i}}|\boldsymbol{v_1}$)

end for

for all visible units j **do**

• compute P($\boldsymbol{v_{1j}}$ =1|$\boldsymbol{h_1}$) (for binomial units, sigm($\boldsymbol{b_j} + \boldsymbol{\eta_i W_{ij} h_{1i}}$))

• sample $\boldsymbol{v_{1j}} \in \{0,1\}$ from P($\boldsymbol{v_{1j}} = 1|\boldsymbol{h_1}$)

end for

for all hidden units i **do**

• compute P($\boldsymbol{h_{2i}}$=1|$\boldsymbol{v_1}$) (for binomial units, sigm($\boldsymbol{c_i} + \boldsymbol{\eta_j W_{ij} v_{1j}}$))

end for

• W ← W + **η** ($\boldsymbol{h_1}$**v'** − P($\boldsymbol{h_2}\cdot = 1|\boldsymbol{v_1}$)$\boldsymbol{v'_1}$)

• **b**← **b** + **η** (**v** − $\boldsymbol{v_1}$)

• **c** ←**c** + **η** ($\boldsymbol{h_1}$ − P($\boldsymbol{h_2}\cdot = 1|\boldsymbol{v_1}$))

- **Persistent Contrastive Divergence (PCD)**

Whereas CD-k has some disadvantages and is not exact, other methods are proposed in RBM. One of these methods is PCD that is very popular . Unlike CD method that uses training data as initial value for visible units, PCD method uses last chain state in the last update step. In other words, PCD uses successive Gibbs sampling runs to estimate the positive phase. Although all model parameters are changed in each step, but can receive good samples from model distribution with a few Gibbs sampling steps because the model parameters change slightly . Many persistent chains can be run in parallel and we will refer to the current state in each of these chains as new sample or a "fantasy" particle .

- **Free Energy in Persistent Contrastive Divergence (FEPCD)**

In PCD method, as described before, many persistent chains can be run in parallel and we will refer to the current state in each of these chains as a "fantasy" particle. Chain selection in this method is blind and the best one may not be selected. If we can define a criterion for goodness of a chain, samples and therefore computing gradient will be more accurate.

The proposed criterion for selecting the best chain is the free energy of visible sample $v$which is defined as follows :

$$p(v) = \frac{1}{z} e^{-F(v)} = \frac{1}{z} \sum e^{-E(v,h)} \qquad (2.7)$$

where $F(v)$ is free energy. Therefore $F(v)$ can be computed as follows :

$F(v) = -\sum_i v_i a_i - \sum_j q_j I_j + \sum_j (q_j \log q_j + (1-q_j) \log(1 - q_j))$ (2.8)

Where : $I_j = b_j + \sum_i v_i w_{ij}$is equal to sum of inputs to hidden unit j and q_j = g (I_j) is equal to activation probability of hidden unit h_j given v and g is logistic function . An equivalent and simpler equation for computing $F(v)$ is as follows:

$$F(v) = -\sum_i v_i a_i - \sum_j \log(1 + e^{I_j}) \qquad (2.8)$$

An RBM by itself is limited in what it can represent and its true potential emerges when several RBMs are stacked together to form a DBN

4.4 Deep Belief Networks (DBNs)

Deep belief Network (DBN) is a stack of simple networks, such as RBMs, that were trained layer-wise in a unsupervised procedure. Training of a DBN consists of two phase, that allows to learn features hierarchies.

In the first phase, generative unsupervised learning is performed layer-wise on RBMs, first, RBM is trained on the data, second its hidden units are used as input to another RBM , which is trained on them . In the second phase, fine-tuning all the parameters of the network using back-propagation and gradient on a global supervised cost function

4.3.1 Layer-Wise Training of deep Belief Networks

A Deep Belief Network with l layers models the joint distribution between observed vector x and l hidden layers h^k as follows:

$P(x, h^1, \dots, h^l) = \left(\left(\prod_{k=0}^{l-2} P(h^k|h^{k+1})\right) P(h^{l-1}, h^l\right)$ (2.9)

Where x=h^0, P($h^{k-1}|h^k$) is a visible-given-hidden conditional distribution in an RBM associated with level k of the DBN , and $P(h^{l-1}, h^l)$ is the joint distribution in the top-level RBM .This is illustrated in **Figure 2.6**

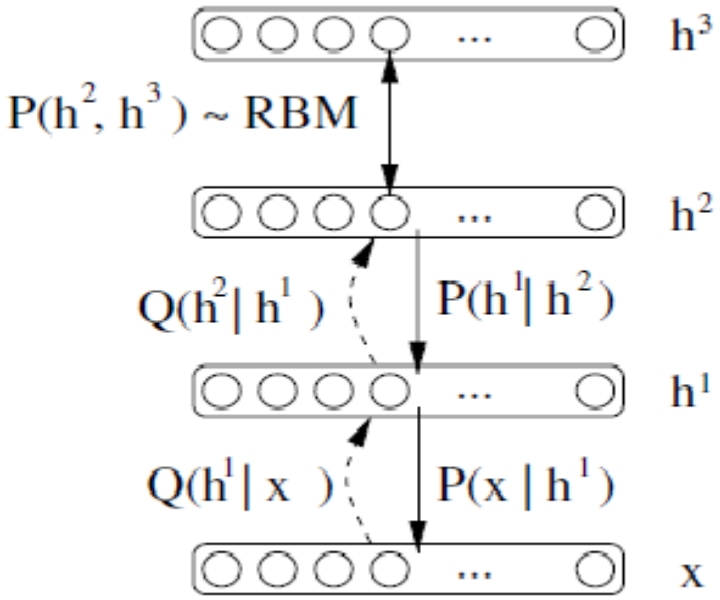


Figure 2.6 Deep Belief Network architecture

The conditional distributions $P(h^k|h^{k+1})$ and the top-level joint an RBM $P(h^{l-1}, h^l)$ define the generative model. In the following we introduce the letter Q for exact or approximate posteriors of that model, which are used for inference and training .

