Università degli Studi di Salerno

Dottorato di Ricerca in Informatica

II Ciclo Nuova Serie

Unsupervised Neural Networks for the Extraction of Scientific Information from Astronomical Data

Antonino Staiano

February 2004

Coordinatore:
Prof. A. De Santis

Relatore:
Prof. R. Tagliaferri

Contents

List of Figures

List of Tables

" I have not failed. I've just found 10.000 *ways that don't work."*
Thomas Alva Edison

"An important scientific innovation
rarely makes its way by gradually winning over
and convincing its opponents:
... what does happen is that the opponents gradually die out"
Max Planck

"... Kids if you want some fun
See what you never have seen
Take off your cheaters and sit right down
Start the projection machine"
Steely Dan
Every one's gone to the movies, 1975

Acknowledgements

This thesis is dedicated
to my nephews Lorenzo and Giampaolo.
I hope this may be an encouragement to pursue their objectives
with tenacity, in whatever field they will choose as a target for their passions.

A lot of people contributed, in one way or another, to let me reach one of the goal I ever dreamt. My deep gratitude goes to my advisor, Prof. Roberto Tagliaferri, for giving me his trust and for the supervision, guidance and kindness he provided when I first met him after my graduation. I am indebted with Prof. Giuseppe Longo who introduced me in one of the most fascinating field of science, astronomy; his enthusiasm and friendship made my research simpler. Part of the work of this thesis was done during a stage at the Space Telescope European Coordinating Facility, in Munich (Germany). I wish to thank its Director Prof. Piero Benvenuti for his warm hospitality and for his guidance. I also thank Prof. Vincenzo Loia who first gave me a chance to become a researcher and Prof. Raffaele Cerulli for his kindness and, above all, for his wonderful printer. A special mention goes to Prof. Witold Pedrycz, Prof. Salvatore Sessa and Prof. Giancarlo Raiconi for their precious help and suggestions. I am grateful to my family who loved and loves me unconditionally: my parents, Anna and Bernardo, whose support has been ever increasing over the years I spent in studying; to my brother Gianni who played a fundamental role for rising in me the passion for science and research, his strong encouragement has been decisive, and last but not least, my gratitude goes to my other brother Paolo who gave

me the amount of rationality which permitted me to win over my laziness and over my "fantasy flight". Moreover, special thanks go to all my dear friends who shared with me fun and working times: Sergio, Angelo Santangelo, Francesca, Umberto, Aniello, Nello, Ciro, Alfredo, Lara Giordano and finally, Angelo Ciaramella who kindly helped me with many scientific suggestions over these years. Maurilio Panella of the *Max Planck Institute for Extraterrestrial Physics* and Alberto Micol of the *European Southern Observatory*, deserve my gratitude for their friendship and for their kindness for providing me with part of the data I used in this thesis and the astronomical notions. There is still a "little" free spot for Lara: her love, patience, comprehension and support were (and I hope will be in future) invaluable and this work would not be the same without her contribute.

Introduction

A paradigm shift is now taking place in astronomy and space science. Astronomy has suddenly become an immensely data-rich field, with numerous digital sky surveys across a range of wavelengths, with many terabytes of pixels and with billions of detected sources, often with tens of measured parameters for each object. Conservative predictions lead to expect that in less then five years, much more than 10 TB of data will be acquired worldwide every night and, due to the ongoing efforts for the implementation of the International Virtual Observatory (IVO)[1], most of these data will become available to the astronomical community worldwide via the network [22]. These huge and heterogeneous data sets will open possibilities which so far were just unthinkable, but it is already clear that their full and effective scientific exploitation will require the implementation of automatic tools capable to perform a large fraction of the routine data reduction, data mining and data analysis work, posing considerable technical and even deeper, methodological challenges, since traditional astronomical data analysis methods are inadequate to cope with this sudden increase in the data volume and especially in the data complexity (ten or hundreds of dimensions of the parameter space) [23]. These challenges, therefore, require strong interdisciplinary activities. Astronomers, for example, already begun to collaborate with statisticians [37], [52]. Non parametric statistical methods, in fact, have great potential for astrophysical data analysis. They provide a way to make inference about complex structures from massive data sets without overlay restrictive assumptions or intractable computations. However, these challenges especially require substantive collaborations and partnerships between researchers in astronomy and computer science, promising to bring relevant advances to both fields. In the last few years, indeed, there has been an increased interest toward astronomical applications of machine learning methodologies, and Neural Networks in particular, even though, in spite of a great variety of problems addressed, most astronomical applications still make use of an handful of neural models only [62],[1],[38]. This thesis is devoted to this fascinating field, carrying on the works [47], [49], [48] started a couple of years ago, and focusing on unsupervised methodologies for probability density estimation.

[1]From the fusion of the European Astrophysical Virtual Observatory (AVO) and of the American National Astrophysical Observatory (NVO), http://www.ivoa.net/.

In the field of pattern recognition any method that incorporates information from training samples employs learning. Learning refers to some form of algorithm for reducing the error on a set of training data. Learning comes in several general forms, and mainly as: a) supervised learning, in which a "teacher" provides a category label or cost for each pattern in a training set, and seeks to reduce the sum of the costs for these patterns; b) unsupervised learning, where there is not any explicit teacher, and the systems form clusters or natural groupings of the input patterns. There are at least five main reasons to be interested in unsupervised procedures:

1. collecting and labelling a large set of sample patterns, as it would be required by the implementation of a training set, can be surprisingly costly. If a learning algorithm can be crudely designed on a small set of labelled samples, and then tuned up by allowing it to run without supervision on a large, unlabelled set, much time and trouble can be saved;

2. one might wish to proceed in the reverse direction: training with a large amount of (less expensive) unlabelled data, and only then use supervision to label the groupings found. This may be appropriate for large data mining applications;

3. in many applications, the characteristics of the patterns can change slowly with time. If these changes can be tracked by a learning system in an unsupervised mode, improved performance can be achieved;

4. we can use unsupervised methods to find features that will be useful for the categorization;

5. in the early stages of investigation it may be valuable to perform exploratory data analysis and thereby gain some insights into the nature or structure of the data. The discovery of distinct subclasses, clusters or groups of patterns whose members are more similar to each other than they are to other patterns, significantly alters our approach to designing the learning system.

Many pattern recognition tasks, such as classification, regression, novelty detection can be viewed in terms of probability density estimation. A powerful approach to probabilistic modelling is to represent the observed variables in terms of a number of hidden, or latent, variables. By defining a joint distribution over visible and latent variables, the

corresponding distribution of the observed variables is then obtained by marginalization. This allows relatively complex distributions to be expressed in terms of more tractable joint distributions over the expanded variable space. Such models may be employed for a number of tasks and there are several successful applications they are involved in [9],[14]. Among all, two of the most successful and well developed latent variable models are the Generative Topographic Mapping [6] and the Probabilistic Principal Surfaces [18]. These models are very appealing for the flexibility they exhibit in a wide range of tasks such as density modelling, classification and data visualization, which are crucial activities for any astronomical data mining process. On the other hand, so far their effectiveness has been tested only on synthetic data sets or on a limited number of complex data sets. Aim of this thesis is to prove their usefulness in the context of scientific astronomical data for density modelling, classification as well as data visualization purposes. The thesis is organized as follows. Chapter 1 provides the general concepts of Knowledge Discovery and Data Mining techniques, and gives details about astronomical data types and data mining tasks in astronomical scientific data analysis. Chapter 2 introduces latent variable models describing in detail the Generative Topographic Mapping and Probabilistic Principal Surfaces models. The models are then evaluated towards classification of complex data. Chapter 3 discusses how to enhance the classification performance of the models by introducing the concept of ensemble methods in machine learning. Afterwards, two combining schemes are proposed and discussed on the basis of experimental results. Chapter 4 addresses the issue of data visualization showing the possibilities offered, in particular, by Probabilistic Principal Surfaces. Finally, concluding remarks and future research directions are provided in Chapter 5.

Chapter 1

Astronomical Data Mining

Across a wide variety of fields, data are being collected and accumulated at a dramatic pace. There is an urgent need for a new generation of computational theories and tools to assist humans in extracting useful information (knowledge) from the rapidly growing volumes of digital data. These theories and tools belong to the field of Knowledge Discovery in Databases (KDD). At an abstract level, the *KDD* field is concerned with the development of methods and techniques aimed at extracting meaning out of data. The basic problem addressed by the *KDD* process is one of mapping low-level data (which are typically too voluminous to be understood and digested easily) into other forms that might be either more compact (for example, a descriptive approximation or model of the process that generated the data), or more useful (for example, a predictive model for estimating the value for future cases). At the core of the process there is the application of specific data mining methods for pattern discovery and extraction [29],[28],[32].

1.1 Why do we need *KDD*?

The traditional method of tuning data into knowledge, relies on manual analysis and interpretation. Be it science, marketing, finance or any other field, the classical approach to data analysis relies fundamentally on one or more analysts becoming intimately familiar with the data and serving as an interface between the data and the users and products. For these (and many other) applications, this type of manual probing of a data set is slow, expensive, and highly subjective. In fact, as data volumes grow dramatically, this type of manual data analysis is becoming unfeasible in many domains. Databases are increasing in size in two ways: (1) the number N of records or objects in the database

and (2) the number d of fields or attributes of an object. Databases containing the order of $N = 10^9$ objects are becoming increasingly common also in astronomy. Who could be expected to digest millions of records, each having hundreds of fields? Since computers have enabled humans to gather more data than what they can digest, it is necessary to rely on computational techniques capable to unearth meaningful patterns and structures from massive volumes of data [29].

1.2 The *KDD* Process

Following the definition given in [31], the KDD process may be defined as: *The non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

Data comprise a set of facts (e.g, cases in a database), and pattern is an expression in some language describing a subset of the data (or a model applicable to that subset). The term process implies there are many steps involving data preparation, search for patterns, knowledge evaluation, and refinement, all repeated in multiple iterations. The process is assumed to be non trivial in that it goes beyond computing closed-form quantities; that is, it must involve search for structure, models, patterns, or parameters. The discovered patterns should be valid for new data with some degree of certainty. We also want patterns to be novel (at least to the system, and preferably to the user) and potentially useful for the user or task. Finally, the patterns should be understandable if not immediately, at least after some postprocessing. This definition implies that we can define quantitative measures for evaluating extracted patterns. In many cases, it is possible to define a measure of certainty (e.g., estimated classification accuracy) or utility. Notions such as novelty and understandability, can be estimated through simplicity. An important notion, called interestingness, is usually taken as an overall measure of pattern value, combining validity, novelty, usefulness and simplicity. The interestingness function can be explicitly defined or can be manifested implicitly through an ordering placed by the *KDD* system on the discovered patterns or models. Data mining is a step in the *KDD* process consisting of an enumeration of patterns (or models) over the data, subject to some acceptable computational-efficiency limitations. Since the patterns enumerable over any finite data set are potentially infinite, and because the enumeration of patterns involves some form of

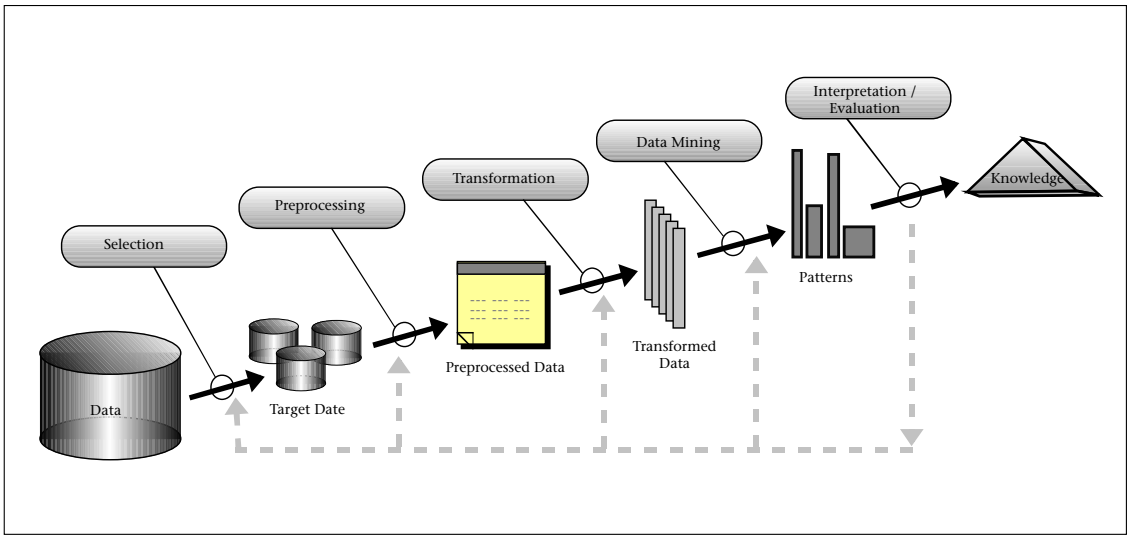

Figure 1.1: Outline of the *KDD* process.

search in a large space, computational constraints place severe limits on the subspace that can be explored by a data mining algorithm. The *KDD* process is outlined in figure 1.1. The *KDD* process is interactive (with many decisions made by the user) and iterative, involving several steps, which can be summarized as:

1. Learning the application domain: includes relevant prior knowledge and the goals of the application;

2. Creating a target data set: includes selecting a data set or focusing on a subset of variables or data samples on which discovery is to be performed;

3. Data cleaning and preprocessing: includes basic operations, such as removing the noise or outliers if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time sequence information and known changes;

4. Data reduction and projection: includes finding useful features to represent the data, depending on the goal of the task, and using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representation for the data;

5. Choosing the function of data mining: includes deciding the purpose of the model derived by the data mining algorithm (e.g., summarization, classification, regression and clustering);

6. Choosing the data mining algorithm: includes selecting methods to be used for searching for patterns in the data, such as deciding which models and parameters may be appropriate and matching a particular data mining method with the overall criteria of the *KDD* process (e.g., the user may be more interested in understanding the model than in its predictive capabilities);

7. Data mining: includes searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, clustering, sequence modelling, dependency and line analysis;

8. Interpretation: includes interpreting the discovered patterns and possibly returning to any of the previous steps, as well as possible visualization of the extracted patterns, removing redundant or irrelevant patterns, and translating the useful ones into terms understandable by the users;

9. Using discovered knowledge: includes incorporating this knowledge into the performance system, taking action based on the knowledge, or simply documenting it and reporting it to interested parties, as well as checking for, and resolving potential conflicts with previously believed (or extracted) knowledge.

We now focus on the data mining component, which has received by far the most attention in literature, nevertheless, all the steps of a *KDD* process are equally important for the successful application of *KDD* to practical cases.

1.3 Data Mining Methods

Data Mining involves fitting models to, or determining patterns of data. The fitted models play the role of inferred knowledge. A wide variety of data mining algorithms are described in literature, from the field of statistics, pattern recognition, machine learning and databases. From a very general viewpoint, data mining techniques can be divided into five classes of methods.

1.3.1 Predictive Modelling

The goal is to predict the value of some fields in a database based on the values of other fields. If the field being predicted is a numeric (continuous) variable (such as a physical measurement) then the prediction problem is a regression problem. If the field is categorical, then it is a classification problem. There is a wide variety of techniques for classification and regression [26]. The problem in general is defined as determining the most likely value of the variable being predicted, given the other fields (inputs), training data (in which the target variable is given for each observation), and a set of assumptions representing one's prior knowledge of the problem. Linear regression combined with non-linear transformation on inputs could be used to solve a wide range of problems. Transformations of the inputs space are typically a difficult problem requiring knowledge of the problem. In classification problems this type of transformation is often referred to as "feature extraction". In classification the basic goal is to predict the most likely state of a categorical variable (the class). This is fundamentally a density estimation problem. If one can estimate the probability that the class $C = c$, given the other field $X = x$ for some feature vector x, then one could derive this probability from the joint density on C and X. However, this joint density is rarely known and very difficult to estimate. Hence one has to study to various estimation techniques. In Chapter 2 and 3 we shall focus on latent variable models for density estimation.

1.3.2 Clustering

Clustering does not specify fields to be predicted but targets separating the data items into subsets that are similar to each other. Since we do not know the number of desired "clusters", clustering algorithms typically employ a two stage search: an outer loop over possible cluster numbers and an inner loop to fit the best possible clustering for a given number of clusters. Given the number K of clusters, clustering methods can be divided into three classes:

1. Metric-distance based methods: a distance measure is defined and the objective becomes finding the best K-way partition such as cases in each block of the partition are closer to each other (or centroid) than to cases in other clusters.

2. Model-based methods: a model is assumed for each of the clusters and the idea is to find the best fit of that model to each cluster. One way to score the fit of a model to a cluster is via likelihood.

3. Partition-based methods: basically enumerate various partitions and then score them by some criterion.

1.3.3 Data Summarization

Sometimes the goal is to extract compact patterns that describe subsets of the data. There are two classes of methods which represent horizontal (cases) or vertical (fields) slices of the data. In the former, one would like to produce summaries of subsets: e.g. producing sufficient statistics, or logical conditions that hold for subsets. In the latter case, one would like to predict relations between fields. The goal, for this class of methods is to find relations between fields. One classical method used in literature is called association rules. Associations are rules that state that specific combinations of values occur with other combinations of values with a forecasted frequency and certainty.

1.3.4 Dependency Modelling

Insight into data is often gained by deriving some causal structure within the data. Models of causality can be probabilistic (as in deriving some statement about the probability distribution governing the data) or they can be deterministic as in deriving functional dependencies between fields in the data. Density estimation methods in general fall under this category.

1.3.5 Change and Deviation Detection

These methods account for sequence information, be it time-series or some other ordering. The distinguishing feature of this class of methods is that ordering of observations is important and must be accounted for.

1.4 The Nature of Astronomical Data

Let us now give some details on the basic features of astronomical data. By its inherent nature, astronomical data are extremely heterogeneous, in both format and content. Astronomers are now exploring all regions of the electromagnetic spectrum, from gamma-rays through radio wavelengths. With the advent of new facilities, previously unexplored domains in the gravitational spectrum will soon be available. Computational advances have enabled detailed physical simulations which rival the largest observational data sets in terms of complexity. In order to truly understand our cosmos, we need to assimilate all of this data, each presenting its own physical view of the Universe, and requiring its own technology. Despite all of this heterogeneity, however, astronomical data and its subsequent analysis can be broadly classified into five domains. In order to clarify later discussions, we briefly discuss these domains and define some key astrophysical concepts.

- Imaging data is the fundamental constituent of astronomical observations, capturing a two-dimensional spatial picture of the Universe within a narrow wavelength region at a particular epoch or instant of time. Astrophysical pictures are generally taken through a specific filter, or with an instrument covering a limited range of the electromagnetic spectrum, which defines the wavelength region of the observation. Astronomical images (see figure 1.2, as an example)[12] can be acquired directly, e.g., with imaging arrays such as CCDs[1], or synthesized from interferometric observations as it is customarily done in radio astronomy.

- Catalogs are generated by processing the imaging data. Each detected source can have a large number of measured parameters, including coordinates, various flux quantities, morphological information, and areal extant. In order to be detected, a source must stand out from the background noise (which can be either cosmic or instrumental in origin). The significance of a detection is generally quoted in terms of σ, which is a relative measure of the strength of the source signal relative to the dispersion in the background noise. We note that the source detection process is generally limited both in terms of the flux (total signal over the background) and surface brightness (intensity contrast relative to the background). Coordinates are

[1] Charge Coupled Device, a digital photon counting device that is superior to photographic images in both the linearity of their response and quantum efficiency

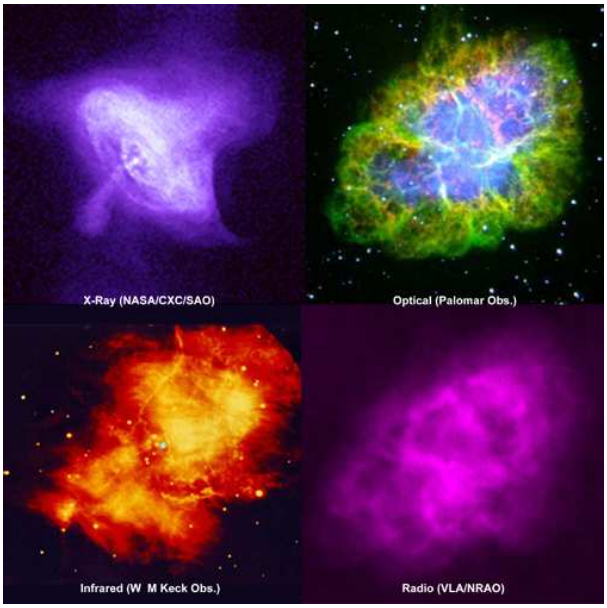

Figure 1.2: A multi-wavelength view of the Crab nebula.

used to specify the location of astronomical sources in the sky. While this might seem obvious, the fact that we are sited in a non-stationary reference frame (e.g, the earth rotates, revolves around the sun, and the sun revolves around the center of our Galaxy) complicates the quantification of a coordinate location. In addition, the Earth's polar axis precesses, introducing a further complication. As a result, coordinate systems, like Equatorial coordinates, must be fixed at a particular instant of time (or epoch), to which the actual observations, which are made at different times, can be transformed. One final caveat is that nearby objects (e.g., solar system bodies or nearby stars) move on measurable timescales. Thus the date or precise time of a given observation must also be recorded. Flux quantities determine the amount of energy that is being received from a particular source. Since different physical processes emit radiation at different wavelengths, most astronomical images are obtained through specific filters. The specific filter(s) used varies, depending on the primary purpose of the observations and the type of recording device. Historically, photographic surveys used filters which were well matched to the photographic material, and have names like *O*, *E*, *J*, *F*, and *N*. Modern digital detectors have different characteristics (including much higher sensitivity), and work primarily with different

filter systems,which have names like U, B, V, R and I, or g, r, i, in the optical, and J, H, K, L, M and N in the near-infrared. In the optical and infrared regimes, the flux is measured in units of magnitudes (which is essentially a logarithmic re-scaling of the measured flux) with one magnitude equivalent to -4 decibels. The zeropoint of the magnitude scale is determined by the star *Vega*, and thus all flux measurements are relative to the absolute flux measurement of this star. Measured flux values in a particular filter are indicated as B= 23 magnitudes, which means the measured B band flux is 100.4×23 times fainter than the star *Vega* in this band.

- Spectroscopy, Polarization, and other follow-up measurements provide detailed physical quantification of the target systems, including distance information (e.g., redshift, denoted by z for extragalactic objects), chemical composition, and measurements of the physical (e.g., electromagnetic, or gravitational) fields present at the source.

- Studying the time domain provides important insights into the nature of the Universe, by identifying moving objects (near -Earth objects and comets), variable sources (pulsating stars), or transient objects (supernovae, and gamma ray bursts). Studies in time domain either require multiple epoch observations of fields (which is possible in the overlap regions of surveys), or dedicated synoptic surveys. In either case, the data volume, and thus the difficulty in handling and analyzing the resulting data, increase significantly.

1.4.1 *Telescopio Nazionale Galileo* Data

The Telescopio Nazionale Galileo (*TNG*), with a primary mirror of $3.58m$, is the national facility of the Italian astronomical community[2], and is located at the Canary Island of La Palma, near the top of the Roque de los Muchachos, at an altitude of $2358m$. It is operated by the Centro Galileo Galilei (CGG) which was created in 1997 by the Consorzio Nazionale per l'Astronomia e l'Astrofisica (CNAA). In 2002 it became a part of the Italian National Institute of Astrophysics (*INAF*) which is ensuring its financial support.
The data collected at the *TNG* are stored together with the telemetry data monitoring the weather condition, the dome and the telescope operational parameters into the Long

[2]http://www.tng.iac.es for more details

Term Archive (*TNG-LTA*). The goal of the present work is to find whether there is any correlation among operational parameters and the quality of the final image. The existence of a such a correlation would play a double role:

1. it would allow to put a quality flag on the scientific exposures;
2. it would allow to asses the quality of the final image while the exposure is being acquired thus avoiding wastes of precious observing time.

1.5 Data Mining, Knowledge Discovery and Astronomical Data

Crucial to maximize the knowledge extracted from the ever-growing quantities of astronomical data, is the successful application of data mining and knowledge discovery techniques. This effort is a step towards the development of the next generation of science analysis tools that will redefine the way scientists interact and extract information from large data sets. In our specific case, the new digital sky survey archives, which are driving the need for a virtual observatory. Such techniques are rather general, and will find several applications outside astronomy and space science. In fact, these techniques can find application in virtually every data-intensive field. Examples of particular studies may include:

Classification methods In order to categorize objects or cluster of objects of interest. Do they objectively found groupings of data vectors correspond to physically meaningful, distinct types of objects? Are the known types recovered, and are there new ones? Can we refine astronomical classifications of object types in an objective manner?

Unsupervised methods Clustering techniques, mixture models to find groups of interest, to come up with descriptive summaries, and to build density estimates for large data set. How many distinct type of objects are present in the data, in some statistical and objective sense? This would be an effective way to group data for specific studies, e.g., some users would want only stars, others only galaxies, etc. They can be useful even to detect rare, anomalous, or somehow unusual objects, e.g., outliers

in the parameter space, to be selected for further investigation. This would include both known but rare classes of objects, e.g, brown dwarf, high redshift quasars, and possibly new and previously unrecognized types of objects and phenomena.

Visualization Effective new data visualization and presentation techniques, which can convey most of the multidimensional information in a way more easily grasped by a human user. Effective and powerful data visualization would be an essential part of any virtual observatory. The human eye and brain are remarkably powerful in pattern recognition, and selection of interesting features. The technical challenge here is posed by the sheer size of the data sets (both in the image and catalog domain), and the need to move through them quickly and to interact with them "on the fly". Here we focus on displaying the information only in the parameter spaces defined in the catalog domain, where each object may be represented by a data vector in tens or even hundreds of dimensions, but only a few can be displayed at any given time (e.g., 3 spatial dimensions, color, shape, and intensity for displayed objects).

The above examples are moving beyond merely providing assistance with handling of huge data set: these software tools may become capable of independent or cooperative discoveries, and their application may greatly enhance the productivity of practicing scientists.

Chapter 2

Probability Density Estimation

In this chapter latent variable models for density estimation are introduced. After a brief introduction on density modelling in general we formally define latent variable models and describe Generative Topographic Mapping and Probabilistic Principal Surfaces.

2.1 Density Modelling

One of the central problem in pattern recognition is that of density estimation, i.e., the construction of a model of a probability distribution given a finite sample of data drawn from that distribution. For now on we consider the problem of modelling the distribution of a set of continuous variables $t_1, \ldots, t_D$ which are denoted collectively by the vector $\mathbf{t}$. A standard approach to the problem of density estimation involves parametric models in which a specific form for the density is proposed which contains a number of adaptive parameters. Values for these parameters are then determined from an observed data set $\mathcal{T} = \{\mathbf{t}_1, ..., \mathbf{t}_N\}$ consisting of N data vectors. The most widely used parametric model is the normal, or Gaussian, distribution given by

$$p(\mathbf{t}|\mu, \mathbf{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(\mathbf{t}-\mu)\mathbf{\Sigma}^{-1}(\mathbf{t}-\mu)^T\right\} \tag{2.1}$$

where μ is the mean, $\mathbf{\Sigma}$ the covariance matrix, and $|\mathbf{\Sigma}|$ denotes the determinant of $\mathbf{\Sigma}$. One technique for setting the values of these parameters is that of maximum likelihood which involves consideration of the log probability of the observed data set given the parameters, i.e.

$$\mathcal{L}(\mu, \mathbf{\Sigma}) = \ln p(\mathcal{T}|\mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} \ln p(\mathbf{t}_n|\mu, \mathbf{\Sigma}) \tag{2.2}$$

in which it is assumed that the data vectors $\mathbf{t}_n$ are drawn independently from the distribution. When viewed as a function of μ and $\mathbf{\Sigma}$, the quantity $p(\mathcal{T}|\mu, \mathbf{\Sigma})$ is called *likelihood* function. Maximization of the likelihood (or log likelihood) with respect to μ and $\mathbf{\Sigma}$ leads to the set of parameter values which are most likely to have given rise to the observed data set. For the normal distribution (2.1) the log likelihood (2.2) can be maximized analytically, leading to the result that the maximum likelihood solutions $\hat{\mu}$ and $\hat{\mathbf{\Sigma}}$ are given by

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{t}_n \tag{2.3}$$

$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{t}_n - \hat{\mu})(\mathbf{t}_n - \hat{\mu})^T \tag{2.4}$$

corresponding to the sample mean and sample covariance, respectively.

While the simple normal distribution (2.1) is widely used, it suffers from some significant limitations. In particular, it can often prove to be too flexible in that the number of independent parameters in the model can be excessive. This problem is addressed through the introduction of continuous latent variables. On the other hand, the normal distribution can also be insufficiently flexible since it can only represent uni-modal distributions. A more general family of distributions can be obtained by considering mixtures of Gaussians, corresponding to the introduction of a discrete latent variable.

