UNIVERSITY OF UDINE - ITALY

Department of Mathematics and Computer Science

Ph.D. Thesis

INSIGHTS INTO MACHINE LEARNING: DATA CLUSTERING AND CLASSIFICATION ALGORITHMS FOR ASTROPHYSICAL EXPERIMENTS

Supervisors:
Prof. ALESSANDRO DE ANGELIS

Prof. GIAN LUCA FORESTI

Candidate:
PRAVEEN BOINEE

Doctorate of Philosophy in Computer Science

XVIII cycle

AY 2005/2006

Abstract

Data analysis domain dealing with data exploration, clustering and classification is an important problem in many experiments of astrophysics, computer vision, bioinformatics etc. The field of machine learning is increasingly becoming popular for performing these tasks. In this thesis we deal with machine learning models based on unsupervised and supervised learning algorithms.

In unsupervised learning category, we deal with Self-Organizing Map (SOM) with new kernel function. The data visualization/exploration and clustering capabilities of SOM are experimented with real world data set problems for finding groups in data (cluster discovery) and visualisation of these clusters.

Next we discuss ensembling learning, a specialized technique within the supervised learning field. Ensemble learning algorithms such as AdaBoost and Bagging have been in active research and shown improvements in classification results for several benchmarking data sets. They grow multiple learner algorithms and combine them for getting better accuracy results. Generally decision trees learning algorithm is used as base classifiers to grow these ensembles. In this thesis we experiment with Random Forests (RF) and Back-Propagation Neural Networks (BPNN) as base classifiers for making ensembles. Random Forests is a recent development in tree based classifiers and quickly proven to be one of the most important algorithms in the machine learning literature. It has shown robust and improved results of classifications on standard data sets. We experiment the working of the ensembles of random forests on the standard data sets available in University of California Irvine (UCI) data base. We compare the original random forest algorithm with their ensemble counterparts and discuss the results.

Finally we deal the problem of image data classification with both supervised (ensemble) learning and unsupervised learning. We apply the algorithms developed in the thesis for this task. These image data are taken from the MAGIC telescope experiment, which collects the images of particle rays coming from the outer universe. We apply the ensembles of RF, BPNN for making a supervised classification of images and compare the performance results. Then we discuss a SOM system, developed for making an automatic classification of images using the unsupervised techniques.

Acknowledgments

I sincerely thank Prof. Alessandro De Angelis for being so friendly guide and giving me lot of opportunities to see many beautiful places in Italy and Europe on my research work. He is a professor with different attitude who is always available, makes constructive remarks and makes things easy and fun. He is more a friend than a professor. Working with him was very motivating. Many thanks to Prof. Gian Luca Foresti for his gigantic patience for listening to me and correcting me in many ways.

Special thanks to Udine University Chancellor Prof. Furio Honsell and Birla Science Center Director Dr. Sidharth in India, because of whom the whole PhD of mine has came into existence.

During my PhD I got lot of suggestions and help from many people. I take this opportunity to thank them. Not in any particular order, I am very much greatful to Marco Frailis, Trieste Longo (Francesco), Abelardo Moralejo, Riccardo Giannitrapani, Fabrizio Barbarino, Oriana Mansutti , Barbara De Lotto, Massimo Pin, Alberto Forti, Reddy, Tiziano, Erica. I am greatful to Alessia Degano for helping me in many titanic burocratic issues.

I had a very special experience with my friends in the multinational - multi cultural, multi-lingual apartment. Many many thanks to my very special spain friends Maria, Cristina, Edurne (for being so beautiful and making me so mad!), Andrea, Javier, Carlos.

Also I am thankful to other Indian students who are also fighting with their PhD's. My friendship with Subbu has gone upto Mt.Blanc levels especially in the last months of my work. Also I am thankful to Reddy, Vijay, Rekha and beautiful Nirmala.

Many thanks to my family who were always with me both in the happy and tough times.

Last but not least a very special acknowledgment to GOOGLE without which it is almost impossible to do this job.

Contents

List of Figures

List of Tables

Introduction

Nature has designed us as an advanced neural- cognitive system to learn, recognize and make decisions. The ease, with which we recognize a face [especially of an opposite sex], learn and understand the spoken languages and several other daily life decision makings are astoundingly complex tasks. These are internally performed by a sophisticated coordination between brain, eye, ear, hands, legs etc. Humans learn from experience, by mistakes or successes. By human nature, it is quite natural that we seek to design and build systems that can also learn to perform these learning, recognition and decision making tasks.

The field of machine learning is concerned with designing machines to learn from experience, learn from examples to make decisions. It deals with the formalization of human learning and decision making. Thus the aim of the machine learning is to automate the process of solving problems, and thereby machine learning is rival to what humans have considered their domain. It has a long history in engineering and science applications. It is quite interdisciplinary and has roots in many disciplines ranging from statistics, biology, philosophy to computational theory, artificial intelligence, data mining, cognitive science etc. It is also used interchangeably as pattern recognition and is used to describe the wide range of problems such as recognition, classification, grouping of patterns. These problems are important in variety of applications such as Signal processing, Physics experiments, Computer vision, Bioinformatics, Genomics, Remote sensing, Financial forecasting etc [1].

After seeing table 1 we can start thinking that our world is now a data-driven one. We are surrounded by very large collections of data from these applications, sometimes with hundreds of millions of individual records stored in centralized data warehouses. This ever growing data requires to be classified or clustered to get analyzed. Machine learning algorithms are proved to be powerful, comprehensive methods used for this purpose. It is an important data mining step which assists in converting the data into information that informs, instructs, answers, or otherwise aids understanding and decision-making. They learn by searching through an n-dimensional space of a given data set to find an acceptable generalization to represent the analysis or knowledge from the data.

In recent years there has been a significant growth of methods in the machine learning field for learning from raw data to classify/cluster them. The proliferation of low-cost computers (for software implementation), low-cost sensors, communications, database technology (to collect and store data), and computer-literature application experts who can pose “interesting” and “useful” application problems, all have contributed for its development.

Problem domain	**Application**	**Input Pattern**	**Pattern Classes**
Physics experiments[Astrophysical, high-energy]	Image classification, Particle separation	Atrributes [image/particle]	Classes of images/particles
Bio-informatics	Sequence analysis	DNA/Protein sequence	Known types of genes/patterns
Data Mining	Searching for meaningful patterns	Points in multidimensional space	Compact and well separated clusters
Document classification	Internet search	Text document	Semantic categories (e.g. business, sports, etc.)
Document image analysis	Reading machine for blind	document image	Alphanumeric characters, words
Industrial automation	Printed circuit board inspection	Intensity or range image	Defective /non-defective nature of product
Multimedia database retrieval	Internet search	Video clip	Video genres (e.g. action, dialogue, etc.)
Biometric recognition	Personal identification	Face, iris, fingerprint	Authorised users for access control
Remote sensing	Forecasting crop yield	Multispectral image	Land use categories, growth pattern of crops
Speech recognition	Telephone directory enquiry without operator assistance	Speech waveform	Spoken words

Table 1: Data explosion centers [2]

Research Objectives

The research aims at developing new machine learning algorithms for data clustering and classification. The developed algorithms are applied for data analysis tasks in MAGIC telescope experiment. It is an international effort with 17 institutes. The telescope collects image data at the rate of 432GB per day. At the end of year we acquired with $\tilde{3}$7800GB of data to be analyzed. The task is to classify the images based on their properties. The work is performed by both supervised and unsupervised classification techniques. Also the algorithms are applied to Gamma ray burst data sets, UCI [1]data sets.

The thesis is written in three parts. First we discuss Self-Organizing Maps (SOM) based on unsupervised learning. New kernel neighborhood functions are proposed and applied to the algorithm. A case study on Gamma Ray Burst (GRB) analysis has performed. In this experiment we use SOM for data exploration, visualsation and cluster discovery in the GRB data sets.

In the second part the work is carried out with supervised classification algorithms, especially of ensemble learning. The ensembles are applied to standard algorithms such as random forests and Back propagation neural nets. The ensembles of random forests are applied on standard UCI data sets to study the performance results compared to the original random forests algorithm.

Finally in the third part, we use the proposed algorithms for image classification in MAGIC. The image classification has done with supervised techniques by using the ensembles of random forest and back propagation neural nets. Then we apply the self organizing map for the classification of images in an automatic way.

Thesis Organization

Chapter 1 discusses the machine learning system in general and introduces various learning modes and algorithms. Chapter 2 discusses the unsupervised learning giving special emphasis to partition clustering algorithms, the basis for SOM. Also various cluster distance measurements are discussed . In chapter 3 Self-organizing maps are discussed with various kernel neighborhood functions. The clustering (quantization) ability of SOM along with its visualization techniques such as U-matrix and component plane visualization are discussed. In this chapter we experiment with gamma ray burst analysis using SOM. Next in chapter 4 ensemble learning techniques are discussed. We especially concentrate on bagging and boosting techniques. In chapter 5 we discuss tree classifier “Random forests” and discuss the making of its ensembles with bagging and boosting. These algorithms are applied on UCI data sets. The results and comparative study are discussed with the original random forests algorithm. In chapter 6 MAGIC telescope project is introduced. We discuss the imaging technique used in MAGIC for image collection and image processing. We apply the

[1]University of California Irvine maintains the international machine learning database repository, an archive of over 100 databases used specifically for evaluating machine learning algorithms.

supervised and unsupervised algorithms that are proposed in the thesis for the image data classification for the MAGIC.

Thesis Contributions

- P. Boinee et.al, *Meta Random Forests*, In International Journal for Computational Intelligence, Volume-II, pages: 138-147 (2005)
- P. Boinee et.al, *Ensembling Classifiers: An application to image data classification from chernkov telescope experiment.* In Proceedings of International Conference on Signal Processing, Enformatika series, Prague (2005)
- P. Boinee et. al, *Self-Organising Networks for Classification: developing Applications to Science Analysis for Astroparticle Physics*, CoRR cs.NE/0402014: (2004)
- P. Boinee et al., *Neural Networks for MAGIC data analysis*, In proceedings of Sixth International Conference on Foundations of Fundamental and Computational Physics, Udine, Italy (2004)
- P. Boinee et al., *Automatic Classification using Self-Organizing Neural Networks in Astrophysical Experiments*, In Proc. of Science with the New Generation of High Energy Gamma-ray Experiments, Perugia, Italy, (2003)

1 Machine Learning System

Ever since computers were invented, we have wondered whether they might be made to learn. If we could understand how to program them to learn-to improve automatically with experience-the impact would be dramatic. The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience. According to [3], *Machine learning is defined as a computer program which learns a problem from experience E with respect to some class of tasks T and performance measure P. The performance at tasks in T, as measured by P, could improve with experience E.*

We do not yet know how to make computers learn nearly as well as people learn. However, algorithms have been invented that are effective for certain types of learning tasks, and a theoretical understanding of learning is beginning to emerge. Many practical computer programs have been developed to exhibit useful types of learning, and significant commercial applications have begun to appear.

It is possible to relate the problem of learning to the general notion of inference in classical philosophy. Every predictive-learning process consists of two main phases:

- Learning or estimating unknown dependencies in the system from a given set of samples,
- Using estimated dependencies to predict new outputs for future input values of the system.

These two steps correspond to the two classical types of inference known as induction (progressing from particular cases-training data-to a general mapping or model) and deduction (progressing from a general model and given input values to particular cases of output values). These processes may be described, formalized, and implemented using different learning methods [4].

A learning method (essentially an algorithm usually implemented in software) estimates an unknown mapping (dependency) between a system's inputs and outputs from the available data set, namely, from known samples. Once such a dependency has been accurately estimated, it can be used to predict the future outputs of the system from the known input values. Learning from data has been traditionally explored in such diverse fields as statistics, engineering, and computer science. In this chapter

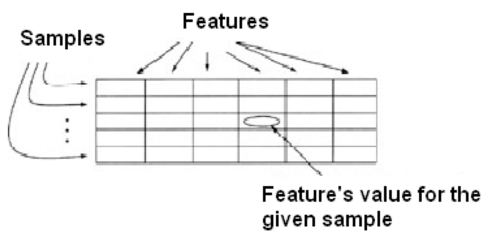


Figure 1.1: Tabular representation of a data set

we discuss the design of the general machine learning system and introduce various learning modes and algorithms.

1.1 System Design

Machine learning system design have been inspired by the learning capabilities of biological systems and, in particular, those of humans. In fact, biological systems learn to cope with the unknown, statistical nature of the environment in a data-driven fashion. Babies are not aware of the laws of mechanics when they learn how to walk, and most adults drive a car without knowledge of the underlying laws of physics. People are not born with such capabilities, but learn them through data-driven interaction with the environment.

Machine learning is also data-driven. The input for the system is taken from a data base, which in general a collection of data sets. Each data set is again a collection of data vectors also called as "*patterns*" representing a collection of measurements called as "*features*" of the learning problem. Thus a pattern can be viewed as a representation of set of n features in the form of a $n-$dimensional vector. These patterns can have optional label, which tells the type of class, a pattern belongs to. Labeling of patterns depends on the type of learning used by the system in the training mode. Supervised learning algorithms uses labeled data sets as input, whereas the unsupervised learning algorithms directly operates on unlabeled patterns to adopt to the underlying data distribution of the patterns.

The data set represents the standard model of structured data with the features uniformly measured over many cases. Usually the pattern vectors in the data set are represented in a tabular form, or in the form of a single relation (term used in relational databases), where columns are features of objects stored in a table and rows are values of these features for specific entities. A simplified graphical representation of a data set and its characteristics is given in Figure 1.1. Many different types of features (attributes or variables)-i.e., fields-in structured data records are common in machine learning.

A machine learning system can be operated in two modes: training and testing

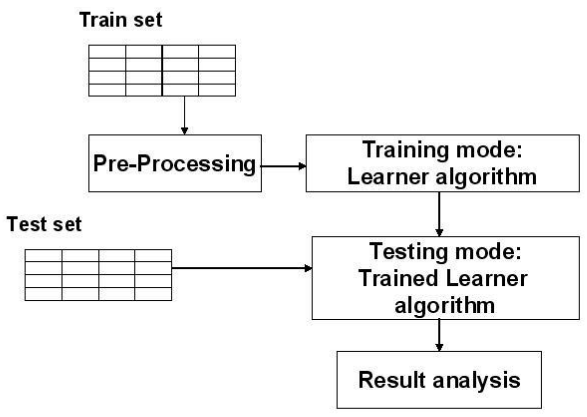

Train set
Pre-Processing
Training mode: Learner algorithm
Test set
Testing mode: Trained Learner algorithm
Result analysis

Figure 1.2: Model for Machine learning System

(figure 1.2). The preprocessing module essentially aims at the feature extraction/selection. It finds the appropriate features for representing the input patterns by segmenting the pattern of interest from the background, removing noise, normalizing the pattern, and any other operation which will contribute in defining a compact representation of the pattern [1].

In the training module, a learner is trained by subjecting each pattern to it. It essentially learns the underlying data distribution in the data set also called as the training set. The decision making process by a learner in bayesian terms can be summarized as follows: A given pattern is to be assigned to one of the k categories $C_1, \ldots, C_k$ based on a vector of n feature values $x = (x_1, \ldots, x_n)$. The features are assumed to have a probability density mass function conditioned on pattern class. thus a pattern vector x belonging to class C_i is viewed as an observation drawn randomly from the class-conditional probability function $p\left(x|C_i\right)$. Now the task is to find the decision boundary that exists among the various class in the input space. A number of well defined decision rules exists in the literature for this purpose. They include bayes decision rule, maximum likelihood rule (a special case of bayes decision rule), and the Neyman-Pearson rule [1].

One the learner is trained, we enter into a testing phase. Typically the system is provided with a test set of patterns. A trained learner tries to form generalizations from particular, true facts, from the training data set [4]. These generalizations are formalized as a set of functions that approximate a system's behavior. The learning machine is capable of implementing a set of functions $f\left(x, w\right)$, $w \in W$, where x is an input, w is a parameter of the function, and W is a set of abstract parameters used only to index the set of functions. Ideally, the choice of a set of approximating functions reflects a priori knowledge, about the system and its unknown dependencies. However, in practice, because of the complex and often informal nature of a priori knowledge, specifying such approximating functions may be, in many cases difficult.

To explain the selection of approximating functions, we can use a graphical interpretation of the inductive-learning process. The task of inductive inference is this: given a collection of samples $(x_i, f(x_i))$, return a function $h(x)$ that approximates $f(x)$. The function $h(x)$ is often called a hypothesis.

The task of learning machine is to select a function from the set of functions, which best approximates the system's responses. The learning machine is limited to observing a finite number of samples n in order to make this selection. The finite number of samples, which we call a training data set, is denoted by (x_i, y_i), where $i = 1, \ldots, n$. The quality of an approximation produced by the learning machine is measured by the loss function $L(y, f(x, w))$ where

- y is the output produced by the system,
- x is a set of inputs,
- $f(x, w)$ is the output produced by the learning machine for a selected approximating function, and
- w is the set of parameters in the approximating functions.

L measures the difference between the outputs produced by the system y_i and that produced by the learning machine $f(x_i, w)$ for every input point x_i. By convention, the loss function is nonnegative, so that large positive values correspond to poor approximation and small positive values close to zero show a good approximation. The expected value of the loss is called the risk functional $R(w)$

$$R(w) = \int \int L(y, f(x, w))\, p(x, y)\, dxdy$$

where $L(y, f(x, w))$is a loss function and $p(x, y)$ is a probability distribution of samples. The $R(w)$ value, for a selected approximating functions, is dependent only on a set of parameters w. Inductive learning can be now defined as the process of estimating the function $f(x, w_{opt})$, which minimizes the risk functional $R(w)$ over the set of functions supported by the learning machine, using only the training data set, and not knowing the probability distribution $p(x, y)$. With finite data, we cannot expect to find $f(x, w_{opt})$ exactly, so we denote $f(x, w_{opt^*})$ as the estimate of parameters w_{opt^*} of the optimal solution w_{opt} obtained with finite training data using some learning procedure.

In a two-class classification problem, where the output of the system takes on only two symbolic values, $y = \{0, 1\}$, corresponding to the two classes, a commonly used loss function measures the classification error.

$$L(y, f(x, w)) = \begin{cases} 0 & ify = f(x, w) \\ 1 & ify \neq f(x, w) \end{cases}$$

Maximum accuracy will be obtained by minimizing the risk functional because, in that case, the approximating function will describe the best set of given samples. The training of the learning machine can be interpreted in terms of "*inductive principle*".

An inductive principle is a general prescription for obtaining an estimate $f\left(x, w_{opt^*}\right)$ in the class of approximating functions from the available finite training data. An inductive principle tells us what to do with the data, whereas the learning method specifies how to obtain an estimate. Hence a learning method or learning algorithm is a constructive implementation of an inductive principle. For a given inductive principle, there are many learning methods corresponding to a different set of functions of a learning machine. The important issue here is to choose the candidate models (approximating functions of a learning machine) of the right complexity to describe the training data.

The result analysis is governed by many factors such as : learning mode, learner algorithm, analysis requirements in the problem domain. For a supervised learning mode typically a learning algorithm outputs the probability of class assignment for each pattern in the test row. These probabilities are inturn used to perform the result analysis like classification accuracies rates (or error rates) of the learner. In Unsupervised learning mode the learner algorithm can be used to group the data set into clusters. These clusters can be analyzed by various cluster analysis techniques (chapter 2). Also some specialized unsupervised learning algorithms such as Self-Organizing Maps (SOM) can be used for cluster visualization, data exploration tasks (Chapter 3).

1.2 Learning modes

In practice, the choice of a learner algorithm is a difficult task and driven by the learning problem at the hand. The learning modes can be broadly classified in to three types depending on the availability of a labeled training set.

Unsupervised learning

Unsupervised classification is a branch of machine learning designed to find natural groupings, or clusters, in multidimensional data, based on measured or perceived similarities among the patterns. It is used in a wide variety of fields under a wide variety of names, the most common of which are cluster analysis. It has no explicit teacher, and the system forms [26] clusters or “natural groupings” of the input patterns. In conceptual terms, we may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples. The environment with its characteristics and model is, however, unknown to the learning system. It operates on unlabeled data sets to discover the natural groups in the data set. “Natural” is always defined explicitly or implicitly in the clustering system itself, and given a particular set of patterns or cost function; different clustering algorithms lead to different clusters. Often the user will set the hypothesized number of different clusters ahead of time. It is often suitable for real world problems as it is difficult to obtain the labeled data sets. We discuss more on unsupervised learning in chapter 2.

Supervised learning

Supervised learning is used to estimate an unknown dependency from known input-output samples. It assumes the existence of a teacher-fitness function or some other external method of estimating the proposed model. The term “supervised” denotes that the output values for training samples are known (i.e., provided by a “teacher”) [7]. In conceptual terms, we may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples. The environment with its characteristics and model is, however, unknown to the learning system. The parameters of the learning system are adjusted under the combined influence of the training samples and the error signal. The error signal is defined as the difference between the desired response and the actual response of the learning system.

Supervised learning algorithms work by searching through a space of possible functions, called hypotheses, to find a function, h, that is the best approximation to the unknown function f. To determine which hypothesis h is best, a learning algorithm can measure how well h matches f on the training data points, and it can also assess how consistent h is with any available prior knowledge about the problem.

As a performance measure for the system, we may think in terms of the mean-square error or the sum of squared errors over the training samples. This function may be interpreted as a multidimensional error surface, with tile free parameters of the learning system as coordinate. Any learning operation under supervision is represented as a movement of a point on the error surface. For the system to improve the performance over time and therefore learn from the teacher, the operating point on an error surface has to move down successively toward a minimum of the surface [2]. An adequate set of input-output samples will move the operating point toward the minimum, and a supervised learning system will be able to perform such tasks as pattern classification and function approximation. Different techniques support this kind of learning, and some of them such as neural networks, support vector machines, decision trees etc. Supervised learning can be applied to many problems like signal processing, image classification, handwriting recognition, medical diagnosis, and part-of-speech tagging in language processing [3].

One of the most active areas of research in supervised learning has been to study methods for constructing good ensembles of classifiers. Ensemble learning algorithms work by running a “base learning algorithm” multiple times and forming a vote out of the resulting hypotheses [62]. This area is referred to by different names in the literature - committees of learners, mixtures of experts, classifier ensembles, multiple classifier systems. In general, an ensemble method is used to improve on the accuracy of a given learning algorithm. Practically it has proven that combining the predictions of ensembles of classifiers produces more accurate predictions than the single classifiers.

Ensembles or committee machines can be useful in many ways. The committee might exhibit a test set performance unobtainable by an individual committee member on its own. The reason for this is that the errors of the individual committee members cancel out to some degree when their predictions are combined. The surprising discovery of this line of research is that even if the committee members were trained on data derived from the same data set, the predictions of the individual committee members might be sufficiently different such that this combining process is beneficial in improving the classification accuracy. Ensemble learning is described in more detail in chapter 4.

Reinforcement learning

The most typical way to train a classifier is to present an input, compute its tentative category label, and use the known target category label to improve the classifier. In reinforcement learning or learning with a critic, no desired category signal is given; instead, the only teaching feedback is that the tentative category is right or wrong. This is analogous to a critic who merely states that something is right or wrong, but does not say specifically how it is wrong [26]. (Thus only binary feedback is given to the classifier; reinforcement learning also describes the case where a single scalar signal, say some number between 0 and 1, is given by the teacher.) In pattern classification, it is most common that such reinforcement is binary either the tentative decision is correct or it is not.

1.3 Some learning algorithms

In this section, we discuss some standard learning algorithms in the machine learning literature. Here we mainly make the study based on learning modes, especially of supervised and unsupervised. In supervised we deal with 4 types of models. First we discuss Back-propagation neural network (BPNN), a standard algorithm available under neural network category. In this thesis we develop the ensembles of BPNN for image data classification for MAGIC telescope. Next we discuss k-nearest neighbor algorithm (K-NN) which performs an instant learning from pattern by pattern in the training set. Then we discuss decision trees a tree based classifier, which forms the basis for making random forests a specialized trees based classifier. Finally we discuss kernel based learners with support vector machines as an example.

Under unsupervised technique there exists a variety of algorithms like Self-Organising Maps (SOM) [33], Fixed weight competitive nets [15], Adaptive resonance theory models [19]. Here we concentrate on SOM models that are discussed in detail in Chapter 3.

1.3.1 Back-Propagation Neural Network (BPNN)

Neural networks has great advantages of adaptability, flexibility, and universal nonlinear functional approximators [14]. One of the most popular algorithms in neural networks category is the back-propagation algorithm [13]. The back-propagation algorithm performs supervised learning on a multilayer feed-forward neural network. It is based on Perceptron algorithm introduced by Rosenblatt [11] that is considered as one of the first machine learning algorithms. Perceptron algorithm express linear

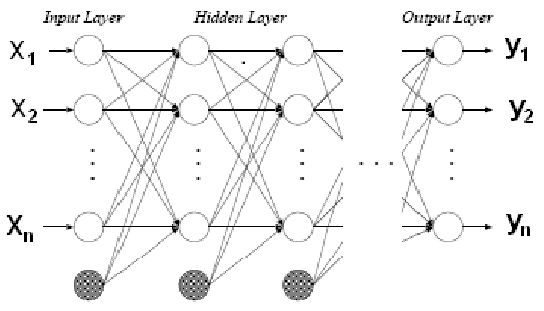


Figure 1.3: Backpropagation neural network architecture.

decision surfaces. In contrast, the kind of multilayer networks [12] learned by BACKPROPAGATION algorithm [15] are capable of expressing a rich variety of nonlinear decision surfaces. Its discovery played made the neural networks as a tool for solving a wide variety of problems ranging from speech recognition to complex task of particle separation in high energy physics experiments.

A multilayer neural network as shown in the figure 1.3 consists of sensory units (neurons or nodes) divided into 3 layers - a input layer (X), one or more hidden layers (Z) and an output layer (Y). The units in layers are fully connected with the units in the next layer. These connections have weights associated with them. Each signal traveling along the link is multiplied by the connection weight. The input signal propagates through the network in a forward direction, on a layer -by-layer basis.