The Q posteriors are all approximate except for the top-level Q(h^l, h^{l-1}) which is equal to the true P(h^l, h^{l-1}) because (h^l, h^{l-1}) from an RBM, where exact inference is possible.[17] The training of the DBN in a greedy layer-wise, as illustrated with the following pseudo-code algorithm.

Algorithm 2

`TrainUnsupervisedDBN`($\widehat{P}, \epsilon, \ell, W, \mathbf{b}, \mathbf{c}$, `mean_field_computation`)

Train a DBN in a purely unsupervised way, with the greedy layer-wise procedure in which each added layer is trained as an RBM (e.g., by Contrastive Divergence).

$\widehat{P}$ is the input training distribution for the network

ϵ is a learning rate for the RBM training

ℓ is the number of layers to train

W^k is the weight matrix for level k, for k from 1 to ℓ

$\mathbf{b}^k$ is the visible units offset vector for RBM at level k, for k from 1 to ℓ

$\mathbf{c}^k$ is the hidden units offset vector for RBM at level k, for k from 1 to ℓ

`mean_field_computation` is a Boolean that is true iff training data at each additional level is obtained by a mean-field approximation instead of stochastic sampling

```
for k = 1 to ℓ do
  • initialize W^k = 0, b^k = 0, c^k = 0
  while not stopping criterion do
    • sample h^0 = x from P̂
    for i = 1 to k − 1 do
      if mean_field_computation then
        • assign h^i_j to Q(h^i_j = 1|h^{i−1}), for all elements j of h^i
      else
        • sample h^i_j from Q(h^i_j|h^{i−1}), for all elements j of h^i
      end if
    end for
    • RBMupdate(h^{k−1}, ε, W^k, b^k, c^k) {thus providing Q(h^k|h^{k−1}) for future use}
  end while
end for
```

Figure 2.7 DBN algorithm

5. Difficulty of training Deep belief Network

In general, the more units and layers, the higher a network's expressional power. This comes with a more complex cost function. Learning will then easily get stuck in a local minimum, leading to high generalization errors and overfitting. There are different methods to reduce overfitting, for example smaller networks, more training data, better optimization algorithms, or regularization, which is good explained in [18,19]**.**

5.1 Regularization

Regularization allows to reduce overfitting of neural networks and learning algorithms in general. This section covers different regularization methods.

L_2 and L_1 regularization

Large parameter values often allow models to match training data well, but do not generalize to new data. L2 and L1 regularization "penalize" large parameter values, by adding them to the cost function. L2 regularization also known as weight decay is used predominantly and adds the squared weights to the cost function:

$$J_{reg}(\Theta) = J(\Theta) + \lambda \sum_{l=1}^{L-1} \sum_{i=1}^{s_{l+1}} \sum_{j=1}^{s_l} \left(\Theta_{ij}^{(l)}\right)^2$$

The magnitude of the regularization can be controlled by the regularization parameter , whose value is often subject to model selection.

An alternative approach is L1 regularization, which only adds the linear weights to the cost function:

$$J_{reg}(\Theta) = J(\Theta) + \lambda \sum_{l=1}^{L-1} \sum_{i=1}^{s_{l+1}} \sum_{j=1}^{s_l} \left|\Theta_{ij}^{(l)}\right|$$

5.2 Early stopping

Another popular regularization method is early stopping, in which for every training iteration, not only the training error, but also a validation error is calculated. The training error is usually a monotonic function, that decreases further in every iteration.

In contrast, the validation error usually drops off first, but then increases, indicating the model starting to overfit. In early stopping, training is stopped at the lowest error on the validation set.

5.3 Invariance

Neural networks can be made invariant of transformations, such as rotation or translation. A simple method is to apply these transformations on the training data and then to train the neural network on the modified training data as well. A different approach is tangent propagation [16] which embeds a regularization function into the cost function. The regularization function is high for non-invariant network mapping functions and zero when the network is invariant under the transformation.

5.4 Dropout

Dropout regularization presented in [32] is a general form of regularization for neural networks. It is motivated by genetic algorithms covered in [31], which combine the genes of parents and apply some mutation in order to produce an offspring. The idea behind dropout regularization is to thin a neural network by dropping out some of its units, as visualized in **Figure 2.8**. The lower number of parameters reduces overffitting of the network. Each unit is retained with probability p, where p is fixed and set for all units. p. can be found in model selection. Alternatively, it can be set to 0.5, which is a near-optimal choice for most applications, as evaluated in [32]. It should be set to greater than 0.5 and closer to 1 for the input units, in order to preserve most input units. It has been empirically determined that a value of 0.8 is a good value for the input units. For a network with n units, there are 2^n different thinned networks, which share most weights, and the total number of parameters is still O(n^2). For each training example, all thinned networks are sampled and trained. The weights of the thinned networks can be combined in order to construct a network with n units. This process allows to filter out noise in the training data, in order to prevent overfitting.