Before starting with the discussion concerning with the latent variable models, it is worth stressing that to model the probability densities from finite data sets in high dimensionality spaces is an extremely complex task which can be sketched by the following example: let $p(\mathbf{t})$ be a probability density function in D dimensions, which is function only of radius $r = \|\mathbf{t}\|$ and which has a Gaussian form

$$p(\mathbf{t}) = \frac{1}{(2\pi\sigma^{1/2})}exp\left(-\frac{\|\mathbf{t}\|^2}{2\sigma^2}\right). \tag{2.5}$$

The probability mass inside a thin shell of radius r and thickness ϵ is given (by expressing variables from Cartesian to polar coordinates) by $\rho(r)\epsilon$ where

$$\rho(r) = \frac{S_D r^{D-1}}{(2\pi\sigma^2)^{1/2}}exp\left(-\frac{r^2}{2\sigma^2}\right)$$

and S_D is the surface area of a unit sphere in D dimensions. Moreover, $\rho(r)$ has a single maximum which, for large values of D, is located at $\hat{r} \simeq \sqrt{D}\sigma$. Now, by considering

$\rho(\hat{r}+\epsilon)$, where $\epsilon << \hat{r}$, we have that for large D

$$\rho(\hat{r}+\epsilon) = \rho(\hat{r}) exp\left(-\frac{3\epsilon^2}{2\sigma^2}\right),$$

which means that $\rho(r)$ decays exponentially away from its maximum at $\hat{r}$ with length scale σ. Since $\sigma << \hat{r}$ at large D, we see that most of the probability mass is concentrated in a thin shell at large radius. By contrast, the value of the probability density itself is $exp(D/2)$ times bigger at the origin than at the radius $\hat{r}$, as can be seen by comparing $p(\mathbf{t})$ in (2.5) for $\|\mathbf{t}\| = 0$ with $p(\mathbf{t})$ for $\|\mathbf{t}\|^2 = \hat{r}^2 = \sigma^2 D$. Thus, the bulk of the probability mass is located in a different part of space from the region of high probability density. With finite data sets, there may be few, if any, data points associated with the region of high probability density near the origin, this is consequence of the well known *curse of dimensionality*[4].

2.1.1 Latent Variable Models

The goal of a latent variable model is to express the distribution $p(\mathbf{t})$ of the variable $\mathbf{t} = (t_1, \ldots, t_D)$ in terms of a smaller number of latent variable $\mathbf{x} = (x_1, \ldots, x_Q)$ where $Q < D$. This is achieved by first decomposing the joint distribution $p(\mathbf{t}, \mathbf{x})$ into the product of the marginal distribution $p(\mathbf{x})$ of the latent variables and the conditional distribution $p(\mathbf{t}|\mathbf{x})$ of the data variables given the latent variables. It is convenient to express the conditional distribution as a factorization over the data variables, so that the joint distribution becomes

$$p(\mathbf{t}, \mathbf{x}) = p(\mathbf{x})p(\mathbf{t}|\mathbf{x}) = p(\mathbf{x}) \prod_{d=1}^{D} p(t_d|\mathbf{x}). \tag{2.6}$$

Next the conditional distribution $p(\mathbf{t}|\mathbf{x})$ is expressed in terms of a mapping from latent variables to data variables, so that

$$\mathbf{t} = \mathbf{y}(\mathbf{x}; \mathbf{w}) + \mathbf{u} \tag{2.7}$$

where $\mathbf{y}(\mathbf{x}; \mathbf{w})$ is a function of the latent variable $\mathbf{x}$ with parameters $\mathbf{w}$, and $\mathbf{u}$ is an $\mathbf{x}$-independent noise process. If the components of $\mathbf{u}$ are uncorrelated, the conditional distribution for $\mathbf{t}$ will factorize as in (2.6). Geometrically the function $\mathbf{y}(\mathbf{x}; \mathbf{w})$ defines a manifold in data space given by the image of the latent space, as shown in figure 2.1.

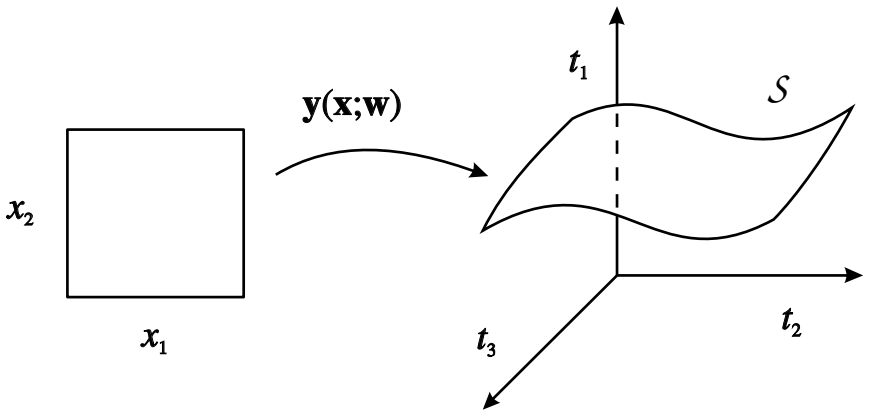

Figure 2.1: The non-linear function $\mathbf{y}(\mathbf{x}; \mathbf{W})$ defines a manifold S embedded in data space given by the image of the latent space under the mapping $\mathbf{x} \rightarrow \mathbf{y}$.

The definition of the latent variable model is completed by specifying the distribution $p(\mathbf{u})$, the mapping $\mathbf{y}(\mathbf{x}; \mathbf{w})$, and the marginal distribution $p(\mathbf{x})$. The type of the mapping $\mathbf{y}(\mathbf{x}; \mathbf{w})$ determines the particular latent variable model. The desired model for the distribution $p(\mathbf{t})$ of the data is obtained by marginalizing over the latent variables

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{2.8}$$

This integration will, in general, be analytically intractable except for specific forms of the distributions $p(\mathbf{t}|\mathbf{x})$ and $p(\mathbf{x})$.

2.1.2 Mixture Distributions

The density models we have considered so far are clearly very limited in terms of the variety of probability distributions which they can model since they can only represent distributions which are uni-modal. However, they can form the basis of a very general framework for density modelling, obtained by considering mixtures of M simpler parametric distributions. This leads to density models of the form

$$p(\mathbf{t}) = \sum_{m=1}^{M} \pi_m p(\mathbf{t}|m) \tag{2.9}$$

in which the $p(\mathbf{t}|m)$ represent the individual components of the mixture and might consist, for example, of normal distributions of the form (2.1) each one with its own independent mean μ_m and covariance matrix $\mathbf{\Sigma}_m$. The parameters π_m in (2.9) are called mixing coefficients and satisfy the requirements $0 \leq \pi_m \leq 1$ and $\sum_m \pi_m = 1$ so that $p(\mathbf{t})$ will be non-negative and will integrate to unity (assuming the individual component densities also have these properties). The mixing coefficients can be interpreted as prior probabilities for the values of the label m. For a given data point $\mathbf{t}_n$ we can then use Bayes' theorem to evaluate the corresponding posterior probabilities, given by

$$R_{nm} \equiv p(m|\mathbf{t}_n) = \frac{\pi_m p(\mathbf{t}_n|m)}{\sum_j \pi_j p(\mathbf{t}_n|j)}. \tag{2.10}$$

The value of $p(m|\mathbf{t}_n)$ can be regarded as the responsibility which component m takes for explaining data point $\mathbf{t}_n$. The log likelihood for the mixture distribution takes the form

$$\mathcal{L}(\{\pi_m, \mu_m, \mathbf{\Sigma}_m\}) = \sum_{n=1}^{N} \ln \left\{ \sum_{m=1}^{M} \pi_m p(\mathbf{t}_n|m) \right\}. \tag{2.11}$$

Maximization of this log likelihood is more complex then for a single component due to the presence of the sum inside the logarithm. An elegant and powerful technique for performing this optimization is the expectation-maximization (EM) algorithm [19]. The EM algorithm is based on the observation that, if we were given a set of indicator variables z_{nm} specifying which component m was responsible for generating each data point $\mathbf{t}_n$, then the log likelihood would take the form

$$\mathcal{L}_{comp}(\{\pi_m, \mu_m, \mathbf{\Sigma}_m\}) = \sum_{n=1}^{N} \sum_{m=1}^{M} z_{nm} \ln\{\pi_m p(\mathbf{t}_n|m)\} \tag{2.12}$$

and its optimization would be straightforward, with the result that each component is fitted independently to the corresponding group of data points, and the mixing coefficients are given by the fractions of points in each group.

The $\{z_{nm}\}$ are regarded as "missing data", and the data set $\{\mathbf{t}_n\}$ is said to be "incomplete". Combining $\{\mathbf{t}_n\}$ and $\{z_{nm}\}$ we obtain the corresponding "complete" data set, with a log likelihood given by (2.12). However, the values of $\{z_{nm}\}$ are unknown, but their posterior distribution can be computed using Bayes' theorem, and the expectation of z_{nm} under this distribution is just the set of responsibilities R_{nm} given by (2.10). The EM algorithm is based on the maximization of the expected complete-data log likelihood given from (2.12)

by

$$\langle \mathcal{L}_{comp}(\{\pi_m, \mu_m, \mathbf{\Sigma}_m\}) \rangle = \sum_{n=1}^{N} \sum_{m=1}^{M} R_{nm} \ln\{\pi_m p(\mathbf{t}_n|m)\}. \tag{2.13}$$

It alternates between the E-step, in which the R_{nm} are evaluated using (2.10), and the M-step in which (2.13) is maximized with respect to the model parameters to give a revised set of parameters values. At each cycle of the EM algorithm the true log likelihood is guaranteed to increase unless it is already at a local maximum.

The EM algorithm can also be applied to the problem of maximizing the likelihood for a single latent variable model of the kind discussed in section 2.1.1. The log likelihood for such a model takes the form

$$\mathcal{L}(\mathbf{W}, \mu, \mathbf{\Psi}) = \sum_{n=1}^{N} \ln p(\mathbf{t}_n) = \sum_{n=1}^{N} \ln \left\{ \int p(\mathbf{t}_n|\mathbf{x}_n) p(\mathbf{x}_n) d\mathbf{x}_n \right\}. \tag{2.14}$$

Again, this is difficult to treat because of the integral inside the logarithm. In this case the values of $\mathbf{x}_n$ are regarded as missing data. Given the prior distribution $p(\mathbf{x})$ we can consider the corresponding posterior distribution obtained through Bayes' theorem

$$p(\mathbf{x}_n|\mathbf{t}_n) = \frac{p(\mathbf{t}_n|\mathbf{x}_n) p(\mathbf{x}_n)}{p(\mathbf{t}_n)} \tag{2.15}$$

and the sufficient statistics for this distribution are evaluated in the E-step. The M-step involves maximization of the expected complete-data log likelihood and is generally much simpler than the direct maximization of the true log likelihood.

In the next sections we shall see how the concepts of latent variables and mixture distributions can be used in a fruitful partnership to obtain a range of powerful algorithms for density modelling, pattern classification and data visualization.

2.2 Non-linear Latent Variable Models

2.2.1 Generative Topographic Mapping

The *GTM* defines a non-linear, parametric mapping $\mathbf{y}(\mathbf{x}; \mathbf{W})$ from a Q-dimensional latent space ($\mathbf{x} \in R^Q$) to a D-dimensional data space ($\mathbf{t} \in R^D$), where normally $Q < D$. The mapping is defined to be continuous and differentiable. $\mathbf{y}(\mathbf{x}; \mathbf{W})$ maps every point in the latent space to a point into the data space. Since the latent space is Q-dimensional, these points will be confined to a Q-dimensional manifold non-linearly embedded into

the D-dimensional data space. If we define a probability distribution over the latent space, $p(\mathbf{x})$, this will induce a corresponding probability distribution into the data space. Strictly confined to the Q-dimensional manifold, this distribution would be singular, so it is convolved with an isotropic Gaussian noise distribution, given by

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) = \left(\frac{\beta}{2\pi}\right)^{\frac{D}{2}} exp\left\{-\frac{\beta}{2}\sum_{d=1}^{D}(t_d - y_d(\mathbf{x}, \mathbf{W}))^2\right\} \tag{2.16}$$

where $\mathbf{t}$ is a point in the data space and β^{-1} denotes the noise variance.
By integrating out the latent variable, we get the probability distribution in the data space expressed as a function of the parameters β and $\mathbf{W}$,

$$p(\mathbf{t}|\mathbf{W}, \beta) = \int p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta)p(\mathbf{x})d\mathbf{x}. \tag{2.17}$$

This integral is generally not analytically tractable. However, by choosing $p(\mathbf{x})$ to have a particular form, a set of M equally weighted delta functions on a regular grid,

$$p(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\delta(\mathbf{x} - \mathbf{x}_m), \tag{2.18}$$

the integral in (2.17) turns into a sum,

$$p(\mathbf{t}|\mathbf{W}, \beta) = \frac{1}{M}\sum_{m=1}^{M}p(\mathbf{t}|\mathbf{x}_m, \mathbf{W}, \beta). \tag{2.19}$$

Now we have a model where each delta function center (from now on we shall refer to these as latent points) maps into the center of a Gaussian which lies in the manifold embedded in the data space, as illustrated in figure 2.2.

Note that, provided the mapping function $\mathbf{y}(\mathbf{x}; \mathbf{w})$ is smooth and continuous, the projected points $\mathbf{y}(\mathbf{x}_m; \mathbf{w})$ will necessarily have a topographic ordering in the sense that any two points $\mathbf{x}_A$ and $\mathbf{x}_B$ which are close in latent space will map to points $\mathbf{y}(\mathbf{x}_A; \mathbf{w})$ and $\mathbf{y}(\mathbf{x}_B; \mathbf{w})$ which are close in data space. What we have is a constrained mixture of Gaussians, since the centers of the mixture components can not move independently of each other, but all depend on the mapping $\mathbf{y}(\mathbf{x}; \mathbf{W})$ (see figure (2.3)). Moreover, all components of the mixture share the same variance, and the mixing coefficients are all fixed to $\frac{1}{M}$.
Given a finite set of independent and identically distributed (i.i.d.) data points, $\{\mathbf{t}_n\}_{n=1}^{N}$, we can write down the likelihood function for this model,

$$\mathcal{L} = \prod_{n=1}^{N}p(\mathbf{t}_n|\mathbf{W}, \beta) = \prod_{n=1}^{N}\left[\frac{1}{M}\sum_{m=1}^{M}p(\mathbf{t}_n|\mathbf{x}_m, \mathbf{W}, \beta)\right], \tag{2.20}$$

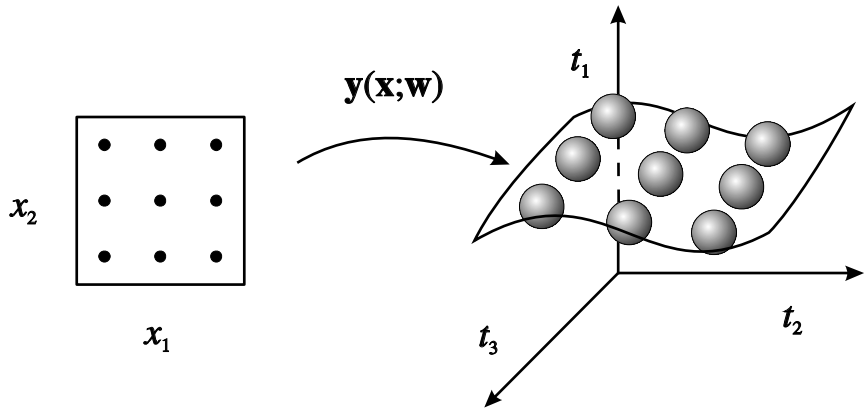

Figure 2.2: In order to formulate a tractable non linear latent variable model, we consider a prior distribution $p(\mathbf{x})$ consisting of a superposition of delta functions, located at the nodes of a regular grid in latent space. Each node $\mathbf{x}_m$ is mapped to a corresponding point $\mathbf{y}(\mathbf{x}_m; \mathbf{w})$ in data space, and forms the center of a corresponding Gaussian distribution.

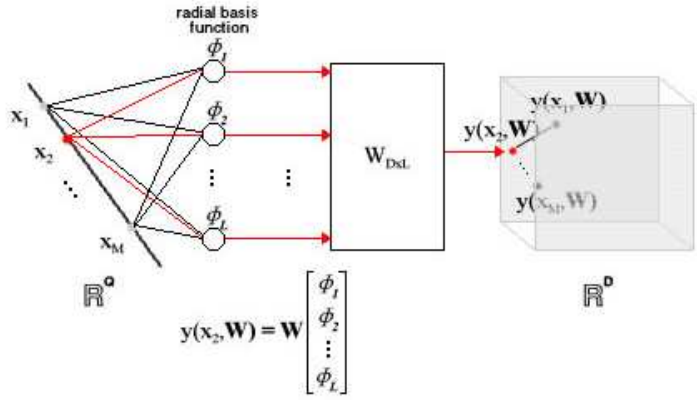

Figure 2.3: A *GTM* example with $D = 3$, $Q = 1$, $L = 4$ and $W_{3\times4}$. An RBF network with 4 hidden units maps input latent node x_m to the corresponding output node $\mathbf{y}(x_m; \mathbf{W}) = \mathbf{W}\mathbf{\Phi}(x_m)$.

and maximize it with respect to $\mathbf{W}$ and β. However, is normally more convenient to work with the log likelihood function,

$$\ell = \sum_{n=1}^{N} \ln \left(\frac{1}{M} \sum_{m=1}^{M} p(\mathbf{t}_n | \mathbf{x}_m, \mathbf{W}, \beta) \right). \tag{2.21}$$

Since *GTM* is a form of mixture model it is natural to seek an EM algorithm for maximizing the corresponding log likelihood. By choosing a particular form for the mapping $\mathbf{y}(\mathbf{x}; \mathbf{w})$ we can obtain an EM algorithm in which the M-step has a simple form. In particular we choose $\mathbf{y}(\mathbf{x}; \mathbf{w})$ to be given by a generalized linear regression model of the form

$$\mathbf{y}(\mathbf{x}; \mathbf{w}) = \mathbf{W}\phi(\mathbf{x}) \tag{2.22}$$

where the elements of $\phi(\mathbf{x})$ consist of L fixed basis functions $\{\phi_l(\mathbf{x})\}_{l=1}^{L}$, and $\mathbf{W}$ is a $D \times L$ matrix. Generalized linear regression models possess the same universal approximation capabilities as multi-layer adaptive networks, provided the basis functions are chosen appropriately. The usual limitation of such models, however, is that the number of basis functions must typically grow exponentially with the dimensionality Q of the latent space. In the present context this is not a significant problem since the dimensionality is governed by the number of latent variables which will typically be small. In fact for data visualization applications we generally use $Q = 2$.

GTM for visualization

An important potential application for the *GTM* is visualization. To see how this works, note that a *GTM*, for which we have found suitable parameter values $\mathbf{W}^*$ and β^*, by (2.16) and (2.18) defines a probability distribution in the data space conditioned on the latent variable, $p(\mathbf{t}|\mathbf{x}_m), m = 1, \ldots, M$. We can, therefore, use Bayes' theorem, in conjunction with the prior distribution over latent variable, $p(\mathbf{x})$, given in (2.18), to compute the corresponding posterior distribution in latent space for any given point in data space, $\mathbf{t}$, as

$$p(\mathbf{x}_m|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{x}_m, \mathbf{W}^*, \beta^*)p(\mathbf{x}_m)}{\sum_{m\prime=1}^{M} p(\mathbf{t}_n|\mathbf{x}_{m\prime}, \mathbf{W}^*, \beta^*)p(\mathbf{x}_{m\prime})}. \tag{2.23}$$

Provided that the latent space has no more than two, or possibly three, dimensions, $p(\mathbf{x}_m|\mathbf{t})$ against $\mathbf{x}_m$ can be plotted. However, in order to visualize whole sets of data, less rich descriptions must be used. Two possibilities are, for each data point $\mathbf{t}_n$, to plot

- the mode of the posterior distribution in latent space,

$$\mathbf{x}_n^{mode} = argmax_{\mathbf{x}_m} p(\mathbf{x}_m|\mathbf{t}_n),$$

  which is called posterior-mode projection;

- the mean of the posterior distribution in latent space,

$$\mathbf{x}_n^{mean} = \sum_{m=1}^{M} \mathbf{x}_m p(\mathbf{x}_m|\mathbf{t}_n)$$

called posterior-mean projection.
One of the motivations for the development of the *GTM* algorithm was to provide a principled alternative to the Self Organizing Maps (*SOM*) algorithm [43, 44]. In fact, while the *SOM* has achieved many success in practical applications, it also suffers from some significant deficiencies: the absence of a cost function, the lack of any guarantee of topographic ordering, the absence of any general proofs of convergence, and the fact that the model does not define a probability density. These problems are all absent in *GTM* [9, 61].

Computational complexity

When updating the parameters, the *GTM* requires the inversion of a $L \times L$ matrix, where L is the number of basis functions. This computation requires $\mathcal{O}(L^3)$ operations. Furthermore some matrix multiplications are involved and these require $\mathcal{O}(MND)$ operations. Note that the computation of the probabilities $p(\mathbf{t}|\mathbf{x})$ requires (assuming small Q) $\mathcal{O}(D)$ operations. This last consideration will be useful when comparing the generative Topographic Mapping and Probabilistic Principal Surfaces models.

2.2.2 Probabilistic Principal Surfaces

Probabilistic Principal Surfaces (*PPS*) were proposed in [16, 17, 18] as a unified probabilistic model for feature extraction to approximate principal surfaces in order to address a number of issues [18] associated with principal surfaces algorithms [39, 46, 64]. The *PPS* share the same formulation as the *GTM*, except for an oriented covariance structure for nodes in R^D. This means that data points projecting near a principal surface node have

higher influences on that node than points projecting far away from it. This is illustrated in figure (2.4).

Therefore, each node $\mathbf{y}(\mathbf{x};\mathbf{w})$, $\mathbf{x} \in \{\mathbf{x}_m\}_{m=1}^{M}$, has covariance

$$\mathbf{\Sigma}(\mathbf{x}) = \frac{\alpha}{\beta}\sum_{q=1}^{Q}\mathbf{e}_q(\mathbf{x})\mathbf{e}_q^T(\mathbf{x}) + \frac{(D-\alpha Q)}{\beta(D-Q)}\sum_{d=Q+1}^{D}\mathbf{e}_d(\mathbf{x})\mathbf{e}_d^T(\mathbf{x}), \quad 0 < \alpha < \frac{D}{Q} \tag{2.24}$$

where

- $\{\mathbf{e}_q(\mathbf{x})\}_{q=1}^{Q}$ is the set of orthonormal vectors tangential to the manifold at $\mathbf{y}(\mathbf{x};\mathbf{w})$,
- $\{\mathbf{e}_d(\mathbf{x})\}_{d=Q+1}^{D}$ is the set of orthonormal vectors orthogonal to the manifold in $\mathbf{y}(\mathbf{x};\mathbf{w})$.

The complete set of orthonormal vectors $\{\mathbf{e}_d(\mathbf{x})\}_{d=1}^{D}$ spans R^D. The unified *PPS* model reduces to *GTM* for $\alpha = 1$ and to the manifold-aligned *GTM* [7] for $\alpha > 1$

$$\mathbf{\Sigma}(\mathbf{x}) = \begin{cases} 0 < \alpha < 1 & \perp \text{ to the manifold} \\ \alpha = 1 & I_D \text{ or spherical} \\ 1 < \alpha < D/Q & \parallel \text{ to the manifold} \end{cases}$$

As $\alpha \to 0$, the support of each node becomes increasingly concentrated along the orthogonal hyperplane at each node. Figure 2.5 shows the unit Mahalanobis distance loci of $\mathbf{\Sigma}(\mathbf{x})$ for various values of α.

Estimation of the *PPS* Parameters

The EM algorithm can be used to estimate the *PPS* parameters. First, the complete log likelihood for the *PPS*, assuming equal and constant prior probabilities $P(\mathbf{x}_m) = 1/M, m = 1, \ldots, M$, is written as

$$\mathcal{L}_{comp} = \sum_{n=1}^{N}\sum_{m=1}^{M} z_{mn} \ln\left[p(\mathbf{t}_n|\mathbf{x}_m)\frac{1}{M}\right], \tag{2.25}$$

where the binary variable z_{mn} indicates whether component m is responsible for generating sample point $\mathbf{t}_n$. Since z_{mn} is unknown or "missing", the complete log likelihood (2.25) cannot be evaluated. Therefore in the E-step of the EM algorithm, the expected value of $\mathcal{L}_{comp}$ with respect to $P(z|\mathbf{t})$ is evaluated instead at the k-th iteration, leading to the following expression

$$\mathcal{L} = \langle \mathcal{L}comp \rangle = \sum_{n=1}^{N}\sum_{m=1}^{M} r_{mn}^{(k)} \ln\left[p(\mathbf{t}_n|\mathbf{x}_m)\frac{1}{M}\right], \tag{2.26}$$

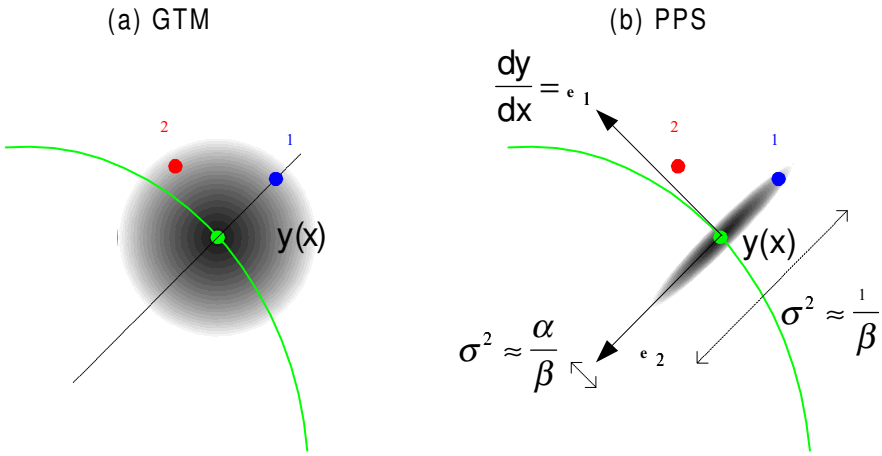

Figure 2.4: Under a spherical Gaussian model of the *GTM*, points 1 and 2 have equal influences on the center node $y(\mathbf{x})$ (a) *PPS* have an oriented covariance matrix so point 1 is probabilistically closer to the center node $y(\mathbf{x})$ than point 2 (b).

where the responsibility parameter

$$r_{mn}^{(k)} = p(\mathbf{x}_m|\mathbf{t}_n) = \frac{p(\mathbf{t}_n|\mathbf{x}_m)P(\mathbf{x}_m)}{\sum_{m\prime=1}^{M} p(\mathbf{t}_n|\mathbf{x}_{m\prime})P(\mathbf{x}_{m\prime})} = \frac{p(\mathbf{t}_n|\mathbf{x}_m)}{\sum_{m\prime=1}^{M} p(\mathbf{t}_n|\mathbf{x}_{m\prime})}, \tag{2.27}$$

is computed by substituting the "old" (k) parameter values $\mathbf{W}^{(k)}, \beta^{(k)}, \alpha^{(k)}$ into the conditional probabilities $p(\mathbf{t}_n|\mathbf{x}_m)$. In the M-step, the expected log likelihood function (2.26) is maximized with respect to $\mathbf{W}, \beta$ and α, thereby giving the corresponding new $(k+1)$-th values. However, for simplicity, the clamping factor α is assumed to be constant and the approximation of the M-step is accomplished through the original *GTM* equations. A simply description of the algorithm follows below:

Initialization Assuming that the latent nodes $\{\mathbf{x}_m\}_{m=1}^{M}$are arranged in a uniform topology within a hypercube in R^Q,
initialize $\mathbf{W}^{(0)}$ to the solution of the following least square problem,

$$[\mathbf{y}(\mathbf{x}_1)\ldots\mathbf{y}(\mathbf{x}_m)] = \mathbf{W}[\mathbf{\Phi}(\mathbf{x}_1)\ldots\mathbf{\Phi}(\mathbf{x}_M)], \tag{2.28}$$

where $\{\mathbf{y}(\mathbf{x}_m)\}_{m=1}^{M}$ are the set of nodes on a hyper-grid in R^D spanned by the Q principal components $\{\mathbf{e}_q\}_{q=1}^{Q}$.

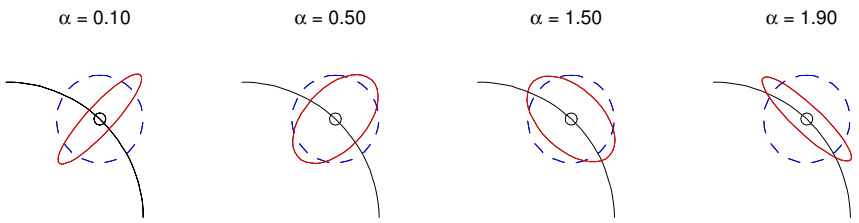

Figure 2.5: Un-oriented covariance $\alpha = 1$ (dashed line) and oriented covariances (solid line) for $\alpha = 0.10, 0.50, 1.50, 1.90$. The valid range for α is $0 < \alpha < 2$ for $D = 2, Q = 1$ in this example.

With $\mathbf{W}^{(0)}$ initialized, an initial distribution of the *PPS* nodes in R^D can be computed as $\mathbf{y}^{(0)}(\mathbf{x}_m) = \mathbf{W}^{(0)}\mathbf{\Phi}(\mathbf{x}_m)$, $m = 1, \ldots, M$. Initialize $1/\beta^{(0)}$ to the median of the squared Euclidean distances between adjacent nodes $\{\mathbf{y}^{(0)}(\mathbf{x}_m)\}_{m=1}^{M}$ in the data space.

At the ***k*-th iteration**:

1. Expectation (E-step): compute the responsibility matrix $\mathbf{R}_{M\times N}^{(k)}$ whose entries are given by (2.27).