Algorithm

The training of a network by back-propagation involves three stages [14]: The feed-forward of the input training pattern, the calculation and back-propagation of the associated error, and the adjustment of the weights. After training, application of the net involves only the computations of the feed-forward phase. Even if the training is slow, a trained net can produce its output very rapidly.

Let $\mathbf{x} \in R^n$ be the Input training vector, $\mathbf{t} \in R^m$ be the Output target vector. δ_k be the error correction weight for the output unit Y_k, δ_j be the error correction weight for the hidden unit Z_j, α be the learning rate. During the feed-forward, each input unit (X_i) receives an input signal and broadcasts this signal to the each of the hidden units $Z_1, \ldots, Z_p$. Each hidden unit then computes its activation and sends its signal (Z_j) to each output unit. Each output unit (Y_k) computes its activation (y_k) to form the response of the net for the given input pattern. During training, each output unit compares its computed activation (y_k) with its target value (t_k) to determine the associated error for that pattern with that unit. Based on this error, the factor $\delta_k (k = 1, \ldots, m)$ is computed. δ_k is used to distribute the error at output unit Y_k back to all units in the previous layer (the hidden units that are connected to Y_k). It is also used later to update the weights between the hidden layer and the input layer. Listing 1.1 shows the algorithm.

Listing 1.1: BPNN algorithm

```
Step 0: Intialize weights with small random values
Step 1: While stopping condition is false **do** steps 2-9.
Step 2: **for** each tarining pair, **do** steps 3-8
        Feedforward
        Step 3: Each input unit $(X_i, i = 1, \dots, n)$
                receives input signal $x_i$ and broadcasts
                this signal to all units to hidden layer
        Step 4: Each hidden unit $(Z_j, j = 1, \dots, p)$
                sums its weighted input signals,
                $z - in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$
                applies its activation function to
                compute its output signal
                $z_j = f(z - in_j)$,
                and sends this signal to all units
                to the output units.
        Step 5: Each output unit $(Y_k, k = 1, \dots, m)$
                sums its weighted input signals,
                $y - in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$
                and applies its activation function to compute
                its output signal,
                $y_k = f(y - in_k)$.
        Back-propagation of error:
        Step 6: Each output unit $(Y_k, k = 1, \dots, m)$
                receives a target pattern corresponding to the input
                pattern, computes its error information term,
                $\delta_k = (t_k - y_k) f'(y - in_k)$
                calculates its bias corerction term
                used to update w_{0k} later,
                $\Delta w_{0k} = \alpha \delta_k$
                and sends the $\delta_k$ to units in the layer below.
        Step 7: Each hidden unit $(Z_j, j = 1, \dots, p)$
                sums its delta inputs from units in layer above.
                $\delta - in_j = \sum_{k=1}^{m} \delta_k w_{jk}$
                multiplies by the derivative of its activation
                function to calculate its error information term,
                $\delta_j = \delta - in_j f'(z - in_j)$
                calculate its weight correction term
                used to update $v_{ij}$ later,
                $\Delta v_{ij} = \alpha \delta_j x_i$
                and calculates its bias corerction term
                used to update $v_{0j}$ later,
                $\Delta v_{oj} = \alpha\delta_j$ .
        Update weights and biases:
        Step 8: Each output unit $(Y_k, k = 1, \ldots, m)$
                updates its bias and weights $(j = 0, \ldots, p)$:
                $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$
                Each hidden unit $(Z_j, j = 1, \ldots, p)$
                updates its bias and weights $(i = 0, \ldots, n)$:
                $v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$
Step 9: Test stopping condition.
```

After all of the δ factors that have been determined, the weights for all the layers are adjusted simultaneously [14]. The adjustment to the weight w_{jk} from the hidden unit Z_j to the output unit Y_k is based on the factor δ_k and the activation function z_j of the hidden unit Z_j. The adjustment to the weight v_{ij} form the input unit X_i to hidden unit Z_j is based on the factor δ_j and the activation x_i of the input unit. An activation function for a backpropagation net should have several important characteristics: It should be continuous differentiable, and monotonically non-decreasing. Furthermore, for computational efficiency, it is desirable that its derivative be easy to compute. For the most commonly used activation functions, the value of the derivative can be expressed in terms of the value of the function. Usually the function is expected to *saturate*, *i.e*, approach finite maximum and minimum values asymptotically. In this thesis the BPPNN network is developed using a hyperbolic tangent function, which has range of (-1, 1) and is defined as

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

1.3.2 k-nearest neighbors

Instance-based learning methods such as nearest neighbor are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions. Learning in these algorithms consists of simply storing the presented training data [3]. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance.

One important instance-based method is the k-nearest neighbors algorithm. This algorithm assumes all instances correspond to points in the n-dimensional space R^n. The nearest neighbors of an instance are defined in terms of the standard euclidean distance. More precisely, let an arbitrary instance x be described by the feature vector $(a_1(x), \ldots, a_n(x))$, where $a_r(x)$ denotes the vaue of the r^{th} atrribute of insyance x. Then the distance between two instances x_i and x_j is defined to be $d(x_i, x_j)$ where $d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$

In nearest-neighbor learning the target function may be either discrete-valued or real-valued. For learning discrete-valued target functions of the form $f : R^n \to V$, where V is the finite set $\{v_1, \ldots, v_s\}$. The k-nearest neighbor algorithm returns $\hat{f}(x_q)$

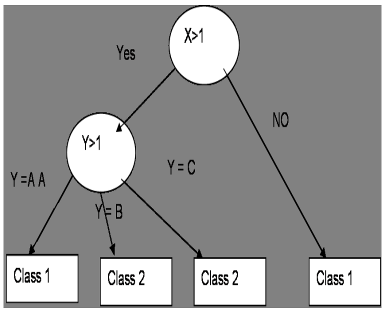


Figure 1.4: A decision tree with the tests on attributes X and Y

for approximating the discrete-valued target function $f(x_q)$, which is given by:

$$\hat{f}(x_q) \longleftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a,b) = 1$ if $a = b$ and where $\delta(a,b) = 0$ otherwise. As shown here, the value $\hat{f}(x_q)$ returned by this algorithm as its estimate of $f(x_q)$ is just the most common value of f among the k training examples nearest to x_q. If we choose $k = 1$, then the 1-nearest neighbor algorithm assigns to $\hat{f}(x_q)$ the value $f(x_i)$ where x_i is the training instance nearest to x_q. For larger values of k, the algorithm assigns the most common value among the k nearest training examples.

1.3.3 Decision Trees

The decision-tree representation is the most widely used logic method for efficiently producing classifiers from the data. There is a large number of decision-tree induction algorithms described primarily in the machine-learning and applied-statistics literature. The decision tree algorithm is well known for its robustness and learning efficiency with its learning time complexity of $O(nlog2n)$, n being the number of rows in the train set. The output of the algorithm is a decision tree, which can be easily represented as a set of symbolic rules $(IF \ldots THEN)$. The symbolic rules can be directly interpreted and compared with the existing domain knowledge, providing the useful information for the domain experts.

A typical decision-tree learning system adopts a top-down strategy that searches for a solution in a part of the search space. It guarantees that a simple, but not necessarily the simplest, tree will be found. A decision tree consists of nodes that where attributes are tested. The outgoing branches of a node correspond to all the possible outcomes of the test at the node. A simple decision tree for classification of samples with two input attributes X and Y is given Figure 1.4.

All samples with feature values $X > 1$ and $Y = B$ belong to Class2, while the samples with values $X < 1$ belong to Class1, whatever the value for feature Y. The samples, at a non leaf node in the tree structure, are thus partitioned along the branches and each child node gets its corresponding subset of samples. Decision trees that use univariate splits have a simple representational form, making it relatively easy for the user to understand the inferred model; at the same time, they represent a restriction on the expressiveness of the model. In general, any restriction on a particular tree representation can significantly restrict the functional form and thus the approximation power of the model.

A well-known tree-growing algorithm for generating decision trees based on univariate splits is Quinlan's ID3 with an extended version called C4.5, C5.0 [101]. Greedy search methods, which involve growing and pruning decision-tree structures, are typically employed in these algorithms to explore the exponential space of possible models and to remove unnecessary preconditions and duplication. They apply a divide and conquer strategy to construct the tree. The sets of instances are accompanied by a set of properties. Each node in the tree performs a test on the values of an attribute, and the leaves represent the class of an instance that satisfies the tests. The tree will return a 'yes' or 'no' decision when the sets of instances are tested on it. Rules can be derived from the tree by following a path from the root to a leaf and using the nodes along the path as preconditions for the rule, to predict the class at the leaf. Random forests, an ensemble decision trees uses the trees that randomly choose a subset of attributes at each mode. We discuss the random forest in chapter 5 in more detail.

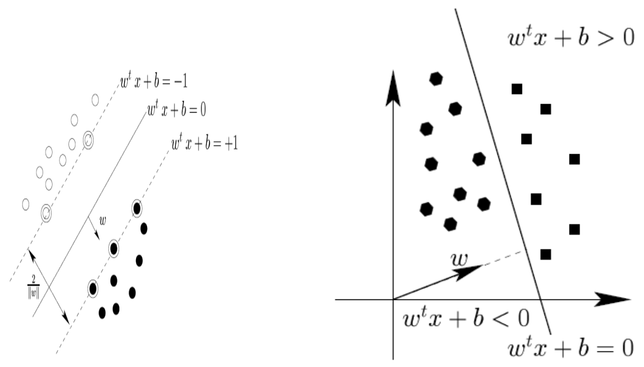
1.3.4 Support vector machines

SVMs are kernel based learning algorithms used in machine learning and computer vision research communities. However, SVMs have two possible limitations in real world applications. First, the training of SVMs is slow, especially for large data sets problems. Second, SVM training algorithms are complex, subtle, and sometimes difficult to implement [17].

The basic training for SVMs involves finding a function which optimizes a bound on the generalization capability, i.e., performance on unseen data. We are guven N observations $x^{(i)} \in R^L$ with associated labels $y^{(i)}, i = 1, \ldots, N$. These set of training data $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}$ is linearly separable if there exists a hyperplane (w^t, b) for which the positive examples lie on one side and the negative examples on the other. Let us define the *margin* as twice the distance of the closest training example to the hyperplane (w^t, b) (figure 1.5 (a)). The structural risk minimization principle [18] states that a hyperplane which classifies the training set accurately with the largest margin will minimize a bound on the generalization error and will generalize best, regardless of the dimensionality of the input space.

Provided that all of the training examples are linearly separable, the goal is then to find the optimal separating hyperplane (w^t, b) in the sense of maximizing the margin. Let $d_{w,b}\left(x^{(s)}\right)$ be the distance of the closest training example $x^{(s)}$ to a separating hyperplane (w^t, b):

$$d_{w,b}\left(x^{(s)}\right) = \frac{\mid w^t x^{(s)} + b \mid}{\parallel w \parallel}$$



(a) Maximal margin classifier (b) Linear SVM classifier

Figure 1.5: Linear separation problem with SVM

If we constrain $\mid w^t x^{(s)} \mid = 1$, then $d_{w,b}\left(x^{(s)}\right) = \frac{1}{\|w\|}$ and the margin associated to (w^t, b) is equal to $\frac{2}{\|w\|}$. The optimal separating hyperplane (w^t, b) is found by minimizing the legrangians multipliers L_2 norm of w, under the constraint that the training set is well separated by (w^t, b) .

$$f(x) = \sum_{i=1} L w_j x_j + b$$

where L is the legrangian norms defined for coordinates for data point x and the normal vector w. Thus the solution for linear decision function gives the optimal separation for hyperplane (w^t, b) (fig 1.5 (b)).

2 Unsupervised Learning

2.1 Introduction

Unsupervised learning refers to situations where the objective is to construct decision boundaries based on unlabeled training data to find the natural groups or clusters that exist in the data set[7]. Unsupervised classification or clustering is a very difficult problem because data can reveal clusters with different shapes and sizes. To compound the problem further, the number of clusters in the data often depends on the resolution with which we view the data.

In unsupervised learning a higher-order statistical model is learnt from a set of examples, with the aim of revealing hidden causes and density estimation. Applications range from data visualization [21] to data mining and knowledge discovery [22]. These techniques are strongly related to the statistical field of cluster analysis, where over the years large number of clustering methods have been proposed [23] [24] [25]. In cluster analysis observed data are organized into meaningful structures or taxonomies. The objective is to sort samples into clusters or groups, so that the degree of association is strong between members of the same cluster and weak between members of different clusters. A fair amount of research has been done on cluster analysis giving many ad hoc methods to search for these groupings, or clusters.

There are at least five basic reasons for interest in unsupervised Procedures. First, collecting and labeling a large set of sample patterns can be surprisingly costly. For the MAGIC telescope experiment discussed in chapter 6, particle image data collection is virtually free, but accurately labeling the images of particles, i.e., marking the particles with labels of gamma, hadrons, muons particle types is very expensive, time consuming and error pruned . If a classifier can be crudely designed on a small set of labeled samples, and then tuned up by allowing it to run without supervision on a large, unlabeled set, much time and trouble can be saved. Second, one might wish to proceed in the reverse direction: train with large amounts of (less expensive) unlabeled data, and only then use supervision to label the groupings found. This may be appropriate for large data mining applications where the contents of a large database are not known beforehand. Again this can be useful for the MAGIC telescope as we have giga bytes of data to be classified. The possible way of working it out is to group the unlabeled data automatically by unsupervised technique, Self-Organizing Map

(SOM). The grouped data is then subjected to a supervised technique for identifying the labels. Third, in many applications the characteristics of the patterns can change slowly with time, for example in automated food classification as the seasons change. If these changes can be tracked by a classifier running in an unsupervised mode, improved performance can be achieved. Fourth, we can use unsupervised methods to find features that will then be useful for categorization. There are unsupervised methods that represent a form of data-dependent *smart preprocessing* or *smart feature extraction*. Lastly, in the early stages of an investigation it may be valuable to gain some insight into the nature or structure of the data. The discovery of distinct subclasses or similarities among patterns or of major departures from expected characteristics may suggest we significantly alter our approach to designing the classifier.

In this chapter we discuss some fundamental issues and state-of-art related to unsupervised learning. After discussing cluster analysis we describe the types of clustering algorithms available in the literature with special emphasis on k-means clustering, the basis for self-organizing map (SOM) discussed in next chapter.

2.2 Cluster Analysis

Cluster analysis is a very important and often required in real world problems. The speed, reliability, and consistency with which a clustering algorithm can organize large amounts of data constitute overwhelming reasons to use it in applications such as data mining [2], information retrieval [9], image segmentation , signal processing [10]. As a consequence, several clustering algorithms have been proposed in the literature and new clustering algorithms continue to appear. Most of these algorithms are based on a) iterative squared-error clustering or b) agglomerative hierarchical clustering. Hierarchical techniques organize data in a nested sequence of groups which can be displayed in the form of a dendogram or a tree. Squared-error partitional algorithms attempt to obtain that partition which maximizes the between-cluster scatter.

It has a variety of goals. All relate to grouping or segmenting a collection of objects into subsets or clusters, such that those within each cluster are more closely related to one another than objects assigned to different clusters. An object can be described by a set of measurements, or by its relation to other objects. In addition, the goal is sometimes to arrange the clusters into a natural hierarchy. This involves successively grouping the clusters themselves so that at each level of the hierarchy, clusters within the same group are more similar to each other than those in different groups. Cluster analysis is also used to form descriptive statistics to ascertain whether or not the data consists of a set distinct subgroups, each group representing objects with substantially different properties. This latter goal requires an assessment of the degree of difference between the objects assigned to the respective clusters.

2.3 Cluster distance measures

Central to all of the goals of cluster analysis algorithms is the notion of the degree of similarity (or dissimilarity) between the individual objects being clustered. A clustering method attempts to group the objects based on the definition of similarity supplied to it. This can only come from subject matter under considerations [20]. In this section we discuss the variety of distance measures used in clustering algorithms.

2.3.1 Proximity measures

Sometimes the data is represented directly in terms of the proximity (alikeness or affinity) between pairs of objects. These can be either similarities or dissimilarities (difference or lack of affinity). For example, in social science experiments, participants are asked to judge by how much certain objects differ from one another. Dissimilarities can then be computed by averaging over the collection of such judgments. This type of data can be represented by an $N \times N$ matrix D, where N is the number of objects, and each element $d_{ii'}$ records the proximity between the ith and $i'th$ objects. This matrix is then provided as input to the clustering algorithm. Most algorithms presume a matrix of dissimilarities with non- negative entries and zero diagonal elements: $d_{ii} = 0, i = 1, \dots, N$. If the original data were collected as similarities, a suitable monotone-decreasing function can be used to convert them to dissimilarities. Also, most algorithms assume symmetric dissimilarity matrices, so if the original matrix D is not symmetric it must be replaced by $\left(D + D^T\right)/2$. Subjectively judged dissimilarities are seldom distances in the strict sense, since the triangle inequality $d_{ii'} \le d_{ik} + d_{i'k}$ for all $k \in \{1, \dots, N\}$ does not hold. Thus, some algorithms that assume distances cannot be used with such data.

2.3.2 Dissimilarities Based on Attributes

Most often we have measurements x_{ij} for patterns $i = 1, \dots, N$, on features $j = 1, \dots, p$. Since most of the popular clustering algorithms take a dissimilarity matrix as their input, we must first construct pairwise dissimilarities between the observations. In the most common case, we define a dissimilarity $d_j\left(x_{ij}, x_{i'j}\right)$ between values of the jth variable, and then define

$$D\left(x_i, x_i'\right) = \sum_{j=1}^{p} d_j\left(x_{ij}, x_{i'j}\right) \tag{2.3.1}$$

as the dissimilarity between objects i and i'. By far the most common choice is squared distance

$$d_j\left(x_{ij}, x_{i'j}\right) = \left(x_{ij} - x_{i'j}\right)^2 \tag{2.3.2}$$

However, other choices are possible, and can lead to potentially different results. For non quantitative attributes (e.g., categorical data), squared distance may not be appropriate. In addition, it is sometimes desirable to weigh attributes differently. Here we discuss alternatives in terms of the attribute type:

- *Quantitative variables.* Measurements of this type of features or variables or attributes are represented by continuous real-valued numbers. It is natural to define the "error" between them as a monotone-increasing function of their absolute difference
$$d\left(x_i, x_i'\right) = l\left(\mid x_i - x_i' \mid\right) \tag{2.3.3}$$
Besides squared error loss $\left(x_i - x_{i'}\right)^2$, a common choice is the identity (absolute error). The former places more emphasis on larger differences than smaller ones. Alternatively, clustering can be based on the correlation
$$\rho\left(x_i, x_{i'}\right) = \frac{\sum_j \left(x_{ij} - \bar{x}_i\right)\left(x_{i'j} - \bar{x}_{i'}\right)}{\sqrt{\sum_j \left(x_{ij} - \bar{x}_i\right)^2 \sum_j \left(x_{i'j} - \bar{x}_{i'}\right)^2}} \tag{2.3.4}$$
with $\bar{x}_i = \sum_j x_{ij}/p$. If the inputs are first standardized, then $\sum_j \left(x_{ij} - x_{i'j}\right)^2 = 2\left(1 - \rho\left(x_i, x_{i'}\right)\right)$. Hence clustering based on correlation (similarity) is equivalent to that based on squared distance (dissimilarity).

- *Ordinal variables:* The values of this type of variable are often represented as contiguous integers, and the realizable values are considered to be an ordered set. Examples are academic grades (A, B, C, D, F), degree of preference (can't stand, dislike, OK, like, terrific). Rank data are a special kind of ordinal data. Error measures for ordinal variables are generally defined by replacing their M original values with
$$\frac{i - 1/2}{M}, i = 1, \ldots, M \tag{2.3.5}$$
in the prescribed order of their original values. They are then treated as quantitative variables on this scale.

- *Categorical variables:* With unordered categorical (also called nominal) variables, the degree-of-difference between pairs of values must be delineated explicitly. If the variable assumes M distinct values, these can be arranged in a symmetric $M \cdot M$ matrix with elements $L_{rr'} = L_{r'r}, L_{rr} = 0, L_{rr'} \geq 0$. The most common choices is $L_{rr'} = 1$ for all $r \neq r$, while unequal losses can be used to emphasize some errors more than others.

2.3.3 Object Dissimilarity

Next we define a procedure for combining the p-individual attribute dissimilarities $d_j\left(x_{ij}, x_{i'j}\right), j = 1, \ldots, p$ into a single overall measure of dissimilarity $D\left(x_i, x_i'\right)$ between two objects or observations $\left(x_i, x_{i'}\right)$ possessing the respective attribute values. This is nearly always done by means of a weighted average (convex combination)

$$D\left(x_i, x_i'\right) = \sum_{j=1}^{p} w_j . d_j\left(x_{ij}, x_{i'j}\right) \tag{2.3.6}$$

Here w_j is a weight assigned to the jth attribute regulating the relative influence of that variable in determining the overall dissimilarity between objects. This choice should be based on subject matter considerations. If the goal is to discover natural groupings in the data, some attributes may exhibit more of a grouping tendency than others. Variables that are more relevant in separating the groups should be assigned a higher influence in defining object dissimilarity. Giving all attributes equal influence in this case will tend to obscure the groups to the point where a clustering algorithm cannot uncover them.

Although simple generic prescriptions for choosing the individual attribute dissimilarities $d_j\left(x_{ij}, x_{i'j}\right)$ and their weights w_j can be comforting, there is no substitute for careful thought in the context of each individual problem. Specifying an appropriate dissimilarity measure is far more important in obtaining success with clustering than choice of clustering algorithm. This aspect of the problem is emphasized less in the clustering literature than the algorithms themselves, since it depends on domain knowledge specifics and is less amenable to general research. Finally, often observations have missing values in one or more of the attributes. The most common method of incorporating missing values in dissimilarity calculations as in equation 2.3.6 is to omit each observation pair $x_{ij}, x_{i'j}$ having at least one value missing, when computing the dissimilarity between observations x_i and x_i'. This method can fail in the circumstance when both observations have no measured values in common. In this case both observations could be deleted from the analysis. Alternatively, the missing values could be imputed using the mean or median of each attribute over the non-missing data. For categorical variables, one could consider the value missing as just another categorical value, if it were reasonable to consider two objects as being similar if they both have missing values on the same variables.

2.4 Clustering Algorithms

Clustering algorithms divide, or partition, data into natural groups of objects. By natural it means that the objects in a cluster should be internally similar to each other, but differ significantly from the objects in the other clusters. Most clustering algorithms produce crisp partitionings, where each data sample belongs to exactly one cluster. To reflect the inherently vague nature of clusterings, there are also some algorithms where each data object may belong to several clusters to a varying degree. Another way to deal with the complexity of real data sets is to construct a cluster hierarchy. Clustering may depend on the level of detail being observed, and thus a cluster hierarchy may, at least in principle, be better at revealing the inherent structure of the data than a direct partitioning 2.4.

Thus clustering algorithms can be fundamentally divided into two types: hierarchical and partitional [24]. Hierarchical clustering algorithms find clusters one by one. The hierarchical methods can be further divided to agglomerative and divisive algorithms, corresponding to bottom-up and top-down strategies. Agglomerative clustering algorithms merge clusters together one at a time to form a clustering tree which finally consists of a single cluster, the whole data set. The algorithms consist

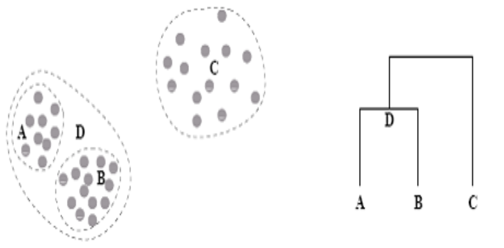


Figure 2.1: Interesting clusters may exist at several levels. In addition to A, B and C, also the cluster D, which is a combination of A and B, is interesting.

of the following steps, Listing 2.1:

Listing 2.1: hierarchical clustering

```
1. initialize: assign each vector to its own cluster , or use
some initial partitioning provided by some other clustering algorithm
2. Compute distances $d(C_i, C_j)$ between all clusters
3. merge the two clusters that are closest to each other
4. return to step 2 until there is only one cluster left
```

The problem of partitional clustering can be formally stated as follows: Given N patterns in a d-dimensional metric space, determine a partition of the patterns into k clusters, such that the patterns in a cluster are more similar to each other than to patterns in different clusters [8]. Under partitional clustering we concentrate on K-means algorithm.

The K-means algorithm is intended for situations in which all variables are of the quantitative type, and squared Euclidean distance is chosen as the dissimilarity measure.

$$d\left(x_i, x_{i'}\right) = \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 = \parallel x_i - x_i' \parallel^2 \tag{2.4.1}$$

Since the goal is to assign close points to the same cluster, a natural loss (or energy) function would be

$$W\left(C\right) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d\left(x_i, x_i'\right) \tag{2.4.2}$$

The above equation characterizes the extent to which observations assigned to the same cluster tend to be close to one another. The minimum value for the equation can be obtained by assigning the N observations to the K clusters in such a way that within each cluster the average dissimilarity of the observations from the cluster mean, as defined by the points in that cluster, is minimized.

Listing 2.2: K-means clustering algorithm

```
1. For a given cluster assignment $C$, the total cluster variance
   is minimized with respect to $\{m_1, \ldots, m_K\}$ yielding
   the means of the currently assigned clusters
2. Given a current set of means $\{m_1, \ldots, m_K\}$, eqn 2.3.4
   is minimized by assigning each observation to the closest cluster mean.
   That is , $C(i) = argmin_{1 \leq k \leq K} \parallel x_i - m_k \parallel^2$
3. First two steps are iterated until the assignments do not change.
```

An iterative descent algorithm for solving

$$C^* = min_c \sum_{k=1}^{K} \sum_{C(i)=k} \parallel x_i - \bar{x}_k \parallel^2 \tag{2.4.3}$$

can be obtained by noting that for any set of observations S

$$\bar{x}_S = argmin_m \sum_{i \in S} \parallel x_i - m \parallel^2 \tag{2.4.4}$$

Hence we can obtain C^* by solving the enlarged optimization problem

$$min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} \sum_{C(i)=k} \parallel x_i - m_k \parallel^2 \tag{2.4.5}$$

This can be minimized by an alternating optimization procedure given by K-means clustering algorithm in Listing 2.2. Each of steps 1 and 2 reduces the value of the criterion (eqn 2.4.5, so that convergence is assured. However, the result may represent a suboptimal local minimum. In addition one should start the algorithm with many different random choices for the starting means, and choose the solution having smallest value of the objective function.