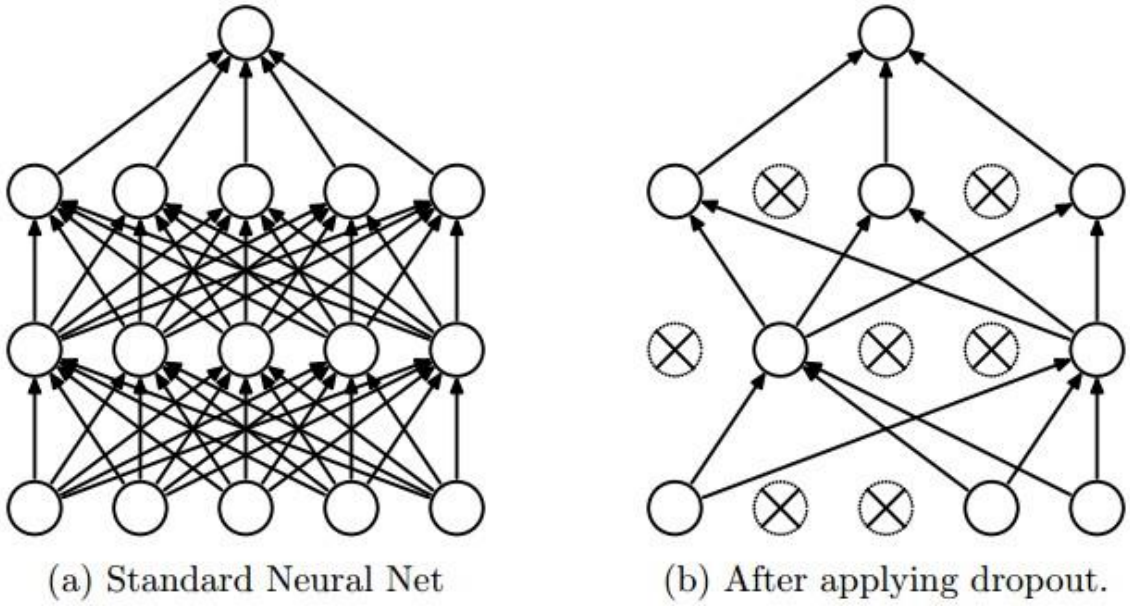


Figure 2.8 Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

6. Conclusion

The approach of DNNs holds great promise for AI tasks like human-level object recognition and speech perception, in which networks with shallow architecture lead to poor results, because they are not able to learn multiple layers of representation.

A significantly benefit of deep networks in comparison to shallow networks is that they enable to capture highly nonlinear functions with a compact set of parameters, especially if labeled training data are scarce or nonexistent.

By learning layer-by-layer a different representation of the input, the network discovers features in the data automatically that enable good generation. After this pre-training phase (the unsupervised greedy layer-wise pre-training algorithm CD) the supervised algorithm Back-propagation fine-tunes the weights in order to improve the recognition ability. This training procedure achieves a great accuracy in many complex applications, even if there is just a small number of labeled training data available. In this chapter we have focused of the architecture of the Deep belief Network, the concept of training for dataset, the importance to add a hyper parameter such as, dropout in the setting of neurons for complex task. The idea behind this is to learn a model that is able to generate data, what decreases the number of required labeled training samples.

in this work, we have being interesting to study the effect of dropout on the performance of DBN, and therefore the space of hyper parameters may use is reduced in the training of DBN. In the next chapter, we will make a series experiences to show the advantage of DBN instead of ML(MLP, SVM,..etc) on two public dataset. Moreover, the manner of learning data in DL is called without handcraft features, so, to demonstrate it we will be apply only a DBN on the dataset of face ORL.

CHAPTER 3

Implementation and experimentation

1. Introduction

This chapter applies DBN method of the previous chapter to real learning problems. this method is compared on two databases (MNIST and ORL) and their results are presented here.

2. Development environment and tools

2.1 Matlab

MATLAB is one of the high-performance languages that is generally used for the purpose of technical computing. By integrating the visualization, and programming, MATLAB creates an easy-to-use environment where understandable mathematical notations are used to express the problems and solutions. This language is generally used in the development of algorithms, and computation and maths.

The decision to use Matlab was made because The framework itself is meant to run on a desktop computer. Two Matlab toolboxes are used: The Parallel Computing, and Neural Network Toolbox. The Neural Network Toolbox provides algorithms, functions, and apps to create, train, visualize, and simulate neural networks .The toolbox includes deep learning algorithms for image classification and feature learning tasks. To speed up training of large data sets, you can distribute computations and data across multicore processors, clusters using Parallel Computing Toolbox.

3. Available databases

The features and size of the images used for training a model are important in that the learning rate is acceptable and accurate results can be obtained. A large number of images used for training affects the learning of the model positively. The availability of many image sources on the Internet and the progress of large data technology allow the creation of many large data sets that are open for access .This section briefly covers the two databases that are of interest for the experiments of this work .

3.1 MNIST

The Mixed National Institute of Standards and Technology (MNIST) data-base is a collection of handwritten digits with different levels of noise and distortions. It was initially created by LeCun for his research on convolutional neural networks in the 1980s. It has 60000 training examples and 10000 test examples. Each example contains (28*28)=784 pixel gray-scale values,. These pixels are a dimensional vector represents the features in the MNIST data set. Moreover, each pixel in the MNIST

data set is represented by a value between 0 and 255, where 0 is black and 255 is white and anything in between is represented by a different shade of gray .

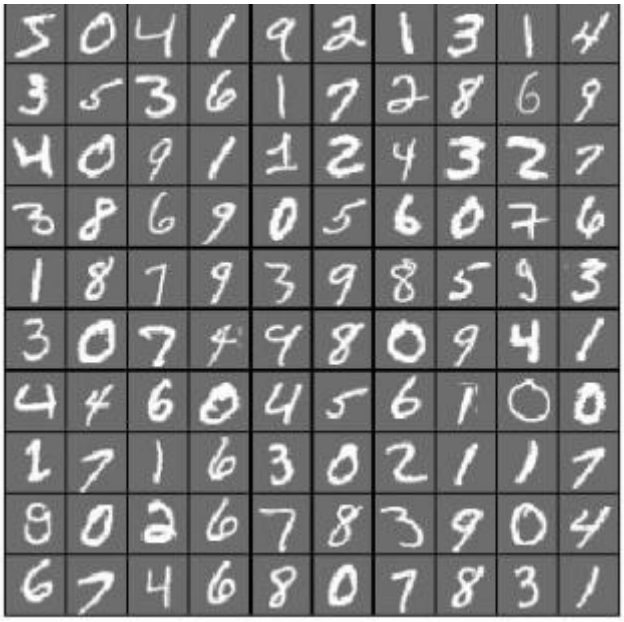


Figure 3.1 MNIST dataset

3.2 ORL

We used the ORL face database composed of 400 images of size 112 x 92. There are 40 persons, 10 images per each person. The images were taken at different times, lighting and facial expressions. The faces are in an upright position in frontal view, with a slight left-right rotation. In example we performed factorization on reduced face images by constructing a matrix of shape 2576 (pixels) x 400 (faces) and on original face images by constructing a matrix of shape 10304 (pixels) x 400 (faces).