2. Maximization (M-step): compute updated parameters $\mathbf{W}^{(k+1)}$ and $\beta^{(k+1)}$

   (a) Compute $\mathbf{W}^{(k+1)}$ as the solution to the following linear matrix equation, $(\mathbf{\Phi}^T\mathbf{G}^{(k)}\mathbf{\Phi})\mathbf{W}^T = \mathbf{\Phi}^T\mathbf{R}^{(k)}\mathbf{T}$, where

$$\begin{aligned} \mathbf{\Phi}_{L\times M} &: \quad \Phi_{lm} = \Phi_l(\mathbf{x}_m), \\ \mathbf{G}_{M\times M}^{(k)} &: \quad g_{mm}^{(k)} = \sum_{n=1}^{N} r_{mn}^{(k)}, \\ \mathbf{T}_{N\times D} &= \quad [\mathbf{t}_1 \ldots \mathbf{t}_N]^T, \end{aligned}$$

   (b) Compute β^{k+1} as,

$$\frac{1}{\beta^{k+1}} = \frac{1}{ND}\sum_{n=1}^{N}\sum_{m=1}^{M} r_{mn}^{(k)}\|\mathbf{W}^{(k+1)}\mathbf{\Phi}(\mathbf{x}_m) - \mathbf{t}_n\|^2.$$

(c) Compute the new *PPS* nodes in R^D as,

$$\mathbf{y}^{(k+1)}(\mathbf{x}_m) = \mathbf{W}^{(k+1)}\mathbf{\Phi}(\mathbf{x}_m), \quad m = 1, \ldots, M.$$

3. Evaluate the change in log likelihood,

$$\Delta\mathcal{L} = \left\| \frac{\mathcal{L}^{(k+1)} - \mathcal{L}^{(k)}}{\mathcal{L}^{(k)}} \right\|.$$

4. Terminate if $\Delta\mathcal{L}$ falls below some threshold ϵ, otherwise increment counter k and go to step 1.

Computational Complexity

The *PPS* incurs in two additional computations over the *GTM*: (1) computation of the $D \times Q$ tangential matrix $\mathbf{E}_{\|}(\mathbf{x})$, which is obtained by concatenating the tangential manifold vectors $\{\mathbf{e}_q(\mathbf{x})\}_{q=1}^{Q}$ and (2) evaluation of the full Gaussian class-conditional probabilities $p(\mathbf{t}|\mathbf{x}_m)$. The set of Q tangential vectors $\{\mathbf{e}_q(\mathbf{x})\}_{q=1}^{Q}$ can be estimated from the partial derivative of the latent basis activations at $\mathbf{x}$:

$$\mathbf{e}_q^{'}(\mathbf{x}) = \mathbf{W}\frac{\vartheta\phi(\mathbf{x})}{\vartheta x_q},$$

where the constant latent basis derivative $\frac{\vartheta\phi(\mathbf{x})}{\vartheta x_q}$ need to be evaluated only once. Furthermore, since neither the row space of $\mathbf{W}$ nor the set $\{\frac{\vartheta\phi(\mathbf{x})}{\vartheta x_q}\}_{q=1}^{Q}$ is orthogonal in general, the resulting $\{\mathbf{e}^{'}(\mathbf{x})\}_{q=1}^{Q}$ will not be orthonormal, and thus must be orthonormalized via the Gram-Schmidt procedure in order to satisfy the conditions (2.24). The matrix $\mathbf{E}_{\|}(\mathbf{x})$ is updated once per EM training epoch, which requires $\mathcal{O}(LQD)$ operations for the matrix multiplication and $\mathcal{O}(Q^2D)$ operation for the orthonormalization. It is worth noting that it is not necessary to compute the Gram-Schmidt procedure for the set of orthogonal manifold vectors $\{\mathbf{e}_d(\mathbf{x})\}_{d=Q+1}^{D}$ in (2.24) since a proposition in [17] shows the (2.24) can be expressed in terms of the tangential manifold vectors only. Definitely, evaluation of the conditional probabilities $p(\mathbf{t}|\mathbf{x})$ requires $\mathcal{O}(QD^2)$ operations with respect to $\mathcal{O}(D)$ complexity of the *GTM*, however for complex mappings this overhead is attenuated by the fact that the *PPS* converge faster than *GTM*.

2.2.3 Spherical *PPS*

If 1 or 2-dimensional latent spaces are considered (i.e, $Q = 1$ or $Q = 2$) then the corresponding manifold will be a curve or a nonlinear plane. In [17] it is shown that for characterizing high-D data, a spherical manifold (in this case $Q = 3$) is a more appropriate tool. Spherical *PPS* are very effective for data visualization purposes (we shall address this issue in chapter 4) and classification tasks. Basically, randomly distributed data in high-D space tend to be sparse and concentrated at the periphery. This is a consequence of the *curse of dimensionality* [4, 17] which causes the number of samples in a training set is always sparse with respect to the dimensionality, and it is a major cause of error in function approximation, density estimation, and classification [34].

A spherical manifold (see figure 2.6 [17]) can be constructed using a *PPS* with nodes $\{\mathbf{x}_m\}_{m=1}^{M}$ arranged regularly on the surface of a sphere in R^3 latent space, with the latent basis functions evenly distributed on the sphere at a lower density. The only modification required with respect to $1-D$ and $2-D$ manifolds is the initialization procedure, which initialize the manifold to a hyper-ellipsoid in R^D defined by the 3 largest eigenvectors of the data. This is achieved by solving for $\mathbf{W}$ in the following least squares equation,

$$[\mathbf{s}_1 \dots \mathbf{s}_M] = \mathbf{W}[\mathbf{\Phi}(\mathbf{x}_1) \dots \mathbf{\Phi}(\mathbf{x}_M)],$$

where

$$\mathbf{s_m} = [\sqrt{\xi_1}\mathbf{e_1}\ \sqrt{\xi_2}\mathbf{e}_2\ \sqrt{\xi_3}\mathbf{e}_3]\mathbf{x}_m, \quad m = 1, \dots, M,$$

are the coordinates of the hyper-ellipsoid in data space with $\{\mathbf{e}_q\}_{q=1}^{3}$ denoting the three largest eigenvectors (scaled by the corresponding eigenvalues $\{\xi_q\}_{q=1}^{3}$) of the data covariance matrix. After initialization, the standard *PPS* iteration procedure described in the previous section follows.

Spherical *PPS* for data visualization

The spherical manifold can be used as an unsupervised high-D data visualization tool. To visualize the data, a spherical manifold is first fitted to the data, effectively capturing its structure. Next, the data is projected onto the manifold in R^D, and the projected locations along with the manifold are plotted in R^3 as points on a sphere.

The method adopted for projecting data onto a spherical manifold for visualization is the

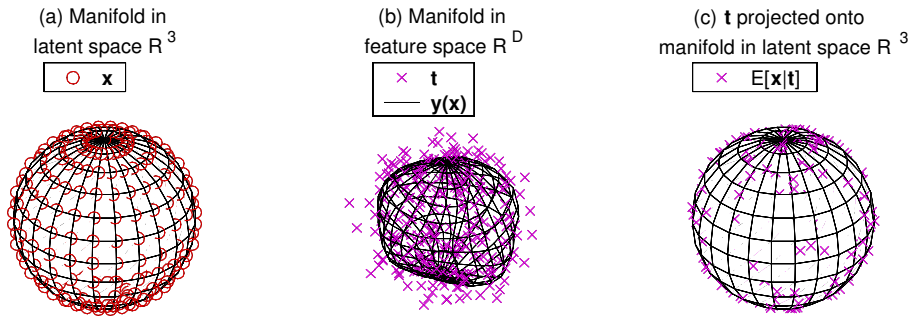

Figure 2.6: (a) The spherical manifold in R^3 latent space. (b) The spherical manifold in R^3 data space. (c) Projection of data points $\mathbf{t}$ onto the latent spherical manifold.

probabilistic projection. The probabilistic projection computes the latent manifold coordinates $\hat{\mathbf{x}}_n$ of each data point $\mathbf{t}_n$ as the mean of the induced probability density function in R^3. In practice, the projected latent coordinate is computed as a linear combination of all latent nodes weighted by the responsibility matrix (2.27),

$$\hat{\mathbf{x}}_n \equiv \langle \mathbf{x}|\mathbf{t}_n \rangle = \int \mathbf{x} p(\mathbf{x}|\mathbf{t}) d\mathbf{x} = \sum_{m=1}^{M} r_{mn}\mathbf{x}_m. \tag{2.29}$$

For a spherical manifold, $\|\mathbf{x}_m\| = 1$ for $m = 1, \ldots, M$ and $\sum_m r_{mn} = 1$ for $n = 1, \ldots, N$, therefore, expression (2.29) implies that all projections lie within the sphere, i.e. $\|\mathbf{x}_m\| \leq 1$.

Spherical *PPS* for Classification

The spherical *PPS* is used as a "reference manifold" for classifying high-D data. A reference spherical manifold is computed for each class during the training phase. In test phase, an unseen data is classified to the class of its nearest spherical manifold. Obviously, the concept of "nearest" implies a distance computation between a data point $\mathbf{t}$ and nodes onto the manifold. Before doing this computation the data point $\mathbf{t}$ must be linear projected onto the manifold. Since a spherical manifold consists of triangular and square patches each defined by three or four manifold nodes, what really happens here is an approximation of the distance. *PPS* framework provides three approximation methods:

- Nearest Neighbor (NN): finds the minimal square distance to all manifold nodes.

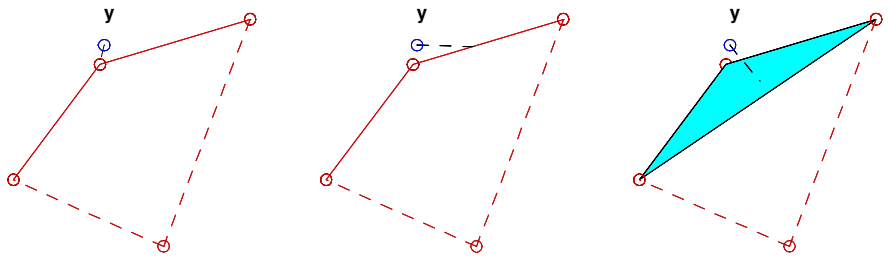

Figure 2.7: From left to right: *NN*, *GP* and *NT* projection approximations on a four node manifold patch.

- Grid projections (GP): finds the shortest projection distance to a manifold grid.
- Nearest triangulation (NT): finds the nearest projection distance to the possible triangulations.

It is worth noting that in the probabilistic framework the distance between a data point $\mathbf{t}$ and the function of the mean of its induced distribution $y(\langle \mathbf{x}|\mathbf{t} \rangle)$ on the manifold is computed. Clearly, the distance may not be the shortest in the Euclidean sense, so the distance using linear projection onto the manifold is computed. Figure 2.7 shows the three methods just described. Another way for employing *PPS* as classifiers consists in choosing the class C with the maximum posterior class probability for a given new input $\mathbf{t}$. Formally, suppose we have N labelled data points $\{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$, with $\mathbf{t}_i \in R^D$, $i = 1, \ldots, N$ and labels *class* in the set $\{1, \ldots, C\}$. The posterior probabilities may be derived from the class-conditional density $p(\mathbf{t}|class)$ via Bayes theorem:

$$P(class|\mathbf{t}) = \frac{p(\mathbf{t}|class)P(class)}{p(\mathbf{t})} \propto p(\mathbf{t}|class)P(class).$$

In order to approximate the posterior probabilities $P(class|\mathbf{t})$ we estimate $p(\mathbf{t}|class)$ and $P(class)$ from the training data. Finally, an input $\mathbf{t}$ is assigned to the class with maximum $P(class|\mathbf{t})$.

In [17] it is shown that spherical *PPS* classifier reaches better performance then k-nearest neighbor classifier and Gaussian Mixture Models classifier on several benchmark data sets.

2.2.4 Experimental results

In this section we examine three classification tasks concerning three different astronomical data sets: (1) *Star/Galaxy* catalog, (2) *GOODS* catalog, and finally (3) Telescopio Nazionale Galileo (*TNG*) telemetry data. The first is a synthetic catalog while the remaining two contain real-world data. These data sets will be used in all the experiments described in this thesis, therefore they are detailed in appendix 6.1.
All the experiments are implemented under the *Matlab* computing environment exploiting the *LANS Pattern Recognition Matlab Toolbox*[1] and *Netlab* Toolbox [51].
Classification is accomplished by using the *PPS* models to

1. compute the reference manifolds for each class (we denote this classifier as *PPSRM*),

2. compute the posterior class probability (hereinafter denoted as *PPSPR*).

In all the experiments, the classifiers are run 25 times for each of which new training and test data partitions (60% for training and 40% for testing, except when differently stated) are generated. Obviously, a good parameters setting is a key point for the overall system performance achievable through various experiments, but at the same time this is not an easy task. Most of the values of the parameters are very problem-dependent and must be determined in an empirical fashion across trials. For this aim, for each training/test partitions, ten *PPS* models are fitted to the data in order to evaluate the best clamping factor α value, where α ranges in the set $\{0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$. Furthermore, fundamental are the number of latent variables or nodes, which determines the manifold resolution and the number of basis functions which control the manifold complexity. Their settings are fixed on the basis of the size and complexity of each data set at hand. Each run is allowed a maximum of 100 epochs with early stopping triggered whenever the change in log-likelihood goes below the fixed threshold. Finally, the only preprocessing made on the data is a normalization to zero mean and unit variance (the whitening was tried as well but with less accurate results).

[1] http://www.lans.ece.utexas.edu/~lans/lans/

Synthetic catalog

The catalog contains 20000 objects equally divided into two classes composed by 10000 stars and galaxies, respectively. Each object is described by eight features or parameters, namely the magnitudes in the corresponding eight optical filters. In general *star-galaxy* classification is a complex task for astronomers and for a catalog of this nature the usually adopted methodologies lead to a classification error about 10%[2]. Parameter setting (the most meaningful) shared by the ten *PPS* models is shown in table 2.1. Figure 2.8 depicts the error bars deriving from 10 *PPS* models used as *reference manifold* classifiers. For each of the 10 fixed values of α, the error bars are computed over 25 iterations of the *PPS* learning algorithm. As can be seen even from table 2.2, where the mean classification error and standard deviation are reported, the best results, in terms of the mean classification error, are obtained with values of α ranging between 0.6 and 1.4, with best models α corresponding to the extremes of this interval. For $\alpha = 0.6$, however, the standard deviation is quite high, whereas for $\alpha = 1.4$ we have the minimum standard deviation as well. Figure 2.9 and table 2.3 show error bars and mean-standard deviation values for *PPS* used to compute a posteriori class probabilities. In this case a different behavior is observed in which the best results are obtained for increasing values of α with the overall best model α fixed to 2.0. Even though a more stable behavior (minimum standard deviation) is obtained for $\alpha = 1.2$. Table 2.4 shows confusion matrices corresponding to both *PPSRM* and *PPSPR* with minimum classification errors. Therefore, by observing the results obtained we can state that *PPSRM* is a more stable classifier whereas *PPSPR* are able to gain the lower peaks in term of classification errors. This instability could be explained for the effect of *overtraining* which is a plague of mixture models, and hence of probabilistic principal surfaces, which converge in different local minima. In fact, it is well known that maximizing the likelihood can lead to over-fitting which is particularly severe in density estimation due to singularities in the log-likelihood function.

[2]personal communication of the synthetic catalog author

Parameter	*Value*	*Description*
M	266	number of latent variables
L	83	number of basis functions
L_{fac}	1	basis functions width
$class$	NT	projection method for classification
$iter$	100	maximum number of iteration
ϵ	0.01	early stopping threshold

Table 2.1: *Synthetic* Catalog: parameter setting for *PPSRM* and *PPSPR*.

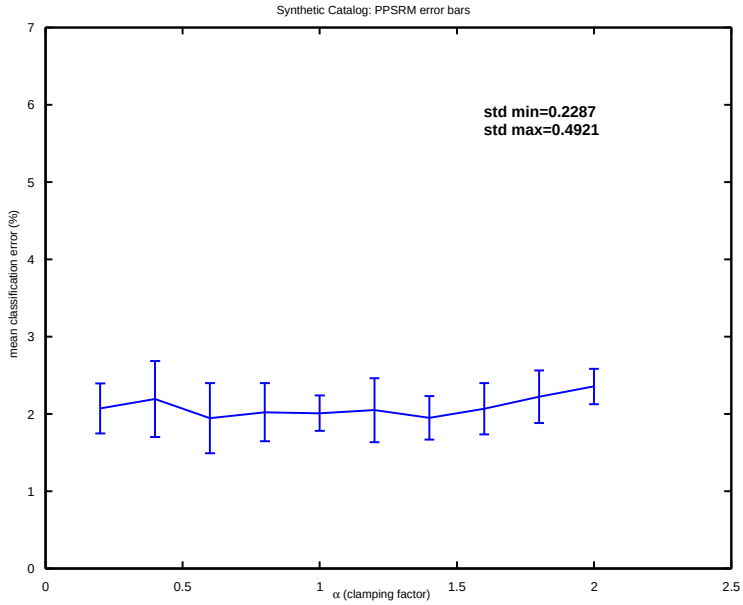

Figure 2.8: *Synthetic* Catalog: error bars for *PPSRM* (errors averaged over 25 iterations for fixed α).

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	2.07	0.3222
0.4	2.19	0.4921
0.6	**1.94**	**0.4535**
0.8	2.02	0.375
1.0	2.00	0.2290
1.2	2.04	0.4133
1.4	**1.94**	**0.2818**
1.6	2.06	0.3320
1.8	2.22	0.3400
<u>2.0</u>	2.35	<u>0.2287</u>

Table 2.2: *Synthetic* Catalog: mean classification error (%) for *PPSRM* (errors averaged over 25 iterations for fixed α). In bold are presented the lower mean classification errors. The lower standard deviation is underlined.

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	5.37	0.7523
0.4	4.35	0.5722
0.6	3.60	0.3228
0.8	3.22	0.4988
1.0	1.88	0.3188
1.2	2.13	0.2547
1.4	1.98	0.4521
1.6	1.34	0.5240
1.8	1.25	0.4443
2.0	**1.10**	**0.4669**

Table 2.3: *Synthetic* Catalog: mean classification error (%) for *PPSPR* (errors averaged over 25 iterations for fixed α). In bold is presented the lower mean classification error. The lower standard deviation is underlined.

Classifier type- Error (%)	*Confusion Matrix*			*Best model* α
$PPSRM - 1.34$		*Star*	*Galaxy*	1.4
	Star	3920	28	
	Galaxy	80	3972	
$PPSPR - 0.4$		*Star*	*Galaxy*	2.0
	Star	3976	8	
	Galaxy	24	3992	

Table 2.4: *Synthetic* Catalog: confusion matrices computed by *PPSRM* and *PPSPR* best models.

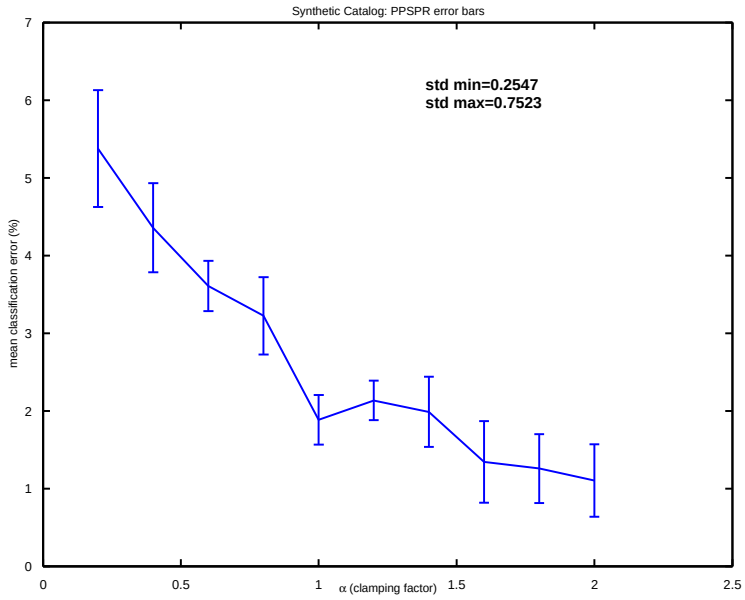

Figure 2.9: *Synthetic* Catalog: errors bars for *PPSPR* (errors averaged over 25 iterations for fixed α).

GOODS catalog

GOODS catalog is a star-galaxy catalog composed by 28405 objects. Each object is detected in 7 optical bands, namely *U,B,V,R,I,J,K* bands. For each band 3 different parameters (i.e., *Kron* radius, *Flux* and *Magnitudes*) are considered summing to a total number of 21 parameters. The catalog contains about 27000 galaxies and about 1400 stars. Moreover, there is a further peculiarity in the data contained in the catalog: the majority of the objects are "drop outs" which are objects not detectable in a given optical band. Among this type of objects there are groups which are not detectable in only one band, two bands, three bands and so on. In order to define the classification problem, we decided to split the data in four classes, namely star, galaxy, star which are drop outs and galaxy which are drop outs (we do not care about the number of bands for which an object is a drop out) and indicated these classes by *Star*, *Galaxy*, *StarD* and *GalaxyD*. The characteristics of the data make the classification task very difficult because the class of "dropped" galaxies dominates over the remaining classes as long as it contains about the 90% of the objects. Therefore for any classifier, a problem of this nature tends to

Parameter	*Value*	*Description*
M	266	number of latent variables
L	103	number of basis functions
L_{fac}	1	basis functions width
$class$	NT	projection method for classification
$iter$	100	maximum number of iteration
ϵ	0.001	early stopping threshold

Table 2.5: *GOODS* Catalog: parameter setting for *PPSRM* and *PPSPR*.

recognize near all the objects as dropped galaxy.

We can now give a look at the results. First of all, we used a *PPS* model a little bit more complex as it can be seen in table 2.5 (a greater number L of basis functions). Figures 2.10, 2.11 and tables 2.6, 2.7 show the error bars for *PPSRM* and *PPSPR*, and mean-standard deviation classification errors for *PPSRM* and *PPSPR*, respectively. Here the differences between *PPSRM* and *PPSPR* classifiers become wider in terms of mean classification errors. In fact, while *PPSRM* reaches a mean classification error ranging between a minimum of 7.48% and a maximum of 9.81%, *PPSPR* reaches its minimum at 2.90% and the maximum at 5.84%. In front of these results, the stability of *PPSRM* (standard deviations between 0.2613 and 0.8378 for *PPSRM* and between 0.1893 and 2.061 for *PPSPR*) assumes less importance with respect to *PPSPR*. Furthermore for $\alpha = 1.8$ *PPSPR* has the best overall mean classification error and standard deviation. In table 2.8 are listed the confusion matrices relative to the best models minimum classification error. Even though these values are different for *PPSRM* and *PPSPR*, respectively, it is interesting to note that both classifiers have the majority of misclassification between the same classes (*Star* as *Galaxy* and viceversa, *StarD* as *GalaxyD* and viceversa).

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	**7.48**	**0.5536**
0.4	8.45	0.6020
0.6	7.83	0.5773
0.8	8.60	0.3933
1.0	8.34	0.5703
1.2	9.55	0.2613
1.4	9.42	0.6117
1.6	9.09	0.3896
1.8	9.18	0.4548
2.0	9.81	0.8378

Table 2.6: *GOODS* Catalog: mean classification error (%) for *PPSRM* (errors averaged over 25 iterations for fixed α). In bold is presented the lower mean classification error. The lower standard deviation is underlined.

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	5.23	0.8695
0.4	5.84	1.6258
0.6	4.37	0.6221
0.8	4.56	1.5156
1.0	4.99	0.5105
1.2	3.19	0.3239
1.4	3.53	0.7930
1.6	4.38	2.0610
1.8	**2.90**	**0.1893**
2.0	3.35	0.2951

Table 2.7: *GOODS* Catalog: mean classification error (%) for *PPSPR* (errors averaged over 25 iterations for fixed α). In bold is presented the lower mean classification error. The lower standard deviation is underlined.

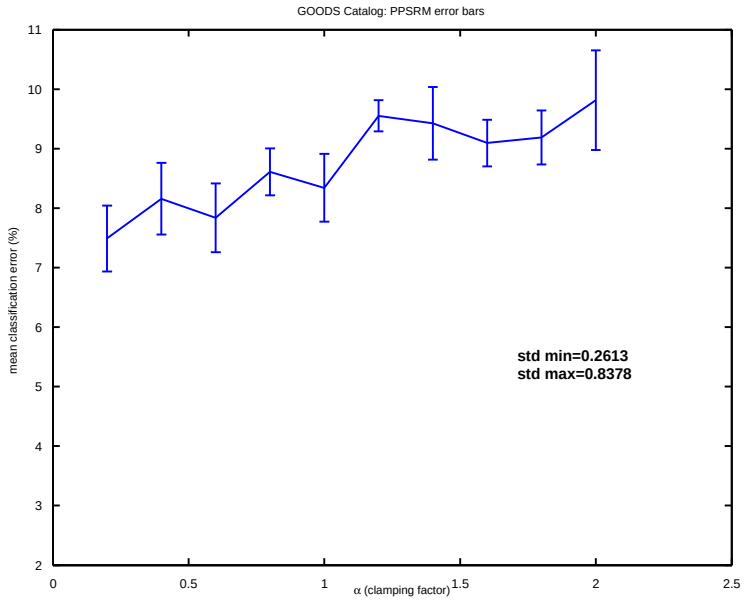

Figure 2.10: *GOODS* Catalog: error bars for *PPSRM* (errors averaged over 25 iterations for fixed α).

Classifier type - Error (%)	*Confusion Matrix*					*Best model* α
		Star	*Galaxy*	*StarD*	*GalaxyD*	
PPSRM - 6.51	*Star*	124	140	2	2	0.2
	Galaxy	40	1080	2	20	
	StarD	4	26	98	416	
	GalaxyD	0	0	88	9322	
		Star	*Galaxy*	*StarD*	*GalaxyD*	
PPSPR – 2.63	*Star*	90	10	2	4	1.8
	Galaxy	78	1216	2	18	
	StarD	0	0	62	42	
	GalaxyD	0	20	124	9696	

Table 2.8: *GOODS* Catalog: confusion matrices computed by *PPSRM* and *PPSPR* best models.

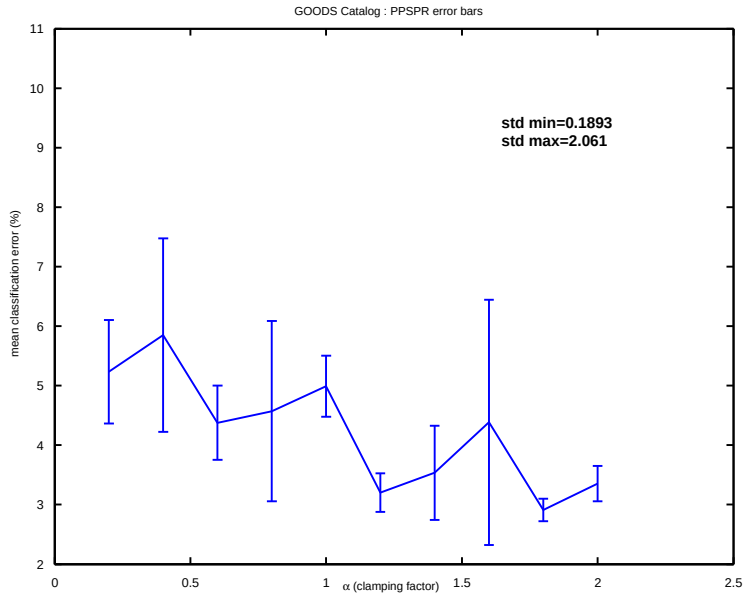

Figure 2.11: *GOODS* Catalog: errors bars for *PPSPR* (errors averaged over 25 iterations for fixed α).

TNG telemetry data

Here we have data coming from sensors of (TNG), collected into three separate observation sessions. Each session is associated to the quality of the images acquired by *TNG*. After the preprocessing phase described in appendix 6.1.3, the data set is composed by three classes corresponding to good, medium and bad quality images, respectively. Each image is described by a vector of 17 values corresponding to the parameters of different groups of sensors of *TNG*. Our experiment was devoted to find whether there was any correlation among the telemetry data and the quality (in terms of tracking, seeing, etc.) of the data. The existence of such a correlation would allow both to put a quality flag on the scientific exposures, and (if real time monitoring is implemented) to interrupt potentially bad exposure in order to avoid waste of precious observing time.

Before starting the *PPS* training steps, we randomly divided the data set in 50% for training and 50% for testing. The *PPS* parameter setting is listed in table 2.9 and it can be seen that we employee a *PPS* of reduced complexity (only 33 latent nodes and 6 basis functions) with respect to the two previous cases studied so far. This is justified

Parameter	*Value*	*Description*
M	33	number of latent variables
L	6	number of basis functions
L_{fac}	1	basis functions width
$class$	NT	projection method for classification
$iter$	100	maximum number of iteration
ϵ	0.001	early stopping threshold

Table 2.9: *TNG* Data: parameter setting for *PPSRM* and *PPSPR*.

from a preliminary analysis of the parameter values selected for training. In fact, they are very different for each of the three classes, therefore we expect a good separation between classes. Indeed our expectation is validated by the results: figure 2.12 and table 2.10 say that *PPSRM* obtained very high performances whose mean classification errors over the ten different α values range between a minimum of 0.031 (for $\alpha = 1.8$) and a maximum of 0.061 and even the standard deviation values are very low (in the range $0.0147 - 0.0290$). *PPSPR* instead, exhibits an inverse behavior, i.e. it perform worse with respect to *PPSRM*, compared to the previous cases. In fact, the minimum mean classification error obtained for $\alpha = 1.6$ and $\alpha = 0.6$ is 0.131 and the maximum is fixed to 0.587. Moreover, in table 2.12, it can be seen that *PPSRM* obtains an exact classification (no errors) for different values of α $(0.2, 0.4, 1.8, 2.0)$. It is worth noting here that in order to consider the *PPS* classifier (both reference manifold and probabilistic) as an important tool to asses whether there is any correlation among the telemetry data and the quality of the data it is necessary to have a greater number of images and a greater number of different representative cases for each class of images quality. In other words, each class may have different sensor value configurations which could be fixed as template for the given class.