K-means algorithm can be extended to perform unsupervised learning using neural networks. These networks will look for regularities, or trends in the input signals, and make adaptations according to the function of the network. Even without being told whether it is right or wrong, the network still must have some information about how to organize itself. This information is built into the network topology and learning rules. Competition between processing elements could form a basis for learning. Training of competitive elements could amplify the responses of specific groups to specific stimuli. As such, it would associate those groups with each other and with a specific appropriate response. Such a procedure was developed by Teuvo Kohonen termed as Self-Organizing Maps (SOM) neural networks, and was inspired by learning in biological systems [33].

3 Self-Organizing Maps

3.1 Introduction

Self-Organising Map (SOM) [33] is a neural network model that is based on unsupervised learning. It is an effective method for clustering and visualization of high-dimensional data. It proved to be a valuable tool in data mining and Knowledge Discovery in Databases (KDD)[1] [36]. SOM has applications in pattern recognition, image analysis, process monitoring [42], organization of document collections [43] etc. In [48] a number of data analysis cases related to economics are presented in which the SOM has been an important tool. More examples of fruitful usage of the SOM in various engineering tasks can be found for example in [54] [55]. A comprehensive bibliography of SOM research has been compiled by Kaski et al. [56].

A SOM consists of neurons organized on a regular low-dimensional grid. The number of neurons can vary from a few dozen upto several thousand. Each neuron is represented by a d-dimensional weight vector also called as prototype vector or codebook vector , where d is equal to the dimension of the input vectors. The neurons are connected to adjacent neurons by a neighborhood relation, which dictates the topology, or structure, of the map. The topology can be broadly divided to two factors: local lattice structure and the global map shape [57]. Examples of rectangular and hexagonal lattice structures are shown in figure 3.1, and examples of different kinds of map shapes in figure 3.2.

The SOM training algorithm resembles the vector quantization (VQ) algorithms, such as k-means [58]. The important distinction is that in addition to the best-matching weight vector is stretched towards the presented training sample, as in figure 3.3. The end result is that the neurons on the grid become ordered: neighboring neurons have similar weight vectors.

Since the weight vectors of the SOM have well-defined low dimensional coordinates on the map grid, the SOM is also a vector projection algorithm [59]. Together the prototype vectors and their projection define a low dimensional map of the data manifold.

[1]The non-trivial extraction of implicit, unknown, and potentially useful information from data

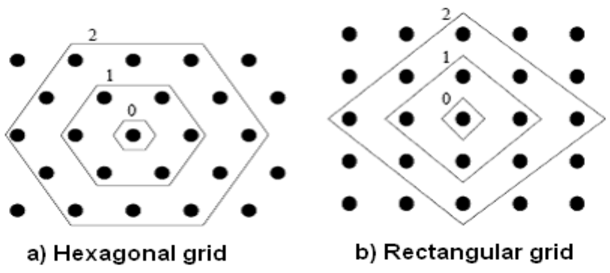


Figure 3.1: : Discrete neighborhoods (size 0, 1 and 2) of the centermost unit: a) hexagonal lattice, b) rectangular lattice. The innermost polygon corresponds to 0-neighbourhood, the second to the 1-neighbourhood and the biggest to the 2-neighbourhood

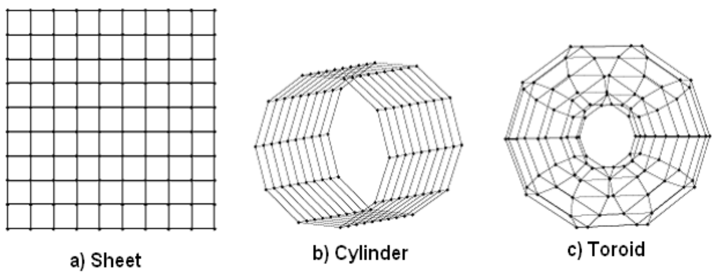


Figure 3.2: Different map shapes. The default shape (a), and two shapes where the map topology accommodates circular data: cylinder (b) and toroid (c).

3.2 Algorithm

The training algorithm is simple, robust to missing values, and - perhaps most importantly - it is easy to visualize the map. These properties make SOM a prominent tool in data mining, data exploration and cluster visualisation phase. The learning process of the SOM is as follows:

1. **Initialisation phase:** Initialise all the neurons in the map with the input vectors randomly.

2. **Data normalization:** For a better identification of the groups the data have to be normalized. We employed the 'range' method where each component of the data vector is normalized to lie in the intravel [0,1].

3. **SOM Training:** Select an input vector x from the data set randomly. A best matching unit (BMU) for this input vector, is found in the map by the following metric
$$\|x - m_c\| = \min_i \{\|x - m_i\|\}$$
where m_i is the reference vector associated with the unit i.

4. **Updating Step:** The reference vectors of BMU and its neighborhood are updated according to the following rule
$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t) \cdot h_{ci}(t) \cdot [x(t) - m_i(t)], & i \in N_c(t) \\ m_i(t), & i \notin N_c(t) \end{cases}$$
where
$h_{ci}(t)$ is the kernel neighborhood around the winner unit c.

   t is the time constant.

   $x(t)$ is an input vector randomly drawn from the input data set at time t.

   $\alpha(t)$ is the learning rate at time t.

   $N_c(t)$ is the neighborhood set for the winner unit c.
The above equation make BMU and its neighborhood move closer to the input vector. This adaptation to input vector forms the basis for the group formation in the map.

5. **Data groups visualisation:** steps 3 and 4 are repeated for selected number of trials or epochs. After the trails are completed the map unfolds itself to the distribution of the data set finding the number of natural groups exist in the data set. The output of the SOM is the set of reference vectors associated with the map units. This set is termed as a codebook. To view the groups and the outliers discovered by the SOM we have to visualize the codebook. U-Matrix is the technique typically used for this purpose.

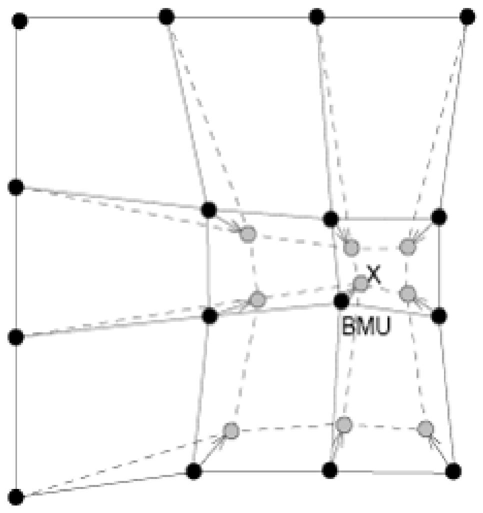


Figure 3.3: Updating the best matching unit (BMU) and its neighbors towards the input sample with x. The solid and dashed lines correspond to situation before and after updating, respectively

In the thesis, the distance computation is modified based on the two factors:

- Missing Values: In the thesis, these are represented by the value of NAN in the vector or data matrix. Missing components are handled by simply excluding them from the distance calculation. It is assumed that their contribution to the distance $\|x - m_i\|$ is zero. According to [60], this is a valid solution as the same variables is ignored in each distance calculation over which the minimum is taken.
- Mask: Each variable has an associated weighting factor. This is primarily used in binary form for excluding certain variables from the BMU-finding process (1 for include, 0 for exclude). However, the mask can get any values, so it can be used for weighting variables according to their importance.

With these changes, the distance measure becomes:

$$\|x - m_i\| = \sum_{k \in K} w_k \left(x_k - m_k\right)^2 \tag{3.2.1}$$

where K is the set of known variables of sample vector x, x_k and m_k are the k^{th} components of the sample and weight vectors and w_k is the k^{th} mask value (mask(k)). After finding the BMU, the weight vectors of the SOM are updated so that the BMU is moved closer to the input vector in the input space. The topological neighbors of the BMU are treated similarly. This adaptation procedures stretches the BMU and

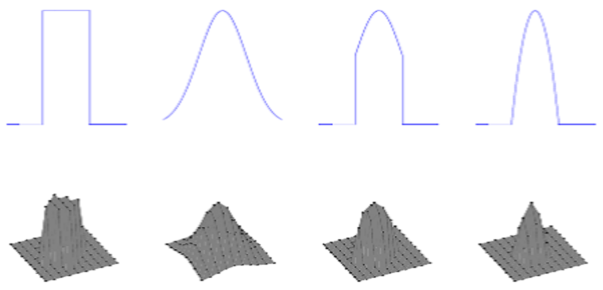


Figure 3.4: Different neighborhood functions. From the left 'bubble' $h_{ci}(t) = 1(\sigma_t - d_{ci})$, 'gaussian' $h_{ci}(t) = e^{-d_{ci}^2/2\sigma_t^2}$, 'cut-gauss' $h_{ci}(t) = e^{-d_{ci}^2/2\sigma_t^2} 1(\sigma_t - d_{ci})$, and 'ep' $h_{ci}(t) = max\left\{0, 1 - (\sigma_t - d_{ci})^2\right\}$, $d_{ci} = \|r_c - r_i\|$ is the distance between map units c and i on the map grid and $1(x)$ is the step function: $1(x) = 0$ if $x < 0$ and $1(x) = 1$ if $x \geq 0$. The top row shows the function in 1-dimensional and the bottom row on a 2-dimensional map grid. The neighborhood radius used is $\sigma_t = 2$.

its topological neighbors towards the sample vector as shown in figure 3.3. As listed in the SOM algorithm, the SOM update rule for the weight vector of unit i is:

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t) [x(t) - m_i(t)] \tag{3.2.2}$$

The input vector $x(t)$ is an input vector randomly drawn from the input data set at time t, $h_{ci}(t)$ the neighborhood kernel around the winner unit c. The neighborhood kernel is a non-increasing function of time and of the distance of unit i from the winner c. It defines the region of influence that the input sample has on the SOM. Different kernel neighborhoods that can be used with SOM are summarized in Figure 3.4, $\alpha(t)$ represents the learning rate, which is also a decreasing function of time used to converge the map to the groups discovered by SOM in the data set. Figure 3.5 shows the different learning rates and their distributions.

For faster learning, SOM can be trained in a batch mode. Batch training algorithm is also iterative, but instead of using a single data vector at a time, the whole data set is presented to the map before any adjustments are made - hence the name "batch". In each training step, the data set is partitioned according to the Voronoi regions of the map weight vectors, i.e each data vector belongs to the data set of the map unit to which it is closest. After this, the new weight vectors are calculated as

$$m_i(t+1) = \frac{\sum_{j=1}^{n} h_{ic}(t) x_j}{\sum_{j=1}^{n} h_{ic}(t)} \tag{3.2.3}$$

3.3 SOM as clustering technique

SOM clusters the data set based on its quantization capability on the given input data set. Quantization reduces the original data set to a small representative set

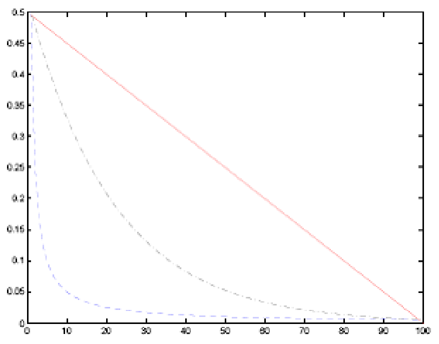


Figure 3.5: Different learning rate functions: $'linear'$ (solid line) $\alpha(t) = \alpha_0(1 - t/T)$, $'power'$ (dot-dashed) $\alpha(t) = \alpha_0(0.005/\alpha_0)^{t/T}$ and $'inv'$ (dashed) $\alpha(t) = \alpha_0/(1 + 100t/T)$, where T is the training length and α_0 is the initial learning rate.

of prototypes to work with. The representative set of prototypes can be utilized in computationally intensive tasks, like clustering or projection, to get approximative results with reduced computational cost [45]. This reduction is important especially in data exploration. In addition, since the prototypes are formed as averages of the data samples, the effect of zero-mean noise as well as outliers are reduced [46].

Vector quantization algorithms, try to find a set of prototype vectors $m_i = 1, \ldots, M$ which reproduce the original data set as well as possible. The best known algorithm to find these prototypes is the k-means alogorithm [53]. Quantization algorithms finds a set of $M = k$ prototype vectors which minimize the quantixation error E_q, used to measure the quatization property of the algorithm.

$$E_q = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{d} |x_{ij} - m_{b_i j}|^r \tag{3.3.1}$$

where b_i is the index of the best-matching prototype, r is the distance norm. The point density of the prototypes follows the density of the training data. Asymptotically it holds that:

$$p(m) \propto p(x)^{\frac{d}{d+r}} \tag{3.3.2}$$

where d is the dimension and $p(x)$ and $p(m)$ are the probability density functions of the input data and the protoype vectors respectively.

The SOM is closely related to the k-means algorithm. If the neighborhood kernel value is one for the BMU and zero elsewhere ($h_{b_i j} = \delta(b_i, j)$ in Eq. 3.2.2, the SOM reduces to the adaptive k-means algorithm. Also batch map reduces to batch k-means.

The difference between classical vector quantization and SOM is that the SOM performs local smoothing in the neighborhood of each map unit. This smoothing creates the ordering of the prototypes, but when the neighborhood radius σ is decreased

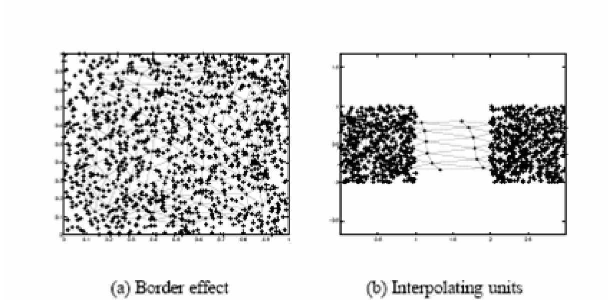


Figure 3.6: Two side effects caused by the neighborhood function: (a) border effect and (b) interpolating units. The + are the training data, and the connected grid of circles is the map.

during the the training, it also implements a simulated annealing type of learning scheme that makes the quantization process more robust. There are also two side effects (fig. 3.6).

- Border effect: The neighborhood definition is not symmetric on the borders of the map. Therefore, the density estimation is different for the border units than for the center units of the map [47]. In practice, the map is contracted on the borders. This has the effect that the tails of the marginal distributions of variables are less well presented than their centers. In some cases, this may help to reduce the effect of outliers, but in general, this is a weakness of the SOM.
- Interpolating units: When the data cloud is discontinuous, interpolating units are the data distribution. However, in case of some analysis tools, for example single linkage clustering, these may give false cues of the shape of the data manifold and may need to be deemphasized or completely left out of analysis.

If the input data set considered as the set of stochastic data variables x that is distributed according to a probability density function $p\,(x)$ then the SOM forms an approximation to this probability density function $p\,(x)$, using a finite number of centroid vectors $m_i\,(i = 1, \ldots, k)$. Now these point density of the prototypes follows roughly the probability density of the data. Once the centroid is chosen, the approximation of x means finding the centroid vector m_c closest to x vector in the input space. The optimal solution of m_i minimizes the average expected value E_{avg} of the quantization error E_q, defined as

$$E_{avg} = \int \|x - m_c\|^2 p\,(x)\,dx \tag{3.3.3}$$

According to [61] , if the number of centroid vectors is large, the optimal selection of m_i values is such that their point density (Eq. 3.3.2) approximates to

$$p\,(m) \propto p\,(x)^{\frac{2}{3} - \frac{1}{3\sigma^2 + 3(\sigma+1)^2}} \tag{3.3.4}$$

Variables play an important role in quantization properties of the SOM. The importances of variables defines the viewpoint of the quantization. When quantization has a central role in data analysis, it is important to know what is this viewpoint, because any analysis based on the quantization will reflect how well the variables are represented.By adding, removing, or rescaling variables, a different quantization result is acquired because the quantization error function E_q changes correspondingly. How well each variable is represented in the quantization depends on how strongly the variable effects the total quantization error.

The quantization error E_q can be expressed in terms of variable-wise errors E_j:

$$E_q = \sum_{j=1}^{d} \frac{1}{N} \sum_{i=1}^{N} |x_{ij} - m_{b_i j}|^2 = \frac{1}{N} \sum_{j=1}^{d} E_j \tag{3.3.5}$$

In order to measure the granularity of the quantization with respect to each variable, the errors E_j can be compared to quantizations performed on each variable separately with increasing number of quantization points $E_j(k)$, $k = 1, \ldots, N$. Depending on the distribution characteristics of the variable, the quantization error decreases at different rates. For example, for a uniformly distributed variable, the quantization error reduces according to formula $E_j(k) = \sigma_j^2 k^- 2$, where σ_j is the standard deviation of the variable. we use this SOM clustering capability to discover the groups of Gamma ray bursts data sets in the case study of GRB analysis described in this chapter. In chapter 6 we describe a SOM based system for automatic classification of images based on the clustering property of SOM

3.4 SOM as visualization technique

One of the most important property of the SOM is that it is an efficient method for visualization of high-dimensional data. The SOM is thus an excellent tool in exploratory data analysis [50]. Visualization methods try to find low-dimensional coordinates that preserve the distances (or the order of distances) between the originally high-dimensional objects. A classical projection method is multi-dimensional scaling (MDS) [49] which tries to preserve pairwise distances between all objects while reducing the dimension. The error function to be minimized is:

$$E_{mds} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(d_{ij} - d'_{ij}\right)^2 \tag{3.4.1}$$

where d_{ij} is the distance between data samples i and j in the input space $\|x_i - x_j\|$, and d'_{ij} is the corresponding distance between the projection coordinates in the output space. There is also a non-metric version of MDS which tries to preserve the rank order of the distances. other prominent projection techniques are sammons' mapping [51], Curvilinear Component Analysis(CCA) [52]

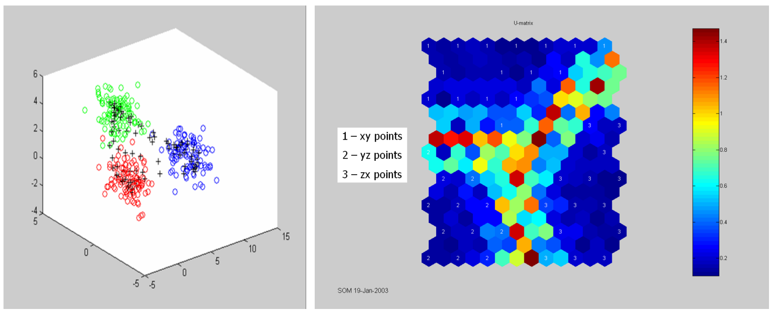
$$sammons'mapping : E_{sam} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(d_{ij} - d'_{ij}\right)^2 / d_{ij} \tag{3.4.2}$$

$$CCA : E_{cca} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(d_{ij} - d'_{ij}\right)^2 e^{-d'_{ij}} \tag{3.4.3}$$

For SOM the error function minimization is given by

$$E_{som} = \sum_{i=1}^{N} \sum_{j=1}^{M} h\left(d'_{ij}\right)\left(d^2_{ij}\right) \tag{3.4.4}$$

where $h\left(.\right)$ is the neighborhood kernel function. Since it is monotonically decreasing function of d'_{ij}, small distances in the outer space are emphasized, also d'_{ij} is dependent on the density distribution of the input data set. Thus, the definition of locality in SOM tunes to input data density. Rather than try to preserve the original distances, the SOM orders prototype vectors on a predefined map grid such that local neighborhood sets in the projection are preserved. SOM is especially good at maintaining the trustworthiness of the projection: if two data samples are close to each other in the visualization, they are more likely to be close in the original high-dimensional space as well. The visualization techniques which are mainly used in this thesis are described below.



(a) Points in threed - space (b) SOM grouping with U-matrix

Figure 3.7: SOM classification of points in threed space

3.4.1 U-matrix

U-matrix (unified distance matrix) representation of the Self-Organizing Map visualizes the distances between the map units or neurons. An U-Matrix displays the local distance structure of the data set.It is a standard tool for the display of the distance structures of the input data on SOM [50]. The distance between the adjacent neurons is calculated and presented with different colorings between the adjacent nodes. In U-matrix color coding scale is used to distinguish various clusters. The clusters and their outliers, boundaries are represented by different colors. This can be a helpful presentation when one tries to find clusters in the input data without having any a priori information about the clusters. Teaching a SOM and representing it with

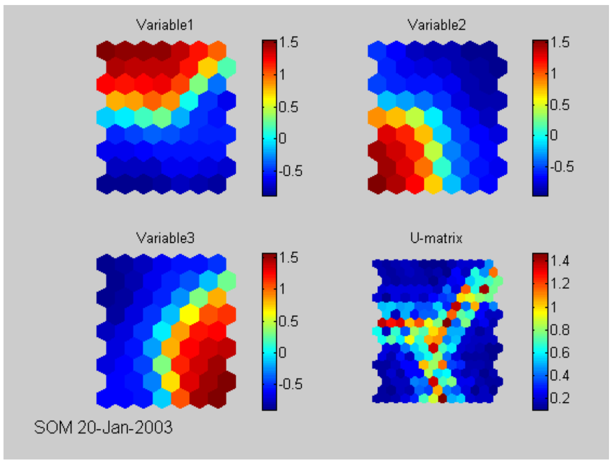


Figure 3.8: Component planes for the data points in 3-D space

the U-matrix offers a fast way to get insight of the data distribution without human intervention. The U-Matrix is constructed on top of the map. The color coding of a map unit is based on the factor “U-height”. The unified distance matrix (U-matrix) visualizes all distances between each map unit and its neighbors. This is possible due to the regular structure of the map grid: it is easy to position a single visual marker between a map unit and each of its neighbors. The map prototypes follow the probability density function of the data, the “u-height” distances are inversely proportional to the density of the data. Thus, cluster borders can be identified as different colors separating the map units of low distances that are with in the cluster. This interpretation can also be used in clustering. Let i be a unit on the map, $NN\left(i\right)$ be the set of immediate neighbors on the map, $m\left(i\right)$ the weight vector associated with neuron i, then

$$U - height\left(i\right) = \sum_{j \in NN(i)} d\left(m\left(i\right) - m\left(j\right)\right) \tag{3.4.5}$$

where $d\left(m\left(i\right) - m\left(j\right)\right)$ is the distance used in the SOM algorithm to construct the map. Thus U-Matrix is a display of the U-heights on top of the grid positions of the neurons on the map. Once the u-heights are determined contrast color coding are assigned to various clusters in the following way:

- U-height(i) = mean$(U - heights) \Rightarrow (clustercenter)$
- U-height$(i) <$ mean$(U - heights) \Rightarrow (intercluster)$
- U-height$(i) >$ mean$(U - heights) \Rightarrow (intracluster)$
- U-height$(i) <$ min$(U - heights) \Rightarrow$ U-height(i) $= 0$ $(boundary)$

Properties of the U-Matrix:

- the position of the projections of the input data points reflect the topology of the input space, this is inherited from the underlying SOM algorithm
- weight vectors of neurons with large U-heights are very distant from other vectors in the data space
- weight vectors of neurons with small U-heights are surrounded by other vectors in the data space
- The U-Matrix realizes the emergence of structural features of the distances within the data space.
- Outliers, as well as possible cluster structures can be recognized for high dimensional data spaces.
- The proper setting and functioning of the SOM algorithm on the input data can also be visually checked.

Figure 3.7 shows the SOM classification of points distributed in threed space. The data set is constructed from the random vectors taken from a cube in 3D space. The prototype vectors represented by '+' in 3.7(a) are chosen at random to assign to the SOM map. SOM discovered the three groups (XY, YZ, ZX plane points), the U-matrix shows these groups in blue color well separated by the boundaries 3.7 (b).

3.4.2 Component plane visualisation

Component plane representation displays the values of each model vector element, i.e. values of each variable, on the map grid. In figure 3.8, the three component planes (XY, YZ, ZX) are shown. For each visualized variable, or vector component, one SOM grid is visualized such that the colors (or for example sizes) of the map unit markers change according to the visualized values. Relationships between variables can be seen as similar patterns in identical places on the component planes: whenever the values of one variable change, the other variable changes, too. Although any kind of projection could be used to link the component planes together, the SOM grid works particularly well in this task. Because of the dynamic focus of the map, the behavior of the data can be seen irrespective of the local scale. By inspecting all the component planes simultaneously, one may possibly observe relationships between variables and even roughly distinguish structure of the input data. Ordering of the component planes makes it easier to investigate a large number of component planes simultaneously. The basic idea is to arrange the component planes in such a way that similar planes (that is, interrelated variables) lie close to each other; this organization is also carried out using the SOM algorithm. Coloring of the SOM was used with the difference that the changes in the colors between neighboring units were chosen to reflect cluster structure of the model vectors of the map.

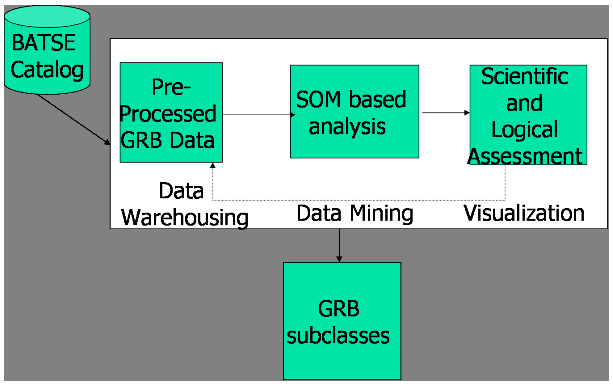


Figure 3.9: SOM based system for GRB Data Analysis

3.5 Case study on Astrophysical experiment: GRB Data Analysis

Gamma ray bursts (GRBs) were arguably the biggest mystery in high-energy astronomy. They've been the target of intense research and speculation by astronomers. Although 3 decades have passed since their discovery, very little is known about the fundamental physical mechanisms behind these phenomena [34]. They are considered to be short-lived bursts of gamma-ray photons, the most energetic form of light. At least some of them are associated with a special type of supernovae, the explosions marking the deaths of especially massive stars. Lasting anywhere from a few milliseconds to several minutes, gamma-ray bursts (GRBs) shine hundreds of times brighter than a typical supernova and about a million trillion times as bright as the Sun, making them briefly the brightest source of cosmic gamma-ray photons in the observable Universe. GRBs are detected roughly once per day from wholly random directions of the sky. GRBs exhibit great morphological diversity and complex temporal behavior: To analyze their properties it is beneficial to find the classes that exists in the GRB data. Studies based on GRB bulk properties (such as burst duration and spectral hardness) have proved to be fruitful, and the discovery of distinct classes in the GRB population might lead to some new astrophysical insight.