Figure 3.2 ORL Database of Faces.

4. Experiments

DBN uses Restricted Boltzmann Machine (RBM) of unsupervised learning networks as building blocks for the multi-layer learning systems and uses a supervised learning algorithm named BP (back-propagation) for fine-tuning after pre-training

4.1 Classification on MNIST

4.1.1 Sampling methods

We did a variety of experiments, using different sampling methods types, these sampling methods are Gibbs, CD, PCD and FEPCD that we have implemented, and **Table3.1** shows a classification experiment using MNIST dataset for a DBN composed of RBMs (784-500-500-2000), after training each RBM, the DBN was fine-tuned in 200 epochs using back-propagation method, as we can see in **Figure3.3**.

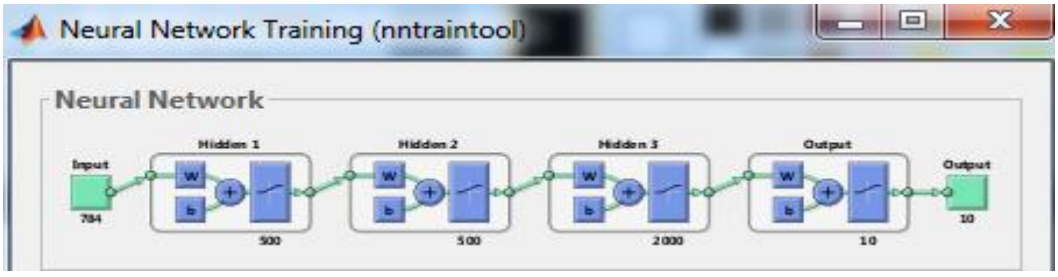


Figure 3.3 Architecture of DBN on MNIST dataset

Learning Methods	Gibbs sampling	CD	PCD	FEPCD
Error	0.8865	0.0173	0.0119	0.0123

Table 3.1 Classification error on MNIST dataset for a DBN (784-500-500-2000)

4.1.2 Dropout

The evaluation of Deep Learning approaches has been divided into two parts, training and testing. The dataset have been divided for example into 80 percent for training and 20 percent for testing. The approaches of DL which is, Dropout have been trained and tested using different values of values p (0 to 1) .

The DBN algorithm was chosen for its accuracy and compatibility with deep learning. It is necessary to measure the accuracy of the neural network. The Root Mean Square Error (RMSE) was employed for classification is given in the following equation:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_1 - y_i)^2}{n}}$$

For classification problems, the RMSE is the difference between the predicted and actual probabilities of each class. For a classification problem, the expected probability is 1.0 for the correct class and 0.0 for all other classes.

Parameters	Default value	Tested values
Learning rate	1.0	0.25, 0.5, 0.75, 1.0, 1.25,
Momentum	0	0.01, 0.02, , 0.1, 0.15, 0.2,
Output unit type	sigmoid	Sigmoid, softmax
Batch size	100	25, 50, 100, 150, 200, 400
Hidden layers	[100,100]	[50, 50],[16,8], [500, 500]
Dropout	0	0, 0.125, 0.25, 0.5

Table 3.2 Model selection values for MNIST

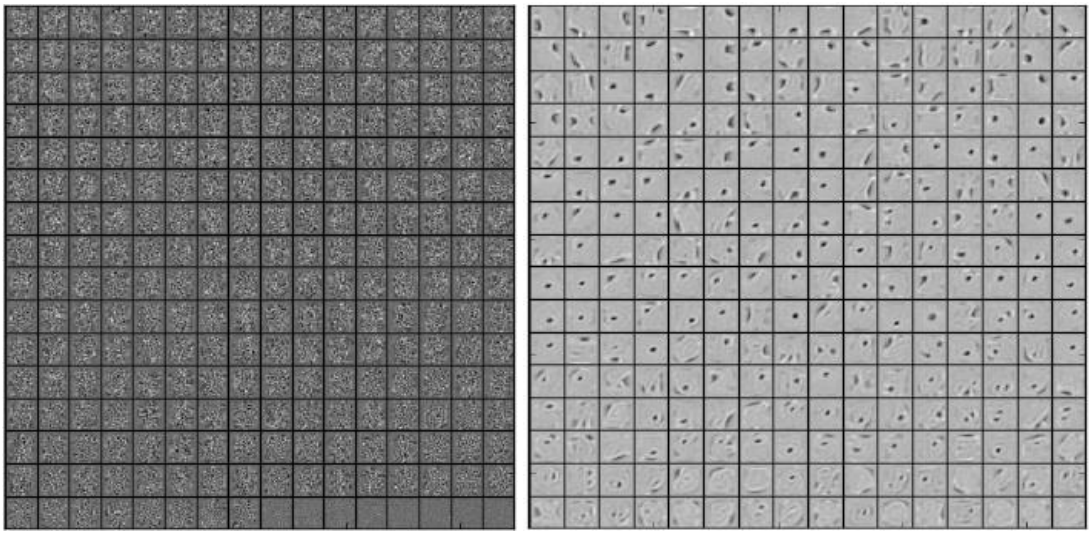
The parameters are optimized independently for computational performance reasons. During the optimization, the other parameters are set to the default values in **Table3.2.** The same procedure is followed for the other parameters with 100 epochs for both pre-training and fine-tuning. **Table3.3** includes the optimal values and respective test errors for DBN. Training method achieve the best error rate improvement for a change of the number of hidden units from one layer of 100 units to two layers of [400,400] units .

Momentum

Using momentum when updating parameters is a useful tool to avoid local minima during training. In a model using momentum, the gradient for each update effects the current velocity of a parameter instead of the parameter itself. The momentum also decays over time to prevent excessive and counterproductive parameter updates, at a rate determined by a hyper-parameter

Parameters	Values	Test error
MaxIterartion	100	0.5771
Learning rate	0.5	0.2155
Momentum	0.5	0.5369
Output unit type	sigmoid	0.0373
Batch size	50	0.3003
Hidden layers	[16,8]	0.4010
Dropout	0.5	0.5771

Table 3.3 Model selection for DBN on MNIST



(a) Without dropout p=0 (b) Dropout with p = 0:5.