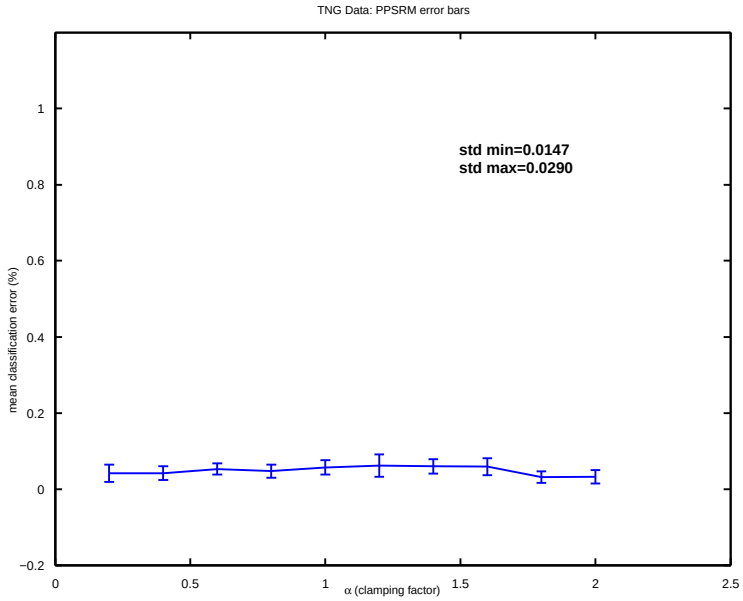

Figure 2.12: *TNG* Data: error bars for *PPSRM* (errors averaged over 25 iterations for fixed α).

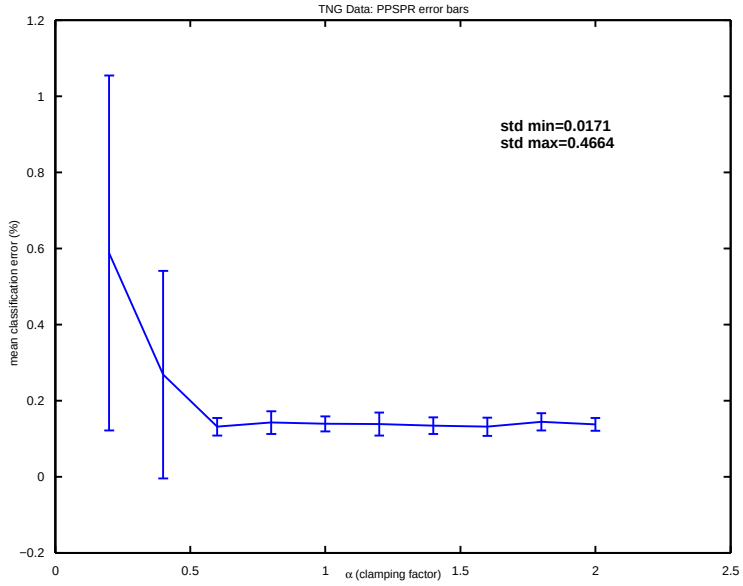

Figure 2.13: *TNG* Data: errors bars for *PPSPR* (errors averaged over 25 iterations for fixed α).

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	0.041	0.0229
0.4	0.041	0.0180
0.6	0.052	0.0149
0.8	0.047	0.0170
1.0	0.057	0.0187
1.2	0.061	0.0290
1.4	0.059	0.0188
1.6	0.059	0.0222
1.8	**0.031**	**0.0147**
2.0	0.032	0.0174

Table 2.10: *TNG* Data: mean classification error (%) for *PPSRM* (errors averaged over 25 iterations for fixed α). In bold is presented the lower mean classification error. The lower standard deviation is underlined.

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	0.587	0.4664
0.4	0.268	0.2729
0.6	**0.131**	**0.0231**
0.8	0.142	0.0299
1.0	0.138	0.0196
1.2	0.138	0.0301
1.4	0.134	0.0215
1.6	**0.131**	**0.0238**
1.8	0.144	0.0228
2.0	0.137	0.0171

Table 2.11: *TNG* Data: mean classification error (%) for *PPSPR* (errors averaged over 25 iterations for fixed α). In bold is presented the lower mean classification error. The lower standard deviation is underlined.

Classifier type - Error(%)	*Confusion Matrix*				*Best model* α
$PPSRM$ - 0		*Good*	*Medium*	*Bad*	1.8
	Good	2230	0	0	
	Medium	0	3680	0	
	Bad	0	0	6140	
$PPSPR - 0.082$		*Good*	*Medium*	*Bad*	$0.6, 1.6$
	Good	2230	0	10	
	Medium	0	3680	0	
	Bad	0	0	6130	

Table 2.12: *TNG* Data: confusion matrices computed by *PPSRM* and *PPSPR* best models.

Summary of experiments with *PPSRM* and *PPSPR*

From the experiments seen so far we can state that *PPS* classifier perform very well on real complex astronomical data. If the data sets are more complex (more overlapping classes) *PPS* classifier used by computing the a posteriori class probability maybe could be more appropriate as it leads to the lowest mean classification errors despite a less stability with respect to the *PPS* reference manifold classifier, even though the superiority of the *PPSPR* should be proved by further experimental evidences. However, our aim here is not to demonstrate better performance between different *PPS* classifier methods but rather the overall viability of *PPS* to address complex astronomical data classification. Furthermore, our results confirm the results shown in [17], i.e *PPS* with $\alpha < 1$ lead to better performance with respect to *GTM* model as can be seen from *PPS* models, with α fixed to 1.0, performances; on the other hand, however, we obtained cases in which the *aligned-GTM* performs better than *PPS* with $\alpha < 1$. Finally, we observed faster convergence in *PPS* models with $\alpha \neq 1$.

Chapter 3

Committee of Probabilistic Principal Surfaces

Ensemble or committee of learning systems is a way to construct learning machines which could obtain better generalization performance with respect to a single model in both regression and classification tasks. In this chapter an overview to this research area is provided giving also its underlying motivations by introducing the well known bias-variance dilemma or "trade-off" for regression and classification problems. Furthermore, two combining schemes for constructing committees of probabilistic principal surfaces are proposed and their effectiveness is demonstrated in the experimental section for classification purposes.

3.1 Bias and Variance

While constructing a learning model one has two ways for measuring its "match" or "alignment" to the problem, being it a regression or a classification problem: the bias and variance. The bias measures the accuracy or quality of the match, in other words, high bias implies a poor match. The variance measures the precision or the specificity of the match, i.e., high variance implies a weak match. Bias and variance can be adjusted in several ways, but the important bias-variance relation says that the two terms are not independent. Fixed a loss function, they obey a sort of "conservation law". Now we discuss theoretically this issue.

3.1.1 Bias-Variance Decomposition for Regression

The mathematical treatment of the bias-variance decomposition is based on the work described in [36]. It is convenient to consider the particular case of a model trained using

a sum of squares error function. The sum of square error, in the limit of an infinite data set, can be written as [4]

$$E = \frac{1}{2}\int \{y(\mathbf{t}) - \langle k|\mathbf{t}\rangle\}^2 p(\mathbf{t})d\mathbf{t} + \frac{1}{2}\int \{\langle k^2|\mathbf{t}\rangle - \langle k|\mathbf{t}\rangle^2\} p(\mathbf{t})d\mathbf{t} \tag{3.1}$$

in which $p(\mathbf{t})$ is the unconditional density of the input data, $y(\mathbf{t})$ is the model function and $\langle k|\mathbf{t}\rangle$ denotes the conditional average, or regression, of the target data given by

$$\langle k|\mathbf{t}\rangle \equiv \int k p(k|\mathbf{t})dk \tag{3.2}$$

where $p(k|\mathbf{t})$ is the conditional density of the target variable k conditioned on the input vector $\mathbf{t}$. Similarly

$$\langle k^2|\mathbf{t}\rangle \equiv \int k^2 p(k|\mathbf{t})dk \tag{3.3}$$

The second term in (3.1) is independent of the network function $y(\mathbf{t})$ and hence is independent of the network weights. The optimal network function $y(\mathbf{t})$, in the sense of minimizing the sum of squares error, is the one which makes the first term in (3.1) vanish, and is given by $y(\mathbf{t}) = \langle k|\mathbf{t}\rangle$. The second term represents the intrinsic noise in the data and sets a lower limit on the error which can be achieved. In a practical situation we must deal with the problems arising from a finite size data set. Suppose we consider a training set D consisting of N patterns which we use to determine the network model $y(\mathbf{t})$. Now consider a whole ensemble of possible data sets, each containing N patterns, and each taken from the same fixed joint distribution $p(\mathbf{t}, k)$. A measure of how close the actual mapping function $y(\mathbf{t})$ is to the desired one is given by the integrand of the first term in (3.1)

$$\{y(\mathbf{t}) - \langle k|\mathbf{t}\rangle\}^2. \tag{3.4}$$

The value of this quantity will depend on the particular data set D on which it is trained. We can eliminate this dependence by considering an average over the complete ensemble of data sets,

$$\mathcal{E}_D[\{y(\mathbf{t}) - \langle k|\mathbf{t}\rangle\}^2] \tag{3.5}$$

where $\mathcal{E}_D[\cdot]$ denotes the expectation, or ensemble average. If the network function were always a perfect predictor of the regression function $\langle k|\mathbf{t}\rangle$ then this error would be zero. The (3.5) in a different mathematical form,

$$\{y(\mathbf{t}) - \langle k|\mathbf{t}\rangle\}^2 = \{y(\mathbf{t}) - \mathcal{E}_D[y(\mathbf{t})] + \mathcal{E}_D[y(\mathbf{t})] - \langle k|\mathbf{t}\rangle\}^2 =$$

$$= \{y(\mathbf{t}) - \mathcal{E}_D[y(\mathbf{t})]\}^2 + \{\mathcal{E}_D[y(\mathbf{t})] - \langle k|\mathbf{t}\rangle\}^2 + 2\{y(\mathbf{t}) - \mathcal{E}_D[y(\mathbf{t})]\}\{\mathcal{E}_D[y(\mathbf{t})] - \langle k|\mathbf{t}\rangle\}. \quad (3.6)$$

In order to compute the expression in (3.5) we take the expectation of both sides of (3.6) over the ensemble of data sets D. We see that the third term on the right-hand side of (3.6) vanishes, and we are left with

$$\mathcal{E}_D[\{y(\mathbf{t}) - \langle k|\mathbf{t}\rangle\}^2] =$$

$$= \underbrace{\{\mathcal{E}_D[y(\mathbf{t})] - \langle k|\mathbf{t}\rangle\}^2}_{(bias)^2} + \underbrace{\mathcal{E}_D[\{y(\mathbf{t}) - \mathcal{E}_D[y(\mathbf{t})]\}^2]}_{variance}. \quad (3.7)$$

In the expression (3.7) the bias measures the extent to which the average (over all data sets) of the network function differs from the desired function $\langle k|\mathbf{t}\rangle$. Conversely, the variance measures the extent to which the network function $y(\mathbf{t})$ is sensitive to the particular choice of data set. The meaning of the bias and variance terms can be illustrated by considering two extreme limits for the choice of functional form $y(\mathbf{t})$. We shall suppose that the target data for network training is generated from a smooth function $h(\mathbf{t})$ to which zero mean random noise ϵ is added, so that

$$k = h(\mathbf{t}) + \epsilon. \quad (3.8)$$

The optimal mapping function in this case is given by $\langle k|\mathbf{t}\rangle = h(\mathbf{t})$. One choice of the model for $y(\mathbf{t})$ would be some fixed function $g(\mathbf{t})$ which is completely independent of the data set D. It is clear that the variance term in (3.7) will vanish, since $\mathcal{E}_D[y(\mathbf{t})] = g(\mathbf{t}) = y(\mathbf{t})$. However, the bias term will typically be high since no attention at all was paid to the data, and so unless we have some prior knowledge which help us to choose the function $g(\mathbf{t})$ we are making a bad guess. The opposite extreme is to take a function which fits the training data perfectly, such as a simple exact interpolant. In this case the bias term vanishes at the data points themselves since

$$\mathcal{E}_D[y(\mathbf{t})] = \mathcal{E}_D[h(\mathbf{t}) + \epsilon] = h(\mathbf{t}) = \langle k|\mathbf{t}\rangle$$

and the bias will typically be small in the neighborhood of the data points. The variance, however, will be significative since

$$\mathcal{E}_D[\{y(\mathbf{t}) - \mathcal{E}_D[y(\mathbf{t})]\}^2] = \mathcal{E}_D[\{y(\mathbf{t}) - h(\mathbf{t})\}^2] = \mathcal{E}_D[\epsilon^2]$$

which is just the variance of the noise on the data, which could be substantial. We see that there is a natural trade-off between bias and variance. A function which is closely

fitted to the data set will tend to have a large variance and hence give a large expected error. We can decrease the variance by smoothing the function, but if this is taken too far then the bias becomes large and the expected error is again large. The bias-variance dilemma can be illustrated in the domain of regression (figure 3.1 taken from (taken from [26])): each column represents a different model, and each row represents a different set of $N = 6$ training points, $\mathcal{D}_i$, randomly sampled from the true function $h(\mathbf{t})$ with noise. Probability functions of the mean-square error of $\mathcal{E}_D[\{(y(\mathbf{t}) - h(\mathbf{t})\}^2]$ are shown at the bottom. Column a) shows a very poor model: a linear $y(\mathbf{t})$ whose parameters are held fixed, independent of the training data. This model has high bias and zero variance. Column b) shows a somewhat better model, though it too is held fixed, independent of the training data. It has a lower bias than in column a) and has the same zero variance. Column c) shows a cubic model, where the parameters are trained to best fit the training samples in a mean-square-error sense. This model has low bias and a moderate variance. Column d shows a linear model that is adjusted to fit each training set; this model has intermediate bias and variance. If these models were instead trained with a very large number N of points, the bias in column c) would approach a small value (which depends upon the noise), while the bias in column d would not; the variance of all models would approach zero.

3.1.2 Bias-Variance Decomposition for Classification

While the bias variance decomposition and dilemma are simpler to understand in the case of regression under the mean-squared loss function, we are most interested in their relevance to classification. Several suggestions have been made in literature for other loss functions [11, 40, 41, 24]. Here the discussion is based on the decomposition proposed in [35] where the classification task is casted in the regression framework. Let us consider a two-class classification problem, where an output variable k assumes values in $\{0, 1\}$. In this context the mean-squared error does not appear the proper one, but we can proceed as it follows.

The goal of a classification procedure is to predict the output value given the set of input variables $\mathbf{t} = \{t_1, \ldots, t_D\}$. It is often the case that at a particular point $\mathbf{t}$ the value of k is not uniquely determinable. It can assume both its values with respective probabilities

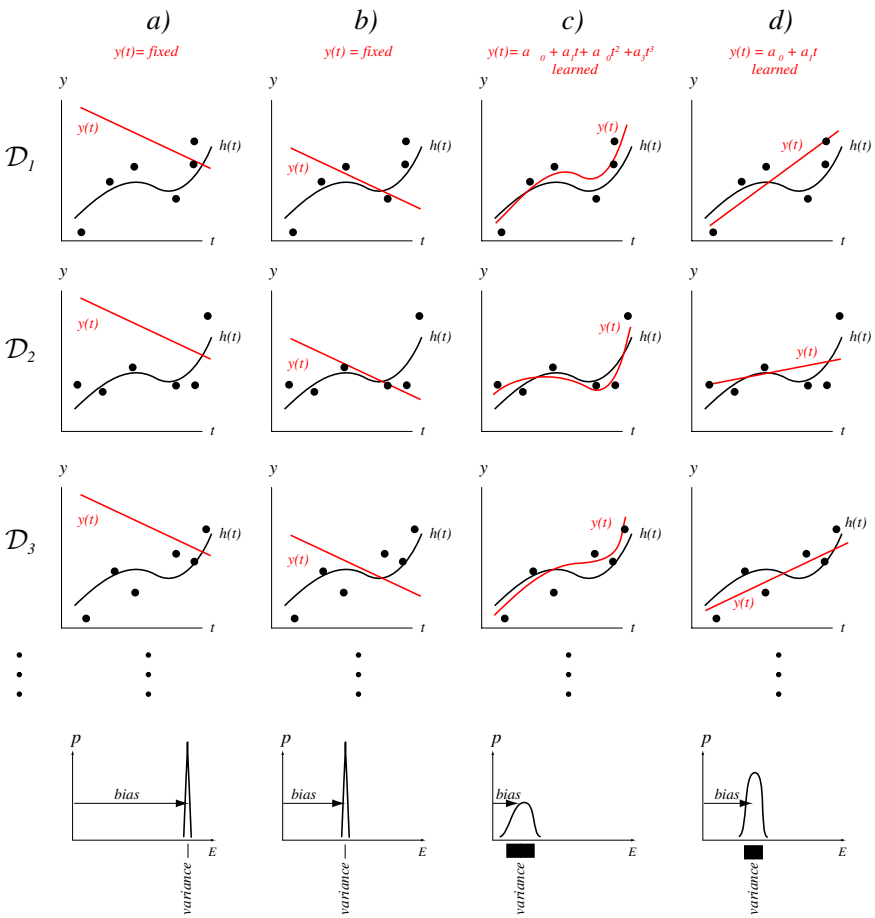

Figure 3.1: The Bias-Variance Dilemma for regression.

that depend on the location of the point $\mathbf{t}$ in the input space

$$h(\mathbf{t}) = Pr[k = 1|\mathbf{t}] = 1 - Pr[k = 0|\mathbf{t}]. \tag{3.9}$$

$h(\mathbf{t})$ is a single value deterministic function that at every point $\mathbf{t}_i$ specifies the probability that k assumes value 1. The role of a classification procedure is to produce a rule that makes a prediction $y(\mathbf{t}) \in \{0, 1\}$ for the correct class label k at every input point $\mathbf{t}$. The goal is to choose $y(\mathbf{t})$ to minimize inaccuracy as characterized by the misclassification "risk"

$$r(\mathbf{t}) = l_1 h(\mathbf{t}) 1(y(\mathbf{t}) = 0) + l_0(1 - h(\mathbf{t}))1(y(\mathbf{t}) = 1). \tag{3.10}$$

where l_0 and l_1 are the losses incurred for the respective misclassifications, and $1(\cdot)$ is an indicator function of the truth of its argument

$$1(\eta) = \begin{cases} 1 & if \quad \eta \; is \; true \\ \\ 0 & otherwise \end{cases}$$

The misclassification risk (3.10) is minimized by the Bayes rule

$$k_B(\mathbf{t}) = 1\left(h(\mathbf{t}) \geq \frac{l_0}{l_0 + l_1}\right) \tag{3.11}$$

which by definition achieves the lowest possible risk

$$r_B(\mathbf{t}) = min(l_1 h(\mathbf{t}), l_0(1 - h(\mathbf{t}))). \tag{3.12}$$

For simplicity we take $l_0 = l_1$ in (3.12) so that the threshold in the indicator function is $1/2$ and the Bayes decision boundary is the set of points for which $h(\mathbf{t}) = 1/2$. Now the classification task can be cast in the regression framework setting by considering the expected value of k. To do so, we consider a discriminant function

$$k = h(\mathbf{t}) + \epsilon, \tag{3.13}$$

where ϵ is a zero mean, random variable, for simplicity here assumed to be a centered binomial distribution with variance $Var[\epsilon|\mathbf{t}] = h(\mathbf{t})(1 - h(\mathbf{t}))$. The target function can thus be expressed as

$$h(\mathbf{t}) = \mathcal{E}[k|\mathbf{t}] = \langle k|\mathbf{t}\rangle,$$

and now the goal is to find an estimate $y(\mathbf{t})$ that minimizes a mean-square error, such as (3.4),

$$\mathcal{E}_{\mathcal{D}}[(y(\mathbf{t}) - \langle k|\mathbf{t}\rangle)^2].$$

In this way the regression method seen before, can yield an estimate $y(\mathbf{t})$ used for classification. For a given training set $\mathcal{D}$, if the classification error rate $Pr[y(\mathbf{t}) = k]$, averaged over predictions at $\mathbf{t}$, agrees with the Bayes discriminant,

$$Pr[y(\mathbf{t}) = k] = Pr[k_B \neq k] = min[h(\mathbf{t}), 1 - h(\mathbf{t})]$$

then indeed we have the lowest error. If not, then the prediction yields an increased error

$$Pr[y(\mathbf{t})] = max[h(\mathbf{t}), 1 - h(\mathbf{t})] = |2h(\mathbf{t}) - 1| + Pr[k_B = k].$$

We average over all data sets of size N and find

$$Pr[y(\mathbf{t}) \neq k] = |2h(\mathbf{t}) - 1|Pr[y(\mathbf{t}) \neq k_B] + Pr[k_B \neq k]. \tag{3.14}$$

Equation (3.14) shows that classification error rate is linearly proportional to $Pr[y(\mathbf{t}) \neq k_B]$, which can be considered a boundary error in that it represents the incorrect estimation of the optimal (Bayes) Boundary. Because of random variations in training sets, the boundary error will depend upon $p(y(\mathbf{t}))$, the probability density of obtaining a particular estimate of the discriminant given $\mathcal{D}$. This error is merely the area of the tail of $p(y(\mathbf{t})$ on the opposite side of the Bayes discriminant value $1/2$:

$$Pr[y(\mathbf{t}) \neq k_B] = \begin{cases} \int_{\frac{1}{2}}^{\infty} p(y(\mathbf{t}))dy & if \quad h(\mathbf{t}) < \frac{1}{2} \\ \\ \int_{-\infty}^{\frac{1}{2}} p(y(\mathbf{t}))dy & if \quad h(\mathbf{t}) \geq \frac{1}{2} \end{cases}$$

If we make the assumption that $p(y(\mathbf{t}))$ is a Gaussian, we find

$$Pr[y(\mathbf{t}) \neq k_B] = \Phi\left[\underbrace{Sgn[h(\mathbf{t}) - 1/2][\mathcal{E}_{\mathcal{D}}[y(\mathbf{t})] - 1/2]}_{boundary\ bias}\underbrace{Var[y(\mathbf{t})]^{-\frac{1}{2}}}_{variance}\right] \tag{3.15}$$

where

$$\Phi[u] = \frac{1}{\sqrt{2\pi}}\int e^{-\frac{1}{2u^2}}du$$

The boundary error is expressed in terms of a boundary bias, in analogy with the bias-variance decomposition for regression. Equation (3.15) shows that the effect of the variance

term on the boundary error is highly non linear and depends on the value of the boundary bias . Furthermore when the variance is small, this effect is particularly sensitive to the sign of the bias. In regression the estimation error is additive in bias and variance, whereas for classification there is a nonlinear and multiplicative interaction. In classification the sign of the boundary bias affects the role of the variance in the error. For this reason low variance is generally important for accurate classification, while low boundary bias need not to be. Said in another way, in classification variance dominates bias. Figure 3.2 (taken from [26]) provides an example to graphically understand the Bias-Variance dilemma: the (boundary) bias-variance trade-off in classification is illustrated with a two-dimensional Gaussian problem. The figure at the top shows the true distributions and the Bayes decision boundary. The nine figures in the middle show different learned decision boundaries. Each row corresponds to a different training set of $N = 8$ points selected randomly from the true distributions and labelled according to the true decision boundary. Column a) shows case of a Gaussian model with fully general covariance matrices trained by maximum-likelihood. The learned boundaries differ significantly from one data set to the next; this learning algorithm has high variance. Column b) shows the decision boundaries resulting from fitting a Gaussian model with diagonal covariances; in this case the decision boundaries vary less from one row to another. This learning algorithm has a lower variance than the one at the left. Finally, column c) shows decision boundaries learning by fitting a Gaussian model with unit covariances (i.e., a linear model); notice that the decision boundaries are nearly identical from one data set to the next. This algorithm has low variance.

3.2 Committee Machines

In committee machines, an ensemble of estimators is generated by means of a learning process and the prediction of the committee for a new input is generated in form of a combination of the predictions of the individual committee members. Committee machines can be useful in many ways, as listed below

1. the committee might exhibit a test set performance unobtainable by an individual committee member on its own. The reason is that the errors of the individual committee members cancel out to some degree when their predictions are combined.

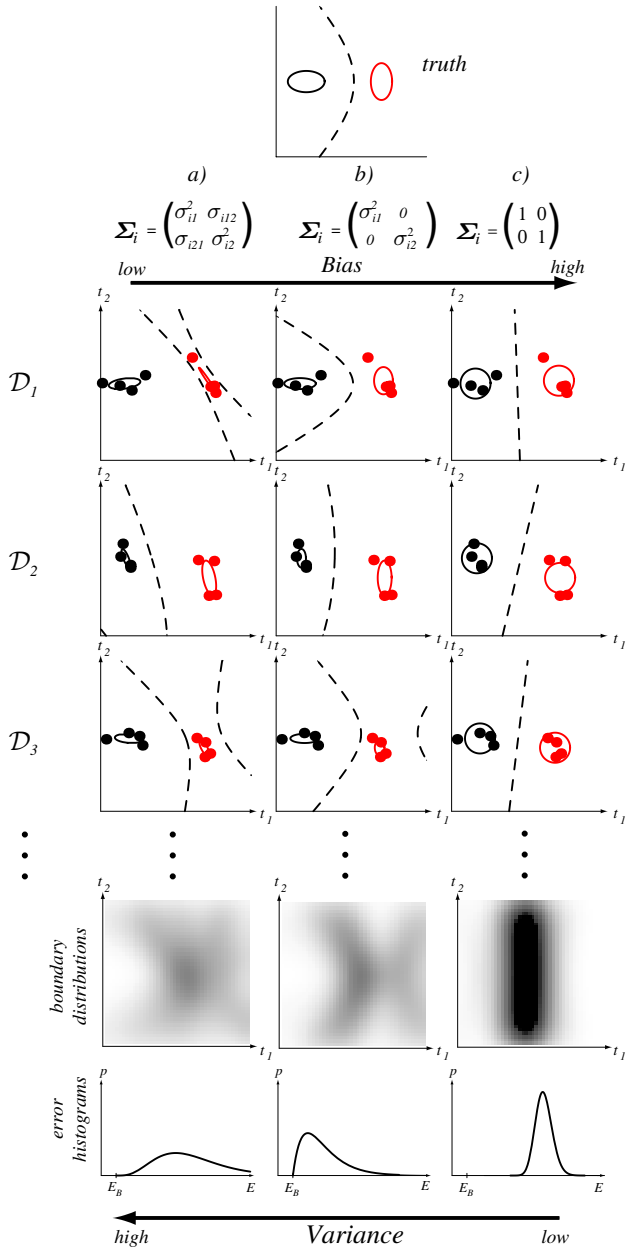

Figure 3.2: The Bias-Variance Dilemma for classification.

Even if the committee members were trained on "disturbed" version of the same data set, the predictions of the individual committee members might be sufficiently different such that this averaging process take place and is beneficial;

2. modularity. It is sometimes beneficial if a mapping from input to target is not approximated by one estimator but several estimators, where each estimator can focus on a particular region of input space. The prediction of the committee is obtained by a locally weighted combination of the predictions of the committee members. In some applications the individual members self-organize in a way such that the prediction task is divided into meaningful modules;

3. reduction of computational complexity. Instead of training one estimator using all training data it is computationally more efficient for some types of estimators to partition the data set into several data sets, train different estimators on the individual data sets and then combine the predictions of the individual estimators. By using a committee machine approach, the computational complexity increases only linearly with the size of the training data set.

3.2.1 Averaging, Bagging and Stacking

The idea is to train a committee of estimators and combine the individual predictions with the goal of achieving improved generalization performance if compared to the performance achievable with a single estimator. In regression, the committee prediction for a test input $\mathbf{t}$ is achieved by forming a weighted sum of the predictions of the M committee members

$$\hat{k}(\mathbf{t}) = \sum_{i=1}^{M} \alpha_i y_i(\mathbf{t})$$

where $y_i(\mathbf{t})$ is the prediction of the i-th committee member at input $\mathbf{t}$ and where α_i are weights which are required to be positive and to sum to one. In classification, the combination is typically implemented as a voting scheme. The committee assigns the pattern to the class which obtains the majority of the vote

$$\hat{class}(\mathbf{t}) = argmax_j \sum_{i=1}^{M} \alpha_i y_{i,class=j}(\mathbf{t})$$

where $y_{i,class=j}(\mathbf{t})$ is the output of the classifier i for class j. The output typically either corresponds to the posterior class probability $y_{i,class=j}(\mathbf{t}) \in [0,1]$ or to a binary decision $y_{i,class=j}(\mathbf{t}) \in \{0,1\}$.