In 1991 NASA launched the Compton Gamma Ray Observatory carrying an instrument called the Burst and Transient Source Experiment, or "BATSE" for short. BATSE was designed specifically for the study of the enigmatic gamma ray bursts and has led to a new understanding of their origin and distribution in the universe. For the GRB analysis we used the data sets taken from the BATSE catalog data[35]. The gamma ray burst data sets are organised in in a series of burst catalogs containing all burst data from the start of the mission. The catalogs follow the naming convention "BATSE nB Gamma Ray Burst Catalog", where n is an integer beginning at 1. The latest published catalog as of 1999 March is the 4B Catalog. Each catalog contains all burst data from the start of the mission. For example, the 2B catalog contains all the data from the 1B catalog, the 3B catalog contains all data from the 2B (an thus from the 1B), etc [35]. The BATSE GRB Team has the most recent published catalog available on the World Wide Web. As of 1999 March, this corresponds to the 4B Catalog.

GRB class properties are indistinct, as overlapping characteristics of individual bursts are convolved with effects of instrumental and sampling biases [38]. Table 3.1 shows some important attributes useful for classifying gamma ray bursts.

Attribute	**Description**
Flux	The rate of flow of photons. The peak flux times are expressed in decimal seconds relative to the burst trigger time for the end of the interval in which the flux was calculated. The fluxes are taken on three time scales: 64 ms, 256ms, 1024ms.
Fluence	The product (or integral) of radiation flux and time. The four energy channels 1,2,3 and 4 cover the fluence with energy ranges 20-50 keV, 50-100 keV, 100-300 keV, and $E > 300$ keV respectively.
Burst duration Parameter (T90)	Time it takes for 90% of the total burst flux to arrive, taken from duration table of BATSE catalog
Burst duration Parameter (T50)	Time it takes for 50% of the total burst flux to arrive, taken from duration table of BATSE catalog
Hardness ratio	Ratio of fluence in different channels

Table 3.1: Important Attributes of BATSE catalog used in GRB classification by various experiments

Previous Studies

Attempts to find the classes in the data sets have been done with standard statistical analysis techniques [37], with machine learning approach based on decision trees [38] and with self-organising maps [40].

Mukherjee et.al,. has revealed the presence of the 3 GRB sub classes performing statistical analysis on BATSE 3B data [37]. Three attributes have used in this experiment S23 fluence (time integrated flux in the 50 and 300 KeV range), T90 duration (time interval during which 90% of the bursts emission is received) and HR321 hardness ratio (the fluence in the 100 to 300 KeV band divided by the fluence in the 25 to 100 keV band).

A more extensive study of gamma-ray burst classes using AI techniques was conducted by Hakkila et al. [38]: the supervised decision tree classifier C4.5 was used for this purpose. The data set consists of six basic parameters: T90 duration, flux, fluence, and three hardness ratios (HR21, HR32 and HR43). A natural division of two classes is expected based on duration (short bursts have durations < 2 seconds; long ones have durations ≥ 2 seconds). The results of the classification is shown in the figure 3.10(a). It should be noted that a clean division does not exist between these subclasses; there is considerable overlap in the distributions. This overlap represents the existence of new class which is identified as a new bias in the BATSE instrument, suggesting that the third class may be an instrumental effect and not a true separate source population. Rajaneimi et.al has performed experiments using self-organising map on the BATSE 3B Catalog [40]. the experiment is conducted using flux, duration and fluence parameters. The results of the SOM algorithm using a gaussian kernel are shown in the figure 3.10(b). The SOM map could able to classify the class 1 from class2, but the class 3 is not clearly separated from class 1. Map units belonging to class 3 are mostly placed on the border between classes 1 and 2.

In this thesis, experiments are conducted with the SOM system with cut-gaussian kernel developed during the PhD Course. Two case studies are performed using different parameter sets.

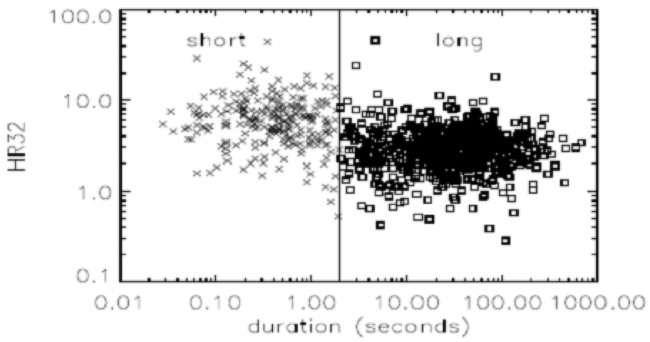
Case Study 1:

We first experimented with a data set of 5 dimensional parameter space: which represents the burst arrival times along with their uncertainties at 50 and 90 seconds and the flux of the burst arrival on 64ms timescale. We used a SOM with 10×10 map units, randomly initialized weight vectors, and hexagonal topology and cut-gaussian kernel neighborhood. Training is done on 100 randomly initialized maps, and the map with the smallest average quantization error was chosen for further examination. The map was then labeled using the training data set for map calibration. Figure 3.11 shows the component data visualisation for the data set.

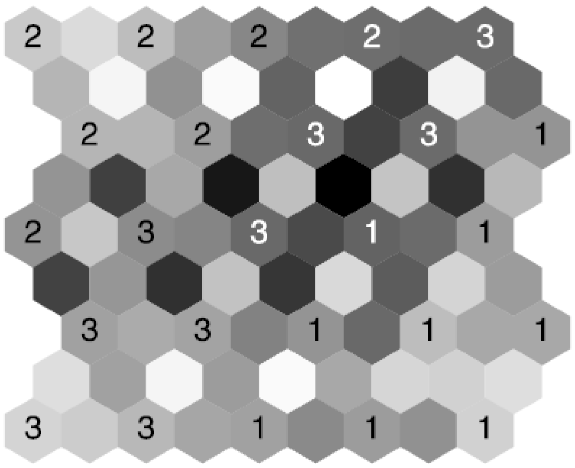
Figure 3.10(c)shows the data groups discovered by SOM in the BATSE catalog data. Basically it found 3 groups. classes 1 and 2 are visually distinct, in the map. However class 3 is not properly distinguishable from class 1. Notably, the map misclassified class 3 bursts far more often than class 1 or class 2 bursts. This supports the Hakkila et.al., hypothesis that basically there exists 2 groups of GRB's, class 1 and 2, which are well separated in the parameter space. The formation of class 3 is basically due to the bias in the instrument.

Case Study 2:

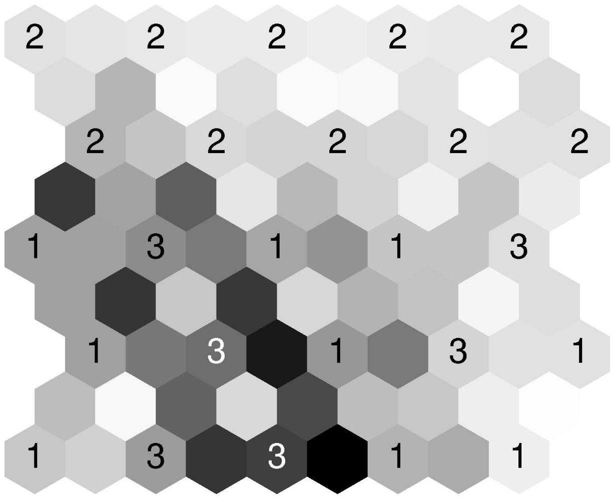
Bagoly et al. study [39] indicated only three variables are needed to characterize the relationships between the bursts in the database, namely the fluence, burst duration (T90), and flux. In this case study we used these 3 parameters to study the GRB classification. We used a SOM with 10×10 map dimension, with a hexagonal topology and cut-gaussian kernel neighborhood. Training is done on 100 randomly initialized maps, and the map with the smallest average quantization error was chosen for further examination. SOM now visualizes two different groups separated by well defined boundary figure 3.12.



(a) Hakkila classification: HR32 vs. duration diagram. Long and short bursts train the supervised classifier C4.5, even though there is not a clean subclass separation. [38]



(b) Rajniemi Classification: SOM Map classification of BATSE 3B Catalog [40]



(c) Our Classification: SOM Map classification of BATSE 4B Catalog

Figure 3.10: GRB Classification models

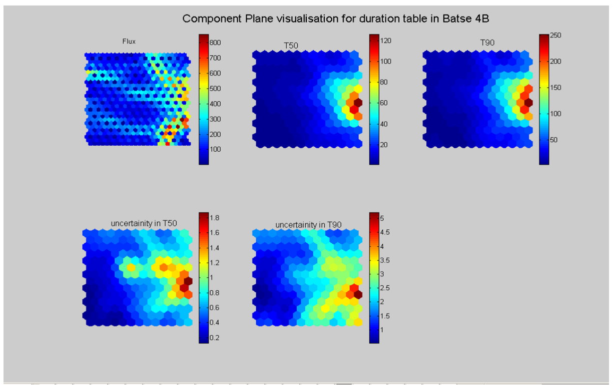


Figure 3.11: Component plane distribution for BATSE data

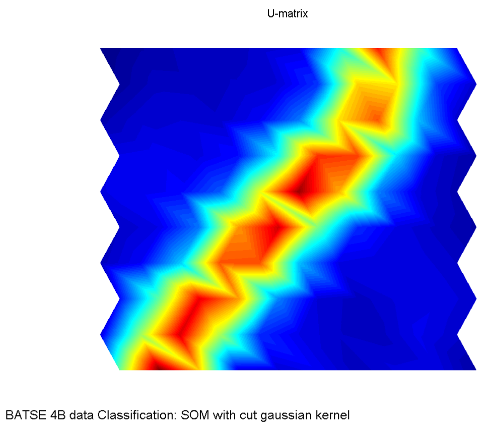


Figure 3.12: BATSE data classification: SOM with cut-gaussian kernel, the two groups are visually distinct[blue color codes] with a well separated boundary

4

Ensemble Learning

4.1 Introduction

In this chapter we introduce ensemble learning techniques in more detail. Experiments on several benchmark data sets and real world data sets showed an improved classification results from ensemble learning techniques. It is the more active areas of research within machine learning and is concerned with methods for improving accuracy in supervised learning [62]. These algorithms combine "base classifiers" to predict the label for the new data points (fig 4.1). The base classifiers can be any of the standard supervised learning algorithms such as neural networks, decision trees, support vector machines, instant based learners such as k-nearest neighbors [K-NN]. In this thesis we concentrate on 2 ensembles techniques AdaBoost and Bagging. The following machine learning algorithms are used as basis for making these ensembles:

- Random forests, a tree based classifier based on decision trees.
- Back-propagation neural network [BPNN] of neural network category.

Ensemble methods aim at improving the predictive performance of a given statistical learner or classifier. These algorithms construct a set of hypotheses (sometimes called a "committee" or "ensemble") to explain the data. These hypotheses are then combined in some fashion to predict the label of new data points [64]. The training data set is a collection of the data points associated with labels. The data points, usually a vector of features (x).

To formalize the things lets start from the supervised learning technique. In supervised learning, a classifier is given a set of training examples of the form $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, for some unknow function $y = f(x)$. The description x_i is usually vector of the form $\langle x_{i,1}, x_{i,2}, \ldots, x_{i,k} \rangle$ whose components are real or discrete values, such as height, weight, age, eye-color, and so on. These components of the description are often referred to as the features or attributes of an example. The values of y are typically drawn from a discrete set of classes Y in the case of classification or from the real line in the case of regression. Our work is primarily focused on the classification task. A learning algorithm L, is trained on a set of training examples, to produce a hypothesis h, also called as classifier. Given a new example x, the hypothesis predicts the

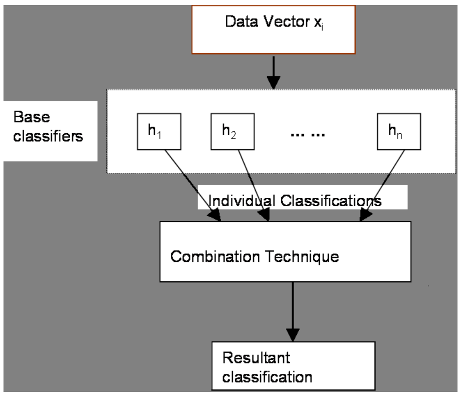


Figure 4.1: The basic architecture of an ensemble.

corresponding y value. The aim of the classification task is to learn a hypothesis that minimizes the error in predictions on an independent test set of examples (generalisation error). For classification, the most common measure for error is the 0/1 loss function given by:

$$error_{C,f} = \begin{cases} 0: & \text{if } C(x) = f(x) \\ 1: & \text{otherwise} \end{cases}$$

Supervised algorithms search for a best possible hypothesis h to f that can be applied to assign labels to new x values. Ensemble learning algorithms construct a set of hypothesis $\{h_1, h_2, \ldots, h_n\}$ and construct a combined classifier $H^*(x) = T(h_1(x), h_2(x), \ldots, h_n(x))$ to predict the label of new data points, where T a criterion to combine the hypothesis. Experimental evidence has shown that ensemble methods are often much more accurate than any single hypothesis [95].

According to Dietterich [63], learning algorithms that output only a single hypothesis suffer from three problems: the statistical problem, the computational problem, and the representation problem. These can be partly overcome by ensemble methods

- The statistical problem - In searching a large hypothesis space given a certain number of available training examples, the possibility arises where a single hypothesis within this space will not predict future data points well. A vote on several equally good classifiers might reduce this risk.

- The computational problem - The learning algorithm may not be able to find the best hypothesis within the hypothesis space. Certain algorithms rely on heuristic methods to approximate hypotheses, as finding the hypothesis that best fits the training data is sometimes computationally intractable. Ensembles may be seen as a way of compensating for the use of these imperfect search methods.

- The representation problem - The hypothesis space may not contain any hypothesis that are good approximations to the true target function that maps any given training example Ai to its respective class Ci. Combining several viable functions (that are only fair approximations) may expand the space of functions and thus form a more accurate approximation to the true target function.

A learning algorithm that suffers from the statistical problem is said to have high "variance". An algorithm that exhibits the computational problem is sometimes described has having "computational variance", and a learning algorithm that suffers from the representational problem is said to have high "bias". Hence, ensemble methods can reduce both the bias and the variance of learning algorithms.

4.2 Ensemble Mechanics

Ensembles can be generalized in various directions. In [67], ensembles are formalized in terms of Bayesian learning theory. Bayesian learning theory is used for decision making using the knowledge of prior events to predict future events. Bayes first proposed his theorem in his 1763 work (published two years after his death in 1761), An Essay Towards Solving a Problem in the Doctrine of Chances. Bayes' theorem provided, for the first time, a mathematical method that could be used to calculate, given occurrences in prior trials, the likelihood of a target occurrence in future trials. According to Bayesian logic, the only way to quantify a situation with an uncertain outcome is through determining its probability. Bayes' Theorem is a means of quantifying uncertainty. Based on probability theory, the theorem defines a rule for refining an hypothesis by factoring in additional evidence and background information, and leads to a number representing the degree of probability that the hypothesis is true.

Bayesian theory stipulates that in order to maximize predictive accuracy, instead of using just a single learning algorithm, one should ideally employ all hypotheses in the hypothesis space. Assume that a statistical model allows the inference about the variable y in the form of the predictive probability density $P(y|x)$, where x is a vector of model parameters. Furthermore, assume that we have a data set D which contains information about the parameter vector x in the form of the probability density $P(x|D)$. We then obtain

$$P(y|D) = \int P(y|x)\, P(x|D)\, dx \approx \frac{1}{M} \sum_{i=1}^{M} P(y|x_i)$$

where M samples $\{x_i\}_{i=1}^{M}$ are generated from the distribution $P(x|D)$. This approximation tells us that for Bayesian inference [68], one should average the predictions of a committee of estimators.

The motivation for pursuing ensembles can be understood by analyzing the prediction error of the combined system, which is particularly simple if we use a squared error cost function. The expected squared difference between the prediction of a committee member h_i and the unknown target t cab be written as

$$
\begin{aligned}
E\left(h_i - t\right)^2 &= E\left(h_i - m_i + m_i - t\right)^2 \\
&= E\left(h_i - m_i\right)^2 + E\left(m_i - t\right)^2 + 2E\left(\left(h_i - m_i\right)\left(m_i - t\right)\right) \\
&= var_i + b_i^2
\end{aligned}
\tag{4.2.1}
$$

decomposes into the variance $var_i = E\left(h_i - m_i\right)^2$ and the square of the bias $b_i = m_i - t$, with $m_i = E\left(h_i\right)$. $E\left(.\right)$ stands for the expected value, which is calculated with respect to different data sets of the same size and possible variations in the training procedure, parameters used for making the base classifier for ex: different initializations of the weights in the neural network. Now t can be estimated by forming a linear combination of the h_i which can be given as:

$$\hat{t} = \sum_{i=1}^{M} g_i h_i = g' f$$

where $f = \left(f_1, \ldots, f_M\right)'$ is the vector of the predictions of the committee members and $g = \left(g_1, \ldots, g_M\right)'$ is the vector of weights. The expected error of the combined system is

$$
\begin{aligned}
E\left(\hat{t} - t\right)^2 &= E\left(g'f - E\left(g'f\right)\right)^2 + E\left(E\left(g'f\right) - t\right)^2 \\
&= E\left(g'\left(f - E\left(f\right)\right)\right)^2 + E\left(g'm - t\right)^2 \\
&= g'\Omega g + \left(g'm - t\right)^2
\end{aligned}
\tag{4.2.2}
$$

where Ω is an $M \times M$ covariance matrix with

$$\Omega_{ij} = E\left[\left(f_i - m_i\right)\left(f_j - m_j\right)\right]$$

and $m = \left(m_1, \ldots, m_M\right)'$ is the vector of the expected values of the predictions of the committee members. Here $g'\Omega g$ is the variance of the committe and $g'm - t$ is the bias of the committee. If we simply average the predictors, i.e., set $g_i = \frac{1}{M}$, the last expression simplifies to

$$E\left(\hat{t} - t\right)^2 = \frac{1}{M^2}\sum_{i=1}^{M}\Omega_{ii} + \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\Omega_{ij} + \frac{1}{M^2}\Big(\sum_{i=1}^{M}\left(m_i - t\right)\Big)^2 \tag{4.2.3}$$

If we now assume that the mean $m_i = mean$, the variance $\Omega_{ii} = var$ and the inter-member covariances $\Omega_{ij} = cov$ are identical for all members, we obtain

$$E\left(\hat{t} - t\right)^2 = \frac{1}{M}var + \frac{M^2 - M}{M^2}cov + \left(mean - t\right)^2 \tag{4.2.4}$$

It is apparent that the bias of the combined system $\left(mean - t\right)$ is identical to the bias of each member and is not reduced. Therefore, estimators should be used which have low bias, and regularization [1]- which introduces bias - should be avoided. Secondly, the estimators should have low covariance, since this term in the error function cannot be reduced by increasing M. The good news is that the term which results from the variances of the committee members decreases as $1/M$. Thus, if we have estimators with low bias and low covariance between members, the expected error of the combined system is significantly less than the expected errors of the individual members. Thus , a committee can be used to reduce both bias and variance: bias is reduced in the design of the members by using little regularization, and variance is reduced by the averaging process which takes place in the committee.

Finally, the generalization error statistic of a committee is a function of the average error of ensemble members and the average variance (or ambiguity) among them [69]. Optiz and Shavlik [70] empirically verified this claim, showing that ensembles that consist of highly correct classifiers that disagree as much as possible generalize well.

The relationship between the error rate of the ensemble and the error rates of the individual can be given by [71]

$$E\left(ensemble\right) = \frac{1+\rho\left(N-1\right)}{N} E\left(individual\right) + E\left(Bayes\right) \tag{4.2.5}$$

where N is the number of classifiers, ρ is the correlation among the classifier errors, $E\left(Bayes\right)$ is the error rate obtained using the Bayes rule assuming that all the conditional probabilities are known. $E\left(ensemble\right)$ and $E\left(individual\right)$ represent the error rate of a classifier ensemble and the error rate of an individual component classifier respectively. $\rho = 0$ means the error of the whole ensemble decreases proportionally to the number of the component classifiers while $\rho = 1$ means the error of the ensemble architecture equals to the error of a single component classifier.

4.3 Methods for constructing the ensembles

Generally speaking, the classifier ensembles can be divided into parallel ensembles and sequential ensembles. The categorization scheme can also be done according to the combining strategies, e.g. the diversity of the classifier ensemble. Jain, Duin and Mao **??** list a number of popular ensemble methods in their review paper on statistical pattern recognition. Dietterich [63] describes the ensemble methods from the point of view of machine learning and we review this categorization scheme in this thesis. According to him there are two main approaches for constructing ensembles.

- Methods for Independently Constructing Ensembles
- Methods for Coordinated Construction of Ensembles

In the first approach each hypothesis is constructed independently in such a way that the resulting set of hypotheses is accurate and diverse that is, each individual hypothesis has a reasonably low error rate for making new predictions and yet the

[1]A numerical method to solve problems of deconvolution by introducing a priori information about the smoothness of the expected result [65].

hypotheses disagree with each other in many of their predictions. If such an ensemble of hypotheses can be constructed, it is easy to see that it will be more accurate than any of its component classifiers, because the disagreements will "cancel out." Such ensembles can overcome both the statistical and computational problems. Bagging is the popular technique belonging to this category.

The second approach to designing ensembles is to construct the hypotheses in a coupled fashion so that the weighted vote of the hypotheses gives a good fit to the data. This approach directly addresses the representational problem discussed above. Boosting is the very popular technique belonging to this category. we experiment with this technique in this thesis.

4.3.1 Bagging

Bagging is a statistical re-sample and combine technique used to improve the classification accuracies of a classifier. It based on bootstrapping and aggregating techniques.

The idea of the bootstrap is to generate more (pseudo) data using the information of the original data. True underlying sample properties are reproduced as closely as possible and unknown model characteristics are replaced by sample estimates. Bagging uses bootstrapping to generate multiple versions of a data set. Each of these data sets have their own classifier. Th predictions from these classifiers are then combined to get the final classification result. In theory this combination should perform better than a single classifier built to solve the same problem.

Bootstrapping is based on random sampling with replacement. It generates the samples more (pseudo) data using the information of the original data. True underlying sample properties are reproduced as closely as possible and unknown model characteristics are replaced by sample estimates. Therefore, taking a bootstrap i.e., (random selection with replacement) of the training set T, one can sometimes avoid or get less misleading training objects in the bootstrap training set. Consequently, a classifier constructed on such a training set may have a better performance.

Aggregating actually means combining classifiers [74]. Often a combined classifier gives better results than individual classifiers, because of combining the advantages of the individual classifiers in the final solution. Therefore, bagging might be helpful to build a better classifier on training sample sets with misleaders. Breiman motivated bagging as a variance reduction technique for a given base procedure, such as decision trees or methods that do variable selection and fitting in a linear model. It has attracted much attention, probably due to its implementation simplicity and the popularity of the bootstrap methodology. At the time of its invention, only heuristic arguments were presented why bagging would work. Later, it has been shown in ([90]) that bagging is a smoothing operation which turns out to be advantageous when aiming to improve the predictive performance of regression or classification trees. In case of decision trees, the theory in confirms Breiman's intuition that bagging is a variance reduction technique, reducing also the mean squared error (MSE). The empirical fact that bagging improves the predictive performance of regression and classification trees is nowadays widely documented ([90], [91], [92]. On average, when taking a bootstrap sample of the training set, approximately 37% of the objects are not presented in the bootstrap sample, meaning that possible 'outliers' in the training set sometimes do not show up in the bootstrap sample. Thus, better classifiers (with a smaller apparent error - classification error on the training data set) may be obtained by the bootstrap sample than by the original training set. These classifiers will be presented 'sharper' in the apparent error than those obtained on the training sets with outliers. Therefore, they will be more decisive than other bootstrap versions in the final judgment. Thus, aggregating classifiers in bagging can sometimes give a better performance than individual classifiers. In Bagging [75] (short for Bootstrap Aggregation Learning) we randomly generate a new training set, based on the original set, for each ensemble member. Given a training set of size N, from this we uniformly at random sample N items with replacement, then train a new ensemble member with this resample. We repeat this for any new ensemble members we wish to create. The resampled sets are often termed bootstrap replicates [77]; Breiman showed that on average 63.2% of the original training set will present in each replicate. Listing 5.1 gives the algorithm.

Listing 4.1: Bagging Algorithm

```
Let $M$ be the final number of predictors required.
Take a training set $T\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}$
for i=1 to M
{
Make a new training set $T_{bag}$ by resampling $T$.
Train a classifier $h_i$ with this set $T_{bag}$ and add it to the ensemble.
}
For any new testing pattern $x$, the bagged output is:
$h_{bag} = \frac{1}{M}\sum_i h_i(x)$
```

Bagging has proved to be a popular technique, applicable to many problems, but the explanation for its success remains controversial. Friedman [78] suggests that Bagging succeeds by reducing the variance component of the error and leaving the bias unchanged;

while [79] shows evidence that Bagging can in fact converge without reducing variance. Domingos [80] gives a Bayesian account, and investigates two hypotheses: Bagging succeeds because

1. it is an approximation to Bayesian Model Averaging [81], and
2. it shifts the prior distribution of the combined estimator to a more appropriate region of the model space; the empirical results on 26 real and artificial datasets all support the second hypothesis and contradict the first.