Figure3.4 Features learned on MNIST with one hidden layer RBM having 256 reconstructed linear units.

Figure3.4 (a) shows features learned by an DBN on MNIST with a three hidden layer of 256 RBM units without dropout. Figure (b) shows the features learned by an identical DBN which used dropout in the hidden layer with p = 0:5. Both DBN had similar test reconstruction errors. However, it is apparent that the features shown in Figure(a) have good learned in order to produce good reconstructions. Each hidden unit on its own does not seem to be detecting a meaningful feature. On the other hand, in Figure(b) , the hidden units seem to detect the significant information in the whole of the image.

4.2 Classification of ORL

Face Recognition includes three key steps: image preprocessing, feature extraction an classification. Image preprocessing is essential process before feature extraction and also is the important step in the process of Face Recognition. Feature extraction is mainly to give an effective representation of each image, which can reduce the computational complexity of the classification algorithm and enhance the separability of the images to get a higher recognition rate. While classification is to distinguish those extracted features with a good classifier. Therefore, an effective face recognition system greatly depends on the appropriate representation of human face features and the good design of classifier .

In our experiment we apply DBN classifier on ORL face Database for face recognition, and we see how it deals with this dataset. There are 40 distinct persons. each person has ten different images labeled with the number 1,2,…..,10, and The size of each image is 112 * 92 pixels.

In this experience, deep belief networks composed of RBMs (DBN) are applied to classify the ORL data face without any handcart feature step. Each data point has 112*92 = 10304 pixels. As there are only 120 training examples, initial experiments returned impractical success rates of 99.98%.

Subsequently, the input is normalized from integer values in [0; 255] to real values in [0; 1] (according to the class of RBM layers) and used throughout the following experiments. Similar to the model selection for MNIST in above, for both training methods, a number of parameters are optimized independently. **Table 3.4** contains the different values of parameters that are tested on both training methods throughout pre-training and fine-tuning. The parameters are also optimized independently for computational performance reasons. During the optimization, the other parameters are set to the default values in **Table 3.4**.

The same procedure is followed for the other parameters with few epochs for both pre-training and fine-tuning. The best contribution has an success rate of 99.977%.

Finally, the optimal values are put together for both training methods in order to train two optimized classifiers.

Training method	*CD (Contrastive Divergence)*
Number of input nodes	10304
Number of hidden layers	5
Number of hidden layer nodes per layer	500 500 2000
Number of output nodes	Determined by dataset (ORL: 40)
Maximum number of epochs	500
Mini batches for stochastic gradient decent	25
batch size	30
momentum	0.5
RBM learning rate	0.01

Table 3.4 Parameters selection for DBN training.

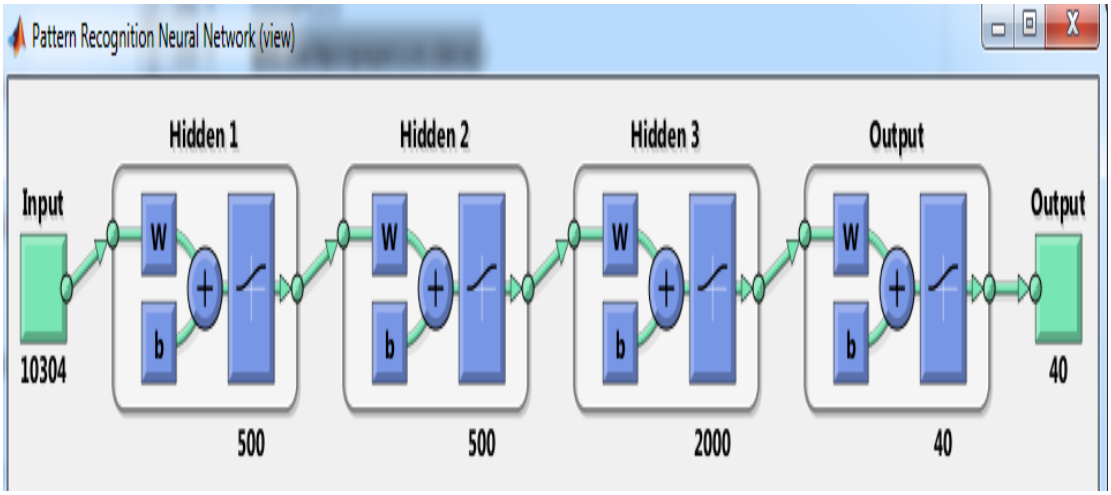


Figure3.5 Architecture of DBN on ORL dataset

Sampling methods

Table3.4 shows a classification experiment using ORL dataset. This table compares different sampling method types that has been implemented.

Learning methods	Gibbs sampling	CD	PCD	FEPCD
Test Error	0.0250	0.0050	0.0250	0.0276

Table 3.5 Classification error on ORL dataset for a DBN

DBN can obtain good features with acceptable discrimination between them. Note that these features has been learnt without using their labels. **Figure3.6** shows extracted features in a DBN on MNIST dataset. The features produced by a 10304-500-500-2000- DBN that maps input data (10304features)