The motivation for pursuing committee methods can be understood by analyzing the prediction error of the combined system, i.e. the bias-variance decomposition (3.7) (for simplicity, we hide the dependence on input $\mathbf{t}$). We are interested in estimating the target k by forming a linear combination of the y_i

$$\hat{k} = \sum_{i=1}^{M} \alpha_i y_i = \alpha^{'} y$$

where $y = (y_1, \ldots, y_M)^{'}$ is the vector of the predictions of the committee members and where $\alpha = (\alpha_1, \ldots, \alpha_M)^{'}$ is the vector of the weights. The expected error of the combined system is

$$\begin{aligned} \mathcal{E}(\hat{k} - k)^2 &= \mathcal{E}(\alpha^{'} y - \mathcal{E}(\alpha^{'} y))^2 + \mathcal{E}(\mathcal{E}(\alpha^{'} y) - k)^2 \\ &= \mathcal{E}(\alpha^{'}(y - \mathcal{E}(y)))^2 + \mathcal{E}(\alpha^{'} m - k)^2 = \alpha^{'} \Omega \alpha + (\alpha^{'} m - k)^2 \end{aligned} \tag{3.16}$$

where Ω is a $M \times M$ covariance matrix with $\Omega_{ij} = \mathcal{E}[(y_i - m_i)(y_j - m_j)]$, and where $m = (m_1, \ldots, m_M)^{'}$ is the vector of the expected values of the predictions of the committee members. The term $\alpha^{'} \Omega \alpha$ denotes the variance of the committee and $\alpha^{'} m - k$ is the bias of the committee. By setting $\alpha_i = 1/M$, we average the predictors, and (3.16) becomes

$$\mathcal{E}(\hat{k} - k)^2 = \frac{1}{M^2} \sum_{i=1}^{M} \Omega_{ii} + \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \Omega_{ij} + \frac{1}{M^2} \left(\sum_{i=1}^{M} (m_i - k) \right)^2 . \tag{3.17}$$

Let us assume now that mean $m_i = mean$, the variance $\Omega_{ii} = var$ and the covariances $\Omega_{ij} = cov$ are identical for all members, then

$$\mathcal{E}(\hat{k} - k)^2 = \frac{1}{M} var + \frac{M^2 - M}{M^2} cov + (mean - k)^2.$$

The last expression says that: 1) the bias of the combined system $(mean - k)$ is identical to the bias of each member and is not reduced. Therefore, estimators should be used which have low bias and regularization, which introduces bias, should be avoided; 2) the estimators should have low covariance, since this term in the error function cannot be reduced by increasing M; 3) the term corresponding to the variances of the committee members decreases as $1/M$. Definitely if we have estimators with low bias and low covariances between members, the expected error of the combined system is significantly

less than the expected errors of the individual members. We can synthesize saying that 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. Anyway, in practical problems things are not so simple. In regression, $t(\mathbf{t})$ corresponds to the optimal regression function and $y_i(\mathbf{t})$ to the prediction of the i-th estimator. Here, the squared error is commonly used and the bias-variance decomposition just described is applicable. In classification $t(\mathbf{t})$ might correspond to the probability of the class one, $1-k(\mathbf{t})$ to the probability for class two, and $y_i(\mathbf{t})$ is the estimate of the i-th estimator for $k(\mathbf{t})$. In this case one can proceed as described in section 3.1.2 or employing other bias-variance decompositions suited for classification.

Averaging

In this approach, committee members are typically neural networks. The neural networks are all trained on the complete training data set. A de-correlation among the neural networks predictions is typically achieved by varying the initial conditions in training the neural networks such that different neural networks converge into different local minima of the cost function. Despite its simplicity, this procedure is surprisingly successful and turns an apparent disadvantage, i.e. local minima in training neural networks, into something useful. This approach was initialized in [54] and drew a lot of attention to the concept of committee machines. Using the the Cauchy inequality in [54] is shown that even for correlated and biased predictors the squared prediction error of the committee machines is equal to or less than the mean squared prediction error of the committee members, i.e.,

$$(\hat{k}-k)^2 \leq \frac{1}{M}\sum_{i=1}^{M}(y_i-k)^2.$$

This means that as long as the committee members have good prediction performance, averaging cannot make things really worse; it is as good as the average model or better. This can be also understood from the work described in [45]. Here it is shown that, in the special case of averaging, $\alpha_i = 1/M$,

$$(\hat{k}-k)^2 = \frac{1}{M}\sum_{i=1}^{M}(y_i-k)^2 - \frac{1}{M}\sum_{i=1}^{M}(y_i-\hat{k})^2$$

which means that the generalization error of the committee is equal to the average of the generalization error of the members minus the average variance of the committee members

(the ambiguity) which immediately leads to the previous bound. In highly regularized neural networks, the ensemble ambiguity is typically small and the generalization error is essentially equal to the average generalization error of the committee members. If neural networks are not strongly regularized the ensemble ambiguity is high and the generalization error of the committee should be much smaller than the average generalization error of the committee members.

Bagging

Bagging (Bootstrap aggregation) [10] is aimed to reduce the correlation between estimators in order to further improve generalization performance. Let us assume that each committee member is trained on a different data set. Then, surely, the covariance between the predictions of the individual members is zero. Unfortunately, we have to work with a fixed training data set. Although it is then impossible to obtain a different training data set for each member, we can at least mimicry this process by training each member on a bootstrap sample of the original data set. Bootstrap data sets [27] are generated by drawing randomly K data points from the original data set of size K with replacement. This means that some data points will appear more then once in a given new data set and some other will not appear at all. The procedure is repeated M times obtaining M non-identical data sets which are then used to train estimators. The output of the committee is then obtained by simple averaging (regression) or by voting (classification). Experimental evidence suggests that bagging typically outperforms simple averaging and voting. A key point for bagging to work properly is that the committee members should be unstable. In fact, for a given bootstrap sample, an instance in the training set has probability $1-(1-\frac{1}{m})^m$ of being selected at least once in the m times instances which are randomly selected from the training set. For large m, this is about $1-\frac{1}{e}=63.2\%$, which means that each bootstrap sample contains only about 63.2% unique instances from the training set. This perturbation causes different estimators to be built if the basic estimators are unstable, and performance can improve if these estimators are good and not correlated; however, bagging may slightly degrade the performance of stable algorithms because effectively smaller training sets are used for training each classifier. Unstable means that estimators should be sensitive to changes in the training data set, e.g. neural networks should not be strongly regularized. But recall that well regularized neural networks in general perform better than

under -regularized neural networks and we are faced with another version of the bias-variance dilemma: if we use under regularized neural networks we start with suboptimal committee members but bagging improves performance considerably. If we start with well regularized neural networks we start with well-performing committee members but bagging does not significantly improve performance. Experimental evidence indicates that the optimal degree of regularization is problem-dependent [63].

Stacked Generalization

In stacking [70], the weights α_i are determined after training the committee members, typically by using leave-one-out cross-validation. Here we have a modular network system, with a set of M 'level 0' networks N_1^0 to N_M^0 whose outputs are combined using a 'level 1' network N^1. The idea is to train the level-0 networks first and examine their behavior when generalizing. This provides a new training set which is used to train the level-1 network. The specific procedure for setting up the stacked generalization system is as follows. Let the complete set of available data be denoted by D. We first leave aside a single data point from D as a validation point, and treat the remainder of D as a training set. All level-0 networks are then trained using the training partition and their outputs are measured using the validation data point. This generates a single pattern for a new data set which will be used to train the level-1 network N^1. The inputs of this network consist of the outputs of all the level-0 networks, and the target value is the corresponding target value from the original full data set. This process is now repeated with a different choice for the data point which is kept aside. After cycling through the full data set of N points we have N patterns in the new data set, which is now used to train N^1. Finally, all of the level-0 networks are re-trained using the full data set D. Predictions on new data can now be made by presenting new input vectors to the level-0 networks and taking their outputs as the inputs to the level-1 network, whose output constitutes the predicted output. In [70] it is suggested that the level-0 networks should contain a wide variety of different models, while the level-1 network should provide a relatively smooth function and hence should have a relatively simple structure.

3.3 Committee Machines for Density Estimation

In literature the main part of the work has been made in the context of supervised learning methods, hence all the techniques described so far were applied to this latter learning paradigm. Less attention, indeed, was paid to unsupervised learning methodologies [72, 60, 68] and even fewer in the field of density estimation [59, 53], where stacking and bagging were properly adapted to the unsupervised density estimation case. More recently boosting [58, 33] is employed for density estimation [57] as well. From now on we shall concentrate on the construction of an ensemble of generative latent variable models, namely an ensemble of probabilistic principal surfaces, and hence for unsupervised density estimation.
Supervised and unsupervised methods share the same motivation we already mentioned when adopting ensemble techniques to combining them. Furthermore, as stated in [59] and [25], in density estimation tasks model uncertainty plays a crucial role in the predictive error in inductive inference. Even when the model class under consideration contains the true density, if we are only given a finite data set, then there is always a chance of selecting the wrong model. Furthermore, even if the correct model is selected, there will typically be an estimation error in the parameters of that model. This can be summarized by writing:

$$P(f|D) = \sum_M \int d\theta_M P(\theta_M|D, M)P(M|D)f_{M,\theta_M}, \tag{3.18}$$

where f is a density function we are assuming generates data set D, M is a model, and θ_M is a set of values for the parameters for model M. The posterior probability $P(M|D)$ reflects model uncertainty, and the posterior $P(\theta_M|D, M)$ reflects uncertainty in setting the parameters even knowing the model. It is worth noting that if we know $P(M, \theta_M)$, the Bayes' theorem allows us to express the posteriors in (3.18) explicitly, so that we explicitly have $P(f|D)$ given by a weighted average of the f_{M,θ_M}. However, calculating the combining weights is a difficult task, therefore it is natural to call for schemes for combining multiple density models in an empirically-driven way.

3.3.1 Stacked *PPS* for Density Estimation: *StPPS*

The combining schema herein described may be seen as an instantiation of the method proposed in [59]. Let us suppose we are given M probabilistic principal surface models (i.e., M density estimators) $\{PPS_m(\mathbf{t})\}_{m=1}^M$, where $PPS_m(\mathbf{t})$ is the m-th PPS model.

Note that in the original formulation given in [59], the M density estimators could also be of different kind, for example finite mixtures with a fixed number of component densities or kernel density estimates with a fixed kernel and a single fixed global bandwidth in each dimension.

Now, going back to our model, each of the M *PPS* models can be chosen to be different enough, i.e. by considering different clamping factors α_m, number of latent variables and latent base functions. To stack the M *PPS* models, we follow the procedure described below (see also figure 3.3):

1. Let D the training data set, with size $|D| = N$. Partition D v times, as in v-fold cross-validation. The v-th fold contains exactly $(v-1)\frac{N}{v}$ training data points and $\frac{N}{v}$test data points both from the training set D. For each fold:

   - (a) fit each of the M *PPS* model to the training portion of D.
   - (b) evaluate the likelihood of each data point in the test partition of D, for each of the M fitted models.

2. At the end of this preliminary steps, we obtain M density estimators for each of the N data points which are organized in a matrix A, of size $N \times M$, where each entry a_{im} is $PPS_m(\mathbf{t}_i)$;

3. Use the matrix A to estimate the combination coefficients $\{\alpha_m\}_{m=1}^{M}$ that maximize the log-likelihood at the points $\mathbf{t}_i$ of a stacked density model of the form:

   $$StPPS(\mathbf{t}) = \sum_{m=1}^{M} \alpha_m PPS_m(\mathbf{t})$$

   which correspond to maximize

   $$\sum_{i=1}^{N} \ln \left(\sum_{m=1}^{M} \alpha_m PPS_m(\mathbf{t}_i) \right),$$

   as a function of the weight vector $(\alpha_1, \ldots, \alpha_M)$. Direct maximization of this function is a non-linear optimization problem. We can apply the EM algorithm directly, by observing that the stacked mixture is a finite mixture density with weights $(\alpha_1, \ldots, \alpha_M)$. Thus, we can use the standard EM algorithm for mixtures, except that the parameters of the component densities $PPS_m(\mathbf{t})$ are fixed and the only parameters allowed to vary are the mixture weights.

4. The concluding phase consists in the parameters re-estimation of each of the m component *PPS* models using all of the training data D. The stacked density model is then the linear combination of the so obtained component *PPS* models, with combining coefficients $\{\alpha_m\}_{m=1}^{M}$. It is worth stressing that this procedure becomes quite heavy in terms of computational performance. Therefore, with increasing size of the learning training sets, it is necessary to keep the number of *PPS* models low. Furthermore the number of folds in the cross-validation procedure needs to be not too high.

3.3.2 Experimental Results

The experiments follow the same organization described in last chapter. The difference is to build a model in which a group of different *PPS* models (in particular, we decided to use six *PPS* models in all the experiments), each of which has a fixed α value, are put together in an ensemble via stacking. All the obtained results are averaged over 25 iterations of the algorithm, in which every time a new training\test partition is randomly built. For *Synthetic* catalog and *GOODS* catalog, the data are split in 60% for training and 40% for testing, while for *TNG* data are split in 50% both for training and testing. An important parameter for stacking is the number v of folds in the cross-validation procedure. In our experiments we tried 5−fold and 10−fold cross-validation.

Synthetic Catalog

As shown in table 3.1, among the six different *PPS* models, not only the parameter α differs but the number M of latent variables and the number L of basis functions too. Moreover, the basis functions widths are set on the base of the number of basis functions. The results depicted in figure 3.4 say that 5−fold cross-validation works better than 10−fold cross-validation, in both the mean classification error (1.34 against 1.84, respectively) and standard deviation (0.2606 against 0.4071, respectively). The minimum error reached is 1.05 as shown in table 3.2 where is also shown the corresponding confusion matrix. The difference between 5−fold and 10−fold cross-validation could be explained by the fact that the size of the training set is quite high, so a 10−fold cross-validation may lead to

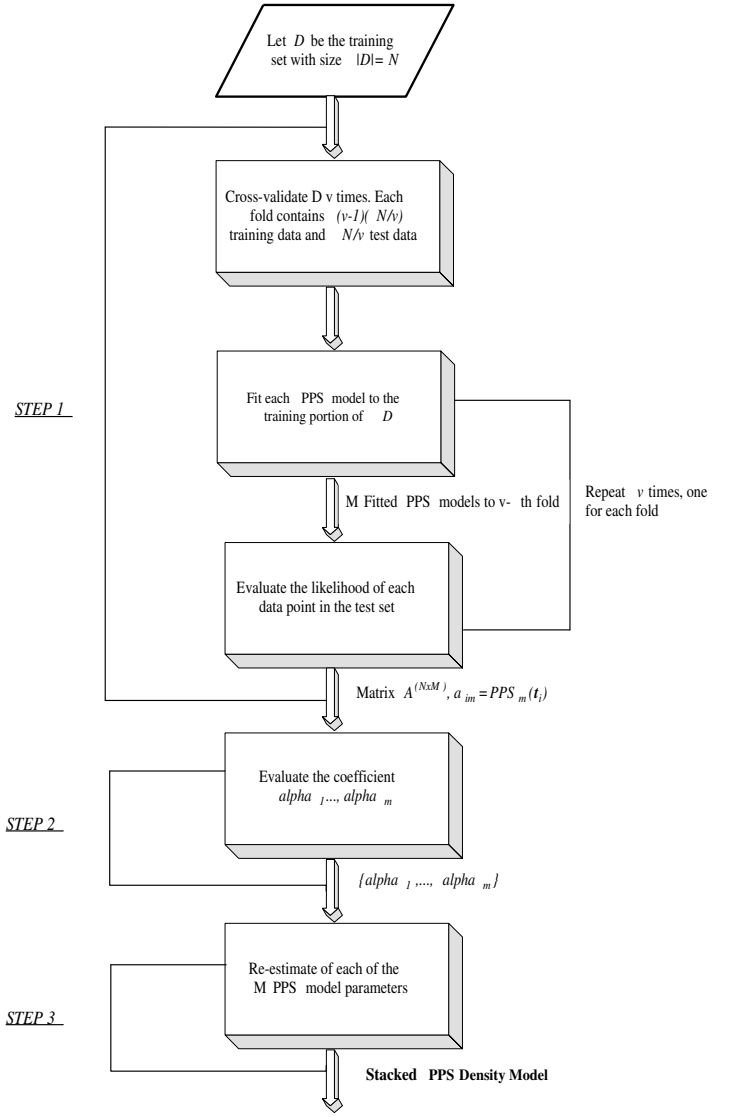

Figure 3.3: Steps of *PPS* stacking procedure.

Parameters	PPS_1	PPS_2	PPS_3	PPS_4	PPS_5	PPS_6
α	1	0.5	3	0.2	0.3	0.8
M	266	266	266	266	266	266
L	18	51	51	51	6	51
L_{fac}	2.2	2	2	2	2.5	2
$iter$	100	100	100	100	100	100
ϵ	0.01	0.01	0.01	0.01	0.01	0.01

Table 3.1: *Synthetic* Catalog: parameter setting for *StPPS* model.

Classifier type - Error(%)	*Confusion Matrix*		
		Star	*Galaxy*
$StPPS - 1.05$	*Star*	3943	27
	Galaxy	57	3973

Table 3.2: *Synthetic* Catalog: confusion matrix computed by *StPPS* best result.

over-fitting problems (recall that in our *PPS* models we don't employe any regularization method).

GOODS Catalog

In GOODS catalog the behavior of the stacked model, for which the parameters are set as in table 3.3, is inverted in terms of $5-$fold and $10-$fold cross-validation. In fact here we have better results for $10-$fold cross-validation (mean classification error 2.87 and standard deviation 0.1344) with respect to $5-$fold cross validation (mean classification error 3.44 and standard deviation 0.4720) as can be seen from figure 3.5. This is reasonable as the number of training data for the first three classes (*Star*, *Galaxy* and *StarD*) are much less than the number of training data for class *GalaxyD*, so a higher number of folds lead to a better fit to data. Confusion matrix corresponding to the minimum error (1.05) is shown in table 3.4.

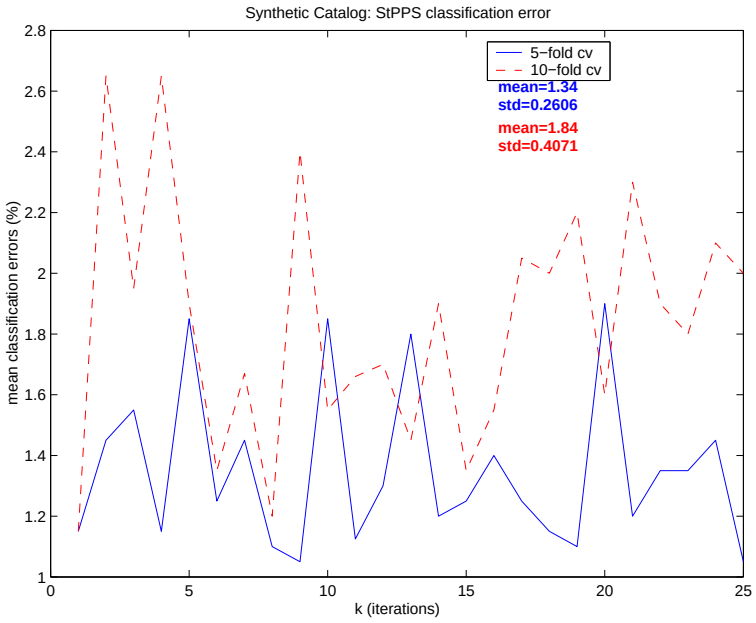

Figure 3.4: *Synthetic* Catalog: errors over 25 *StPPS* iterations.

Parameters	PPS_1	PPS_2	PPS_3	PPS_4	PPS_5	PPS_6
α	1.4	1.2	0.8	0.6	1.6	2.0
M	266	266	266	266	615	615
L	18	83	83	83	83	83
L_{fac}	1	2	1.5	1.1	1.3	2
$iter$	100	100	100	100	100	100
ϵ	0.01	0.01	0.01	0.01	0.01	0.01

Table 3.3: *GOODS* Catalog: parameter setting for *StPPS* model.

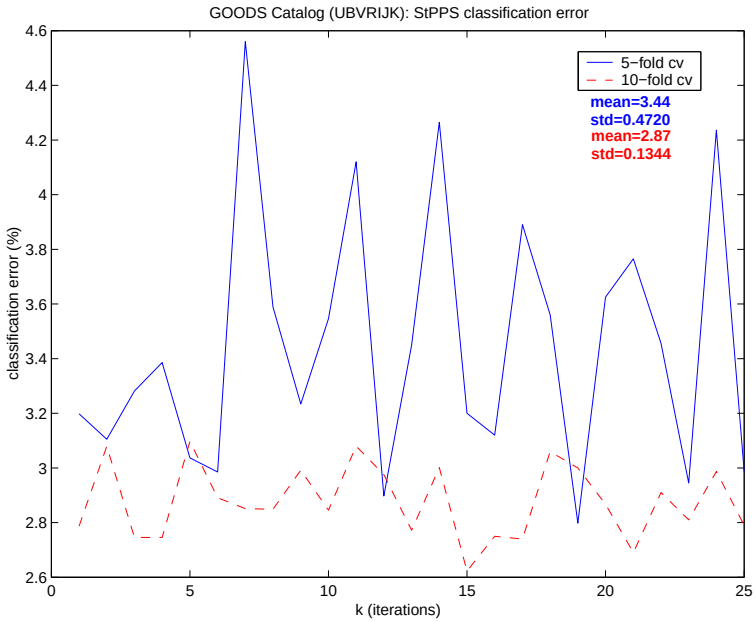

Figure 3.5: *GOODS* Catalog: errors over 25 *StPPS* iterations.

Classifier type - Error(%)	*Confusion Matrix*				
		Star	*Galaxy*	*StarD*	*GalaxyD*
$StPPS - 2.62$	*Star*	92	4	2	0
	Galaxy	76	1234	2	36
	StarD	0	0	52	36
	GalaxyD	0	8	134	9688

Table 3.4: *GOODS* Catalog: confusion matrices computed by *StPPS* best model.

Parameters	PPS_1	PPS_2	PPS_3	PPS_4	PPS_5	PPS_6
α	1	1.4	1.8	2.0	0.8	1.6
M	62	62	62	62	62	62
L	18	18	18	6	11	27
L_{fac}	1	1	1	1	1	1
$iter$	100	100	100	100	100	100
ϵ	0.01	0.01	0.01	0.01	0.01	0.01

Table 3.5: *TNG* Data: parameter setting for *StPPS* models.

Classifier type - Error(%)	*Confusion Matrix*			
		Good	*Medium*	*Bad*
$StPPS - 0.091$	*Good*	2230	0	11
	Medium	0	3680	0
	Bad	0	0	6129

Table 3.6: *TNG* Data: confusion matrix computed by *StPPS* best model.

TNG Telemetry Data

We used a less complex *PPS* models (see table 3.5) as suggested by the experiments on *TNG* data with *PPSRM* and *PPSPR*. The results are depicted in figure 3.6 and table 3.6. For which concerns the stacked model performance using 5−fold and 10−fold cross validation, the trend is similar to *GOODS* catalog results, with better results for 10−fold cross-validation with respect to 5−fold cross-validation, even though the difference between them is reduced.

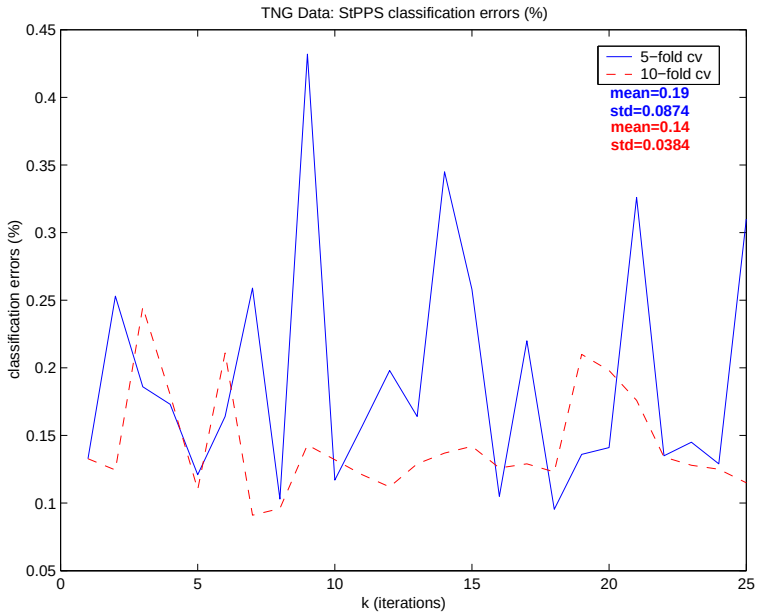

Figure 3.6: *TNG* Data: errors over 25 *StPPS* iterations.

3.3.3 Committee of *PPS* via Bagging: *BgPPS*

The second combining schema here proposed employees bagging as mean to average single density estimators, in our case probabilistic principal surfaces, in a way similar to the model proposed in [53]. All we have to do is to train a number M of *PPS* with M bootstrap replicates of the original learning data set. At the end of this training process, we obtain M different density estimates which are then averaged to form the overall density estimate model. Formally speaking, let D be the original training set of size N and $\{PPS_m\}_{m=1}^{M}$ a set of *PPS* models:

1. create M bootstrap replicates (with replacement) of D, $\{D_{Boot}(m)\}_{m=1}^{M}$ with size N;
2. train each of the M *PPS* models with a bootstrap replicate D_{Boot};
3. at the end of the training we obtain M density estimates $\{PPS_m\}_{m=1}^{M}$;
4. average the M density estimates $\{PPS_m\}_{m=1}^{M}$ as

$$BgPPS(\mathbf{t}) = \frac{1}{M}\sum_{m=1}^{M} PPS_m(\mathbf{t}).$$

As we shall see in the next section looking at the experimental results, we use bagging to build a "bagged" density estimate (possibly, improved) from which computing the posterior class probability.
A natural question using bagging arise, i.e. how many bootstrap replicates we have to use? In [10] Breiman suggested that this number comes from the empirical evidence, and indicated 50 replicates for classification (and 25 for regression) and this number should increase with increasing number of classes. Obviously setting the number of bootstrap replicates faces with the complexity of the models adopted. Neural networks and models like probabilistic principal surfaces require much more training time with respect to other procedures like, for instance, CART used by Breiman. For our model, this number is fixed as a compromise between computational efficiency and accuracy of classification task as it will be showed in the next section.

3.3.4 Experimental Results

For bagging there are two possibilities to build our model. One can decide to use a single *PPS* model with its own parameters setting and to bag it in order to improve its performance. Although this schema employees just one *PPS* model it could still be considered as an ensemble since being the *PPS* learning algorithm based on the *EM* algorithm the Gaussian mixture models differs, since optimization procedure typically terminates in different local minima if different starting points are used. The alternative is to bag a number of different *PPS* models (different α values, variable number of latent nodes and basis function). In our experiments we bag ten *PPS* models (one for each value of $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$) in order to assess the best α value. The *PPS* models are trained on 20 bootstrap replicates of the training data set (hence we have a committee of 20 *PPS* models whose responses are averaged). We average the results on 25 iterations of the algorithm in which a new training\test partition is randomly generated as described in the section 3.3.2 ($60 \setminus 40$ for *Synthetic* and *GOODS* catalogs, $50 \setminus 50$ for *TNG* data).

Parameter	*Value*	*Description*
M	266	number of latent variables
L	102	number of basis functions
L_{fac}	1	basis functions width
iter	100	maximum number of iteration
ϵ	0.01	early stopping threshold

Table 3.7: *Synthetic* Catalog: parameter setting for combined *PPS* via Bagging.

Synthetic Catalog

As almost usual, the parameter setting is shown in table 3.7. On synthetic catalog, bagging performs very well for values of clamping factor α between $[1.0, 2.0]$, where the best mean classification error and standard deviation results are obtained. In particular, for $\alpha = 2.0$ *BgPPS* reaches its minimum mean classification error (0.24) (see figure 3.7 and table 3.8). Table 3.9 shows confusion matrix and corresponding classification error in the best case.

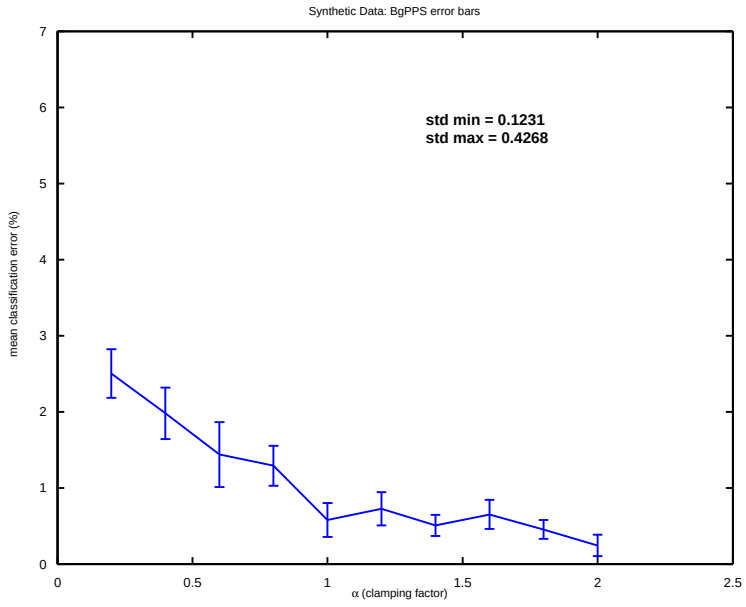

Figure 3.7: *Synthetic* Catalog: error bars for *BgPPS* (errors averaged over 25 iterations for fixed α).

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	2.50	0.3192
0.4	1.98	0.3379
0.6	1.43	0.4268
0.8	1.29	0.2640
1.0	0.57	0.2226
1.2	0.72	0.2186
1.4	0.50	0.1387
1.6	0.65	0.1908
1.8	0.45	0.1231
2.0	**0.24**	**0.1412**

Table 3.8: *Synthetic* Catalog: mean classification error (%) for *BgPPS* (errors averaged over 25 iterations for fixed α).

Classifier type	*Confusion Matrix*			*Best model* α
BgPPS		*Star*	*Galaxy*	2.0
	Star	3996	0	
	Galaxy	4	4000	

Table 3.9: *Synthetic* Catalog: confusion matrix computed by *BgPPS* best model.