4.3.2 Boosting

Boosting algorithms have been proposed in the machine learning literature by Schapire ([82]) and Freund ([83]). These first algorithms have been developed as ensemble methods. Unlike bagging which is a parallel ensemble method, boosting methods are sequential ensemble algorithms where the weights c_k are depending on the previous fitted functions $h_1, \dots, h_{k-1}$. Boosting has been empirically demonstrated to be very accurate in terms of classification, notably the so-called AdaBoost algorithm ([74]).

Boosting algorithms can be viewed as a functional gradient descent techniques or nonparametric optimization algorithm in function space, as first pointed out by Breiman ([74]). This view turns out to be very fruitful to adapt boosting for other problems than classification, including regression and survival analysis. The goal is to estimate a function $h : R^d \to R$, minimizing an expected loss

$$E\left[\ell(Y, g(X))\right] \ , \ \ell(\cdot,\cdot) : R \times R \to R^+$$

based on the data $T = \{(x_i, y_i), (i = 1, \dots, n)\}$. The loss function ℓ is typically assumed to be convex in the second argument. We consider here both cases where the univariate response Y is discrete (classification problem) or continuous (regression problem) , since boosting is potentially useful in both cases. The most popular loss functions, for regression and binary classification, are given in Table 4.1. While the squared error loss is mainly used for regression, the log-likelihood and the exponential loss are for classification problems. In this thesis we deal with Adaboost algorithm.

Boosting	**Loss function**
AdaBoost	$\ell(y,h) = exp\left(-(2y-1)h\right)$
LogitBoost	$\ell(y,h) = log_2\left(1 + exp\left(-2(y-1)h\right)\right)$
L_2Boost	$\ell(y,h) = (y-h)^2$

Table 4.1: The exponential, binomial negative log-likelihood and squared error loss functions for boosting. The miss classification plots are in figure 4.2

AdaBoost algorithm introduced by Freund [83] and Schapire [82] is an extremely effective method for constructing an additive model. Boosting works by repeatedly running a given weak learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier [74]. The first provably effective boosting algorithms were presented It works by incrementally adding one hypothesis at a time to an ensemble. Each new hypothesis is constructed by a learning algorithm that seeks to minimize the classification error on a weighted training data set. The goal is to construct a weighted sum of hypothesis such that $h_{boost}(x_i) = \sum_k w_k h_k(x_i)$ has the same sign as y_i the correct label of x_i.

Boosting resample the data set randomly with a non uniform probability distribution. It is an additive model of ensembles that predicts the class of a new data point by taking an weighted sum of a set of component models. The component models and the weights used in these algorithms fits the data well. Statistically these models resemble generalized additive models, where the choice of one component hypothesis influences the choice of other hypotheses and the weights assigned to them. Boosting works by repeatedly running a learning algorithm on various distributions over the training data, and then combining the classifiers produced by the learner into

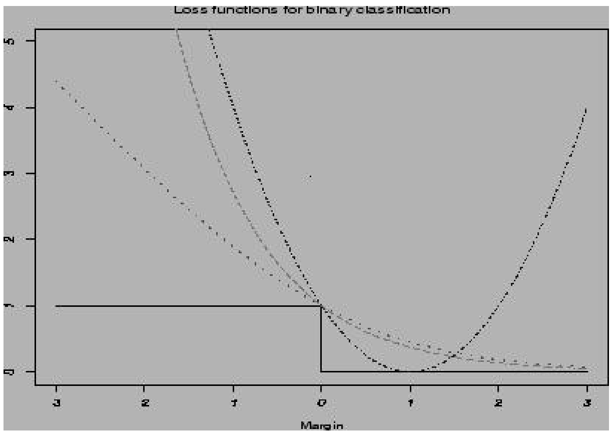


Figure 4.2: Loss functions of the margin for binary classification. Zero-one misclassification loss (solid line), log-likelihood loss (dashed line), exponential loss (dotted line), squared error loss (dashed/dotted).

the single composite classifier. The boosting algorithm takes as input a training set $T = ((x_1, y_1), \ldots, (x_m, y_m))$of m examples, where each instance x_i is a vector of attributes drawn from the input space X and y_i belonging to the finite label set Y is the class label associated with x_i. In boosting classifiers and training sets are obtained in a strictly deterministic way. Both training sets and classifiers are obtained sequentially in the algorithm, in contrast to bagging, where training sets and classifiers are obtained randomly and independently from the previous step of the algorithm. At each step of the boosting, training data are reweighed in such a way that incorrectly classified objects get larger weights in a new modified training set [89]. AdaBoost manipulates the training examples to generate multiple hypotheses. It maintains the probability distribution $p_l(x)$ over the training examples. In each iteration l it weights the training samples with the probability distribution $p_l(x)$. The learning algorithm is then applied to produce the classifier h_l. The error rate ϵ_l of this classifier on the training examples is computed and used to adjust the probability distribution on the training examples. The effect of the change in the weights is to place more weight on training examples that were misclassified by h_l and less weight on examples that were correctly classified in the last stage. In subsequent iterations, therefore, AdaBoost tend to construct progressively more difficult learning problems. The final classifier, h_{boost}, is constructed by a weighted vote of the individual classifiers $h_1, \ldots, h_k$. Each classifier is weighted according to its accuracy for the distribution p_l that it was trained on. Listing 4.2 shows the adaboosting algorithm.

Listing 4.2: Adaboost algorithm

```
Input:
Training set $T = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
with labels $y_i \in Y = \{1, \ldots, k\}$
```

```
basic learn algorithm base−learner
integer $M$ specifying number of iterations
Initialize $D_1(i) = \frac{1}{m}$ for all $i$
for $t = 1, \ldots, M$
{
call base−learner, providing it with distribution $D_t$.
Get back a hypothesis $h_t : X \longrightarrow Y$
Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
if $\epsilon_t > 1/2$
{
  $M = t - 1$
  exit
}
$\beta_t = \epsilon_t / (1 - \epsilon_t)$
update distribution $D_t$:
if $(h_t(x_i) = y_i)$
$D_{t+1}(i) = \frac{D_t(i)}{Z_t} * \beta_t$
else
$D_{t+1}(i) = \frac{D_t(i)}{Z_t}$
here $Z_t$ is a normalization constant
}
Output:
$h_{boost}(x) = argmax_{y \in Y} \sum_{t:h_t(x)=y} log \frac{1}{\beta_t}$
```

There seem to be two separate reasons for the improvement in performance that is achieved by boosting. The first and better understood effect of boosting is that it generates a hypothesis whose error on the training set is small by combining many hypotheses whose error may be large (but still better than random guessing). It seems that boosting may be helpful on learning problems having either of the following two properties. The first property, which holds for many real-world problems, is that the observed examples tend to have varying degrees of hardness. For such problems, the boosting algorithm tends to generate distributions that concentrate on the harder examples, thus challenging the weak learning algorithm to perform well on these harder parts of the sample space. The second property is that the learning algorithm be sensitive to changes in the training examples so that significantly different hypotheses are generated for different training sets. In this sense, boosting is similar to Breiman's bagging which performs best when the weak learner exhibits such "unstable" behavior. However, unlike bagging, boosting tries actively to force the weak learning algorithm to change its hypotheses by changing the distribution over the training examples as a function of the errors made by previously generated hypotheses.

Intuitively, taking a weighted majority over many hypotheses, all of which were trained on different samples taken out of the same training set, has the effect of reducing the random variability of the combined hypothesis. Thus, like bagging, boosting may have the effect of producing a combined hypothesis whose variance is significantly lower than those produced by the weak learner. However, unlike bagging, boosting may also reduce the bias of the learning algorithm, as discussed above [84].

Drucker, Schapire and Simard [85] performed the first experiments using a boosting algorithm. They used Schapire's [83] original boosting algorithm combined with a neural net for an OCR problem. Followup comparisons to other ensemble methods were done by Drucker et al. [86]. In [87] used AdaBoost with a decision-tree algorithm for an OCR task. Jackson and Craven [88] used AdaBoost to learn classifiers represented by sparse perceptrons, and tested the algorithm on a set of benchmarks.

4.3.3 Other techniques

Sharkey [93] points out that a limiting factor in research on combining classifiers is due to a lack of awareness of the full range of available modular structures. One reason for this is that there is as yet little agreement on a means of describing and classifying types of multiple classifier systems in the literature. One possible way of literature survey of ensembles is to study based on the outputs expected by ensembles from individual classifiers [94]. These expectations can be grouped into three levels: a) measurement (or confidence), b) rank and c) abstract. At the confidence level, a classifier outputs a numerical value for each class indicating the belief or probability that the given patterns belong to that class. At the rank level, a classifier assigns a rank to each class with the highest rank being the first choice. Rank value cannot be used in the isolation because the highest rank does not necessarily mean a high confidence in the classification. At the abstract level, a classifier only outputs a unique class label or several class labels. The confidence level conveys the richest information, while the abstract level contains the least amount of information about the decision being made. Following [1] here we summarize a number of popular ensemble methods used in the machine learning literature (table 4.2). This is by no means an exhaustive list.

Model	Info-level	Comments
Bagging	Confidence	Uses bootstrapping to generate many data sets for base classifiers
Boosting	Abstract	Improves margins; unlikely to overtrain; sensitive to mislabels
Neural tree	Confidence	Handles large number of classes
Voting	Abstract	Assumes independent classifiers
Sum, mean, median	Confidence	Robust, assumes independent confidence estimation
Random subspace	Confidence	uses features to generate many training sets
Generalized ensembles	Confidence	Considers error correlation
Adaptive weighting	Confidence	Explores local expertise
Stacking	Confidence	Good utilization of training data
Borda count	Rank	Converts ranks into confidences
Logistic regression	Rank-Confidence	Converts rank into confidence
Dempster-Shafer	Rank-Confidence	Fuses non-probabilistic confidences
Fuzzy integrals	Confidence	Fuses non-probabilistic confidences
Mixture of Local Experts (MLE)	Confidence	Explores local expertise; joint optimization
Hierarchical MLE	Confidence	Same as MLE hierarchical
Associative switch	Abstract	Same as MLE; but no optimization

Table 4.2: Some ensemble techniques

5 Meta Random Forests

5.1 Introduction

Leo Breimans Random Forests (RF) [96] is a recent development in tree based classifiers and quickly proven to be one of the most important algorithms in the machine learning literature. It has shown robust and improved results of classifications on standard data sets. It is throwing very good competition to neural networks, ensemble techniques and support vector machines on various classification problems. Random forest are considered to be special type of ensembles using bagging and random splitting methods for growing multiple trees. In this chapter we experiment with the random forests as themselves acting as the base classifiers for making ensembles with bagging and boosting. These meta random forests are then applied on UCI standard data sets and compared the results with the original random forest algorithm.

The important properties of random forests can be summarized as follows [97]

- Random forests are computationally effective and offer good prediction performance.
- It has proved to be robust to noise,
- Offers possibilities for explanation and visualization of its output.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses and thus does not overfit.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.

In previous chapter we seen the construction of ensembling techniques of bagging and boosting. The success of these ensemble methods is usually explained with the error convergence of base classifiers [108]. To have a good ensemble one needs base classifiers which are diverse (in a sense that they predict differently), yet accurate.

The ensemble mechanism which operates on the top of base learners then ensures highly accurate predictions. Here we discuss the possibilities of making ensembles of random forests and how they perform on standard data sets. WEKA, a very nice machine learning framework is used for this purpose [126]. It facilitates powerful object oriented programming framework for making ensembles. It provides facilities for using various machine learning algorithms as base classifiers for ensembles. In this thesis the ensembles of random forests have been developed by using the WEKA software framework.

The chapter is organised as follows. First we discuss the random forests in section 5.2. Section 5.3 discuss the error convergence property for random forests. Section 5.4, 5.5 discuss the making of meta random forests with bagging and boosting respectively. Section 5.6 shows the experiments with standard data sets from UCI repository. Finally in section 5.7 a discussion on performance results on UCI data sets and comparative studies between meta random forests and the original random forest are discussed.

5.2 Random Forests

Random forests construct a series of tree-based learners. Each base learner receives different training set which are drawn independently with replacement from the original learning set. Statistically speaking two elements serve to obtain a random forest - resampling and random split selection. Resampling is done here by sampling multiple times with replacement from the original training data set. Thus in the resulting samples, a certain event may appear several times, and other events not at all. About $2/3^{rd}$ of the data in the training sample are taken for each bootstrap sample and the remaining one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. The design of random forests is to give the user a good deal of information about the data besides an accurate prediction. Much of this information comes from using the **oob** cases in the training set that have been left out of the bootstrapped training set. Random split selection selects a subset of features to grow the nodes in the tree. It is also used to get estimates of variable importance. Thus random forests perform randomized selection in both rows (by random sampling) and in columns (by random splitting) for better classification results.

According to [96] the formal definition for random forests is as follows,
A random forest is a classifier consisting of a collection of tree structured classifiers $\{h(x, \Theta_k), k = 1, \ldots\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* x.

Each tree is grown as follows:

- If the number of cases in the training set is n then sample n cases at random but with replacement, from the original data. This sample will be the training set for growing the tree.

- If there are M input variables, a number $m << M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

- Each tree is grown to the largest extent possible. There is no pruning [1].

The overall forest error rate depends on two things:

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.
- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

The variable selection is an important criterion in classification problems for RF. One way of variable selection is to to estimate the importance of the m^{th} variable, in the *oob* cases for the k^{th} tree, randomly permute all values of the m^{th} variable. These altered *oob* x-values are put down the tree to get the classification results and new internal error rate. The amount by which this new error exceeds the original test set error is defined as the importance of the m^{th} variable. The other measure criterion could be dealing with the nth case in the data. This nth case data margin at the end of a run is the proportion of votes for its true class minus the maximum of the proportion of votes for each of the other class.

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

To formalize the working of the random forests, Let the forest contain K classifier trees $h_1(\mathbf{x}), \ldots, h_K(\mathbf{x})$ and the joint classifier be $h(\mathbf{x})$. Each training set of n instances is drawn at random with replacement from the training set of n instances. Each learning instance is represented by an ordered pair $(\mathbf{x}, y)$, where each vector of attributes $\mathbf{x}$ consists of individual attributes $A_i, i = 1, \ldots, d$, (d is the number of attributes) and is labeled with the target value $y_j, j = 1, \ldots, c$(c is the number of class values). The correct class is denoted as y, without index. Each discrete attribute A_i has values v_1 through v_{m_i} (m_i is the number of values of attribute A_i).We write $p(v_{i,k})$ for the probability that the attribute A_i has value v_k, $p(y_j)$ is the probability of the class y_j and $p(y_j \mid v_{i,k})$ is the probability of the class y_j conditioned by the attribute A_i having the value v_k.

With this sampling of bootstrap replication, on average 36.8% of training instances are not used for building each tree. These out-of-bag instances come handy for computing an internal estimate of the strength and correlation of the forest. Let

[1]The technique of removing parts of a decision tree that do not capture true features of the underlying pattern. Decision Tree algorithms typically employ prunning to avoid overfitting of the data.

out-of-bag instances for classifier $h_k(\mathbf{x})$ be $O_k(\mathbf{x})$. Let $Q(\mathbf{x}, y_j)$ be the out-of-bag proportion of voted for class y_j at input $\mathbf{x}$ and an estimate of $P(h(\mathbf{x}) = y_j)$:

$$Q(\mathbf{x}, y_j) = \frac{\sum_{k=1}^{K} I\left(h_k(\mathbf{x}) = y_j; (\mathbf{x}, y) \in O_k\right)}{\sum_{k=1}^{K} I\left(h_k(\mathbf{x}); (\mathbf{x}, y) \in O_k\right)} \tag{5.2.1}$$

where I (.) is the indicator function.

Calculate the margin function which measures the extent to which the average vote for the right class y exceeds the average vote for any other class as follows.

$$mr(\mathbf{x}, y) = P(h(\mathbf{x}) = y) - \underset{\substack{j=1 \\ j \neq y}}{\overset{c}{max}} P(h(\mathbf{x}) = y_j) \tag{5.2.2}$$

It is estimated with $Q(\mathbf{x}, y)$ and $Q(\mathbf{x}, y_j)$. Strength is defined as the expected margin, and is computed as the average over the training set:

$$s = \frac{1}{n} \sum_{i=1}^{n} \left(Q(\mathbf{x}_i, y) - \underset{\substack{j=1 \\ j \neq y}}{\overset{c}{max}} Q(\mathbf{x}_i, y_j) \right) \tag{5.2.3}$$

The average correlation is computed as the variance of the margin over the square of the standard deviation of the forest:

$$\overline{\rho} = \frac{var(mr)}{sd(h())^2} = \frac{\frac{1}{n} \sum_{i=1}^{n} \left(Q(\mathbf{x}_i, y) - \underset{\substack{j=1 \\ j \neq y}}{\overset{c}{max}} Q(\mathbf{x}_i, y_j) \right)^2 - s^2}{\left(\frac{1}{K} \sum_{t=1}^{K} \sqrt{p_k + \hat{p_k} + (p_k - \hat{p_k})^2} \right)^2} \tag{5.2.4}$$

where

$$p_k = \frac{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}) = y)}{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}))}$$

is an out-of-bag estimate of $P(h_k(\mathbf{x}) = y)$ and

$$\hat{p}_k = \frac{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}) = \hat{y}_j)}{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}))}$$

is an out-of-bag estimate of $P(h_k(\mathbf{x}) = \hat{y}_j)$ and

$$\hat{y}_j = \underset{\substack{j=1 \\ j \neq y}}{\overset{c}{argmax}} Q\left(\mathbf{x}, y_j\right)$$

is estimated for every instance $\mathbf{x}$ in the training set with $Q\left(\mathbf{x}, y_j\right)$.

Breiman used unpruned decision tress as base classifiers and introduces additional randomness into the trees [99]. The basic approach of the algorithm is to use a splitting criterion to determine the most suitable attribute to be used for splitting the nodes, to grow the tree. The algorithm continually perform this search for finding the attributes to build the branches of the tree until there is no more splits are necessary. Gini Index is the common splitting criterion used in the algorithm. It determines the best attribute to be used for splitting the node, to grow the tree. The attribute with the highest Gini index is chosen as split in that node. The gini index is given by the formula:

$$Gini\left(A_i\right) = -\sum_{i=1}^{c} p(y_i)^2 + \sum_{j=1}^{m_i} p\left(v_{i,j}\right) \sum_{i=1}^{c} p(y_i \mid v_{i,j})^2$$

The other splitting criterion methods used in classification problems are Gain ratio [101], ReliefF [102], MDL [103], and KDM [104]. Random Forests uses the Gini index taken from the CART learning system [105].

5.3 Error convergence in random forests

The generalization error is defined as the probability of margin function defined over the training set space $(\mathbf{X}, Y)$. It is given by

$$PE^* = P_{\mathbf{X},Y}\left(mg\left(\mathbf{X}, Y\right) < 0\right)$$

where $P_{\mathbf{X},Y}$ is the probability over the $\mathbf{X}$, Y Space.

The generalization error for random forests can be expressed in terms of two parameters. One is how accurate the individual classifiers and the other is dependence between them. The interplay between them gives the insights of the working of the random forests. Here we derive the convergence factor for generalization error of random forest. This factor estimates the accuracy of the classifier.

Theorem 5.3.1 *As the number of trees increases, for almost surely all sequences $\Theta_1, \ldots$ generalization error PE^* converges to*

$$P_{\boldsymbol{X},Y}\left(P_\Theta\left(h\left(\Theta, \boldsymbol{X}\right) = Y\right) - max_{j \neq Y} P_\Theta\left(h\left(\Theta, \boldsymbol{X}\right) = j\right) < 0\right)$$

Proof. The theorem can be proved by showing that there exists a set of probability zero C on the sequence space $\Theta_1, \Theta_2, \ldots$ such that outside of C, for all $\mathbf{x}$,

$$\frac{1}{N}\sum_{n=1}^{N} I\left(h\left(\Theta_n, \mathbf{x}\right) = j\right) \rightarrow P_\Theta\left(h\left(\Theta, \mathbf{x}\right) = j\right)$$

For a fixed training set and fixed Θ, the set of all $\mathbf{x}$ such that $h(\Theta, \mathbf{x}) = j$ is a union of hyper-rectangles. For all $h(\Theta, \mathbf{x})$ there is only a finite number K of such unions of hyper-rectangles, denoted by $S_1, \ldots, S_K$. Define $\varphi(\Theta) = k$ if $\{\mathbf{x} : h(\Theta, \mathbf{x}) = j\} = S_k$. Let N_k be the number of times that $\varphi(\Theta_n) = k$ in the first N trials. Then

$$\frac{1}{N}\sum_{n=1}^{N} I\left(h\left(\Theta_n, \mathbf{x}\right) = j\right) = \frac{1}{N}\sum_{k} N_k I\left(\mathbf{x} \in S_k\right)$$

By the law of large Numbers,

$$N_k = \frac{1}{N}\sum_{n=1}^{N} I\left(\varphi\left(\Theta_n\right) = k\right)$$

converges as to $P_\Theta\left(\varphi\left(\Theta_n\right) = k\right)$. Taking unions of all the sets on which convergence does not occur for some value of k gives a set C of zero probability such that outside of C,

$$\frac{1}{N}\sum_{n=1}^{N} I\left(h\left(\Theta_n, \mathbf{x}\right) = j\right) \rightarrow \sum_{k} P_\Theta\left(\varphi\left(\Theta\right) = k\right) I\left(\mathbf{x} \in S_k\right).$$

■

Thus the theorem proves that as the number of trees grows in the *RF* the resultant classifier converges to a probability$P_\Theta\left(h\left(\Theta, \mathbf{x}\right) = j\right)$. This shows that when the *RF* itself used as a base classifier for ensembling, with many trees can produce better convergence factors.

5.4 Bagged Random Forests

In this section we discuss the application of bagging algorithm to grow the ensembles of random forest. The random forest itself is considered to be the varied version of bagged decision trees. Here we experiment with the growth of ensembles of random forests with bagging in order to reduce the overall bias and variance of learning system. **Listing 5.1** describes the bagged random forest algorithm.

Bagging could be used for enhancing the accuracy of random forests. It can be used to give ongoing estimates of the generalization error (PE^*) of the combined ensemble of forests and the trees within the forests, along with the estimates for the strength and correlation.

Listing 5.1: Bagged Random Forests

```
Input:
Training Set $T$ with $n$ number of training samples
Input vector $x_i \in R^d$, $d$ is the dimension of the input vector
Label of the $x_i = y_i$, $y_i \in \{1,\ldots,c\}$
$i^{th}$ Base Random Forest $= h_i$
Number of random forests to be generated $= M$

for each iteration $i = 1 \ldots M$
{
Construct a bootstrap sample by randomly drawing $n$ times
with replacement from the original data set $T$
$T_i^* \langle x_i, y_i \rangle \longleftarrow bootstrapped(T \langle x_i, y_i \rangle)$
Generate a random forest $h_i$ over the bootstrapped sample $T_i^*$
to minimize the bias over the data set.
}
The final bagged ensemble,
$RF_{bag} = \frac{1}{M}\sum_{i=1}^{M} h_i(T_i^*)$
Output: $RF_{bag}$
```

For each trail $i = 1,\ldots,M$, a bootstrap training set of size n is sampled (with replacement), from the original instances. This training set is the same size as the original data, but some instances may not appear in it while others appear more than once, generally one-third of the instances are left out from the original set. The learning system generates a random forest classifier from each of the M bootstrapped samples.

To classify an instance x, a vote for the class c is recorded by every classifier for which $h_i(x) = c$ and the final classifier RF_{bag} is then the class with the the most votes. Since the error rate decreases as the number of combinations increases, the use of several RF could be resulting in the better accuracy results.

5.5 Boosted Random Forests

Here we use random forests as a base learner to grow ensembles of boosting. Listing 5.2 describes the algorithm. The purpose of boosting is to sequentially apply the RF algorithm to repeatedly modified versions of the data, thereby producing a sequence of RF algorithms $h_m(\mathbf{x})$ $i = 1,\ldots,M$. The predictions from all of them are then combined through a weighted majority vote to produce the final prediction:

$$RF_{boost} = sign\Big(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\Big)$$

Listing 5.2: Boosted Random Forests

```
Input:
Training Set $T$ with $n$ number of training samples.
Input vector $x_i \in R^d$, $d$ is the dimension of the input vector
Label of the $x_i = y_i$, $y_i \in \{1, \ldots, c\}$
$m^{th}$ Base Random Forest $= h_m$
$M$ be the number of Random forests to grow
Initialize the observation weights:
for all $i = 1, \ldots,$ n
 $w_i = 1/n$
for $m =$ 1 to $M$
{
Fit a random forest $h_m(x)$ to training-set $T$ using weights $w_i$
Compute error of $h_m$
            $err_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq h_m(x_i))}{\sum_{i=1}^{n} w_i}$
Compute relative importances of $h_m$
            $\alpha_m = log\left((1 - err_m)/err_m\right)$
Set $w_i \leftarrow w_i \cdot exp\left[\alpha_m \cdot I\left(y_i \neq h_m\left(x_i\right)\right)\right]$, $i = 1, \ldots, n$.
}

Output:
$RF_{boost} = sign\left[\sum_{m=1}^{M} \alpha_m h_m(x)\right]$
```

In the algorithm $\alpha_1, \alpha_2, \ldots, \alpha_M$ represent the relative importances, for are each of the random forest classifier, computed by the boosting algorithm. Their effect is to give higher influence to the more accurate classifiers in the sequence. Figure 5.1 shows a schematic of the AdaBoost procedure for random forests, here RF classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.