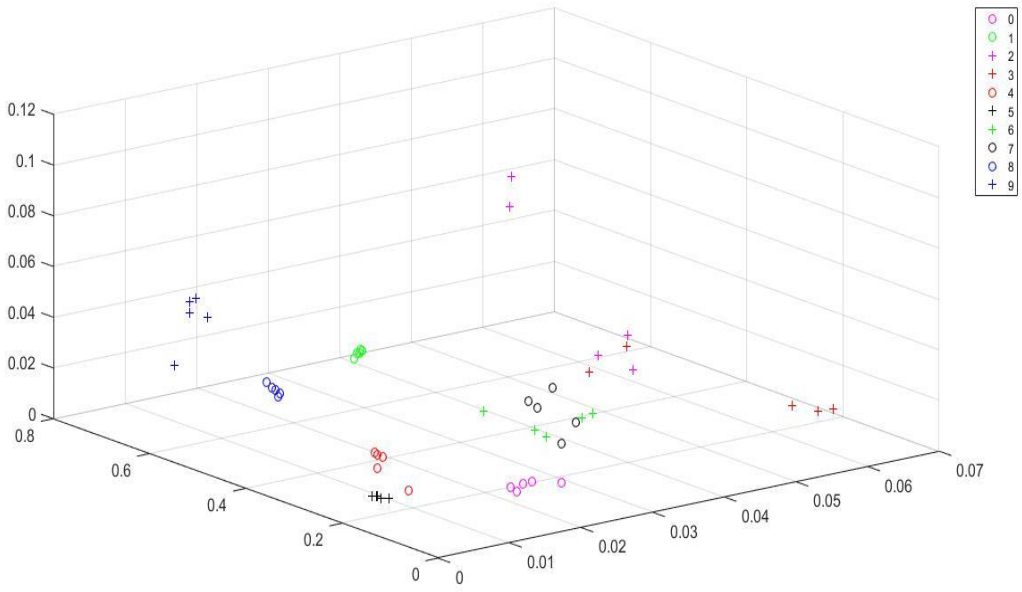


Figure3.6 Extracted features from a DBN on MNIST dataset

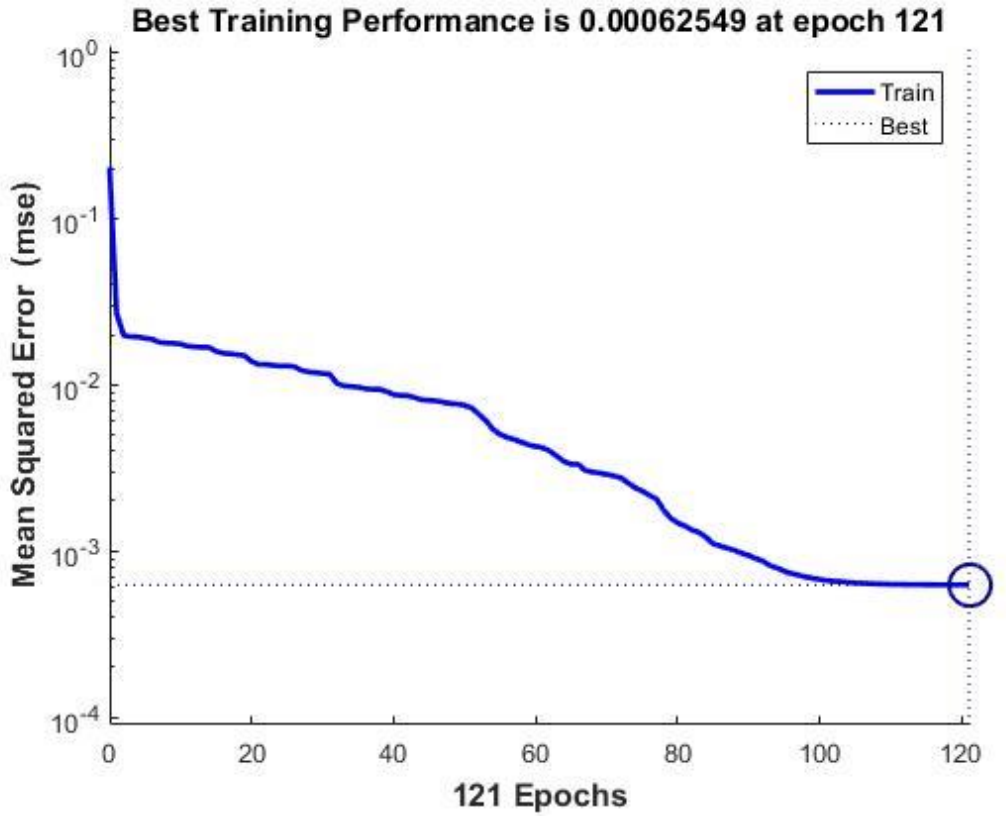
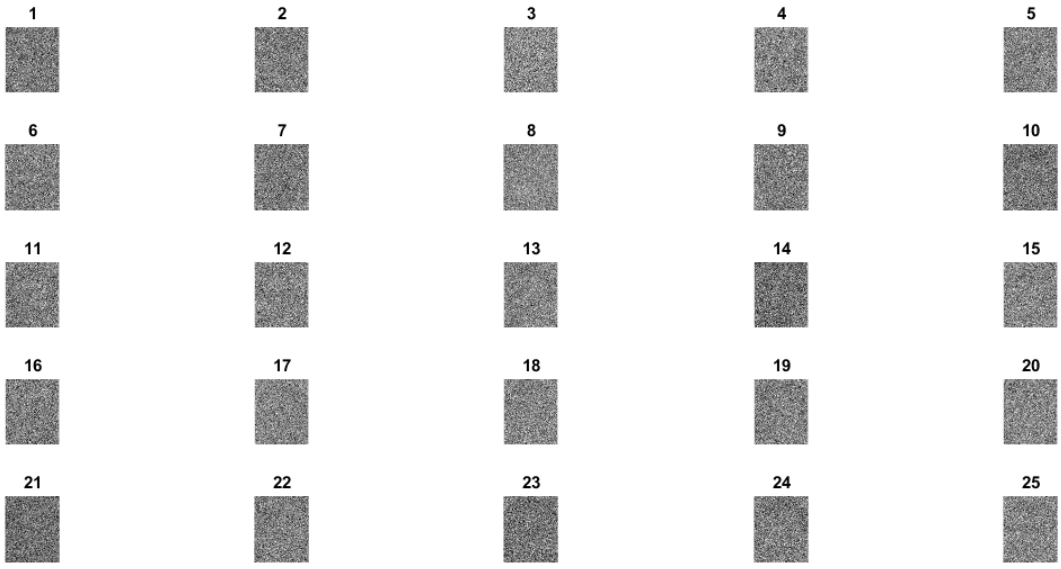
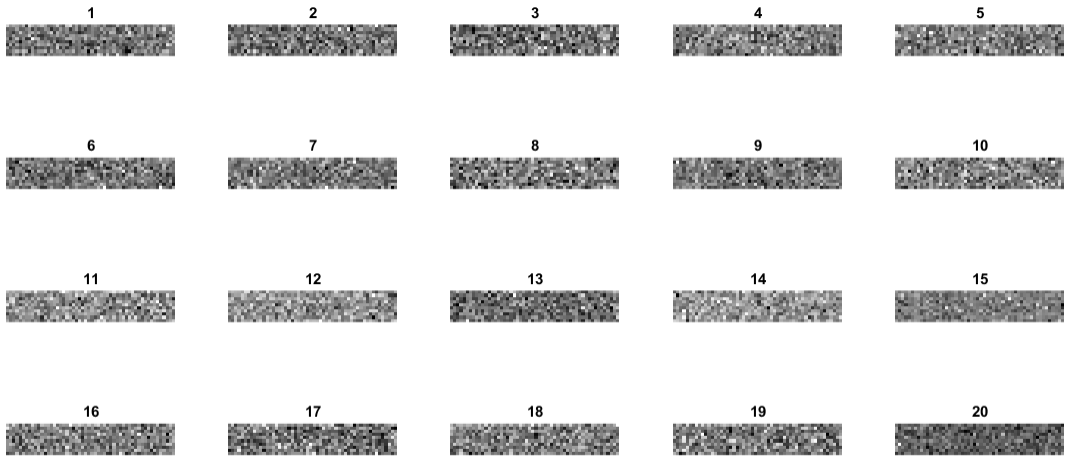