Parameter	*Value*	*Description*
M	266	number of latent variables
L	83	number of basis functions
L_{fac}	1	basis functions width
iter	100	maximum number of iteration
ϵ	0.01	early stopping threshold

Table 3.10: *GOODS* Catalog: parameter setting for combined *PPS* via Bagging.

GOODS Catalog

For *GOODS* catalog the results are more fluctuating for each of the α values. In fact the best results are obtained between the interval $[0.2, 0.6]$ and $[1.4, 2.0]$. The overall best result falls in the second interval, in particular for $\alpha = 1.8$ (mean classification error 2.74 and standard deviation 0.3987) even though *BgPSS* with $\alpha = 0.6$ obtains a lower standard deviation value (0.1725). The minimum classification error with confusion matrix is shown in table 3.12.

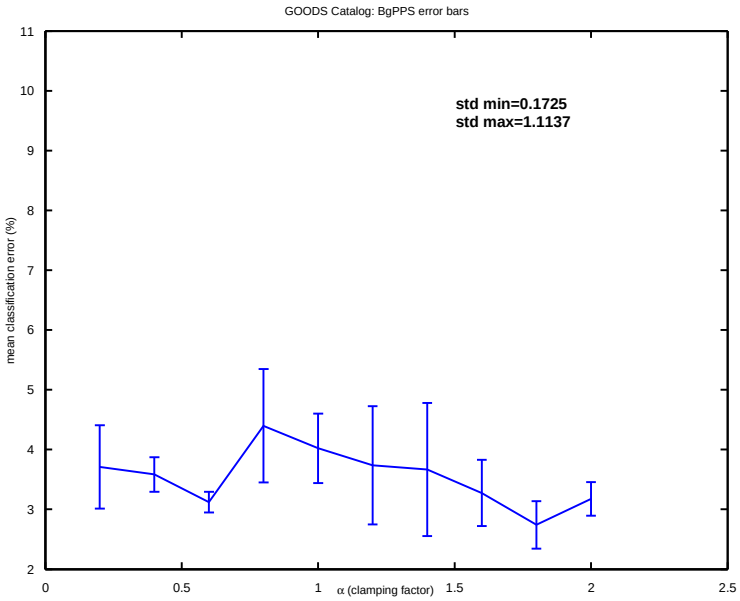

Figure 3.8: *GOODS* Catalog: error bars for *BgPPS* (errors averaged over 25 iterations for fixed α).

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	3.71	0.6973
0.4	3.58	0.2885
0.6	3.12	0.1725
0.8	4.39	0.9490
1.0	4.02	0.5811
1.2	3.73	0.9887
1.4	3.66	1.1137
1.6	3.27	0.5518
1.8	**2.74**	**0.3987**
2.0	3.17	0.2812

Table 3.11: *GOODS* Catalog: mean classification error (%) for *BgPPS* (errors averaged over 25 iterations for fixed α).

Classifier type - Error (%)	*Confusion Matrix*					*Best model* α
		Star	*Galaxy*	*StarD*	*GalaxyD*	
$BgPPS - 2.15$	*Star*	155	35	12	5	1.8
	Galaxy	8	1160	6	8	
	StarD	0	0	64	7	
	GalaxyD	5	51	108	9740	

Table 3.12: *GOODS* Catalog: confusion matrix computed by *BgPPS* best model.

TNG Telemetry Data

For *TNG*, parameter setting is shown in table 3.13. With increasing α values *BgPPS* decreases the obtained mean classification error until $\alpha = 0.8$, then becoming more stable with little fluctuations in the mean classification errors for the remaining values of α. The minimum is reached for $\alpha = 1.4$ (mean classification error equal to 0.12) as can be seen from figure 3.9 and table 3.14. The minimum error and the corresponding confusion matrix is shown in table 3.15. As we shall see in the next section where we compare the performance of each of the model seen so far, *TNG* data is the unique case in which *BgPPS* does not perform better with respect to the other models (in particular with respect to *PPSRM* and *PPSPR*). For this reason a further *BgPPS* model is built whose committee is formed by 25 different *PPS* models each one with the same parameter setting except the values of α which ranges between 0.1 and 4.49 as shown in table 3.16. In substance, we wish to assess if increasing the variability in the *PPS* model components of the committee we gain better results. In table 3.17 are shown the mean classification error and standard deviation over 25 iterations and in table 3.18 the minimum error over the 25 iterations and the corresponding confusion matrix. As expected by introducing more variability in the *PPS* model components we gain something in terms either the mean classification error and standard deviation, even though the difference is very small in mean classification error (0.11% for different α values components against 0.12% for *BgPPS* with best α value) while more sensible in the standard deviation value (0.0118 for different α values components against 0.0225 for *BgPPS* with best α value).

Parameter	*Value*	*Description*
M	33	number of latent variables
L	6	number of basis functions
L_{fac}	1	basis functions width
iter	100	maximum number of iteration
ϵ	0.01	early stopping threshold

Table 3.13: *TNG* Data: parameter setting for combined *PPS* via Bagging.

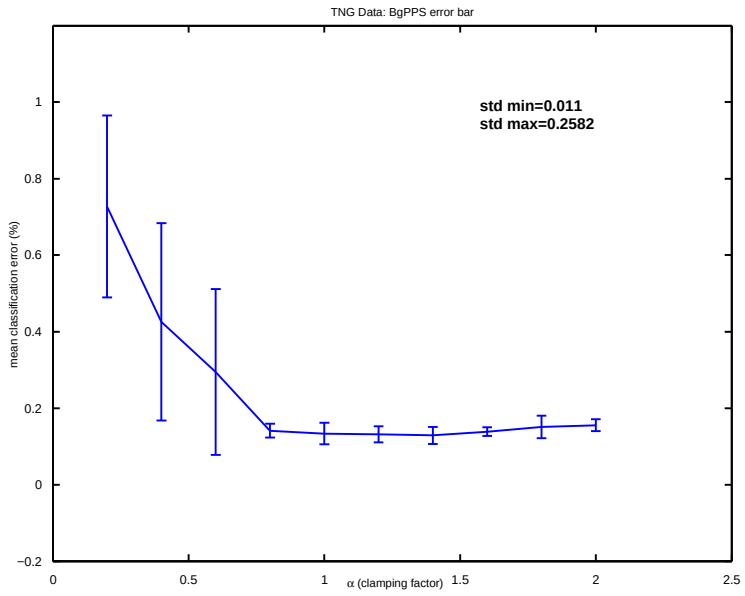

Figure 3.9: *TNG* Data: error bars for *BgPPS* (errors averaged over 25 iterations for fixed α).

α	*Mean Classification Error (%)*	*Standard Deviation*
0.2	0.72	0.2377
0.4	0.42	0.2582
0.6	0.29	0.2170
0.8	0.14	0.0183
1.0	0.13	0.0280
1.2	0.13	0.0209
1.4	**0.12**	**0.0225**
1.6	0.13	0.0110
1.8	0.15	0.0292
2.0	0.15	0.0156

Table 3.14: *TNG* Data: mean classification error (%) for *BgPPS* (errors averaged over 25 iterations for fixed α).

Classifier type - Error (%)	*Confusion Matrix*				*Best model* α
$BgPPS - 0.091$		*Good*	*Medium*	*Bad*	$1.8, 1.4, 1.2, 1.0$
	Good	2230	0	11	
	Medium	0	3680	0	
	Bad	0	0	6129	

Table 3.15: *TNG* Data: confusion matrix computed by *BgPPS* best model.

α	M	L	L_{fac}	$iter$	ϵ
0.1	33	6	1	100	0.01
0.2	33	6	1	100	0.01
0.3	33	6	1	100	0.01
0.4	33	6	1	100	0.01
0.5	33	6	1	100	0.01
0.6	33	6	1	100	0.01
0.7	33	6	1	100	0.01
0.8	33	6	1	100	0.01
0.9	33	6	1	100	0.01
1.0	33	6	1	100	0.01
1.1	33	6	1	100	0.01
1.2	33	6	1	100	0.01
1.3	33	6	1	100	0.01
1.4	33	6	1	100	0.01
1.5	33	6	1	100	0.01
1.6	33	6	1	100	0.01
1.7	33	6	1	100	0.01
2.16	33	6	1	100	0.01
3.33	33	6	1	100	0.01
4.49	33	6	1	100	0.01

Table 3.16: *TNG* Data: parameter setting for combined *PPS* via Bagging (different α values).

Mean	*Std*
0.11	0.0118

Table 3.17: *TNG* Data: *BgPPS* with different α values result (averaged over 25 iterations).

Classifier type - Error(%)	*Confusion Matrix*			
$BgPPS - 0.082$		*Good*	*Medium*	*Bad*
	Good	2230	0	10
	Medium	0	3680	0
	Bad	0	0	6130

Table 3.18: *TNG* Data: confusion matrix computed by *BgPPS* (different α values).

3.3.5 *PPSRM*, *PPSPR*, *StPPS* and *BgPPS* comparison

Having made all the experiments with single *PPS* model classifiers (*PPSRM* and *PPSPR*) and the two committee of *PPS* schemes proposed (*StPPS* and *BgPPS*), what remains to do is to compare them all together. In order to be clear we address the comparison task separately, one for each of the used data sets. For each case two types of figures are shown:

- mean classification error plots for *PPSRM*, *PPSPR* and *BgPPS*, in which the errors are showed for each of the ten α values used;
- bar chart of the best model mean classification errors and standard deviations for *PPSRM*, *PPSPR*, *StPPS* and *BgPPS*. Obviously, saying best model we mean the best model α value only for *PPSRM*, *PPSPR* and *BgPPS*, whereas for *StPPS* we only have the model result over 25 iterations (since in *StPPS* we employee a committee of *PPS* with different α values).

Synthetic Catalog

As can be seen from figure 3.10 *BgPPS* outperforms either *PPSR* and *PPSPR* for near all the values of clamping factor α used. Moreover, from figure 3.11 is clear that *BgPPS*

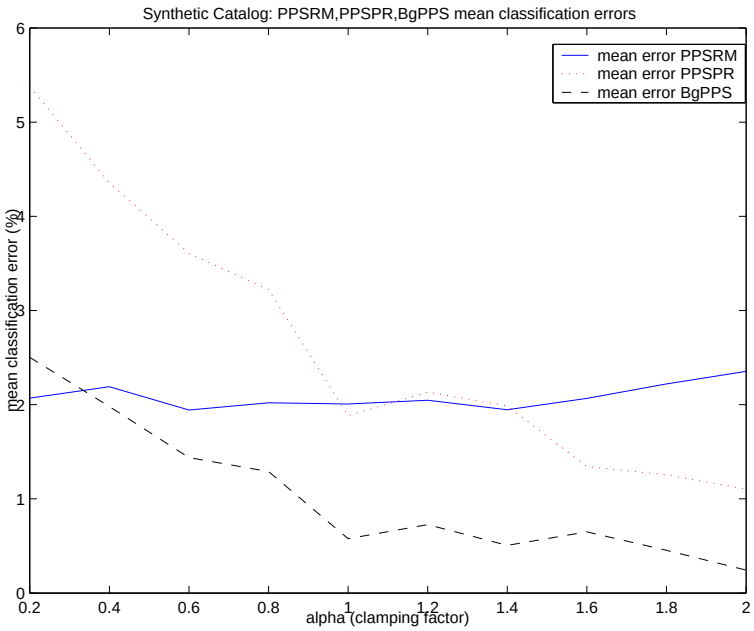

Figure 3.10: *Synthetic* Catalog: mean errors for *PPSRM*, *PPSPR* and *BgPPS* (errors averaged over 25 iterations for fixed α).

outperforms *StPPS*. This latter model performs better of the single model classifiers on average, even though *PPSPR* for just one α value reaches a better result.

GOODS Catalog

GOODS catalog classification task is more complex (we shall discuss this issue from a graphical point of view in the next chapter). This is evident from the results obtained by the different classifiers used. However, even in this case *BgPPS* outperforms all the other models (*PPSRM*, *PPSPR* and *StPPS*). Moreover, Stacked *PPS* here outperforms both *PPSRM* and *PPSPR*. Among the two single *PPS* classifier models, *PPSPR* is still better than *PPSRM* (see figures 3.12 and 3.13).

TNG Data

TNG data in this thesis, provides the simpler classification task. In fact, the classes (we shall see in the next chapter) are well separated, due to the features chosen but, above

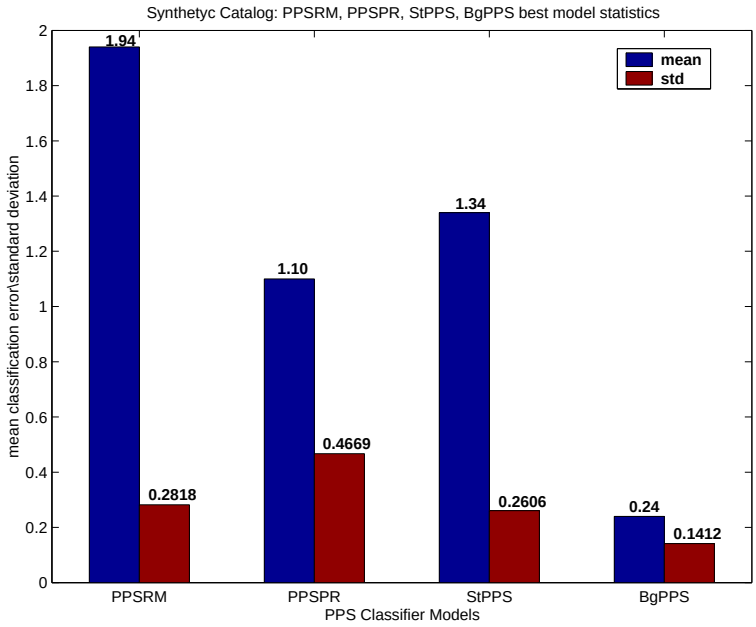

Figure 3.11: *Synthetic* Catalog: bar chart for *PPSRM*, *PPSPR*, *StPPS* and *BgPPS* best models statistics (averaged over 25 iterations).

all, for the limited number of observational sessions used. Actually, the problem is not so easy but more reliable results will be obtained when more observational sessions for each quality class (*Good*, *Medium* and *Bad*) will be available. This is to justify the fact that *PPSRM* outperforms very clearly all the remaining *PPS* classifier models as it can be seen from figures 3.14 and 3.15. *BgPPS* performs better than *PPSPR* and *StPPS*, but the differences are smaller with respect to the previous data sets. We tried to improve *BgPPS* performances by adopting a committee built with *PPS* models with different clamping factors α effectively improving only a little bit the performances. In conclusion it can be stated that the committees of *PPS* perform better than single PPS, eve though this is clear for the ensemble of *PPS* built via bagging. Stacked *PPS*, instead, has less stable results but it seems a promising combining schema after all, since we did few experiments by varying the *PPS* component complexity. We rather focused, on the impact of cross-validation which appears as of primary importance. Last consideration is about the complexity of committee combining schema, which is computationally expensive, since we have to train a number of different *PPS* models. Bagged *PPS*, anyway is less expensive with respect to

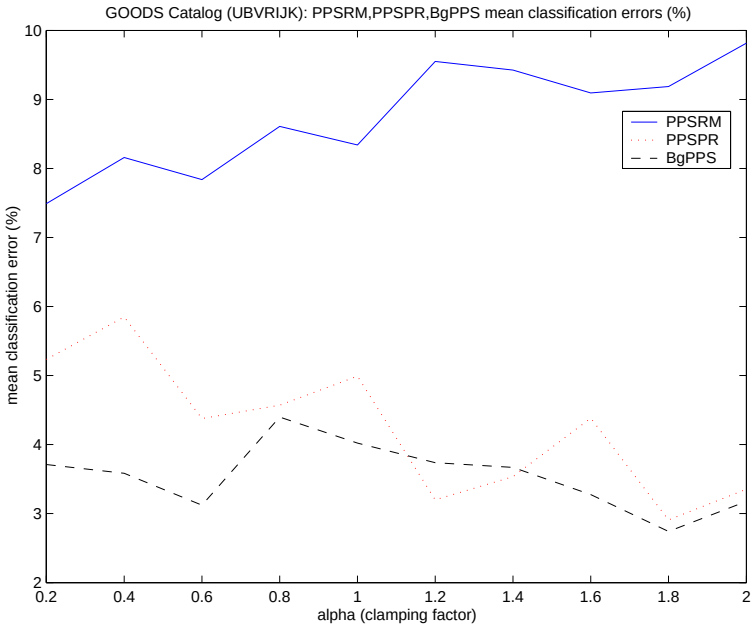

Figure 3.12: *GOODS* Catalog: mean errors for *PPSRM*, *PPSPR* and *BgPPS* (errors averaged over 25 iterations for fixed α).

stacked *PPS*, primarily when a high number o folds in cross-validation for stacked *PPS* is used. Nevertheless, the search for the best model in the case of the single *PPS* classifiers, might involve the training of a considerable number of *PPS*.

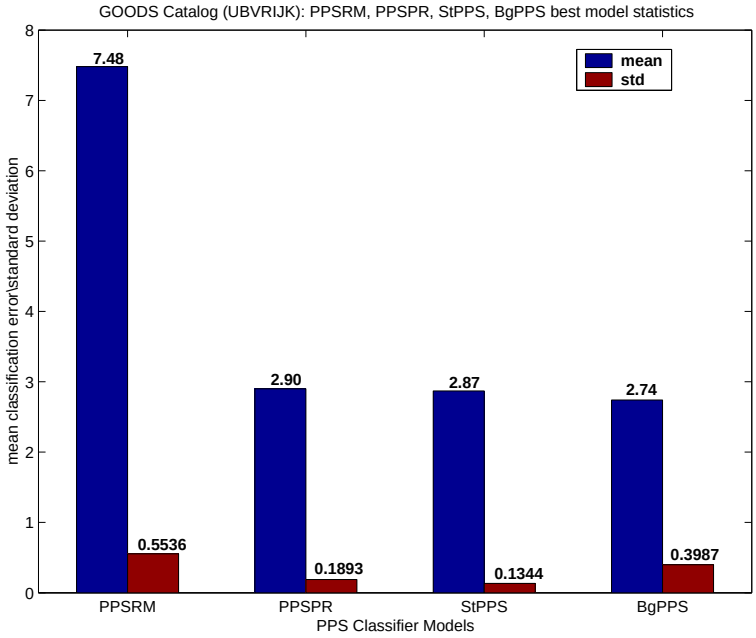

Figure 3.13: *GOODS* Catalog: bar chart for *PPSRM*, *PPSPR*, *StPPS* and *BgPPS* best models statistics (averaged over 25 iterations).

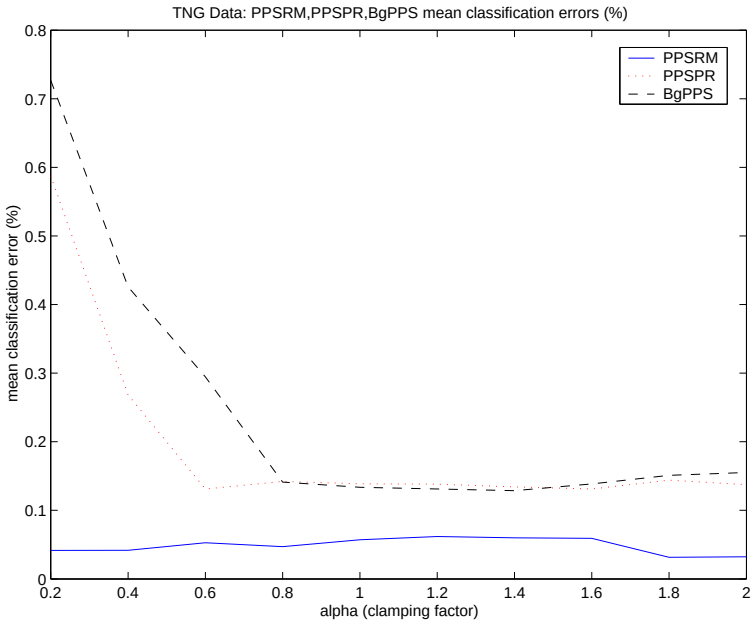

Figure 3.14: *TNG* Data: mean errors for *PPSRM*, *PPSPR* and *BgPPS* (errors averaged over 25 iterations for fixed α).

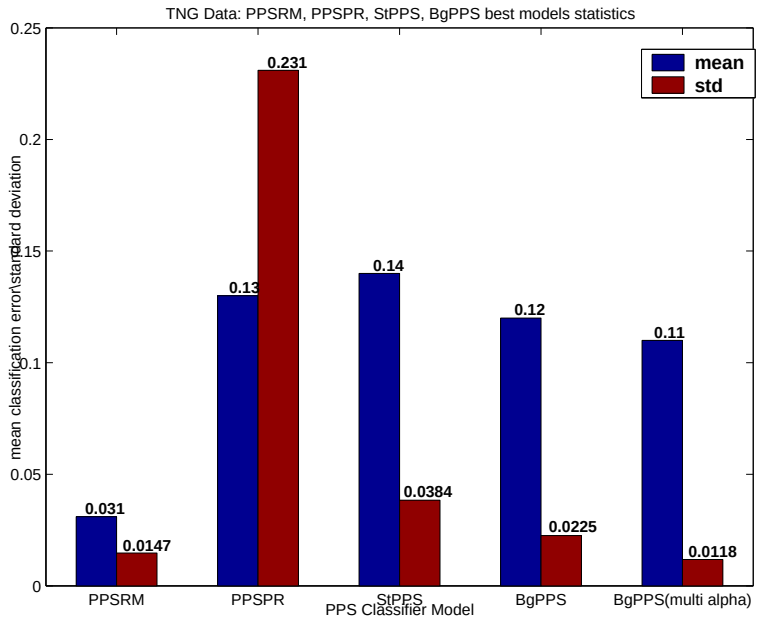

Figure 3.15: *TNG* Data: bar chart for *PPSRM*, *PPSPR*, *StPPS*, *BgPPS* and *BgPPSma* best models statistics (averaged over 25 iterations).

Chapter 4

Spherical PPS Data Visualization

In this chapter an overview of the visualization possibilities offered by the PPS framework is provided. Next, we describe the visualization capabilities added to the system. Finally, we give a brief introduction of our developed easy-to-use graphical user interface which integrates all the functionalities described, then the overall visualization possibilities are illustrated for each of the used data sets (*Synthetic*, *GOODS* and *TNG*).

4.1 Visualizations offered by Spherical Probabilistic Principal Surfaces

As already mentioned in section 2.2.3, the spherical manifold can be used as an unsupervised high$-D$ data visualization tool. After a *PPS* model is fitted to the data, the data themselves are projected into the latent space as points onto a sphere. The latent manifold coordinates $\hat{\mathbf{x}}_n$ of each data point $\mathbf{t}_n$ are computed as

$$\hat{\mathbf{x}}_n \equiv \langle \mathbf{x}|\mathbf{t}_n \rangle = \int \mathbf{x} p(\mathbf{x}|\mathbf{t}) d\mathbf{x} = \sum_{m=1}^{M} r_{mn} \mathbf{x}_m \tag{4.1}$$

and these coordinates lie within a unit sphere. In figure 4.1 an example of such a projection is sketched. From the figure it is clear that the visualization appears confused in the crowded areas because the data points lying on the opposite sides of the sphere are overlapped. Therefore, as we shall see in the next paragraphs, we draw a unit sphere under the data so that the data lying on the opposite side is hidden to the user, who can, eventually, rotate the sphere to look at all the data interactively. Obviously this means that data points lying in the volume of the sphere must be projected on the surface in

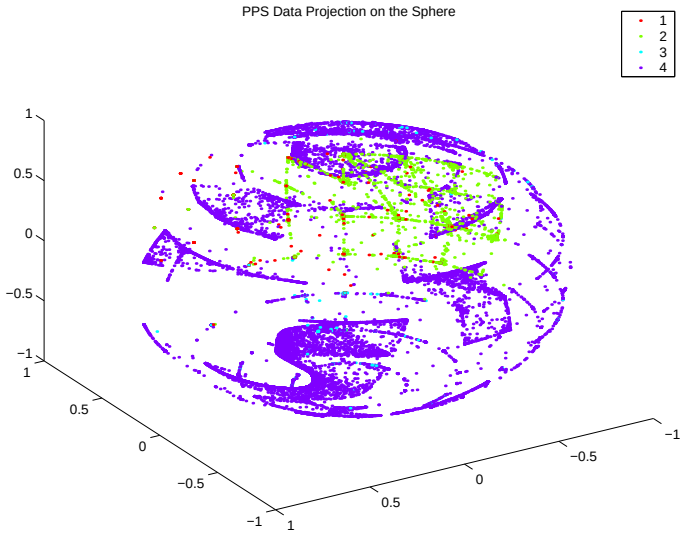

Figure 4.1: A typical data projection on a sphere in the latent space. As it can be seen, even though this representation is already better with respect to other visualization (i.e, PCA) and useful for a first investigation on the data, the data lying on the opposite sides of the sphere can be confused when this regions are particularly crowded.

order to be properly visualized. This is the only type of visualization possible in the latent space. Eventually, one can show the manifold shape into the input space along with the input data in two or at most three dimensions. Obviously the projection of input data into latent space is useful to gain some insights into the class shapes, the overlap amount between classes and so on. However, for data mining goals these visualizations do not appear an efficient tool since there is no interaction at all between the user and the data. For example, for the astronomers it is of primary importance to have the chance to select a data point or a group of data points and to know to what pattern it corresponds in the original catalog in order to make inference about the similarity or dissimilarity between objects or to establish the nature of such a group (Stars, Galaxy, etc.). All these requests can be satisfied by the visualization options we are going to describe in the next section.

4.2 Further visualization capabilities added to PPS

Basically, our aim is to allow the user to:

- easily interact with the data into the latent space, hence with the data onto the sphere in several ways,
- to visualize the data probability density in the latent space so giving a first understanding about the clusters in the data,
- finally to fix a number of clusters and visualize it. Eventually, at the end of this option one could still interact with the data by selecting data points in a given cluster and make a number of comparisons.

4.2.1 Interactively selecting points on the sphere

Having projected the data onto the latent sphere, it is advisable for a data analyzer to localize the most interesting data points (obviously, this depends on the application at hand), for example the ones lying far away from more dense areas, or the ones lying in the overlapping regions between clusters, and to gain some information about them, by linking the data points on the sphere with their position in the catalog which contains all the information about the typology of the data. Eventually, if the images corresponding to

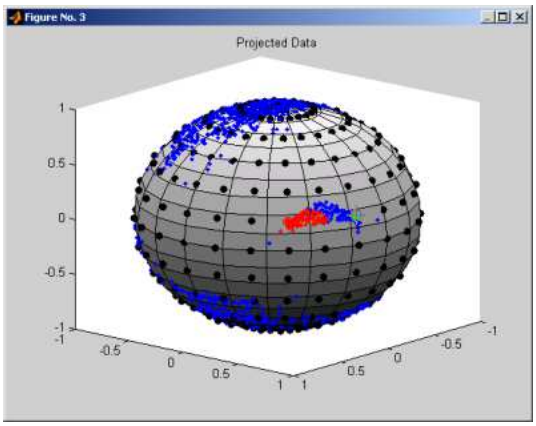

Figure 4.2: Data points selection phase. The bold black circles represent the latent variables; the blue points represent the projected input data points. While selecting a latent variable, each projected point for which the variable is responsible is colored. By selecting a data point the user is provided with information about it: coordinates and index corresponding to the position in the original catalog.

the catalog would be available to the user then he can visualize the object in the original image which corresponds to the data point selected onto the sphere. These possibilities are fundamental for the astronomers who may be able to extract important meanings from the data and for all the data mining activities. In figure 4.2 the functionality just described is depicted. Furthermore, the user is also allowed to select a latent variable and coloring all the points for which the latent variable is responsible.

4.2.2 Visualizing the latent variable responsibilities on the sphere

The only projections of the data points onto the sphere provide only partially information about the clusters inherently present in the data: if the data are strongly overlapped the data analyzer can not derive any information at all. A first insight on the number of agglomerate localized onto the spherical latent manifold is provided by the mean of the responsibility for each latent variable. In details, from equation 4.1, we saw that each latent variable onto the spherical manifold has an associate value which measures its amount of responsibility for the overall input data points. Therefore, if we build a spherical

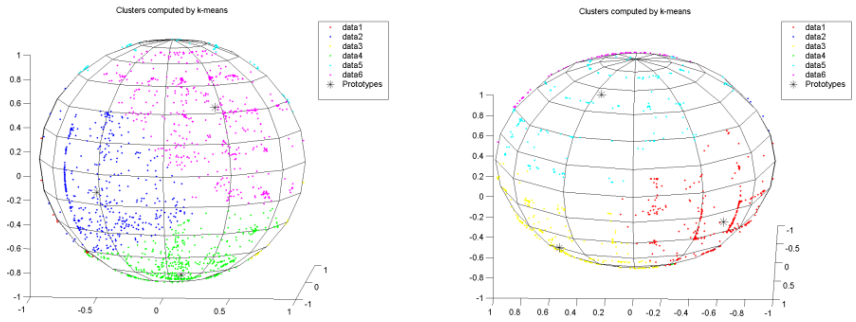

Figure 4.3: Clusters computed by k-means on the spherical latent manifold (left) opposite side of the same sphere (right).

manifold which is composed by a set of faces each one delimited by four vertices (and these latter corresponding to latent variables) we can then color each face with colors varying in intensity on the base of the values of the responsibility associate to each vertex (and hence, to each latent variable). The overall result is that the sphere will contain regions more dense with respect to other and this information is easily visible and understandable. Obviously, what can happen is that a more dense area of the spherical manifold might contain more than one cluster, and this can be validated by further investigations.