5.6 Experiments on UCI data sets

UCI machine learning repository [106] contains data sets that have been in use to evaluate learning algorithms. Each data file contains individual records described in terms of attribute-value pairs. The ensembles of RF along with the original RF have been evaluated on several datasets from the UCI Machine learning Repository. The data sets used in the experiment are summarized in the table 5.1

For the original random forest algorithm, 20 decision trees have been grown for these experiments. In all the experiments at an average a random forest with 20 trees have shown significant improvements in the classification accuracies, further growth of trees have not shown any significant improvements in classification accuracies (Figure 5.6). The experiments are conducted with unpruned trees, and the variable yielding

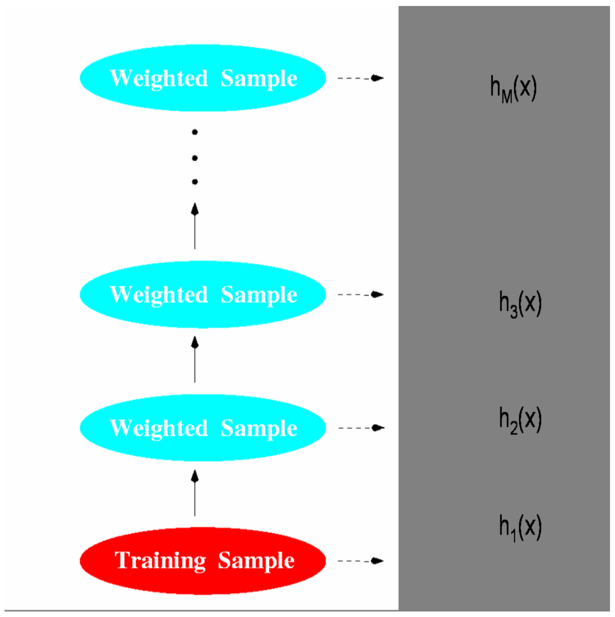


Figure 5.1: Schematic of AdaBoost random forests.

Data set	total cases	Classes	Cont.attribute	Discr.attribute
BreastCancer	699	2	9	0
Heart-c	303	2	8	5
Glass	214	6	9	-
Hepatitis	155	2	6	13
Labour	57	2	8	8
Sonar	208	2	60	-
Vowel	990	11	11	10
Letter	20000	26	16	-
WaveForm	300	3	21	-
Letter	20000	26	16	-

Table 5.1: Description of UCI data sets

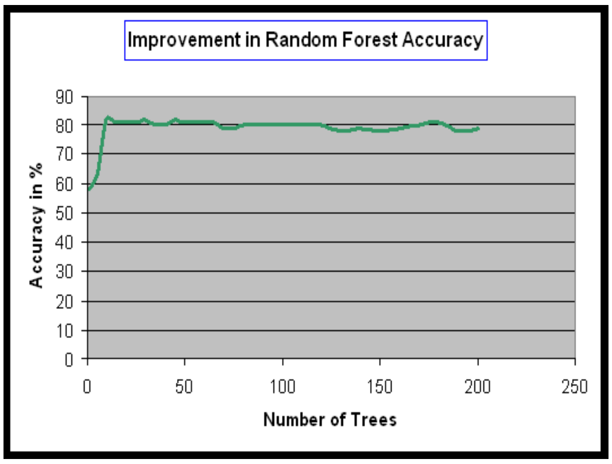


Figure 5.2: Performance of random forests as function of number of trees

the smallest gini-index has been used for splitting, tree building is stopped when the number of instances in a node is 5 or less. RF facilitate to deal with the fractional instances, required when some attributes have missing values, which can be easily adapted to handle the instance weights used in the ensembles.

The following parameters are used for bagging random forest. A 100% bagSizePercent [Size of each bag, as a percentage of the training set size] without any out-of bag errors. We used 20 random forests to grow the bagged ensembles with a random seed of 1. Further increase in random forests have not shown any significant improvements. Most of the improvement from bagging is evident with in few replications, and it is interesting to see the performance improvement that can be bought by a single order of magnitude increase in computation. The boosting random forests have been grown with the same parameters for RF described above. For boosting ensembles, 20 random forests have been grown with a random number seed of 1. The weight threshold for weight pruning has set to 100, only reweighing has been performed without any resamplingof data sets. The data sets are splitted in 66% for training and remaining 34% for testing.

5.7 Discussion

The results of the trials are shown in the table 5.2. The following statistical measures are used in the comparative study:

- **ROC curves:** A Receiver Operating Characteristic (ROC) curve summarizes the performance of a classifier across the range of possible thresholds. It is a standard comparision statistic used in machine learning literature for comparing classifiers, as it does not merely summarize performance at a single arbitrarily selected decision threshold, but across all possible decision thresholds. It plots

	RF			**BagRF**			**BoostRF**		
Data sets	%	MAE	ROC	%	MAE	ROC	%	MAE	ROC
Breast Cancer	69.23	.362	.634	73.46	.377	.683	66.32	.361	.624
Heart-C	82.69	.099	.917	84.61	.099	.937	82.69	.099	.917
Glass	69.86	.108	.804	72.60	.118	.804	67.12	.093	.839
Hepatitis	84.90	.221	.893	84.90	.272	.866	88.67	.113	.900
Labour	90	.205	.998	95	.210	.978	95	.216	.978
Sonar	73.23	.314	.857	80.28	.328	.915	73.23	.314	.857
Vowel	88.13	.054	.999	91.09	.062	1	92.28	.013	1
Letter	93.14	.014	.993	93.14	.014	.993	93.14	.014	.993
Waveform	80.82	.188	.935	84.76	.196	.952	83.05	.113	.944

Table 5.2: The Classification results for Random Forests (RF), Bagged RF, Boosted RF on various UCI data sets. (Notations % for classification accuracy, MAE for Mean Absolute Error, ROC for ROC area)

the sensitivity (true positives) versus one minus the specificity (false negatives). we report the area bounded by these curves as the performance measure. An ideal classifier hugs the left side and top side of the graph, and the area under the curve is 1.0. Figure 5.3 shows ROC curves for various data sets comparing RF along with its ensmebles.

- **Classification accuracy:** The classification accuracy for a classifier is defined as percentage of number of correctly classified samples to the total number of samples. Figure 5.4 shows the classification accuracies as barchart for various UCI data sets.

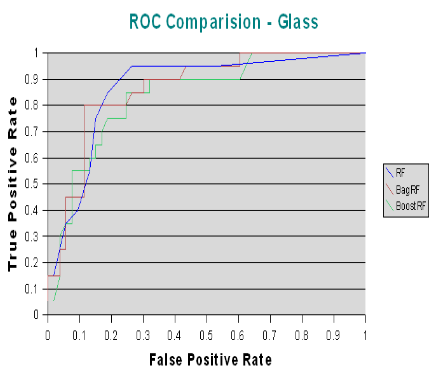
- **Mean Absolute Error:** Mean absolute error is the average of the difference between predicted and actual value. It gives the average prediction error representing a simple measure of overall error in the data distribution.

Bagged random forests have shown clear improvements over the original random forests, whereas boosted technique has ranged from the best to rather mediocre results. When bagging and boosting are compared head to head, boosting leads to greater accuracies for vowel and hepatitis data sets. But it also performed inferior to random forests in data sets like Glass, Breast cancer. Bagging shown consistent performances and been less risky. It proved to more suitable for increasing the random forest accuracies.

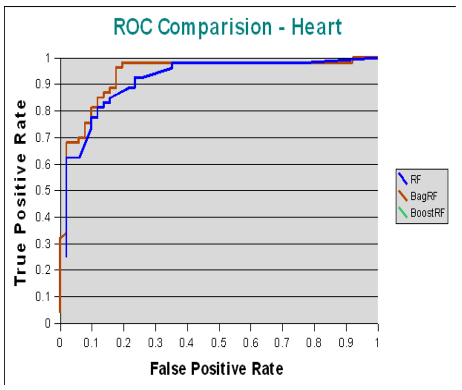
The possible reason for boosting failure is the deterioration in generalization performances. Freund and Schapire study [95] puts it down to overfitting - a large number of trials T allows the composite classifier rf_{boost} to become very complex. AdaBoost stops when the error of any of the base classifier drops to zero, but does not address the possibility that the final classifier rf_{boost} might correctly classify all the training data even though no base classifier does. Further trials in this situation have increased the complexity of the rf_{boost} but cannot improve its performance on the data sets. It also supports the hypothesis that over-fitting is a major factor in explaining boosting's failure on some data sets.

We investigated the possibilities of improving the random forests. Experiments over diverse collection of data sets have confirmed that the ensembles of random forest have improved the performance of classifications. Boosting and Bagging both have a sound theoretical base and also have the advantage that the extra computation they require is known in advance - if T classifiers are generated, then both require T times the computational effort of random forests. In these experiments, a little increase in computation can buys a significant increase in the classification accuracies. In many applications, improvements of this magnitude would be well worth the computational cost.

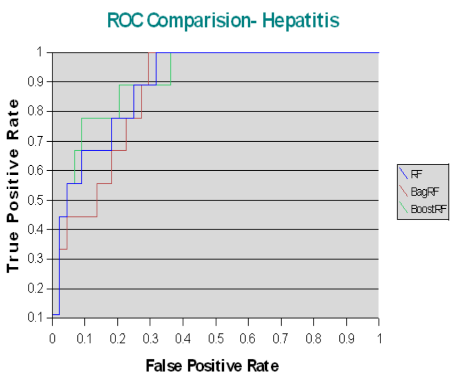
We use these random forest ensembles for the image data classification in MAGIC telescope experiment discussed in the next chapter. Also we discuss the development of BPNN ensembles for the application to MAGIC image data.



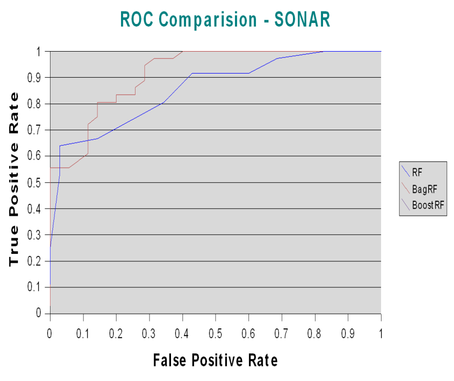
(a) glass



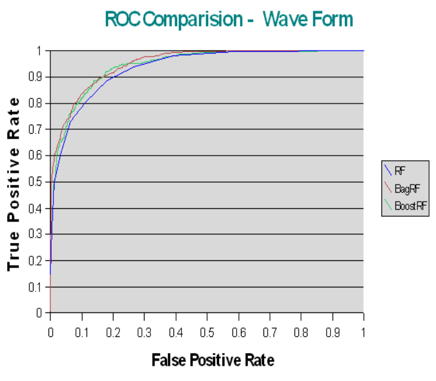
(b) heart



(c) hepatitis



(d) sonar



(e) waveform

Figure 5.3: ROC Comparison for various UCI data sets

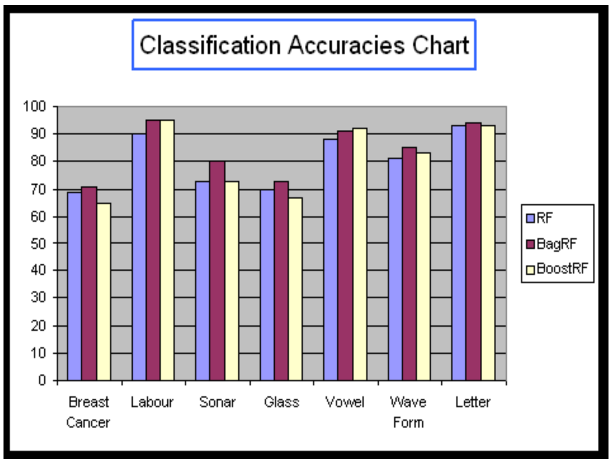


Figure 5.4: Classification accuracies for RF and its ensembles for various UCI data sets

6
MAGIC and its image classification

Figure 6.1: The MAGIC telescope

MAGIC(**M**ajor **A**tmospheric **G**amma **I**maging **T**elescope) is the world's largest imaging air Cherenkov telescope used for detecting the gamma signals from the outer space [109]. It is an international collaboration project with more than 150 Physicists, Computer scientists from 17 institutions. It is designed to provide vital information on several established gamma-ray sources, like Active Galactic Nuclei, Supernova Remnants, Gamma Ray Bursts and Pulsars. It is located in La Palma one of the beautiful Canary islands in Spain at the *Roque de los Muchachos Observatory Center*(28.8°N, 17.9°W), 2200 m above the sea level. The place is well known for its good meteorological features and its altitude allows to make observations of astronomical sources, with an almost always clean night sky. The data taking by the telescope has been started in april 2004, and since then several sources are being observed. In this chapter we discuss the classification of images collected by the MAGIC based on

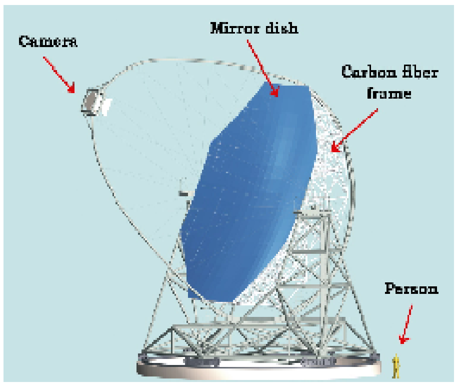


Figure 6.2: Schematic view of $17m\phi$ MAGIC *telescope* [111]

the algorithms discussed in the thesis. First, the experiments are made with supervised (ensemble) techniques based on random forests and back-propogational neural nets. Next we discuss using the unsupervised technique for automatic classification of images based on SOM model.

6.1 MAGIC Infrastructure

The schematic view of the 17m diameter ϕ MAGIC telescope is shown in the figure 6.2. The main parts are the mirror dish to collect the signals from the outer space astronomical sources and the camera which converts these signals into images. The mirror dish has a parabolic shape with an area of 234 m^2. The dish is composed of 956, $49.549.5cm^2$ spherical mirror tiles made of aluminum [110]. The parabolic shape for the dish was chosen to minimize the time spread of the light flashes reflected onto the camera plane, thus the rate of fake events induced by night-sky background light are reduced. The aluminum plate of each tile is diamond-milled to achieve the spherical reflecting surface with the radius of curvature most adequate for its position on the paraboloid. The entire system is mounted on a lightweight ($<$ 10 ton) carbon fiber frame, which enables its altazimuth mount to point to any source in the sky in less than 20 seconds. This feature is important as it is essential for the study of short living events like Gamma ray burst's.

The MAGIC camera is equipped with 576 "6-dynode" compact photomultiplier (PMs). These PMs are arranged into two sections: The inner section contains 396 PMs arranged in a hexagonal grid and the remaining 180 PEs are arranged into 4 concentric rings around this hexagonal grid (figure 6.3). Each PM has 20% average quantum efficiency in the 350-500 nm range and is coupled to a small Winston cone type light collector to maximize the active surface of the camera. The total field of view of the camera is about 4 °, with an angular resolution of 0.1°.

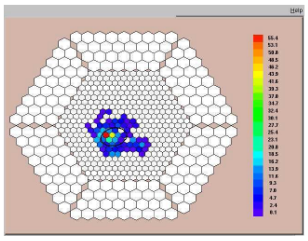


Figure 6.3: A simulated MAGIC Camera

6.2 Image collection in MAGIC

The source observation by MAGIC is carried out during night time with a well organized night data taking schedule for various interesting astronomical objects. The detection of signals from these sources is based on Imaging Atmospheric Cherenkov Technique **(IACT)**. Astronomical sources transmit extremely energetic particles traveling very close to the speed of light or traveling faster than the speed of light “in the medium of the atmosphere” [112]. Infact nothing can travel faster than the speed of light “in a vacuum”, but that the speed of light is reduced when traveling through most media (like glass, water, air, etc.). The resultant polarization of local atoms as the charged particles travel through the atmosphere results in the emission of a faint, bluish light known as ”Cherenkov radiation” [113], named for the Russian physicist who made comprehensive studies of this phenomenon. Depending on the energy of the initial cosmic gamma-ray, there may be thousands of electrons/positrons in the resulting cascade which are capable of emitting Cherenkov radiation. As a result, a large ”pool” of Cherenkov light accompanies the particles in the air shower. This pool of light is pancake-like in appearance, about 200 meters in diameter but only a meter or so in thickness. MAGIC telescope, rely on this pool of light to detect the arrival of a cosmic gamma-ray.

This large pool Cherenkov light is reflected from the mirror of telescope on to the focal plane of one or many photo-multipliers (PMs) of camera, which convert the incoming optical signal into an electronic signal to record the signal “image” or also called as “event” figure 6.3. These images (electronic signals) are essentially analog signals, which are first amplified obtaining the low and high gain signals and then passed to FADC (Flash Analog to Digital Converter) module. This **FADC** module is responsible for digitally sampling the signals into computer readable data which represents the raw image formed on the camera. It is composed of several 8 bit converters each operating at a frequency of 300MHz. Data are then transferred to larger storage arrays and saved them in the raw data format on tapes and hard disks [116]. In this way, a crude image of the Cherenkov light pool is recorded.

MAGIC collaboration has developed a software system called Magic Analysis and Reconstruction Software (**MARS**) for camera simulations and analysis tasks. The software is a collection of C++ classes based on the well known data analysis package ROOT [114]. A software release is documented and available on the web [115]. Figure

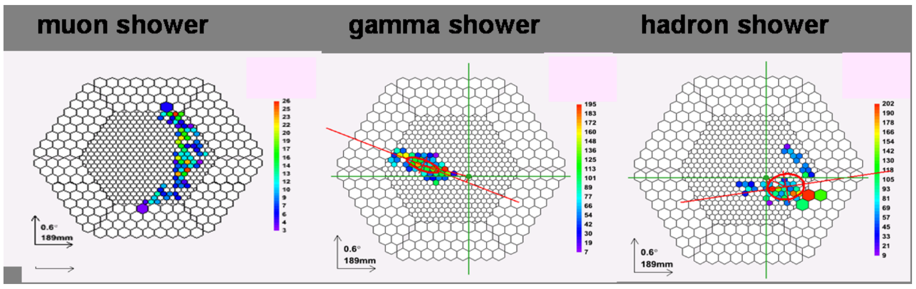


Figure 6.4: Images of various particles recorded on the simulated camera of the magic.

6.4 shows the **MARS** based simulations of the images of various particles formed on the camera. The images formed by the gamma-ray particles differs from that produced by cosmic ray particles in a few fundamental ways. The Cherenkov light collected from a gamma-ray shower has a smaller angular distribution and tends to have an ellipsoidal shape which aligns itself with the direction of the incoming photon. Cosmic-ray induced air showers also called as background events are made of hadron and muon particles. They have Cherenkov light images which are much broader and less well aligned with the arrival direction. By measuring the shape of each shower image, and selecting only those events which are gamma-ray-like in appearance, we can remove the background event contamination, resulting in a much improved ability to detect the gamma signals from the source direction. The light in this pool is very faint and can only be detected cleanly on dark, moonless nights. Even though the total pool passes through the detector in only a few nanoseconds it still helps for imaging this faint signal and to separate it from the ambient night sky. Due to atmospheric radiations, the ground based observation of MAGIC, collects an overwhelming background events containing hadrons and muons[hadron events are interchangeably used for background events in the MAGIC collaboration]. Thus an efficient gamma- hadron separation on the basis of the image analysis is absolutely crucial for understanding the gamma ray sources. In the imaging technique adapted by the MAGIC, the differences in the shower development of gamma and background particles shown up in the recorded images can be taken as the base point for the classification task. In this regard, images are processed to extract some important image parameters for the classification purposes.

6.3 Image processing in MAGIC

Processing of the raw images collected by the telescope is the key in extracting the suitable image parameters for gamma - hadron separation task. The parameters used in astrophysical experiments could be in thousands. A typical astronomical source observation can yield upto 4 million events. Each event is distributed over 577 pixels of camera. Thus total parameters come somewhere near 577*4 million to deal with. For feasible image classification, reduction in these parameter set is mandatory. In this

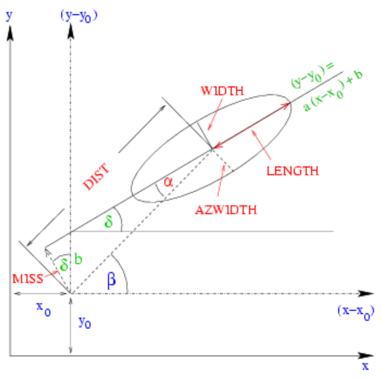


Figure 6.5: Image parameters. (x, y) are coordinates in the original camera system. (x_0, y_0) is a reference point, for example the source position or the center of the camera.

section we describe the various image processing steps used in MAGIC for extracting standard image parameters.

The first step in image processing is to convert the raw data into number of photoelectrons, which are the direct measure of the observed cheronkov photons [117]. This process is called image calibration and is an important step to normalize signals and cleaning image. It makes a synchronization between various pixels of camera to return the same signal. To make this happen, a standard color pixel (ex: Led Green or blue) is taken as a reference and a conversion factor for each pixel is calculated [120]. In the subsequent process, image cleaning technique is applied to the shower images to remove the unusable events and biases created due to night observations and electronics of the telescope. These unusable events are either shower events, which are too small for a reasonable image parameter set calculation or night sky background triggered events. For further analysis it is useful to characterize the image by simple parameters .

Hillas Image parameters are used to describe the light content, shape and orientation of the recorded shower images in the camera [122],[118],[119],[121]. After proper calibration and image cleaning the shower image is represented as the number of Cherenkov photons N_i and its error ΔN_i for each pixel $i\,(1, \ldots, N_{pix})$. The Hillas parameters are derived as the moments of the 2-dimensional distribution of N_i and represents the light distribution in the camera (figure 6.5). These hillas parameters can be categorized based on the shape and orientation of the images . The important hillas parameters are summarized as follows:

Length: It is the spread of light along the major axis of the image. It carries the information of the longitudinal development of the shower. It represents the major half axis of the ellipse [mm] i.e, half length of the shower image.
Width: The spread of light along the minor axis of the image. Carries with it infor-

mation of lateral development of the shower. In the image it represents the minor axis of the ellipse [mm] i.e half width of the shower image.
Size: The total integrated light content of the shower. It represents the total number of photons in the shower.
Conc2: It is the ratio of sum of two highest pixels over size ratio.
Conc1: It is the ratio of the brightest pixel over size.
Asym: Assymetry of the photon distribution along the shower axis.
MDIST: Distance between the center of the shower image and the brightest pixel.
LEAKAGE: Fraction of photons contained in the pixels close to the edge of the camera.
DELTA: angle δ between the shower axis and the $x - axis$ of the camera
DIST: distance of center of shower image from the Reference Point (RP)
MISS: distance of the shower axis from the RP
ALPHA: angle α by which shower axis misses the RP, as seen from the center of the shower image
BETA: angle β between the $x - axis$ of the camera and the line connecting the RP with the center of the shower image
AZWIDTH: half width of the shower image as seen from the RP.

Traditionally the separation task is carried out by statistical techniques such as dynamical cuts. These cuts operate on one variable at a time but often made dependent on other parameters based on logical operators such as AND or OR to partition the given n-dimensional space; the problem gets unwidely even for a low n-space of features, nevertheless this is the widely used technique for many cheronkov experiments for image classification [123]. Now we discuss the usage of the machine learning algorithms both supervised and unsupervised for the image classification task based on these image parameters.

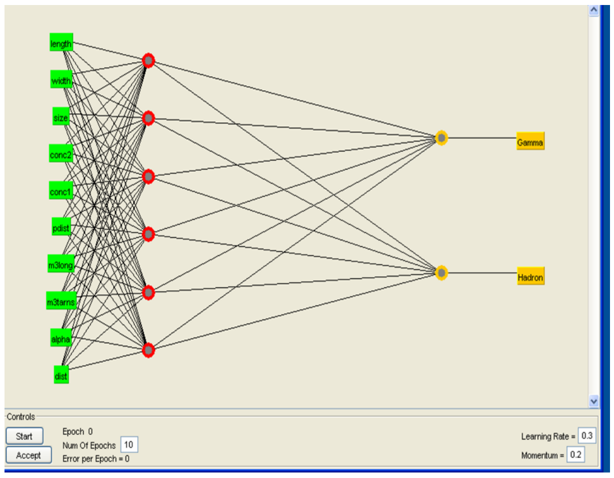
6.4 Supervised classification

Previous classification studies performed on MAGIC data sets shown that random forest, neural networks performed better than other classifiers [124]. Here we experiment with ensembles of random forests, neural networks and study their performance results. The experiment is conducted on crab astronomical source. The data analyzed here were taken during September and October 2004 and in January 2005. The Crab nebula is the remnant of a supernova explosion that occurred in 1054. The γ-emissions from this source was first reported by the Whipple collaboration [136]. It exhibits a stable and strong γ-emission and is frequently used as the "standard candle" in very high energy γ-astronomy [135]. The Crab nebula has been observed extensively in the past over a wide range of wavelengths, covering the radio, optical and X-ray bands, as well as high-energy regions up to nearly 100TeV [137].

Probing the presence of gamma signals from this sources is a challenge for experimenters. A total of 2.8 million images in 2004 and 4.5 million images from the 2005 observations were used . The overall observation time of the sample analyzed corresponds to 13 hours on-source and the data collected [around 25 Giga bytes], is used as a test set for separating gammas from the hadrons. The training data sets for the algorithms have been generated by a monte-carlo program, CORSIKA [125]. It is a package used to perform detail simulation of cheronkov light effect observed by the MAGIC for various sky objects. The CRAB source was simulated with the same parameter set that has been used for the observation. The Monte Carlo raw data has been subjected to image calibration and image cleaning to extract the image parameters. Image cleaning eliminates the biased data and outliers. Subsequently the analysis is simplified with little or no loss of information, by converting the pixels of image into image parameters for both gamma and other background particles. As suggested in [124], the following image parameters have been chosen based on their importance in the image properties : length, width, size, conc2, conc1, pdist, m3long, m3trans, alpha, dist. They are used only for relative comparison of algorithms. At image processing stage, no care has been taken to optimize the data. Also, no particular care has been taken to choose independent parameters since a robust discrimination method should itself take care of correlations between parameters. The sample in the training set contains 122800 background events, 720000 gamma events. Now we discuss the algorithms used for the classification purpose and study the performance results. Random forests, back propagation neural nets and their ensembles are developed for the classification tasks. WEKA, a very nice machine learning framework is used for this purpose[126]. It facilitates powerful object oriented programming framework for making ensembles. It provides facilities for using various machine learning algorithms as base classifiers for ensembles.