Figure3.7 precision and MSE for the model

-



'a'



'b'

Figure3.8 The learned weights ('a': between layers one and 'b' between layers two and three) for the DBN mode on the ORL database.

The proposed method was also compared to other existing methods in the literature, with ORL database is used for this purpose. Table shows that the proposed method significantly outperforms the other previous methods in terms of performance and robustness.

Approach	**a. [35]**	**b. [36]**	**c. [33]**	**d. [34]**	**e. Our work**
Recognition Rate	97.15	94.54	85.00	98.00	99.98

Table3.6 Comparison recognition rate for other existing approach.

Through extensive experiments over two publically available real datasets and with 4 sampling methods (Gibbs, CD, PCD, FEPCD), we investigated the use of DBN algorithm. We did show that the ability of DBN to reach up the higher accuracy on the both dataset.

Having add the Dropout in DBN. This has been shown to bit enhance the rate accuracy and to have reducing the computation cost than the standard version DBN .

As a side effect, the Dropout rule can eliminate a number of input features and in this manner be used as a feature selection method.

In such avoiding overfitting, training algorithm could be enhances. Moreover, we provided a more efficient tool for DBN classification. Our study shows that DBN is both robust and efficient. **Figure 3.3** shows the DBN, which allows us to classify the face image with neither preprocessing step and nor feature extraction step .

5. Conclusion

As shown in the experiments, deep neural network training methods are powerful and achieve very high classification rates in different computer vision problems. Nonetheless, deep learning is not a concept that can be used out of the box, as proper pre-processing of the data and extensive model selection are necessary.

Dropout is a technique for improving neural networks by reducing overfitting. The main idea is to further learn the features through the hidden units. Dropout improves performance of neural nets in a wide variety of application domains including object classification, digit recognition, speech recognition, document classification and analysis of bio-medical data.

It would be interesting to design more specify architecture network to the big data, such as sparse coding or cloud computing. Furthermore, Deep neural networks have been reported in the literature to perform very well on computer vision problems, which would be interesting to study, too. In order to speed up learning, use of GPUs has been successfully used in the literature. These methods also look promising to be applied to the problems covered in this work (face recognition).

CONCLUSION AND PERSPECTIVES

CONCLUSION AND PERSPECTIVES

This research investigated Deep Learning techniques under different attacking conditions in order to find the best parameters that can obtain the lower error rate. The experiments started with training and testing deep learning sampling methods using MNIST and ORL datasets. The error rate of testing deep learning algorithm over toy example datasets showed that PCD was the best sampling method, it gave the lawer error among other method sampling.

The powerful key behind the successfully of deep learning in many tasks is that its ability to learn and extract the data without preprocessing or feature extraction step. Indeed, in this work, we have made an experience to demonstrating these propriety on the ORL dataset.

We have made many experiences on the MINST dataset to show the effect of dropout technique that is applied for each RBM layer on the performance of DBN network. despite its performance, the dropout technique is considered not sufficient to reduce the over fitting phenomena and to have reducing the computation complexity DBN(the hypermeter configuration DBN).

For future work, we propose to apply the evolution algorithm such genetic, pso, …for optimize the parameters of the DBN method such as the number of hidden units, the number of epochs, and the learning rates, which would reduce the error rate and the network training time of complex task.

Bibliography

[1] ssH. Özcan, “Deep Learning Practices in Very Low-Resolution Face Images,” Master thesis, Naval War College, Turkey, (2014)

[2] Shan Suthaharan ,Machine Learning Models and Algorithms for Big Data Classification Thinking with Examples for Effective Learning 123,

[3] Stephanie Thomas, Michael Paluszek, Matlab Machine Learning

[4] https://medium.com/deep-math-machine-learning-ai/different-types-of-machine-learning-and-their-types-34760b9128a2 visited(25/42018)

[5] Taiwo Oladipupo Ayodele, Types of Machine Learning Algorithms, University of Portsmouth United Kingdom .

[6] N.E Sahla , A Deep Learning prediction model for object classification .

[8] Oliver Kramer ,Machine Learning for Evolution Strategies .