4.2.3 A method to visualize clusters on the sphere

Once the user or a data analyzer has an overall idea of the number of clusters on the sphere, he can then exploit this information through the use of classical clustering techniques (such as hard or fuzzy k-means [3]) to find out the prototypes of the clusters and the data therein contained. This task is accomplished by running the clustering algorithm on the projected data. In general, since the data points lie in the volume of the unit spherical manifold and not necessarily on the surface, we need, after the clustering algorithm run, to project each computed prototype onto the surface of the spherical latent manifold. Afterwards, one may proceed by coloring each cluster with a given color (see figure 4.3).

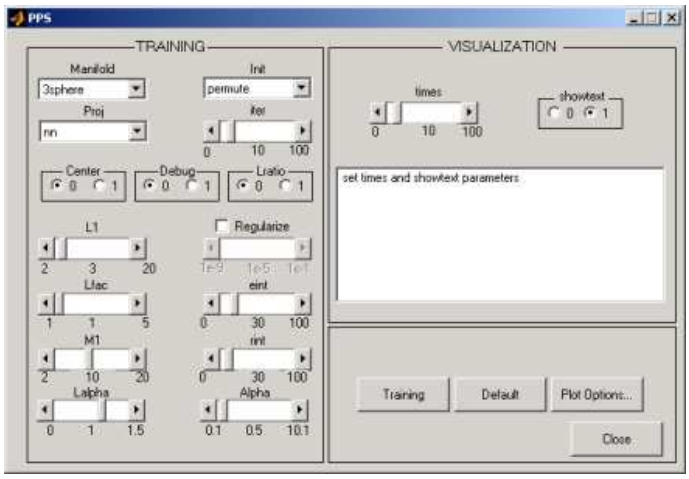

Figure 4.4: The PPS Graphical user interface main window. In the left panel the parameter of the PPS are listed, while in the right panel are shown a text window for the training results, and the buttons for starting the training and the plot options.

4.3 An easy to use interface to *PPS*

The visualization options so far described have been integrated in a user-friendly graphical user interface which provides a unified tool for the training of the *PPS* model, and next, after the completion of the training phase, to accomplish all the functions for the visualization and the investigation of the given data set. Figure 4.4 shows the main interface from which it is possible to carry on the setting of the *PPS* parameters, and from which one can select the plotting options. These plotting options, are grouped in a single vertical toolbar (see figure 4.5). As already done with the classification algorithms described in the previous sections, all the software was implemented in the Matlab computing Environment exploiting and adapting the LANS Matlab Toolbox. This software was used to make all the experiments we are going to describe in the next section.

4.3.1 *Synthetic* Catalog Visualizations

The visualizations showed in the following are computed with the best model *PPS* derived during the classification tasks described in chapter 2. Figure 4.6 shows three different visualizations for the synthetic catalog, namely, $3-D$ *PCA* visualization, the *SOM* U-matrix

Figure 4.5: The plot bar to start the plotting options.

and the spherical *PPS* projections. Recall, that the *SOM* U-matrix [67], visualizes the clustering structure of the *SOM* as distances between neighboring map units: high values of the U-matrix indicate a cluster border, uniform areas of low values indicate clusters themselves. *PPS* projections onto the spherical latent manifold appear far and away more readable than *PCA* where all the data appear as a unique overlapped agglomerate (except a little isolated group), and than *SOM* U-matrix which provides the same information of *PCA*, a large unique cluster: there is a large agglomerate of data points in which, by rotating the sphere, it is possible to localize two main clusters divided by a little region of less dense data points. In figure 4.7 the *PPS* projections with class labels and the corresponding latent variable probability density function are shown. By rotating the sphere with density, two high density regions are highlighted with other few lower density regions. This visualization result confirms the a priori knowledge we have about the data set and the good classification performance exhibited by the *PPS* in chapters 2 and 3. Finally, figure 4.8 displays the variable probability densities from class *Star* and *Galaxy*, respectively, where it is worth noting how these densities are quite different and, hence, representative of these classes.

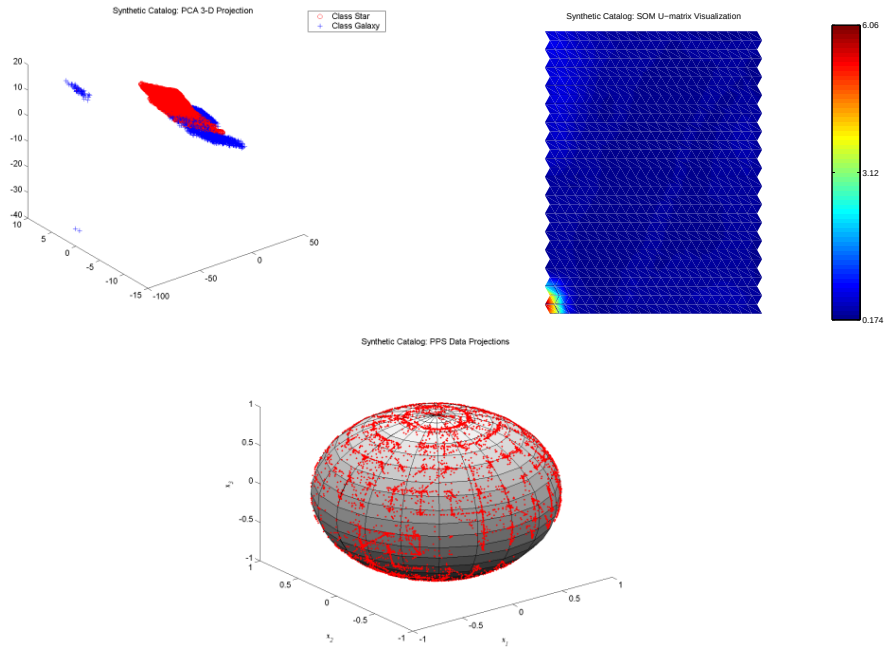

Figure 4.6: *Synthetic* Catalog - clockwise from upper left: $3-D$ *PCA* visualization corresponding to the 3 largest eigenvectors; *SOM* U-Matrix (grid size: 32×22); Projections onto *PPS* latent manifold.

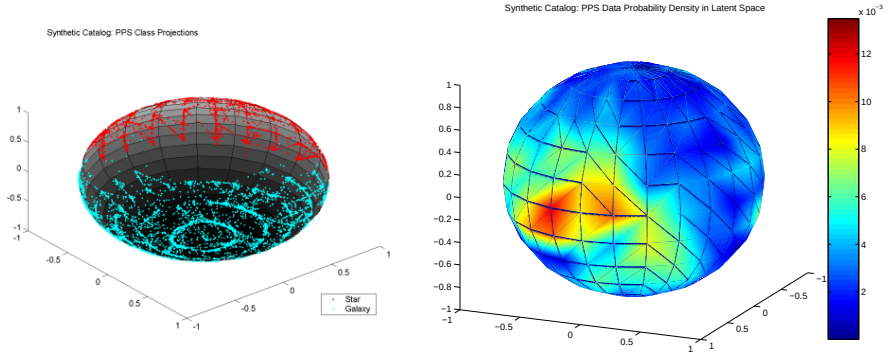

Figure 4.7: *Synthetic* Catalog: (left) input data point projections with class labels (right) the corresponding probability density onto the latent manifold.

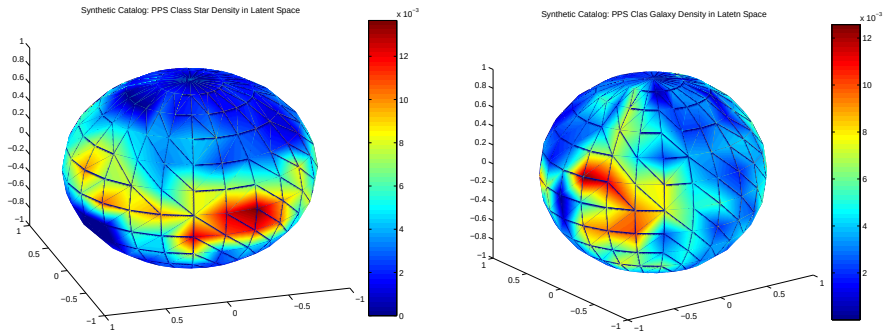

Figure 4.8: *Synthetic* Catalog: (left) class *Star* probability density (right) class *Galaxy* probability density.

4.3.2 *GOODS* Catalog Visualizations

As already told, the GOODS catalog, is a very complex data set which exhibits four strongly overlapping classes. In fact, as it can be seen from figure 4.9, the *PCA* and *SOM* visualizations give no interesting information at all, since they display only a single large group of data. In *PCA*, the class *GalaxyD* (whose objects are yellow colored), which contains the majority of objects (about 24000) is near totally hidden. The *PPS* projections, instead, show a large group consisting of the objects belonging to *GalaxyD* class and overlapping objects of the remaining classes and a well bounded group of *Galaxy* class objects (see the *PPS* projections with and without labels in figure 4.9). It is meaningful to compute the PPS manifold for each one of the 4 classes. The projections onto the latent manifold are displayed in figure 4.10 while in figure 4.11 the corresponding latent variable probability densities are shown. Note, especially, how different these densities appear for each class. It is clear, from these visualizations the reason for which the *PPS* classification performances for the GOODS catalog obtained in chapters 2 and 3 are worse than performances obtained with the synthetic catalog.

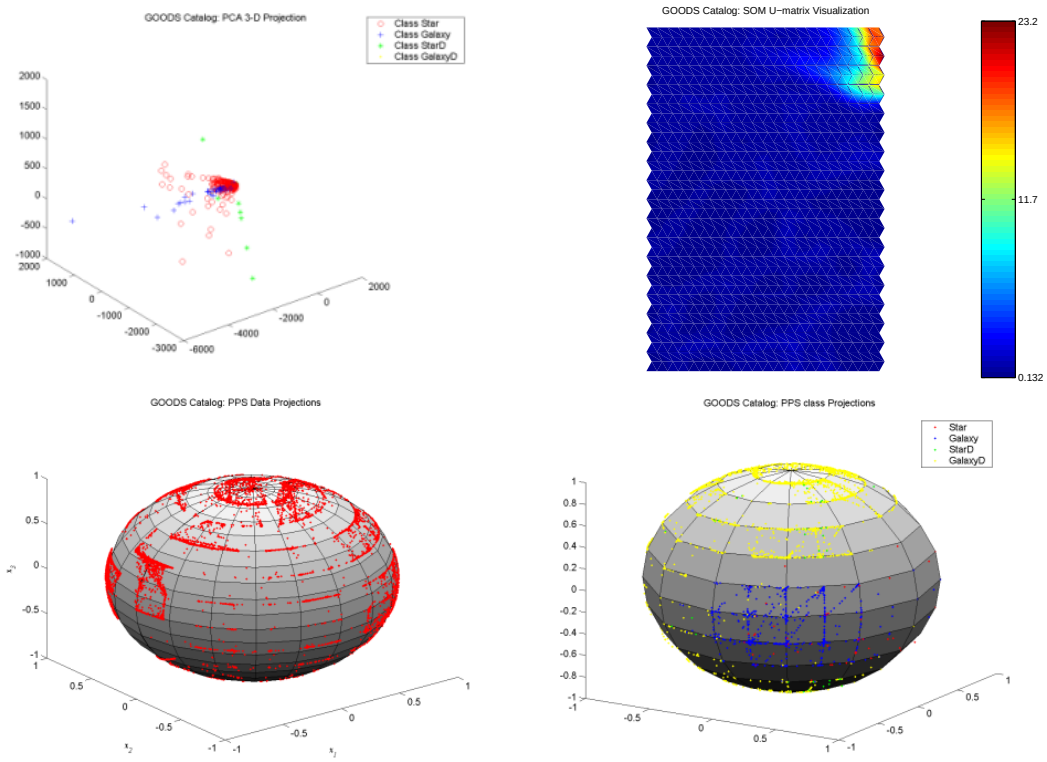

Figure 4.9: *GOODS* Catalog - Clockwise from upper left: $3-D$ *PCA* visualization corresponding to the 3 largest eigenvectors; *SOM* U-matrix (grid size: 37×28); Projections onto the *PPS* latent manifold with class labels and Projections onto *PPS* latent manifold without class labels.

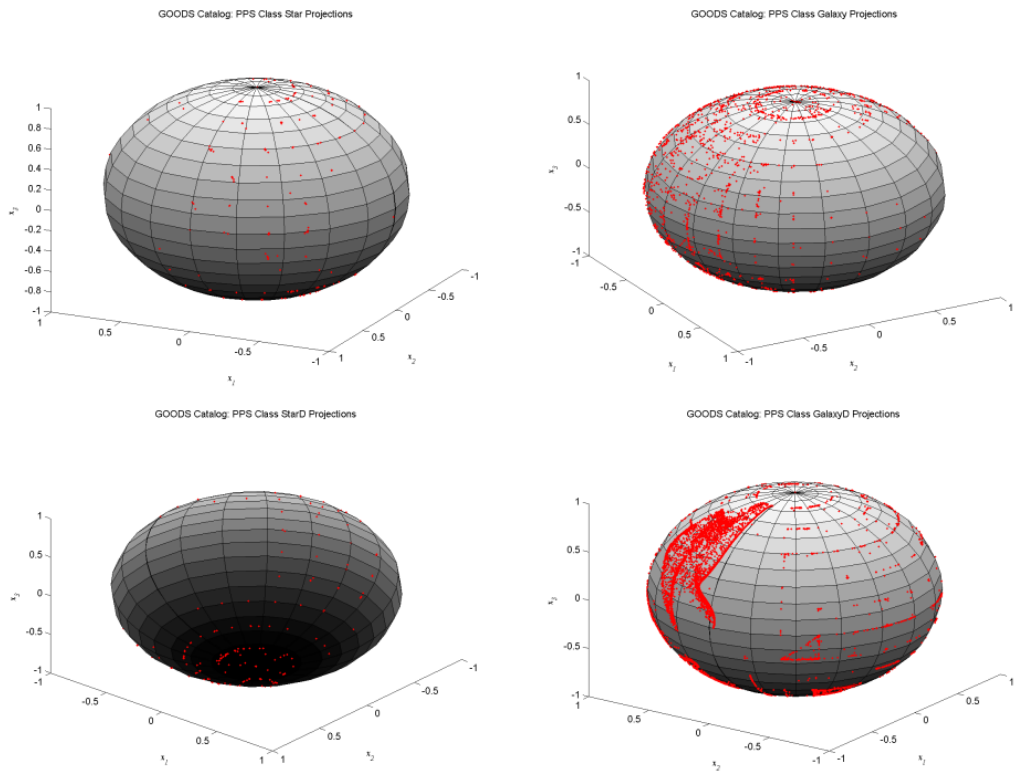

Figure 4.10: *GOODS* Catalog - Clockwise from upper left: input data point projections onto the sphere for classes *Star*, *Galaxy*, *GalaxyD* and *StarD*.

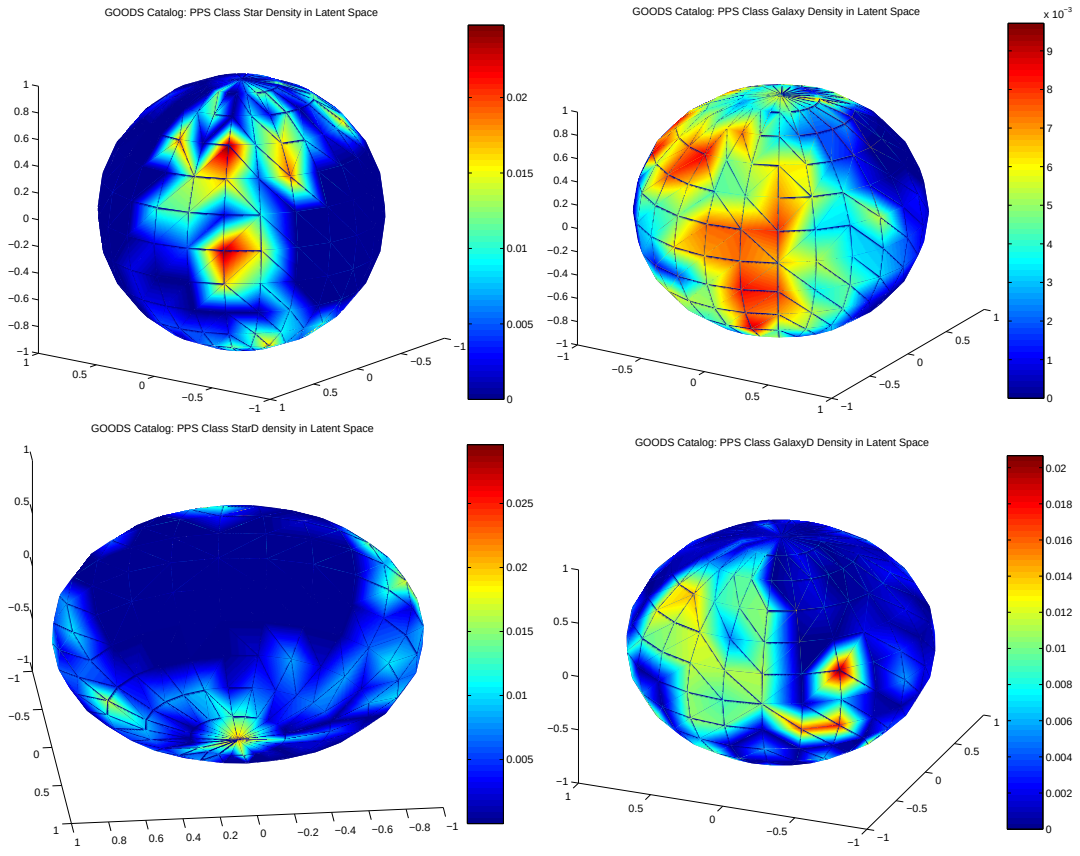

Figure 4.11: *GOODS* Catalog - Clockwise from upper left: probability density functions into the latent space for classes *Star*, *Galaxy*, *GalaxyD* and *StarD*.

4.3.3 *TNG* Data Visualizations

Looking at the plots in figure 4.12 it is clear that the *TNG* data represent a simpler problem for *PPS* and for *PCA* and *SOM* as well. The data appears well separated, and this explains the almost perfect classification results described in chapters 2 and 3. It is interesting to note how, in particular in the *PPS* projections, the data points are superimposed each other (recall that the entire *TNG* data set is composed by 24118 objects). This is due to the parameter selection phase which caused each object belonging to the same class, to have near the same values in all the parameters. The important thing to note here is that each class has its own parameter configuration enough different each other. Figure 4.13 displays the *PPS* data projections with class labels and the corresponding probability density in the latent space. In this latter plot 3 high density regions are evident (by rotating the sphere) which correspond to the 3 data classes. For each class the corresponding latent variable responsibility is also plotted (figure 4.14). For the *TNG* data it is of primary importance to establish the influences of each parameter, therefore, for this aim, a *PPS* model was trained on the data by sequentially eliminating the *Azimuth*, the *Azimuth* and the *Elevation*, and *Azimuth*, *Elevation* and *Rotator* position, respectively, in order to assess their weight in the obtained results, which are depicted in figures 4.15 and 4.16. This preliminary results indicate that the only *Azimuth* has a little influences for discriminating between classes (the corresponding *PPS* projections and responsibility plots are very similar with respect the use of all the parameters) while eliminating the *Elevation* and the *Rotator* position led to lightly different projections on the sphere and latent variable responsibility. At first glance *Elevation* and *Rotator* position seem to have the same importance. As stressed in section 3.3.5, for *TNG* data, there is the need to have more observational sessions with several diversified cases for each kind of class in order to gain more reliable results.

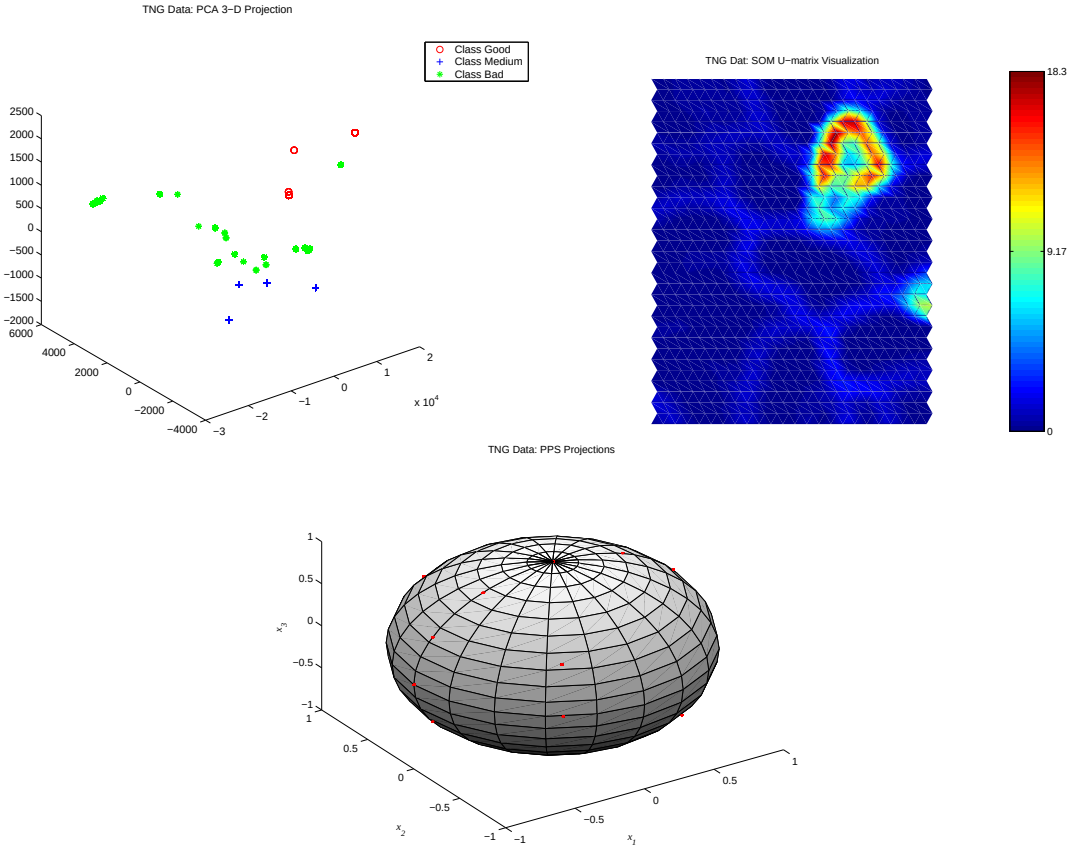

Figure 4.12: *TNG* Data - Clockwise from upper left: $3-D$ *PCA* visualization corresponding to the 3 largest eigenvectors, *SOM* U-Matrix (grid size: 33×24) and *PPS* projections.

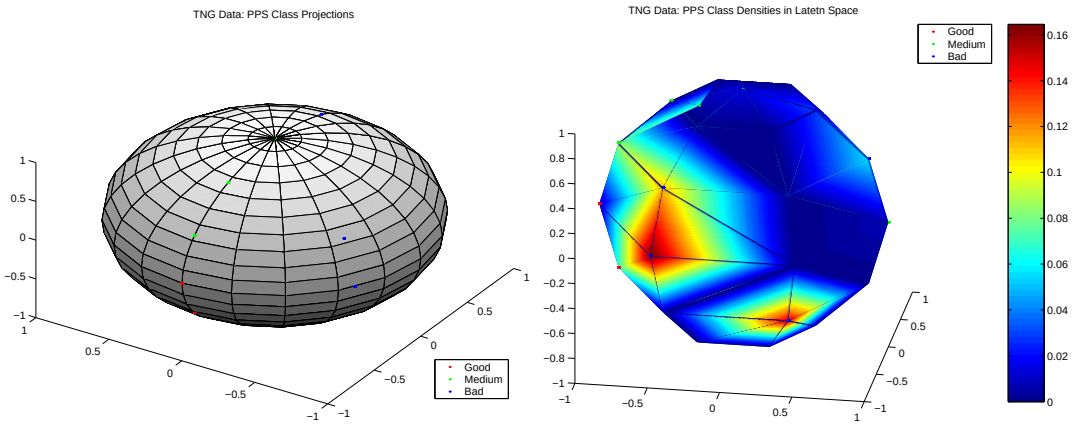

Figure 4.13: *TNG* Data: (left) *PPS* class projections and (right) latent variable responsibilities.

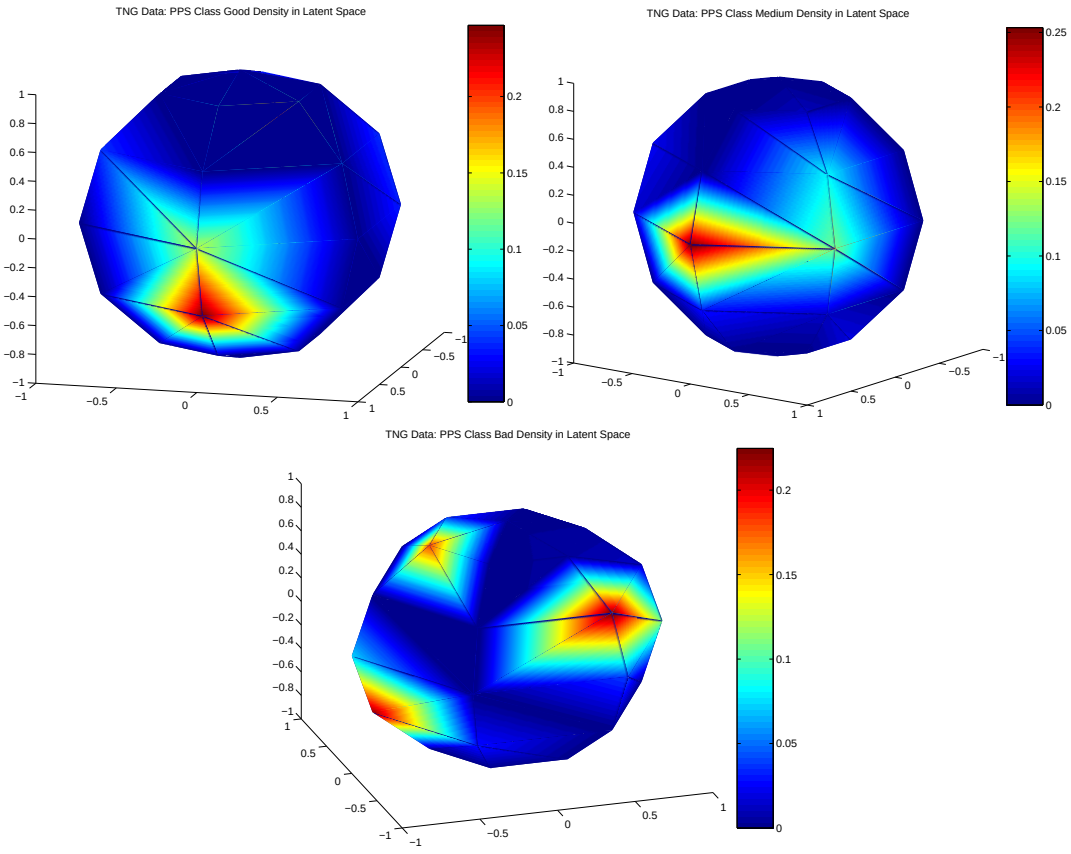

Figure 4.14: *TNG* Data - Clockwise from upper left: latent variable responsibilities for classes *Good*, *Medium* and *Bad*.

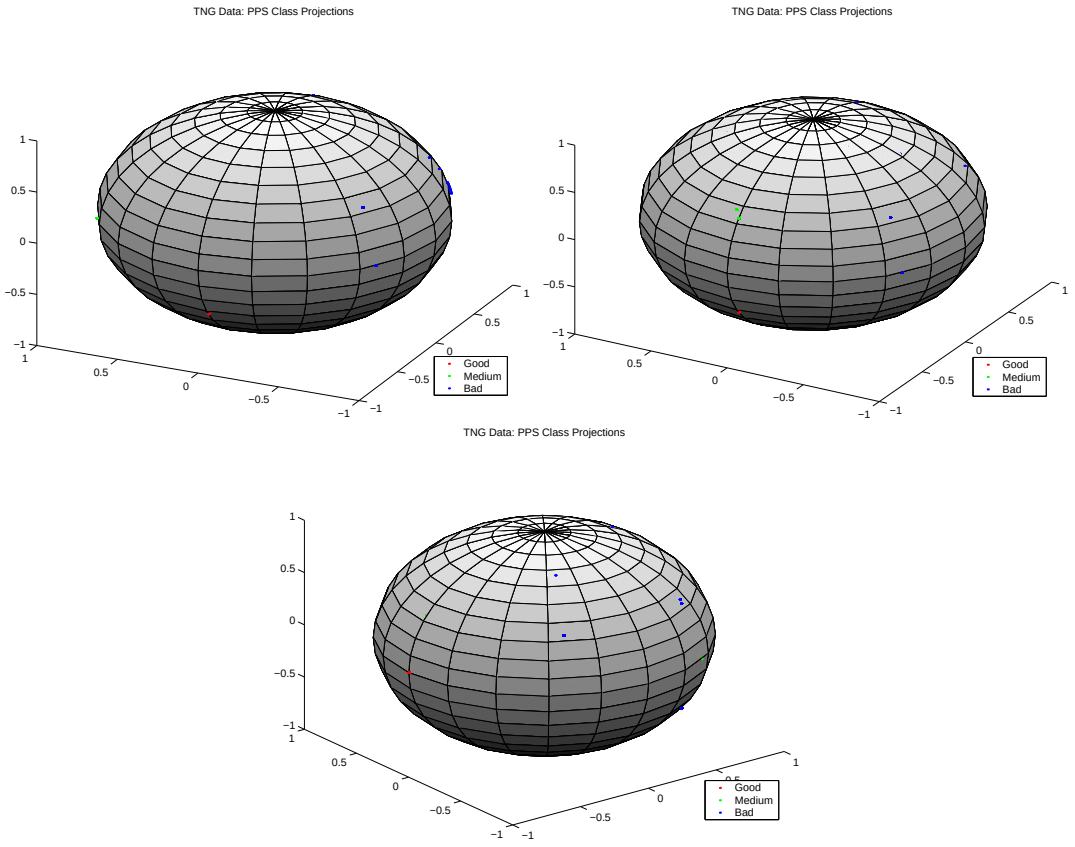

Figure 4.15: *TNG* Data - clockwise from upper left: class projections with all parameters minus the *Azimuth*, all parameters minus *Azimuth* and *Elevation* and all parameters minus *Azimuth*, *Elevation* and *Rotator* position.