6.4.1 Random forests and its ensembles

Random forests and its ensembles have been thoroughly described in chapter "*Meta Random Forests*". Here we used these algorithms for studying the performance results of the MAGIC experiment. The tree growing begins with all cases being contained in the root node. The root node is then split by a cut using one of the image parameters, into two successive nodes to achieve a classification by separation of the classes. When using all of 10 image parameters, three of them are chosen randomly and uniformly distributed. The image parameter yielding the smallest Gini-index among these three is used for splitting. Subsequently, the same procedure is applied to each branch in turn. The tree growing stops only when all nodes contain pure data, i.e. from one class only. These nodes are then called terminal nodes and receive their class label from the training data. No pruning is performed. Using unpruned trees (in general poor classifiers) requires a reasonably large number of them to be combined. For our data, growing 20 trees turns out to be sufficient: the quality of the classification in terms ROC plot remains unchanged after a certain number of trees has been reached. Combining classifications from the trees is simply done by calculating the arithmetic mean from the 20 classifications of all trees (considered as 0 and 1). The results are verified with R statistical software [127], Breiman original FORTRAN sources for the Random Forests technique [128]. Random Forests are themselves proved to be powerful classifiers, and given good results for MAGIC data sets. The ensembles of



Epoch 0
Num Of Epochs 10
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

Figure 6.6: Fundamental BPNN used for ensembling to perform image classification in MAGIC

random forests have further improved the classification accuracy. For both bagging and boosting, the random forests of 20 trees, each constructed while considering four random features have been used as base classifiers and iterated for 10 times.

6.4.2 BPNN and its ensembles

BPNN has been described in "*Chapter 1 - section 1.3.1*" here we discuss the ensembles of it and the application to MAGIC data sets. The fundamental neural network is shown in the figure 6.6 with the following architecture: 10-6-2 [10 input neurons, 6 hidden neurons, 2 output neurons one for gamma and and one for background] with learning rate as 0.2 and momentum factor as 0.3. 10 neural nets are used with each neural network trained for 20 iterations. Both bagging and adaboosting has been done using the same parameter set.

One fundamental weakness of neural networks is that they are very sensitive to the training data sets (i.e.) small changes in training set and/or parameter selection can cause large changes in performance. They are unstable or exhibit variance. The fact that neural networks are high variance means that different networks will produce different results for individual test cases. Two networks tested on the same data may have similar average accuracies but may disagree on several individual samples [131]. This instability is magnified when real-world systems such as MAGIC data sets are modeled as they contain more noise and dominated by only one class events.

In general, neural network classifiers provide more information than just a class label. It can be shown that the network outputs approximate the a-posteriori probabilities of the classes, and it might be useful to use this information rather than to perform a hard decision for one recognized class. This issue is addressed by AdABoost algorithm [129] when the classifier computes the confidence scores for each class. The result of the training the t^{th} classifier is now a hypothesis $h_t : XxY \rightarrow [0,1]$. Furthermore, as described in adaboost algorithm in *"Ensemble Learning"* chapter, we use a distribution $D_t(i,y)$ over the set of all *miss-labels* $B = \{(i,y) : i \in \{1,\ldots,N_v\}, y \neq y_i\}$, where N_v is the number of training examples. Therefore $|B| = N_v(k-1)$. Adaboost modifies this distribution so that the next learner focuses not only on the examples that are hard to classify, but more specifically on improving the discrimination between the correct class and the incorrect class that competes with it. Note that the miss-label distribution D_t induces a distribution over the examples: $P_t(i) = W_i^t / \sum_i W_i^t$ where $W_i^t = \sum_{y \neq y_i} D_t(i,y)$. $P_i(i)$ may be used for resampling the training set. According to [129] the *pseudoloss* of a learning machine is defined as:

$$\epsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i,y)\left(1 - h_t(x_i,y_i) + h_t(x_i,y)\right)$$

It is minimized if the confidence scores of the correct labels are 1.0 and the confidence scores of all the wrong labels are 0.0. The final decision f is obtained by adding together the weighted confidence scores of all the machines (all the hypothesis $h_1, h_2, \ldots$).

Neural networks have trained by minimizing a quadric criterion that is weighted sum of the squared differences $(z_{ij} - \widehat{z}_{ij})^2$, where $z_i = (z_{i1}, z_{i2}, \ldots z_{ik})$ is the desired output vector (with a low target value everywhere except at the position corresponding to the target class) and $\widehat{\mathbf{z}}_i$ is the output vector of the network [130]. A score for class j for pattern i can be directly obtained from the $j-th$ element $\widehat{z}_{ij}$ of the output vector $\widehat{z}_i$. When a class must be chosen, the one with the highest score is selected. Let $V_t(i,j) = D(i,j) / max_{k\neq y_i} D_t(i,k)$ for $j \neq y_i$ and $V_t(i,y_i) = 1$. These weights are used to give more emphasis to certain incorrect labels according to the Pseudo-Loss Adaboost.

Generally decision trees are adaboosted by training the $t-th$ classifier with a fixed training set obtained by resampling with replacement once from the original training set: before starting the $t-th$ classifier, we sample N patterns from the original training set, each time with a probability $P_t(i)$ of picking pattern i. Training is performed for a fixed number of iterations always using this same resampled training set.

The BPNN is adaboosted by using the stochastic version of the above technique.Each t-th network is trained by using a different training set at each epoch(pass of entire train set to the neural network), by resampling with replacement for each epoch: after each epoch, a new training set is obtained by sampling from the original training set with probabilities $P_t(i)$. For an online stochastic gradient case, this is equivalent to sampling a new pattern from the original training set with the probability $P_t(i)$ before each forward/backward pass through the neural network. Training continues until a fixed number of pattern presentations has been performed. The training cost that is minimized for a pattern that is the $i-th$ one from the original training set is $\frac{1}{2}\sum_j V_t(i,j)(z_{ij} - \widehat{z}_{ij})^2$.

Bagged BPNN ensembles are generated by using bootstrap [132], a very popular statistical resampling technique, which is used to generate multiple training sets and

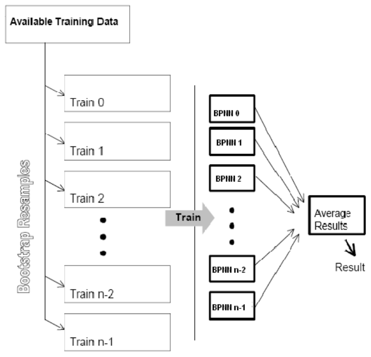


Figure 6.7: A bagged BPNN ensemble. The ensemble is built by using bootstrap re-sampling to generate multiple training sets which are then used to train an ensemble of BPNNs. The predictions generated by each network are averaged to generate more stable bagged ANN ensemble predictions.

networks for an ensemble (see Figure 6.7 for an illustration of this). Although other ensemble techniques such as boosting have been shown to out-perform bagging on some data-sets [133], bagging has shown number of key advantages when applied to real-world tasks for ex: medical decision support. One of the most important is the ease with which confidence intervals can be computed [134]. Another is the robustness and stability of the technique itself. Here we probe the performance of this technique for MAGIC.

6.4.3 Classification results

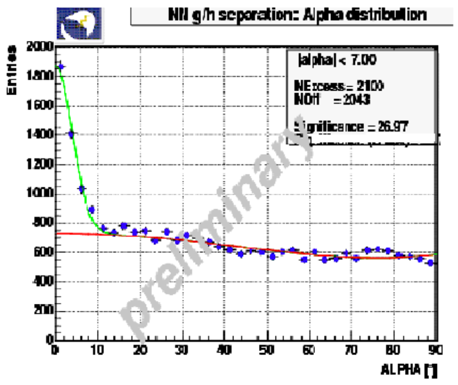
One standard way of data analysis in MAGIC is to obtain the significance of the detected gamma signal from astrophysical source.

If ϵ_γ is the acceptance efficiencies for gammas and ϵ_h are the acceptance efficiency of background (An acceptance efficiency is the conditional probability of an event given the probability of occurrence of all events of its category), then according to [138] significance is determined by

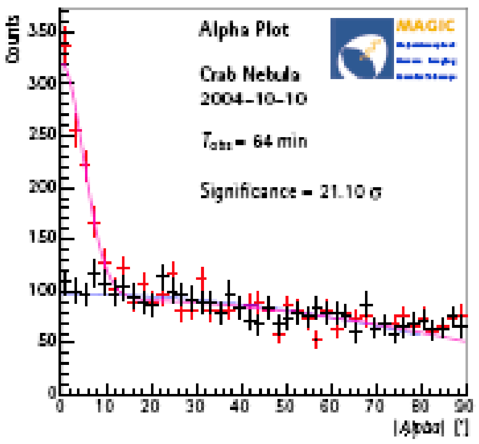
$$S_{LM} = \sqrt{2}\left\{N_{on} ln\left[\frac{1+\alpha}{\alpha}\left(\frac{N_{on}}{N_{on}+N_{off}}\right)\right] + N_{off} ln\left[(1+\alpha)\left(\frac{N_{on}}{N_{on}+N_{off}}\right)\right]\right\}^{1/2}$$

where N_{on} are the total number of gamma events/images, N_{off} are the total number of background images/events.

The parameter α plays a vital role in determining the significance of gamma signal detection from the telescope. The α distribution for gamma events takes a gaussian curvature where as for other events in in principle flat i.e. all α values have equal probability [Figure 6.8(a), (b)]. We got good significance values with BPNN compare to RF.



(a) BPNN



(b) RF

Figure 6.8: Alpha distributions

Experiments are performed in the WEKA software framework, to compare various classification algorithms on image data classification task. The statistical measures used in the comparative study are discussed in the chapter "*Meta random forests*". Table 6.1 shows the classification results for Random forests (RF), Back propagation neural networks [BPNN], along with their bagged and boosted ensembles. Though the ensembles take much time than single classifiers they produce improved classification results. We saw significant improvements with bagged random forests. Adaboosted random forests also showed improved classification results though they are inferior to that of bagged ones. In neural network category, Boosted neural network has shown

Model	Classification Accuracy[%]	ROC Area	Mean Error rate
RF	78.9085	0.818	0.2748
Bagged RF	81.2461	0.8753	0.276
Boosted RF	79.9527	0.8481	0.2006
BPNN	78.28	0.8651	0.2644
Bagged BPNN	80.1893	0.8666	0.2888
Boosted BPNN	81.7508	0.8671	0.2564

Table 6.1: Comparative study of 2 classifiers and their ensembles

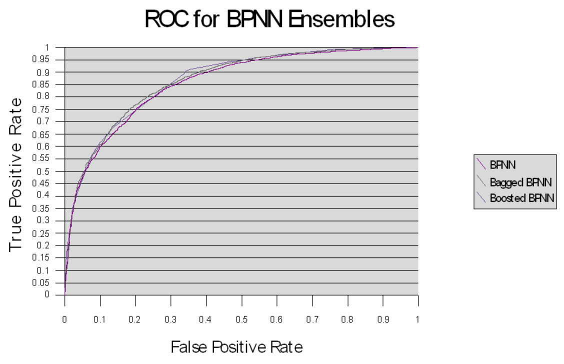


Figure 6.9: ROC curves for BPNN and its ensembles

improved accuracies with bagged MLP standing second.

The ensembles have shown classification improvements for all algorithms, Figure 6.9, 6.10 shows the ROC plots. Though they take more time for training, it is important for MAGIC experiment as analysis of gamma ray events are heavily dependent on classification accuracies. In overall rankings Boosted neural networks, Bagged random forests have shown superior results. Bagging has shown good results with tree based classifiers, especially for random forests. Though neural networks benefited from bagging but they got better accuracies with boosting. The future work will be concentrated on introducing more base classifiers and experimenting with other ensemble techniques such as multi-boost, random committee.

6.4.4 Experiments with R.K. Bock et.al Monte-carlo data sets

R. K. Bock et.al, have performed a case study comparing different classification methods, using as input a set of Monte Carlo data [124]. The data set contains two classes, gamma rays or hadronic showers with 10 image parameters. There are 12332 gamma events and 6688 hadron events. In this paper, 2/3 of both gamma and background events are used as train sample and remaining 1/3 as test sample. The various classification models used in this study are: classification trees, forests, kernel methods, neural networks (Neunet package, neural network with switching units, multi-layer

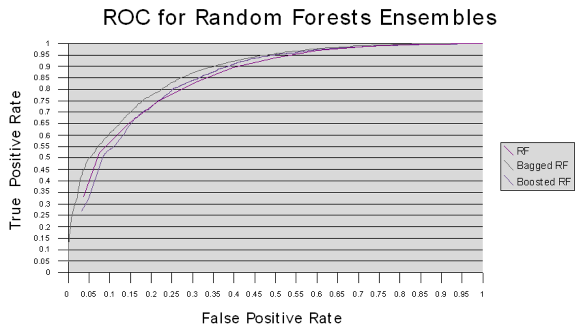


Figure 6.10: ROC curves for RF and its ensembles

perceptrons), nearest neighbor methods, support vector machines. The ROC curves for these classifiers are shown in the figure 6.11(a)

Based on these data sets, in this thesis we performed the classification experiments using ensembles of RF and BPNN. The parameters set used for these algorithms are identical to the description made in the section 6.4.1 and 6.4.2. The ROC curves for the RF ensembles are shown in figure 6.11(b) and for BPNN ensembles in 6.11(c).

6.5 Unsupervised classification

We experimented with SOM networks for making an automatic separation of gamma events from other particle events. The test in conducted again on the crab source. As many real world experiments, producing the labeled data for MAGIC is a costly affair and offered with statistical biases in their data distributions. The main motivation of this work is to make the classification with no monte-carlo data.

Now the train data set comes from off_data set which contains only background events. The data is collected during the MAGIC observation in the off line mode i.e the telescope is pointed away from the actual source path to collect the background events. The test set comes from the data collected directly pointing to the source. This test set termed as on_data set contains the mixture of gamma events from the crab source and background events due to the atmospheric interactions. Now the task is to separate the gamma events from the background events in this on_data set. For this we trained the SOM map only with the off_data set events thus after training the SOM will able to learn the data distribution of the background events. Now the on_data set events are subjected to SOM in the testing phase. Since the SOM knows the off data events, it is self-organised grouping all the background events at one end of the map while all the gamma events are collected at other end of the map. Thus an automatic classification is performed on the on_data set. The whole experimental setup is organized into following steps (figure 6.12)

Parameter set extraction: First the image parameters for the off_data set containing background events and on_dataset containing mixture of gammas, background events are extracted following the image processing steps described in this chapter

Data Normalization: Normalization usually means linear scaling of variables. More generally, it corresponds to selecting a distance metric in the input space. The distance metric sets the viewpoint from which the whole sample analysis looks into the data by defining how important different variables are in the subsequent analysis methods.A well known normalization technique [range method] is used to normalize the components of the input vector so that the component values lie in the range [0, 1]. Each attribute x_i in the event vector is divided by the factor $\sqrt{x_1 + \ldots + x_i + \ldots + x_m}$, m being the dimension of the event vector. This step is useful to make a balanced comparison of euclidean distance among the various map units. The transformation permits a clear differentiation between gamma events and background events.

SOM Training: The normalized train data is then used for SOM training. Each SOM map unit is randomly initialized with an event vector from the off_data set. During the training process, SOM model unfold itself according to the data distribution of the background events. The neighborhood functions used for self-organising in the map are gaussian and cut-gaussian kernels. The training is carried out in 2 phases. The first phase of training was carried out for 150 epochs, with a learning rate of 0.5, while the second phase was 300 epochs, with a learning rate of 0.01. These parameters were retained, after we tried more than 50 trainings, with different architectures. After these two phases, the codebook(which is the collection of the references vectors of all map units) approximates the probability density function $p\left(x\right)$ of the background events. The power law relating codebook vectors and $p\left(x\right)$ of input data set is given by $p\left(x\right)^{\frac{d}{d+r}}$ [140], where d is the dimension and r is the distance norm . Thus SOM follows the density of the input data reducing the computational complexities to the linear functions of the input data samples. Experiments shown that clustering SOM codebook is better then clustering the entire data set [140]. Now the trained map represented by its codebook is ready to be tested and can be used for event classification task.

SOM Testing - Cluster Analysis: Group formation or clustering is one of the main application of SOM. The groups in the the on_data set are detected by comparing its event vectors with every reference vector in the codebook of the SOM map. In [139], [140], [45] different ways to cluster data using SOM has discussed. In this thesis we employed, a *knn based* partitional clustering algorithm **listing 6.1** for finding the groups in on_data set. The algorithm is is based on a cluster distance function $S\left(Q_k\right)$ and a cut value c in $(0, 1)$ interval for labeling the groups it found. If $S\left(Q_k\right) <= c$, it is more likely to be a background event otherwise it is more likely to be a gamma event. Thus the events are got labeled resulting in an automatic classification. The following cluster-distance measures [140] have been used in the experiment .

Let $S\left(Q_k\right)$ be the cluster distance function for a cluster Q_k; here we have only one cluster i.e., the codebook of SOM map.

N_k be the number of codebook vectors,

$c_k = \frac{1}{N_k} \sum_{x_i \in Q_k} x_i$ be the mean of the code book vector,
$x_{i'}$ be the event vector from ON data set.
$x_i \in Q_k$ be the i^{th} codebook vector representing closest match to $x_{i'}$, based on euclidean distance metric.

the various cluster distance functions are:
average distance:

$$S_a = \frac{\sum_{i,i'} \|x_i - x'_i\|}{N_k\,(N_k - 1)}$$

nearest neighbor distance:

$$S_{nn} = \frac{\sum_i min'_i \{\|x_i - x'_i\|\}}{N_k}$$

centroid:

$$S_c = \frac{\sum_i \|x'_i - c_k\|}{N_k}$$

variance:

$$S_v = \sum_i \|x'_i - c_k\|^2$$

Listing 6.1: KNN based clustering

```
Input:
On data set $O = \{x'_i\}$
SOM codebook $Q_k = \{x_i\}$
Distance function $S(Q_k)$
$k$ best matching units for a given $(x_i)$
cutvalue $c$

for each $x'_i \in O$
{
        for each $x_i \in Q_k$
          {
                find the first $k$ Best matching units (BMU)
                $BMU = argmin_i \|x_i - x'_i\|$
                for each of the $k$ BMU
                        Calculate $S_i(Q_k), i = 1, \ldots, k$
                $S_{avg}(Q_k) = \sum_{i=1}^{k} w_i * S_i(Q_k), w_i = \frac{1}{d(x_i, x'_i)^2}$
                if ( $S_{avg}(Q_k) \leq c$ )
                $x'_i \rightarrow backgroundevent$
                else
                $x'_i \rightarrow gammaevent$
          }

}
```

The k BMU's are weighted according to their inverse square distances from the input event vector. The data sets are stored in the secondary memory and data items are transferred to the main memory one at a time for clustering. Only the cluster representations are stored permanently in the main memory to alleviate space limitations. The computational complexity of the nearest-neighbor algorithm both in space (storage of prototypes) and time (search) has received a great deal of analysis. There are a number of elegant theorems from computational geometry on the construction of Voronoi tesselations and nearest-neighbor searches in one- and two-dimensional spaces. However, because the MAGIC is a multidimensional data classification problem we concentrate on the more general d-dimensional case. For n labelled training samples in d- dimensions, and seek to find the closest to a test point x' (k = 1). In the most naive approach we inspect each stored point in turn, calculate its Euclidean distance to x, retaining the identity only of the current closest one. Each distance calculation is $O(d)$, and thus this search is $O\left(dn^2\right)$. One method for reducing the complexity of nearest-neighbor search is to eliminate "useless" codebook vectors during testing, a technique known variously as editing, pruning or condensing. A simple method to reduce the $O(n)$ space complexity is to eliminate prototypes that are surrounded by training points of the same category label. Now the complexity statistics reduces to $O\left(d^3 n^{\lfloor d/2 \rfloor} \mathbf{ln} n\right)$, $\lfloor . \rfloor$ is the "floor" operation.

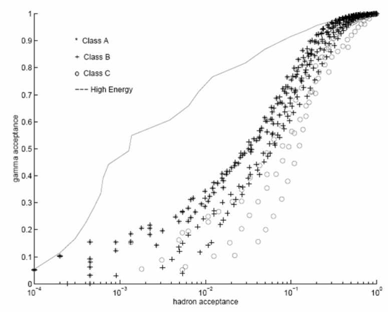
Cluster visulisation: The clusters discovered by SOM are visualised using U-matrix, a matlab tool to visualize these clusters [141]. SOM is an excellent tool for visualisation of high dimensional data. Figure 6.13 displays the data distribution of individual image parameters for crab on_data. These distribution maps can assist analyst to observe the possible relationships that exist between the variables and even roughly distinguish the structure of the input data. The variable distribution planes are organized in such a way that similar planes lie close to each other; this organization is carried out by SOM itself. We experimented with two types of kernels based on gaussian curvature. The group visualisation by these two type of kernel based SOM maps are shown in the figure 6.14, the color coding of the map based on the HSV color system [142]. The value of component H (hue) was dependent on direction from the center of the map, S (saturation) was held constant, and V (value) was inversely proportional to the map unit distance from the center of the map. Thus the groups are formed at the ends of the map with separation boundaries and the clusters concentrated in the center of the map. Figure 6.14 visualizes the classification groups found by the SOM system. The 2 groups found here are represented by blue color. The experiments are performed by using two kernels gaussian and cutgaussian. For gaussian kernel a map of 25X25 network is used and trained for 300 epochs. Cutgaussian kernel map of 40X30 size is trained with 300 epochs. The cut-gaussian kernel shown better performance than that of of gaussian kernel.

The data set is again taken from the CRAB data source, this time with number of off_data events 2.5 million and on_data set contains 7.3 million events [mixture of gamma and background] to be classified. An quantisation error represents the norm of difference of an input vector from the closest reference vector. We used Average Quantization Error **(QE)** of the MAP as the benchmark criteria. The experiments are carried out with various distance functions and the nearest neighbor measure has

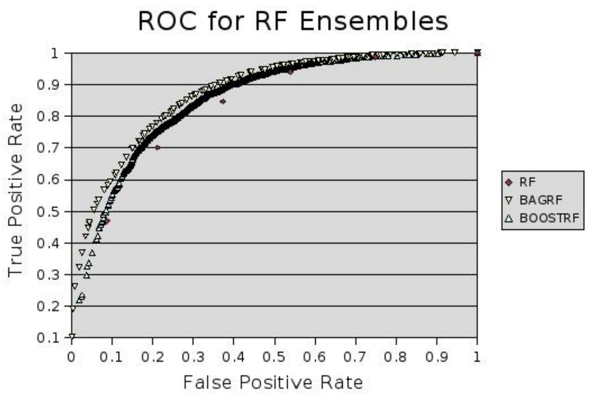
Cluster distance measure	**Total Epochs**	**QE**	**found gammas [%]**
S_a	300	0.43	65
S_{nn}	300	0.35	73
S_c	300	0.47	61
S_v	300	0.39	70

Table 6.2: SOM clustering results for different $S\left(Q_k\right)$

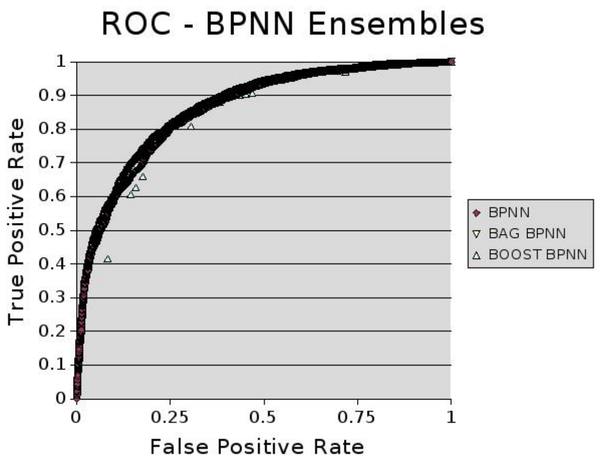
proved to be more effective giving a lowest quantisation error of 0.35. QE at various epochs are shown in figure 6.5. The results of the classification of events based on cut-gaussian SOM and percentage of gammas found are summarized in table 6.2.



(a) Several classification methods compared: a logarithmic axis is used to better show the low-acceptance region. Results are grouped: class A are regression tree and PDE methods, class B contains various Neural Net meth- ods, class C comprises SVM, direct selection, LDA, and composite probabilities. For comparison with present telescopes, a (nearest-neighbour) result for events with incident energies above 120 GeV is also shown. [124]



(b) RF Ensembles



(c) BPNN Ensembles

Figure 6.11: Classifier performances study: Bock et.al classifications, ensemble classifiers used in this thesis

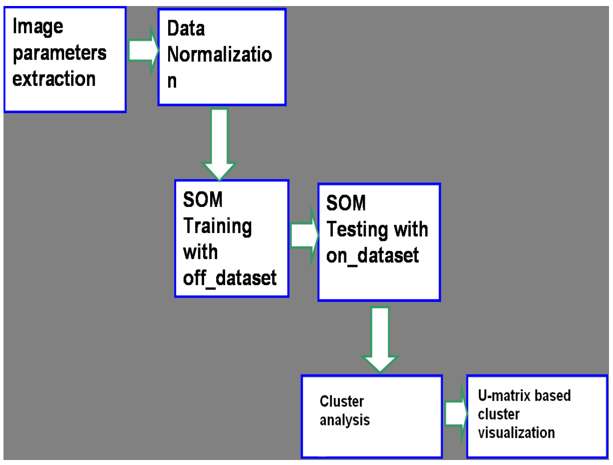


Image parameters extraction
Data Normalizatio n
SOM Training with off_dataset
SOM Testing with on_dataset
Cluster analysis
U-matrix based cluster visualization

Figure 6.12: The SOM based system for automatic separation of gamma events from background events

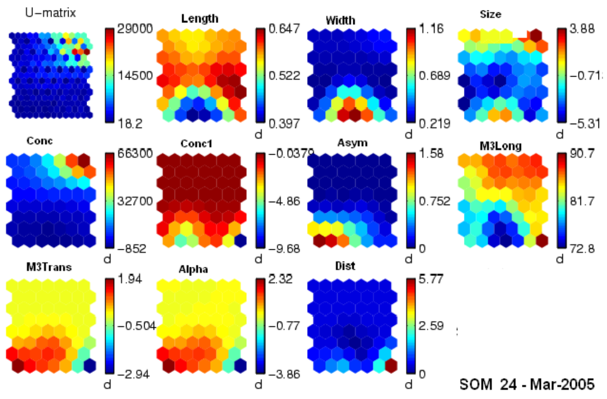
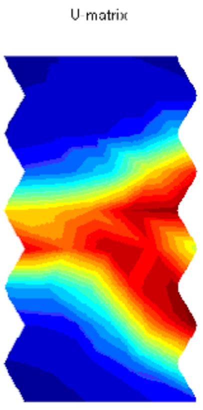
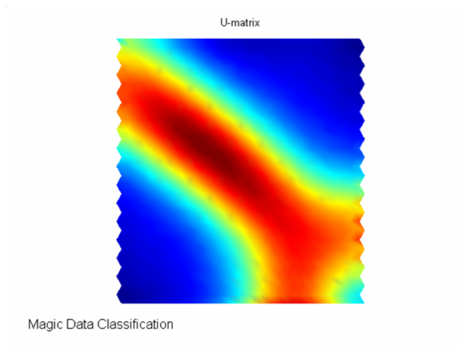


Figure 6.13: Data distributions for image parameters in crab on data set.