[9] Bernardete Ribeiro ,Machine Learning for Adaptive Many-Core Machines – A Practical Approach ,Noel Lopes

[10] Bottou, L. and LeCun. Y. “Large scale online learning,” Proc. NIPS, 2004.

[11] https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/ visited(25/4/2018)

[12] http://www.saedsayad.com/support_vector_machine.htm visited(28/4/2018)

[13] Fiona Nielsen, Neural Networks – algorithms and applications , 4i 12/12-2001

[14] Liliana Perescu-Popescus3,Nikos Mastorakis4, Multilayer Perceptron and Neural Networks, Marius-Constantin Popescu1 Valentina E.Balas2

[15] Mohit Joshi Rol, Implementation of an integrated artificial neural network trained with back-propagation algorithm

[16] Christopher M. Bishop: Pattern Recognition and Machine Learning. Springer. 2007.

[17] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.

[18] N Srivastava, G Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov Dropout: A simple way to prevent neural networks from overfitting ,The Journal of Machine Learning Research 15 (1), 1929-1958

[19] N Srivastava, Improving neural networks with dropout, University of Toronto182

[20] Martens, J. "Deep learning with Hessian-free optimization," Proc. ICML, 2010.

[21] Vinyals O., and Ravuri, S. "Comparing multilayer perceptron to deep belief network tandem features for robust ASR," Proc. ICASSP, 2011.

[22] Hinton, G. and Salakhutdinov, R. "Reducing the dimensionality of data with neural networks," Science, vol. 313. no. 5786, pp. 504 - 507, July 2006.

[23]https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks visited(2/5/2018)

[24] Taweh Beysolow II ,Introduction to deep learning using R

[25] Noel Lopes , Bernardete Ribeiro ,Machine Learning for Adaptive Many-Core Machines – A Practical Approach

[26] Hinton, G.E.: A practical guide to training restricted Boltzmann machines. Technical report, Department of Computer Science, University of Toronto (2010)

[27] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," in Artificial Intelligence and Statistics, 2005, vol. 2005, p. 17.

[28] Dahl, G., Yu, D., Deng, L., and Acero, A. "Context-dependent DBN-HMMs in large vocabulary continuous speech recognition," Proc. ICASSP, 2011.

Mohamed, A., Dahl, G. and Hinton, G. "Acoustic Modeling Using Deep Belief Networks", IEEE Trans. Audio, Speech, & Language Proc. Vol. 20 (1), January 2012.

[29] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., "Deep Neural Networks for Acoustic Modeling in Speech Recognition," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, November 2012.

[30] Melanie Mitchell: An Introduction to Genetic Algorithms. M.I.T. Press. 1996.

[31] Nitish Srivastava, Geo_rey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. The Journal of Machine Learning Research, 15, 1929-1958, 2014.

[32] Akshita Tyagi, Content Based Spam Classification- A Deep Learning Approach, December, 2016 .

[33] Tang, Xiutao , Qu, Cuilu, 2010, Facial image recognition based on fractal image encoding, Bell Labs Technical Journal Alcatel-Lucent, Volume: 15 Issue: 1 pp. 209-2014 ISNN 1089-7089.

[34] Temdee P, Khawparisuth D, Chamnongthai K, 1999, Face recognition by using fractal encoding and back propagation neural network, International Symposium on Signal Processing and its Applications, Brisbane, Australia .

[35] Yousra Ben Jemaa, Ahmed Derbel and Ahmed Ben Jmaa, 2011, 2DPCA fractal features and genetic algorithm for efficient face representation and recognition, EURASIP Journal on Information Security

[36] Ebrahimpour-Komleh, Hossein, 2001, Face recognition using fractal codes, In Proceedings of International Conference on Image Processing, IEEE, Thessaloniki, Greece.

Abstract

The work presented in this report is located in the pattern recognition area, and the specific motivation in this work is to study the manner of training , their architectures, techniques, and methods. Our system is based on Restricted Boltzmann Machines and specifically those who belong to the deep learning. We detail the systems based on Deep belief Networks used for data learning, this approach offers a distinctive architecture that is used in the experimental section to train two databases so that our neural network can recognize the data proposed to the network input.

Keywords : Deep Learning, Neural Network, Classification, Restricted Boltzmann Machines, Deep Belief Network

Résumé

Le travail présenté dans ce rapport se situe dans le domaine de reconnaissance de la forme, et la motivation particulière dans celui-ci est d'étudier les systèmes d'apprentissage de données, leurs architectures, techniques, et méthodes. Notre système s'appuie sur les machines de Boltzmann restreinte et précisément ceux qui font partis d'apprentissage profond. Nous détaillons les systèmes basés sur les réseaux du Boltzmann empiles restreints pour l'extraction des caractéristiques, à savoir, les chiffres et les visages. Cette approche propose une architecture particulière qu'on vient d'utiliser dans la partie expérimentale pour entrainer nos bases de données afin que notre réseau de neurone arrive à reconnaitre les données proposées à l'entrée du réseau.

Mots clés : l'apprentissage profond, Réseaux de neurons, classification, Restricted Boltzmann Machines (RBM), Deep Croyance Networks (DBN).

ملخص

يقع العمل المقدم في هذا التقرير في منطقة التعرف على الأنماط ، وتتمثل الدوافع المحددة في هذا العمل في دراسة طريقة التدريب وهندستها وتقنياتها وأساليبها. يعتمد نظامنا على ماكينات بولتزمان المقيدة وعلى وجه التحديد أولئك الذين ينتمون إلى التعلم العميق. نحن نقوم بتفصيل الأنظمة التي تعتمد على شبكات المعتقد العميق المستخدمة في تعلم البيانات ، حيث يقدم هذا المنهج بنية مميزة يتم استخدامها في القسم التجريبي لتدريب قاعدتي بيانات بحيث تتمكن شبكتنا العصبية من التعرف على البيانات المقترحة لمدخلات الشبكة.

الكلمات المفتاحية: التعلم العميق,الشبكات العصبونية, التصنيف, ماكينات بولتزمان المقيدة, وشبكات المعتقدة العميقة.