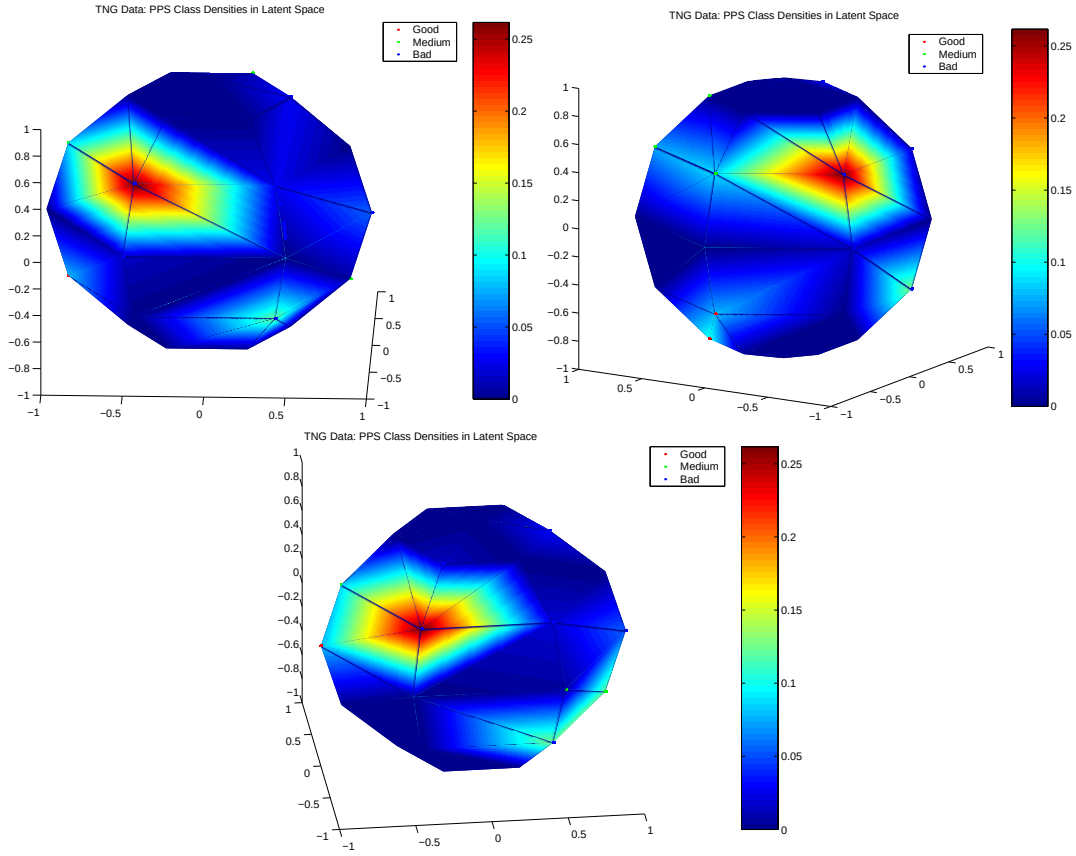

Figure 4.16: *TNG* Data - clockwise from upper left: latent variable responsibilities with all parameters minus the *Azimuth*, all parameters minus *Azimuth* and *Elevation* and all parameters minus *Azimuth*, *Elevation* and *Rotator* position.

Chapter 5

Conclusions

This thesis studied in depth two well known nonlinear latent variable models, namely the Generative Topographic Mapping (GTM) and the Probabilistic Principal Surfaces (PPS), showing how the latter model seems to be the most flexible and efficient in several data mining activities, specially for high-D data classification and visualization. Above all, the spherical PPS, which consists of a spherical latent manifold lying in a three dimensional latent space, is better suitable to high-D data since the sphere is able to capture the sparsity and periphery of data in large input spaces which are due to the curse of dimensionality. Nevertheless, it was also shown that PPS may be enhanced both in classification and visualization tasks:

***PPS* for Classification** PPS builds a probability density function in the input data space which is composed by a mixture of Gaussians whose parameters, fixed the log-likelihood function, are derived through the Expectation-Maximization algorithm (EM). However, specially for high-D data, the EM algorithm is inherently unstable, due to the singularity of the log-likelihood function and for the local optima. Therefore, we developed a committee of PPS taking inspiration by [53], to compute more accurate density models which are averaged over the single density models computed by the PPS components. The way this is done is through two diverse combining schemes, i.e. Bagging and Stacking. By computing the posterior class probability, as expected, the classification performance improved substantially. This work enlarges the area of committee machines applied to unsupervised learning algorithm and for density estimation, usually a less developed field with respect to the supervised case.

***PPS* for Visualization** In [17], some examples of spherical PPS visualization are provided, and while it appeared very appealing, it is not enough for large data mining

applications. Hence, the basic functionalities provided in the *PPS* framework, have been enriched with a number of visualization options which are proved to be very effective for the interpretation of the data at hand:

- Interactive selection of regions of sample points projected onto the sphere for further analysis. This is particularly useful to profile groups of data.
- Visualization of the latent variable responsibilities onto the sphere as a colored surface plot. Specially, useful to localize more and less dense areas to find out a first number of clusters present into the data, and to highlight the regions where lies outliers.
- A method to exploit the information gathered with the previous visualization options through a clustering algorithm to find out the clusters with the corresponding prototypes and data points.

Both the classification and visualization tasks have been proved effective in a complex application domain: astronomical data analysis. Astronomy is a very rich field for a computer scientist due to the presence of a very huge amount of data. Therefore, every day there is the need to resort to efficient methods which often are neural networks-based. In all the used astronomical data sets, a synthetic one, and two real world data sets, the committee of *PPS* classifier performed very well, far and away outperforming the standard methods usually adopted by astronomers. Furthermore, in its own way the spherical *PPS* for visualization represents the first tool for astronomical data mining which gives to the astronomers the possibility to easily interact with the data. Although the study of the methods addressed in this dissertation is devoted to the astronomical applications, the system is general enough to be used in whatever data-rich field to extract meaningful information.

5.1 Future developments

There are many way to further develop the models described in this thesis. In the following we propose two directions:

1. In committee of *PPS* via staking, for example, one way to enhance the stacked model is to make the coefficients α_i, $i = 1, \ldots, M$ dependent on the data as suggested in

[59] as done in the hierarchical mixture of experts model [42]. So doing, the input data space is partitioned into regions for which a *PPS* component is responsible.

2. To build a hierarchical *PPS* for constructing localized nonlinear projection manifold as already done for *GTM* [65] and previously for a linear latent variable model [8]. Following [65], a hierarchy of PPS could be organized in a tree whose root corresponds to the PPS model trained on the entire data set at hand, and whose nodes, built interactively in a top-down fashion, represent *PPS* models trained in localized regions of the data input chosen in the ancestor plot *PPS* by the user, interactively. In all the sub-models one might exploit all the visualization options developed in this thesis.

Chapter 6

Appendix

6.1 Astronomical Data Sets used in the thesis

6.1.1 Stars/Galaxies Synthetic data

The catalogs contain 10000 simulated objects each. The photomteric information (i.e. the magnitudes) for each object was obtained by the convolution of some template spectral energy distributions (SEDs) with the chosen photometric system (i.e. the filters). Our simulated observation is performed with VLT filters (FORS + ISAAC instruments) therefore our fake observations are very deep.

In table 6.1 the completeness magnitudes for each filter (i.e. the maximum magnitude at which we expect to see all objects in a specific filter and to have an error of about 0.1) are listed. The photometric errors are introduced as Gaussian noise plus a zeropoint error, added quadratically, that takes into account any systematic error.

The STARS catalog

The stellar spectra flux library we used was published by Pikles [55] and consists of 131 flux-calibrated spectra, encompassing all normal spectral types and luminosity classes at solar abundance, and metal-weak and metal-rich F-K dwarf and G-K giant components. Each spectrum of the library was formed by combining data from several sources overlapping in wavelength coverage. The library has a complete spectral coverage from 1150 to 10620 Angstrom for all the components and to 25000 Angstrom for about half of them, mainly later types of solar abundance. Because we need to have photometry from ultraviolet (UV) to near infrared (NIR), i.e. from 3000 to 20000 Angstrom, this library is an ideal

Filter	s/n	Mag
U	10	27
B	10	27.5
V	10	27.5
R	10	27.5
I	10	28
J	10	24
H	10	24
Ks	10	24

Table 6.1: Completeness magnitudes for each filter

tool for our goal. The simulated sources were selected in the R band as those having a R magnitude in the range between 22 and 25, no reddening was applied and a minimum error of 0.05 was quadratically added.

The GALAXIES catalog

The construction of a fake galaxies catalog is a more tricky job in the sense that more paremeters are involved. For this purpose we used the Bruzual - Charlot code [13]. The code, $GISSEL98$ (Galaxy Isochrone Synthesis Spectral Evolution Library), provides spectral synthesis models. The stellar population synthesis models are based on stellar tracks libraries: then, a spectral energy distribution is assigned to all stars on the evolutionary tracks. Then, the Initial Mass Function (IMF) and the Star Formation Rate (SFR) must be specified to follow the evolution of the integrated spectrum. In a few words, the IMF specifies the distribution in mass of newly formed stellar population and the SFR says how many stars are formed as function of time, the birthrate of stars. While for the IMF the standard Miller and Scalo law [50] is adopted, for the SFR one has to choose different laws in order to end up with different types of galaxies. We have choosen different SFRs in order to have all the different galaxies spectra we know in nature: from Ellipticals to the Irregular Starbursts passing trough all kind of Spiral Galaxies. The sample is selected again in the R band as those having a R magnitude in the range between 22 and 25.

All the galaxies are formed at a redshift of 10 and randomly taken in the redshift range from 0 to 3 to fill up the catalog. A Lambda CDM in considered for the age (redshift) estimate with $H0 = 70$ (Hubble constant), $OMEGA_M = 0.3$ (matter density parameter) and $OMEGA_V = 0.7$ (Lambda density parameter). A reddening, due to extragalactic absorption, in the form of the Calzetti law is considered [15]. A minimum error of 0.05 was quadratically added. Two aspects are worth noting:

1. The number counts are obviously in no way representative of the real number counts one could obtain from a real survey (GOODS), in the sense that you will never end up with the same number of galaxies and stars in a catalog for a normal extragalactic survey;

2. Because of the tricky job of creating a galaxy catalog, there are a lot of different possible solutions: this is only one of the many. However, the important thing in our case is to have simulated objects having different physical properties as it happens in the two catalogs.

6.1.2 GOODS Catalog

The Great Observatories Origin Deep Survey (GOODS)[1], covers at several wavelengths the Chandra Deep Field South (CDF-S) [56]. The available catalogs provide photometric multi-wavelength data reduced to a common system: optical broad band UBVRI photometry obtained using the Wide Field Imager (WFI) at the ESO/MPG $2.2m$ telescope, near infrared (JHK) photometry obtained with the SOFI imager at the ESO/NTT. Additional information on the X and radio fluxes are also available but since they will not be used here, we shall neglect them. Object catalogues were extracted using the package S-Extractor [2] from each co-added image were combined in a multi-color list: the UBVRIJK catalogue contains more than 28000 sources (WFI + SOFI) in an area of approximately 0.25 square degrees. One of the main problems is posed by the so called "dropouts", namely objects which are below the detection threshold in at least one of the available bands. This problem is especially relevant in the GOODS catalog due to the very different sensitivities of the instruments (24872 in the UBVRIJK catalog). Table 6.2 shows the parameters used to build the GOODS catalog.

[1] http://www.stsci.edu/science/goods

No.	Parameters	Description
1	SeqNr	Object sequence number
2	ALPHA-J2000	Right ascension of barycenter (J2000)
3	DELTA-J2000	Declination of barycenter (J2000)
4	FLUX-AUTO-B842	Flux within a Kron-like elliptical aperture
5	FLUX-AUTO-I845	Flux within a Kron-like elliptical aperture
6	FLUX-AUTO-J998	Flux within a Kron-like elliptical aperture
7	FLUX-AUTO-K999	Flux within a Kron-like elliptical aperture
8	FLUX-AUTO-R844	Flux within a Kron-like elliptical aperture
9	FLUX-AUTO-U877	Flux within a Kron-like elliptical aperture
10	FLUX-AUTO-V843	Flux within a Kron-like elliptical aperture
11	KRON-RADIUS-B842	Kron apertures in units of A or B
12	KRON-RADIUS-I845	Kron apertures in units of A or B
13	KRON-RADIUS-J998	Kron apertures in units of A or B
14	KRON-RADIUS-K999	Kron apertures in units of A or B
15	KRON-RADIUS-R844	Kron apertures in units of A or B
16	KRON-RADIUS-U877	Kron apertures in units of A or B
17	KRON-RADIUS-V843	Kron apertures in units of A or B
18	MAG-ISO-B842	std
19	MAG-ISO-I845	std
20	MAG-ISO-J998	std
21	MAG-ISO-K999	std
22	MAG-ISO-R844	std
23	MAG-ISO-U877	std
24	MAG-ISO-V843	std
25	CLASS STAR	S/G class: 1 point source, 0 extended

Table 6.2: Parameters used in the *UBVRIJK GOODS* Catalog

6.1.3 Telescopio Nazionale Galileo Telemetry Data

TNG is provided with three mirrors (M_1, M_2, M_3) whose primary mirror (M_1) has 78 axial actuators and 24 lateral supports. Moreover, *TNG* (see paragraph 1.4.1) is equipped with five instruments which are permanently operating on his foci and offers a large variety of observing modes covering the optical and near infrared wavelength ranges and spanning from broad band imaging to high resolution spectroscopy.
The Long Term Archive of the Telescopio Nazionale Galileo (*TNG-LTA*) contains both the raw data and the telemetry data collecting a wide set of monitored parameters such as, for instance, the atmospheric and dome temperatures, the operating conditions of the telescope and of the focal plane instruments, etc.
The images come from two different optical instruments (namely, *Dolores* and *OIG*), and are divided into five different observational sessions. Some sessions have strongly elliptical images maybe due to bad tracking or incorrect aberrations. However, generally the quality of the images provided by *TNG* is better. We can now list in details the five sessions:

EUIB images acquired by Dolores, in imaging mode. The images belong to good quality images.

EVNJ images acquired by Dolores, in imaging mode. Medium quality images.

EXOH Images acquired by Dolores in imaging mode. Images belonging to bad quality class. In fact, they are very elliptical as a differential tracking was used (since the target was fixed to a nearby object, i.e. asteroids or comets, and therefore TNG had a tracking as well as a relative speed).

FISE Images acquired by *OIG*. Medium-bad quality images, in which the causes are unknown.

FJDX Images acquired by *OIG*. Medium-bad quality images, in which the causes are unknown.

However, in the experiments described in the thesis, we only used the first three sessions. We extracted from the *TNG-LTA* a set of 278 telemetric data monitored (for a total of 35.000 epochs) during the acquisition of almost 50 images. The images were then randomly examined in order to assess their quality and build a set of labels (we shall limit

Parameters	*Number*
Δ actuators M_1	78
Gravitational field strength lateral actuators	4
Gravitational field strength M_1 actuators	78
M_2 bars extent	6
M_3 X-axes position	1
M_3 Y-axes position	1
M_3 Z-axes position	1
CCD temperature	1
Encoder read Azimuth	1
Encoder read elevation	1
Encoder read rotator position	1

Table 6.3: TNG parameters

ourselves to the case of images with bad tracking (elongated PSF), medium tracking and good tracking (round PSF). From the starting list of 278 parameters we extracted 172 parameters ignoring the ones less significative by eye inspection. These parameters are described in table 6.3

Moreover, a further parameter reduction is accomplished by reducing only the groups of parameters listed in table 6.3 which contains more than one parameter as follows:

Δ actuators M_1: the 78 actuators are divided in four group of 15 elements and one group of 18 elements. For each group the mean is computed, and the obtained five means represent the parameters used for training.

Gravitational field strength lateral actuators: the mean of 4 parameters is computed.

Gravitational field strength M_1 actuators: the same as Δ actuators M_1 preprocessing is carried on this group of parameters.

M_2 bars extent: the mean of 6 parameters is computed.

M_3 **X-axes position,** M_3 **Y-axes position,** M_3 **Z-axes position:** these parameters are grouped in their mean.

Therefore the parameter used for the training of the *PPS* models are reduced to 17 plus the label.

References

[1] C.A.L. Bailer-Jones, R. Gupta, H.P. Singh, *Automated Data Analysis in Astronomy*, In Gupta (Ed.), Astro-ph/0102224, 2001

[2] E. Bertin, S. Arnouts, *SExtractor: Software for Source Extraction*, Astronomy and Astrophysics Supplement, 117, 393-404, 1996

[3] J.C. Bezdek, J. Keller, R. Krisnapuram, N.R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publisher, 1999

[4] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

[5] C.M. Bishop, M. Svensén, and C. K. I. Williams,*GTM: a principled alternative to the Self-Organizing Map*, In C. von der Malsburg, W. von Selen, J. C. Vorbruggen, and B. Sendhoff (Eds.), International Conference on Artificial Neural Networks, ICANN'96, Springer, 1997

[6] C. M. Bishop, M. Svensen, C.K.I. Williams, *GTM: The Generative Topographic Mapping*, Neural Computation, 10(1), 1998.

[7] C.M. Bishop, M. Svensén, and C. K. I. Williams, *Developments of the Generative Topographic Mapping*, Neurocomputing 21,1998.

[8] C.M. Bishop and M.E. Tipping, *A hierarchical latent variable model for data visualization*, IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 281293,1998

[9] C.M. Bishop, *Latent variable models*, In M. I. Jordan (Ed.), Learning in Graphical Models, pp. 371403, MIT Press, 1999.

[10] L. Breiman, *Bagging Predictors*, Machine Learning, 26, 1996

[11] L. Breiman, *Combining Predictors*, in Combining Artificial Neural Nets, A.J.C. Sharkey (Ed.), Springer, 31, 1999

[12] R.J. Brunner, S.G. Djorgovski, T.A. Prince, A.S. Szalay, *Massive Datasets in Astronomy*, in The Handbook of Massive Datasets, 2001.

[13] G. Bruzual, S. Charlot, ApJ, 405, 538 NASA ADS, 1993

[14] R.J. Bullen, D. Cornford, I.T. Nabney, *Outlier Detection in Scatterometer Data: Neural Networks Approaches*, Neural Networks, Special Issue on Applications of Neural Networks to Astrophysics and Geosciences, R. Tagliaferri, G. Longo, D'Argenio B. (Eds.), 2003

[15] D. Calzetti, L. Armus, R.C. Bohlin et al., ApJ, 533, 682, NASA ADS, 2000

[16] K. Chang, J. Ghosh, *Probabilistic Principal Surfaces*, Proc. IEEE International Joint Conference on Neural Networks, 1999

[17] K. Chang, *Nonlinear Dimensionality Reduction Using Probabilistic Principal Surfaces*, PhD Thesis, Department of Electrical and Computer Engineering, The University of Texas at Austin, USA, 2000

[18] K. Chang, J. Ghosh, *A unified Model for Probabilistic Principal Surfaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, NO. 1, 2001

[19] A.P. Dempster, N.M. Laird, D.B. Rubin, *Maximum-Likelihood from Incomplete Data Via the EM Algorithm*, J. Royal Statistical Soc., Vol. 39, NO. 1, 1977

[20] T.G. Dietterich, *Ensemble Methods in Machine Learning*, In J. Kittler and F. Roli (Eds.), Multiple Classifer Systems. First International Workshop, MCS 2000, Cagliari, Italy, Vol 1857 of Lecture Notes in Computer Science, Springer-Verlag, 2000

[21] T.G. Dietterich, *Ensemble Learning*, The Handbook of Brain Theory and Neural Networks, Second Edition, M.A. Arbib (Ed.), Cambridge, MA: The MIT Press, 2002

[22] S.G. Djorgovski, R.J. Brunner, A.A. Mahabal, S.C. Odewahn et al., *Exploration of Large Digital Sky Surveys*, Mining the Sky, Proc. of the MPA/ESO/MPE Workshop, A.J. Banday, S. Zaroubi, M. Bartelmann (Eds.), 2000.

[23] S.G. Djorgovski, A.A. Mahabal, R.J. Brunner, S.C. Odewahn et al., *Searchs for Rare and New Types of Objects*, Virtual Observatory of the Future, ASP Conference Series, Vol 225, R.J. Brunner, S.G. Djorgovski and A.S. Szalay, (Eds.), 2001

[24] P. Domingos, *A Unified Bias-Variance Decomposition for Zero-One and Squared Loss*, Proceedings of the Seventeenth National Conference on Artificial Intelligence (pp. 564-569), Austin, TX: AAAI Press, 2000

[25] D. Draper, *Assessment and Propagation of Model Uncertainty*, Journal of the Royal Statistical Society,B, 57, 1995

[26] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, John Wiley and Sons, 2001

[27] B. Efron, R. Tibshirani, *An introduction to the Bootstrap*, Chapman and Hall, 1993

[28] U. Fayyad, P. Smyth, *From Massive Data Sets to Science Catalogs: Applications and Challenges*, Proc. Workshop on Massive Data Sets, J. Kettenring and D. Pregibon (Eds.), Committe on Applied and Theoretical Statistics, 1995

[29] U. Fayyad, D. Haussler, P. Storloz, *Mining Science Data*, Communications of the ACM 39 (11), 1996.

[30] U. Fayyad, D. Haussler, P. Stolorz, *KDD for Science Data Analysis: Issues and Examples*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96), Menlo Park, CA, AAAI, Press, 1996

[31] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, *From Data Mining to Knowledge Discovery: An Overview*, in AKDDM, AAAI/MIT Press, 1996

[32] U. Fayyad, *Taming the Giants and the Monsters: Mining Large Databases for Nuggets of Knowledge*, Database Programming and Design Magazine, March 1998.

[33] Y. Freund, R.E. Shapire, *A Decision-Theorethic Generalization of on-line Learning and an Application to Boosting*, In Proceedings of the Second European Conference on Computational Learning Theory, Springer-Verlag, 1995

[34] J.H. Friedman, *An overview of predictive learning and function approximation*, From Statistics to Neural Networks, Proc. NATO/ASI Workshop, V. Cherkassky, J.H. Friedman, and H. Wechsler (Eds.), Springer Verlag, 1994

[35] J. Friedman, *On Bias, Variance, 0-1 Loss and the Curse of Dimensionality*, J. Data Mining and Knowledge Discovery, 1, 1997

[36] S. Geman, E. Bienenstock, R. Doursat, *Neural Networks and the Bias-Variance Dilemma*, Neural Computation, Vol 4, 1, 1992

[37] C.R. Genovese, L. Wasserman, R.C. Nichol, A.J. Connolly et al., *Nonparametric Density Estimation: A Brief and Selective Review*, Virtual Observatory of the Future, ASP Conference Series, Vol 225, R.J. Brunner, S.G. Djorgovski and A.S. Szalay, (Eds.), 2001

[38] R.K. Gulati, L. Altamirano, *Artificial Neural Networks in Stellar Astronomy*, In Anguilar (Ed.), Focal Points in LAtin American Astronomy, vol. 85, Revista Mexicana de Astronomia y Astrofisica Serie de Conferencias, 2001

[39] T. Hastie, W. Stuetzle, *Principal Curves*, J. Am. Statistical Assoc., Vol. 84, NO. 406, 1988

[40] T. Heskes, *Bias-Variance decompositions for likelihood-based estimators*, Neural Computation, 10:1425-1433, 1998

[41] G. James, T. Hastie, *Generalizations, of the bias-variance decomposition for prediction error*, Technical report, Department of Statistics, Stanford University, 1997

[42] M.I. Jordan, R.A. Jacobs, *Hierarchical Mixture of Experts and the EM Algorithm*, Neural Computation, 6, 181-214, 1994

[43] T. Kohonen, *Self-organized formation of topologically correct feature maps*, Biological Cybernetics, 43, 1982

[44] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin, 1995

[45] A. Krogh, J. Vadelsby, *Neural Network Ensembles, corss-validation, and active learning*, in Advances in Neural Information Processing Systems 7, G. Tesauro, D.S. Touretzky and T.K. Leen (Eds.), MIT Press, 1995

[46] M. LeBlanc, R. Tibshirani, *Adaptive Principal Surfaces*, J. Am. Statistical Assoc., Vol. 89, NO. 425, 1994

[47] G. Longo, R. Tagliaferri, A. Staiano et al., *Advanced data mining tools for exploring large astronomical databases*, Proc. SPIE 2001, San Diego USA, vol. 4477, pp. 61-75, 2001

[48] G. Longo, R. Tagliaferri, A. Staiano et al., *Artificial Intelligence tools for data mining in large astronomical databases*, ESO/ESA/NASA Conference, Steps Towards an International Virtual Observatory, Garching, June 2002

[49] G. Longo, R. Tagliaferri, A. Staiano et al., *Data Mining of large astronomical databases with neural tools*, Proc. SPIE 2002, Waikoloa Hawai, vol. 4847-49, pp. 265-276, 2002

[50] G.E. Miller, J.M. Scalo, ApJS, 41, 513, 1979

[51] I.T. Nabney, *Netlab: Algorithms for Pattern Recognition*, Springer-Verlag, 2002

[52] R.C. Nichol, A.J. Connolly, C.R. Genovese, L. Wasserman et al., *Computational AstroStatistics: Fast Algorithms and Efficient Statistics for Density Estimation in Large Astronomical Datasets*, Virtual Observatory of the Future, ASP Conference Series, Vol 225, R.J. Brunner, S.G. Djorgovski and A.S. Szalay, (Eds.), 2001

[53] D. Ormoneit, V. Tresp, *Averaging, Maximum Likelihood and Bayesian Estimation for Improving Gaussian Mixture Probability Density Estimates*, IEEE Transaction on Neural Networks, Vol.9, NO. 4, 1998

[54] M.P. Perrone, *Improving regression estimates: averaging methds for variance reduction with extensions to general convex measure optimization*, PhD Thesis, Brown University, 1993

[55] A.J. Pikles, PASP 110, 863, 1998

[56] P. Rosati, P. Tozzi, R. Giacconi et al., Astrophys. J. Suppl., 139, 369-410, 2002

[57] S. Rosset, E. Segal, *Boosting Density Estimation*, In Proceedings of 16th International Conference on Neural Information Processing Systems (NIPS), Vancouver, Canada, 2002

[58] R.E. Shapire, *The Strength of Weak Learnibility*, Machine Learning 5(2), 197-227, 1990

[59] P. Smyth, D.H. Wolpert, *An evaluation of linearly combining density estimators via stacking*, Machine Learning, Vol. 36, 1999.

[60] A. Strehl. J. Ghosh, *Cluster Ensembles - A Knowledge reuse Framework for Combining Multiple Partitions*, Journal of Machine Learning Research, 3, 2002

[61] M. Svensén, *GTM: The Generative Topographic Mapping*, PhD thesis, Aston University, Birmingham, UK, 1998

[62] R. Tagliaferri, G. Longo, A. Staiano et al., *Neural Networks in Astronomy*, Neural Networks, Special Issue on "Neural Network Analysis of Complex Scientific Data: Astronomy and Geosciences", R. Tagliaferri, G. Longo, D'Argenio B. (Eds.), 2003

[63] M. Taniguchi, V. Tresp, *Averaging Regularize Estimators*, Neural Computation,9, 1163, 1997

[64] R. Tibshirani, *Principal Curves Revisited*, Statisitcs and Computing, Vol. 2, 1992

[65] P. Tino, I. Nabney,*Hierarchical GTM: constructing localized non-linear projection manifolds in a principled way*, IEEE Transactions on Pattern Analysis and Machine Intelligence, in print

[66] G. Valentini, F. Masulli, *Ensembles of Learning Machines*, in M. Marinaro, R. Tagliaferri (Eds.), Neural Nets, 13th WIRN, Vietri sul Mare (Sa), Italy, Vol 2486 of Lecture Notes in Computer Science, Springer-Verlag, 2002

[67] J. Vesanto, *Data Mining Techniques based on the Self-Organizing Maps*, PhD Thesis, Helsinki University of Technology, 1997

[68] A. Weingessel, E. Dimitriadou, K. Hornik, *An Ensemble Method for Clustering*, Working Papers, Conference on Distributed Statistical Computing, 2003

[69] J. Welling, M. Derthick, *Visualization of Large Multi-Dimensional Datasets*, Virtual Observatory of the Future, ASP Conference Series, Vol 225, R.J. Brunner, S.G. Djorgovski and A.S. Szalay, (Eds.), 2001

[70] D.H. Wolpert, *Stacked Generalization*, Neural Networks, 5, 241, 1992

[71] D.H. Wolpert, *On bias plus variance*, Neural Computation,9, 1997

[72] Z.-H. Zhou, W. Tang, *Clusterer ensemble*, Technical Report, AI Lab, Computer Science and Technology Department, Nanjing University, Nanjing, China, 2002