(a) with gaussian map



(b) with cut-gaussianmap

Figure 6.14: Classification using SOM

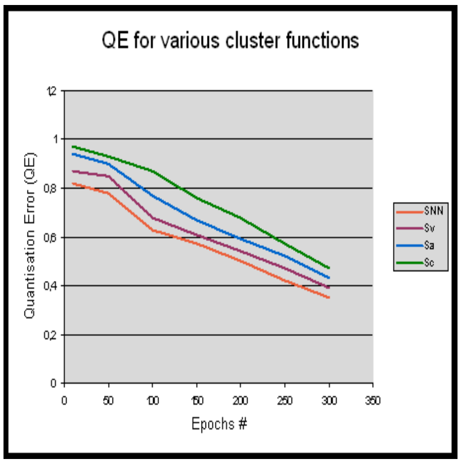


Figure 6.15: QE of various cluster distances

Conclusions

In this thesis experiments on novel machine learning algorithms are performed based on unsupervised and supervised learning techniques. In unsupervised model, we experimented with Self-Organizing Maps (SOM) with new kernel neighborhood functions. The SOM capabilities for data visualization have been explored. Experiments are performed on an astrophysics data catalog (Gamma ray bursts). We used SOM system for data exploration, visualization and cluster discovery in the data sets. The cluster discovery resulted in 3 groups in the data set, which well agrees with the previous studies of GRB studies. The experiment is conducted with two different data sets on the BATSE 4B Catalog. The first case study demonstrates the existence of 3 classes, but the class 3 is not clearly distinguished from class 1 supporting the hypothesis of instrument bias in the misclassification of some events of class 1 as class 3. In the second case study we used the bagoly et.al., suggested attributes . This time the SOM has clearly distinguished the class 2 events from the class1. The cut gaussian kernel makes the winner map unit to get adapted to the data vectors to maximum extent and the adaptation falls away as the distance of map units increases from the winner unit. This gaussian form of adaption to the data set distribution forms the basis for the self organisation with in the map thus assisting in the automatic discovery of the groups that exist in the dataset.

In the ensemble (supervised) learning, we experimented with 2 different machine learning algorithms. BPNN, a neural network model is used to grow ensembles with bagging and boosting. In tree based model we choose random forests to grow ensembles. The methodology is novel and experiments on standard UCI data sets have shown improvements in the classification results compared to the standard random forest algorithm.

The above developed algorithms are used in the image data classification performed with in the MAGIC telescope experiment framework. The research is carried out to perform the classification both in supervised and unsupervised fashion. We used BPNN ensembles and random forest ensembles to perform a supervised classification. The results show that the ensembles have performed superior to the original algorithms with BPNN ensemble giving better performances. All the experiments are carried out using the WEKA software tool for machine learning. One of the important requirements for the MAGIC is to perform the classification of image data in an automatic way. We worked on this task by applying SOM based system. The clusters discovered by SOM are labeled by using a KNN- model algorithm to perform an automatic classification of images. The self-organisation capability of SOM directed by its kernel function, makes the identical types of events to be accumulated at one end of the map where as the other event types are moved toward the other end. Thus the self-organisation capability of the map forms the basis for the automatic classification of events. We used various cluster distance measures for this task and the results are compared.

Future work can be concentrated on working with various image data collections available in the MAGIC experiment. More ensemble learning techniques as described in section 4.3.3 can be introduced to work with the base classifiers introduced in the thesis. On other hand there is a scope of introducing new base classifiers such as K-Nearest Neighbor (K-NN), Radial basis function networks, Probabilistic neural nets, support vector machines etc to grow the ensembles.

In unsupervised learning category, more work can be concentrated on using different map dimensionalities, learning rates. Experiments can be performed by using the SOM variants such as Self-organizing tree algorithm (SOTA) [143], Generative topographic map(GTM) etc. [144].

Bibliography

[1] A. K. Jain, R. P. W. Duin, J. Mao *Statistical Pattern Recognition: A Review*, EEE Transactions on Pattern Analysis and Machine Intelligence (1999)

[2] D. Judd, P. K. McKinley, and A. K. Jain. Large scale parallel data clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8):871876 (1998)

[3] T. M. Mitchell, *Machine Learning*, McGraw Hill (2005)

[4] M. Kantardzic, *Data Mining: Concepts, Models, Methods, and Algorithms*, John Wiley &Sons (2003)

[5] L. Kuncheva, C. Whitaker, *Measures of diversity in classifier ensembles and their relationship with ensemble accuracy*, Machine Learning, pp. 181-207 (2003)

[6] P. Melville, R. J. Mooney, *Constructing Diverse Classifier Ensembles Using Artificial Training Examples*, In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), pp. 505-510, Mexico (2003)

[7] A. D. Gordon. *Classification*, Chapman & Hall/CRC (1999)

[8] H. Frigui, R. Krishnapuram, *A robust competitive clustering algorithm with applications in computer vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5):450465 (1999)

[9] S. K. Bhatia, J. S. Deogun. Conceptual clustering in information retrieval. IEEE Transactions on Systems, Man, and Cybernetics, 28(3):427436 (1998)

[10] C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning, 24(2):95122 (1996)

[11] F. Rosenblatt: *The Perceptron: a perceiving and recognizing automaton,* Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, N.Y., 1957.

[12] J. L. McClelland, D. E. Rumelhart: *Explorations in Parallel Distributed Processing*, Cambridge, MA: MIT Press 1988.

[13] D. E. Rumelhart, G.E. Hinton, R. J. Williams: *Learning Representations by Back-propagating Error*, Nature, 323:533/536. Reprinted in Anderson& Rosenfeld, pp. 696-699, 1988.

[14] S. Haykin: *Neural Networks- A comprehensive Foundation*, Macmillan College Publishing Company, 1994.

[15] L. Fausett: *Fundamentals of Neural Networks*, Prentice Hall, 1994.

[16] K. Hornik, M. Stinchcombe, and H. White: *Multilayer feedforward networks are universal approximators*, Neural Networks, 2:359-366, 1989.

[17] J. Platt, *Fast training of support vector machines using sequential minimal optimization*, In, B. Scholkopf, C. Burges, A. Smola, (eds.): Advances in Kernel Methods - Support Vector Learning, MIT Press (1998)

[18] V. N. Vapnik, *The nature of statistical learning theory*, Springer Verlag (1995)

[19] G. A. carpenter, S. Grossberg, *A massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine*, Computer vision, Graphics and Image processing, 37:54-115 (1987)

[20] J. Friedman, T. Hastie, R. Tibshirani, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer series (2001)

[21] L. Akarun, D. O. zdemir, and O. Yalc, *In. Joint quantization and dithering of color images*, In Proc. IEEE Int. Conf. on Image Processing, pages 557-560 (1996)

[22] M. Aguilar, D. A. Fay, D. B. Ireland, J. P. Racamoto, W. D. Ross, and A. M. Waxman, *Field evaluations of dual-band fusion for color night vision*, In J.G. Verly, editor, SPIE Conference on Enhanced and Synthetic Vision 1999, pages 168-175, (1999)

[23] J. Hartigan, *Clustering Algorithms*, Wiley (1975)

[24] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ (1988)

[25] L. Kaufman, P. J. Rousseeuw, *Finding Groups in Data*, John Wiley & Sons (1990)

[26] R.O. Duda, P. E. Hart, D. G. Stork, Pattern Classification , John Wiley & Sons, Inc (2001)

[27] R. O. Duda, P. E. Hart, *Pattern classification and scene analysis*, John Wiley, New York (1973)

[28] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining* AAAI/MIT Press (1996)

[29] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press (2000)

[30] M. Kamber J. Han, *Data Mining : Concepts and Techniques*, Morgan Kaufmann (2000)

[31] B. D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge (1996)

[32] D. J. Willshaw, C. V. der malsburg, *How patterned neural connections can be setup by self-organization*, Proc.Roy.Soc. Lond. B, 194, 431-445.

[33] T. Kohonen, *Self-Organizing Maps*, third edition Springer-Verlag (2002)

[34] *Website for GRB's*, `http://imagine.gsfc.nasa.gov/docs/science/know_l1/grbs_proof.html`

[35] *Site for GRB data from BATSE:*, `http://www.batse.com/`

[36] *SOM research website:* `http://www.cis.hut.fi/research/som-research/som.shtml`

[37] Mukherjee, S., et al. 1998, ApJ, 508, 314

[38] Hakkila, J., et al. 2000, ApJ, 538, 165

[39] Bagoly, Z., et al. 1998, ApJ, 498, 342

[40] H. J. Rajaneimi, P. Mahonen, *Classifying Gamma-Ray Bursts using Self-Organizing maps*, The Astrophysical Journal, 566:202-209 (2002)

[41] L. Chittaro, L. Ieronutti, *A Visual Tool for Tracing Behaviors of Users in Virtual Environments*, Proceedings of AVI 2004: 6th International Conference on Advanced Visual Interfaces, pp.40-47 ACM Press (2004)

[42] O. Simula, E. Alhoniemi, J. Hollmn, and J. Vesanto. *Monitoring and modeling of complex processes using hierarchical self-organizing maps*, In Proceedings of the 1996 IEEE International Symposium on Circuits and Systems, volume Supplement, pages 73-76. IEEE, (1996)

[43] T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela, *Self organization of a massive document collection* IEEE Transactions on Neural Networks, (11):574-585 (2000)

[44] J. Vesanto, *Importance of Individual Variables in the k-Means Algorithm*, In Proceedings of the Pacific-Asia Conference Advances in Knowledge Discovery and Data Mining (PAKDD2001), Springer-Verlag, pp. 513518 (2001)

[45] J. Vesanto, *Data exploration process based on the SOM*, PhD thesis, Helsinki University of Technology (2002)

[46] T. Kohonen, *Comparison of SOM Point Densities Based on Different Criteria*, Neural Computation, 11(8):20812095 (1999)

[47] J. Lampinen and T. Kostiainen, *Generative probability density model in the Self-Organizing Map*, In U. Seiffert and L. Jain, editors, Self-organizing neural networks: Recent advances and applications, pages 7594. Physica Verlag (2002)

[48] G. Deboeck, T. Kohonen, editors, *Visual explorations in Finance using Self-Organizing Maps*, Springer-Verlag, London (1998)

[49] J. B. Kruskal, *Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis*, Psychometrika, 29(1):127 (1964)

[50] J. Himberg, J. Ahola, E. Alhoniemi, J. Vesanto, and O. Simula, *The selforganizing map as a tool in knowledge engineering*, In N. R. Pal, editor, Pattern recognition in soft computing paradigm, pages 3865. World Scientific (2001)

[51] J. W. Sammon, Jr., *A Nonlinear Mapping for Data Structure Analysis*, IEEE Transactions on Computers, C-18(5):401409 (1969)

[52] P. Demartines, J. Hrault, *Curvilinear Component Analysis: A Self- Organizing Neural Network for Nonlinear Mapping of Data Sets*, IEEE Transactions on Neural Networks, 8(1):148154 (1997)

[53] J. Moody, C. J. Darken, *Fast Learning in Networks of Locally- Tuned Processing Units*, Neural Computation, 1(2):281294 (1989)

[54] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, *Engineering Applications of the Self-Organizing Map*, Proceedings of the IEEE, 84(10):1358-1384, (1996)

[55] O. Simula and J. Kangas, *Neural Networks for Chemical Engineers*, volume 6 of Computer-Aided Chemical Engineering, chapter 14, Process monitoring and visualization using self-organizing maps. Elsevier, Amsterdam (1995)

[56] S. Kaski, J. Kangas, and T. Kohonen, *Bibliography of self-organizing map (SOM) papers: 1981-1997*, Neural Computing Surveys, 1:102-350 (1998)

[57] J. Vesanto, *Neural network tool for data mining: SOM Toolbox*, In Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000), pages 184-196, Oulu, Finland (2000)

[58] R. M. Gray, *Vector quantization*, IEEE ASSP Magazine, pages 4-29, (1984)

[59] S. Kaski, *Data Exploration Using Self-Organizing Maps*, PhD thesis, Helsinki University of Technology, Acta Polytechnica Scandinavica: Mathematics, Computing and Management in Engineering, (1997)

[60] T. Samad, S.A. Harp, *Self-Organization with Partial Data*, Network, 3:205-212, IOP Publishing Ltd, (1992)

[61] H. Ritter, *Asymptotic Level Density for a Class of Vector Quantization Processes*, IEEE Transactions on Neural Networks, 2(1):173175 (1991)

[62] T. G. Dietterich, *Ensemble Methods in Machine Learning*, Proc. of the 1st International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, 1857, 1-15. Springer-Verlag (2000)

[63] T. G. Dietterich, *Ensemble learning*, In Arbib, M. (Ed.), The Handbook of Brain Theory and Neural Networks, 2nd Edition. MIT Press (2002)

[64] T. Dietterich, *Machine learning research: Four current directions*, AI Magazine, 18(4), 97-136 (197)

[65] *Data Analysis hand book website* `http://rkb.home.cern.ch/rkb/AN16pp/node237.html#SECTION000237`

[66] T. G. Dietterich, *Machine Learning*, In Nature Encyclopedia of Cognitive Science, Macmillan, London (2003)

[67] K. Ali, J. Pazzani, *Error reduction through learning multiple descriptions*, Machine, Learning 24(3), 173-202 (1996)

[68] J. M. Bernardo, A. F. M. Smith, *Bayesian Theory*, JohnWiley & Sons, New York (1993)

[69] A. Krogh, J. Vedelsby, *Neural network ensembles, cross validation, and active learning*, In G. Tesauro, D. Touretzky, T. Leen (Eds), Advances in Neural Information Processing System, 7, 231-238. MIT Press (1995)

[70] D. Opitz, J. Shavlik, *Generating accurate and diverse members of a neural-network ensemble*, In D. Touretzky, M. Mozer, M. Hesselmo, (Eds.), Advances in Neural Information Processing Systems, 8, 535-451. MIT Press (1996)

[71] T. G. Dietterich, *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting and randomization*, Machine Learning 40 (2) 139-157 (2000)

[72] Y. H. Hu, J. N. Hwang, *Handbook of Neural Network Signal Processing*, CRC Press (2002)

[73] L. Breiman, *Bagging predictors* Machine Learning, 24(2), 123-140 (1996)

[74] Y. Freund, R. Schapire, *Experiments with a new boosting algorithm* In Saitta, L. (Ed), Proceedings of the 13th International Conference on Machine Learning, 148-156. Morgan Kaufmann (1996)

[75] K. Cherkauer, *Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks*, In P. Chan, editor, Working Notes of the AAAI Workshop on Integrating Multiple Learned Models,, pages 15-21. (1996)

[76] J. Carney, P. Cunningham, *The NeuralBAG algorithm: optimizing generalization performance in Bagged Neural Networks*, In: Verleysen, M. (eds.): Proceedings of the 7th European Symposium on Artificial Neural Networks , pp. 3540 (1999)

[77] B. Efron, R. Tibshirani, *An Introduction to the Bootstrap*, Chapman and Hall (1993)

[78] J.H. Friedman, *Bias, variance, 0-1 loss and the curse of dimensionality*, Technical report, Stanford University (1996)

[79] Y. Grandvalet, *Bagging can stabilize without reducing variance*, In ICANN'01, Lecture Notes in Computer Science. Springer (2001)

[80] P. Domingos, *Why does bagging work? A bayesian account and its implications*, In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, (eds), Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), page 155. AAAI Press (1997)

[81] J. A. Madigan, D. Hoeting, C. T. Raftery, *Bayesian model averaging: A tutorial. Statistical Science*, 44(4):382417 (1999)

[82] R. E. Schapire, *The strength of weak learnability*, Machine Learning, 5:197-227 (1990)

[83] Y. Freund, *Boosting a weak learning algorithm by majority*, Information and Computation, 121(2):256-285 (1995)

[84] E. B. Kong, T. G. Dietterich, *Error-correcting output coding corrects bias and variance*, In Proceedings of the Twelfth International Conference on Machine Learning, pages 313-321 (1995)

[85] H. Drucker, R. Schapire, P. Simard, *Boosting performance in neural networks*, International Journal of Pattern Recognition and Artificial Intelligence, 7(4):705-719 (1993)

[86] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, *Boosting and other ensemble methods*, Neural Computation, 6(6):1289-1301 (1994)

[87] H. Drucker, C. Cortes, *Boosting decision trees*, In Advances in Neural Information Processing Systems 8 (1996)

[88] J. C. Jackson, M. W. Craven, *Learning sparse perceptrons*, In Advances in Neural Information Processing Systems 8 (1996)

[89] M. Skurichina, R. P. W. Duin, *Bagging, Boosting and the Random Subspace Method for Linear Classifiers*, Vol. 5, no. 2: 121-135, Pattern Analysis and Applications (2002)

[90] P. Bhlmann, B. Yu, *Analyzing bagging*, Annals of Statistics 30, 927-961 (2002)

[91] L. Breiman, *Out-of-bag estimation*, Technical Report (1996)

[92] S. Borra, A. Di Ciaccio, *Improving nonparametric regression methods by bagging and boosting*, Computational Statistics & Data Analysis 38, 407-420 (2002)

[93] A. J. C. Sharkey, *Types of multinet system*, In Proc. of the Third International Workshop on Multiple Classifier Systems (2002)

[94] L. Xu, A. Krzyzak, C. Y. Suen, *Methods for combining multiple classifiers and their applications in handwritten character recognition,* IEEE Trans. Syst., Man, Cybern., vol. 22, pp. 418-435, 1992.

[95] Y. Freund, R. E. Schapire, *A decision-theoretic generalization of online learning and an application to boosting*, In Computational Learning Theory: Second European Conference, EuroCOLT '95, pages 23-37, Springer-Verlag (1995)

[96] L. Breiman, *Random Forests Technical Report,* University of California, 2001.

[97] *Website for Random Forests:* `http://www.stat.berkeley.edu/users/breiman`

[98] L. Breiman, *Looking Inside the Black Box,* Wald Lecture II, Department of Statistics, California University (2002)

[99] M. Sikonja, *Improving Random Forests.* , In J.-F. Boulicaut et al.(Eds): ECML 2004, LNAI 3210: pp. 359-370, Springer, Berlin, (2004)

[100] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. *Classification and regression trees.* Wadsworth Inc., Belmont, California (1984)

[101] J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann, San Francisco, (1993)

[102] I. Kononenko. *Estimating attributes: analysis and extensions of Relief.* In Luc De Raedt and Francesco Bergadano, editors, Machine Learning: ECML-94, pages 171-182. Springer Verlag, Berlin (1994)

[103] I. Kononenko. *On biases in estimating multi-valued attributes.* In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95), pages 1034-1040. Morgan Kaufmann (1995)

[104] T. G. Dietterich, M. Kerns, and Y. Mansour. *Applying the weak learning framework to understand and improve C4.5.* In Lorenza Saitta, editor, Machine Learning: Proceedings of the Thirteenth International Conference (ICML'96), pages 96-103. Morgan Kaufmann, San Francisco (1996)

[105] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wadsworth Inc., Belmont, California (1984)

[106] *UCI data sets repository*, `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[107] J.R. Quinlan, *Bagging, Boosting, and C4.5,* In Proceedings, Fourteenth National Conference on Artificial Intelligence (1996)

[108] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. *Boosting the margin: a new explanation for the effectiveness of voting methods.* In D. H. Fisher, editor, Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97), pages 322-330. Morgan Kaufmann (1997)

[109] *Website for MAGIC Telescope:* http://wwwmagic.mppmu.mpg.de

[110] A. Moralejo for the MAGIC Collaboration, *The MAGIC telescope for gamma-ray astronomy above 30 GeV*, Chin. J. Astron. Astrophys. Vol.3 Suppl. 531-538 (2003)

[111] D. Paneque, *The MAGIC Telescope: development of new technologies and first observations*, PhD thesis , MPI Munich, August (2004)

[112] *Nasa website on cernkov radiation:* http://imagine.gsfc.nasa.gov/docs/science/how_l2/Cherenkov

[113] P. A. Cherenkov, *C.R. Acad. Sci. U.S.S.R.*, 2, p. 451 (1934)

[114] *ROOT website:* http://root.cern.ch/

[115] *MARS website:* http://magic.astro.uni-wuerzburg.de/mars/

[116] D. Kranich, *Temporal and spectral characteristics of the active galactic nucleus Mkn -501 during a phase of high activity in the TeV range*, PhD thesis, Max-Plank Institute of Astro-physics, Munich (2002).

[117] T. C. Weekes, M.F. Cawley, D.J. Fegan et al.,*Observation of TeV gamma rays from the Crab nebula using the atmospheric Cherenkov imaging technique*. Astrophysical Journal, 342: 379-395 (1989)

[118] A.M. Hillas, Proc. 19th ICRC, La Jolla 3 (1985) 445

[119] P.T. Reynolds et al., ApJ 404 (1993) 206

[120] F. Barbarino, *Gamma-hadron separation using neural network techniques,* Tesi di Laurea, Udine University (2004)

[121] D.J. Fegan, J.Phys.G: Nucl. Part. Phys. 23 (1997) 1013

[122] W. Wittek, *Image Parameters* , MAGIC-TDAS 02-03, Internal Technical report (2002)

[123] D.J.Fegan,*Gamma/hadron separation at TeV energies*, Topical Review, J.Phys.G: Nucl. Part. Phys. 23 (1997) 1013.

[124] R. K. Bock et al., *Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope*, Nucl. Instr. Methods A516: 511-528 (2004)

[125] D. Heck, et al., *CORSIKA, A Monte Carlo code to simulate extensive air showers*, Forschungszentrum Karlsruhe, Report FZKA 6019 (1998)

[126] Ian H. Witten, E. Frank, *Data Mining: Practical machine learning tools and techniques,* 2nd Edition, Morgan Kaufmann, San Francisco (2005)

[127] *R statistical software:* www.r-project.org/

[128] *Fortran Source code for random forests* `http://www.stat.berkeley.edu/users/breiman/RandomForests`

[129] Y. Freund, R. E. Schapire, *A decision-theoretic generalization of online learning and an application to boosting*, In Computational Learning Theory: Second European Conference, EuroCOLT '95, pages 23-37, Springer-Verlag (1995)

[130] H. Schwenk, Y.Bengio, *Boosting Neural Networks*, Neural Computation, Volume 12, Number 8, pp. 1869-1887(19), MIT Press (2000)

[131] P. Cunningham, J. Carney, S. Jacob, *Stability Problems with Artificial Neural Networks and the Ensemble Soloution* , Technical report, Department of Computer Science, Trinity College Dublin, Ireland (2003)

[132] B. Efron, R. Tibshirani, *An Introduction to the Bootstrap*, Monographs on Statistics and Applied Probability, No 57, Chapman and Hall (1994)

[133] D. Opitz, J. Shavlik, *Generating accurate and diverse members of a neural network ensemble*, in D. Touretzky, M. Mozer and M. Hasselmo, eds.,Advances in Neural Information Processing Systems 8, 535-541, MIT Press (1996)

[134] J. Carney, P. Cunningham, *Confidence and prediction intervals for neural network ensembles,* in proceedings of IJCNN99, The International Joint Conference on Neural Networks, Washington, USA (1999)

[135] R. M. Wagner et al., *Observations of the Crab nebula with the MAGIC telescope*, 29^{th} International Cosmic Ray Conference Pune (2005) 00, 101.106

[136] T. Weekes et al., Astrophys. J. 342 (1998) 379

[137] F. A. Aharonian et al, Astrophys. J. 614 (2004) 897

[138] T. Li, Y. Ma, ApJ, 272, 317 (1983)

[139] E. Alhoniemi, *Unsupervised Pattern Recognition Methods for Exploratory Analysis of Industrial Process Data*, PhD thesis, Helsinki University of Technology (2002)

[140] J. Vesanto, E. Alhoniemi *Clustering of the Self-Organizing Map.* In IEEE Transactions on Neural Networks, Volume 11, Number 3, pp. 586600 (2000).

[141] J. Vesanto, *Neural network tool for data mining: SOM Toolbox*, In Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000), pages 184-196, Oulu, Finland (2000)

[142] A. R. Smith. *Color gamut transform pairs*, Computer graphics, 12(3):1219 (1978)

[143] J. Dopazo, J. M. Carazo, *Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree*, Journal of Molecular Evolution 44, 226-233 (1997)

[144] C. M. Bishop, M. Svensn, C. K. I. Williams, *GTM: The generative topographic mapping. Neural Computation*, 10:215234 (1998)