Chapter 1

Introduction

The subjects of this thesis are unsupervised learning in general, and principal curves in particular. It is not intended to be a general survey of unsupervised learning techniques, rather a biased overview of a carefully selected collection of models and methods from the point of view of principal curves. It can also be considered as a case study of bringing a new baby into the family of unsupervised learning techniques, describing her genetic relationship with her ancestors and siblings, and indicating her potential prospects in the future by characterizing her talents and weaknesses. We start the introduction by portraying the family.

1.1 Unsupervised Learning

It is a common practice in general discussions on machine learning to use the dichotomy of supervised and unsupervised learning to categorize learning methods. From a conceptual point of view, supervised learning is substantially simpler than unsupervised learning. In supervised learning, the task is to guess the value of a random variable Y based on the knowledge of a d-dimensional random vector $\mathbf{X}$. The vector $\mathbf{X}$ is usually a collection of numerical observations such as a sequence of bits representing the pixels of an image, and Y represents an unknown nature of the observation such as the numerical digit depicted by the image. If Y is discrete, the problem of guessing Y is called *classification*. Predicting Y means finding a function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(\mathbf{X})$ is close to Y where "closeness" is measured by a non-negative cost function $q(f(\mathbf{X}), Y)$. The task is then to find a function that minimizes the expected cost, that is,

$$f^* = \arg\min_f E[q(f(\mathbf{X}), Y)].$$

In practice, the joint distribution of $\mathbf{X}$ and Y is usually unknown, so finding f^* analytically is impossible. Instead, we are given $\mathcal{X}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$, a sample of n independent and

identical copies of the pair $(\mathbf{X}, Y)$, and the task is to find a function $\hat{f}_n(\mathbf{X}) = \hat{f}(\mathbf{X}, \mathcal{X}_n)$ that predicts Y as well as possible *based on the data set* $\mathcal{X}_n$. The problem is well-defined in the sense that the performance of a predictor $\hat{f}_n$ can be quantified by its *test error*, the average cost measured on an independent test set $\mathcal{X}'_m = \{(\mathbf{X}'_1, Y'_1), \ldots, (\mathbf{X}'_m, Y'_m)\}$ defined by

$$q(\hat{f}) = \frac{1}{m} \sum_{i=1}^{m} q(\hat{f}(\mathbf{X}'_i), Y'_i).$$

As a consequence, the best of two given predictors $\hat{f}_1$ and $\hat{f}_2$ can be chosen objectively by comparing $q(\hat{f}_1)$ and $q(\hat{f}_2)$ on a sufficiently large test sample.

Unfortunately, this is not the case in unsupervised learning. In a certain sense, an unsupervised learner can be considered as a supervised learner where the label Y of the observation $\mathbf{X}$ is the observation itself. In other words, the task is to find a function $f : \mathbb{R}^d \to \mathbb{R}^d$ such that $f(\mathbf{X})$ predicts $\mathbf{X}$ as well as possible. Of course, without restricting the set of admissible predictors, this is a trivial problem. The source of such restrictions is the other objective of unsupervised learning, namely, to represent the mapping $f(\mathbf{X})$ of $\mathbf{X}$ with as few parameters as possible. These two competing objectives of unsupervised learning are called *information preservation* and *dimension reduction*. The trade-off between the two competing objectives depends on the particular problem. What makes unsupervised learning ill-defined in certain applications is that the trade-off is often not specified in the sense that it is possible to find two admissible functions $\hat{f}_1$ and $\hat{f}_2$ such that $\hat{f}_1$ predicts $\mathbf{X}$ better than $\hat{f}_2$, $\hat{f}_2$ compresses $\mathbf{X}$ more efficiently than $\hat{f}_1$, and there is no objective criteria to decide which function performs better *overall*.

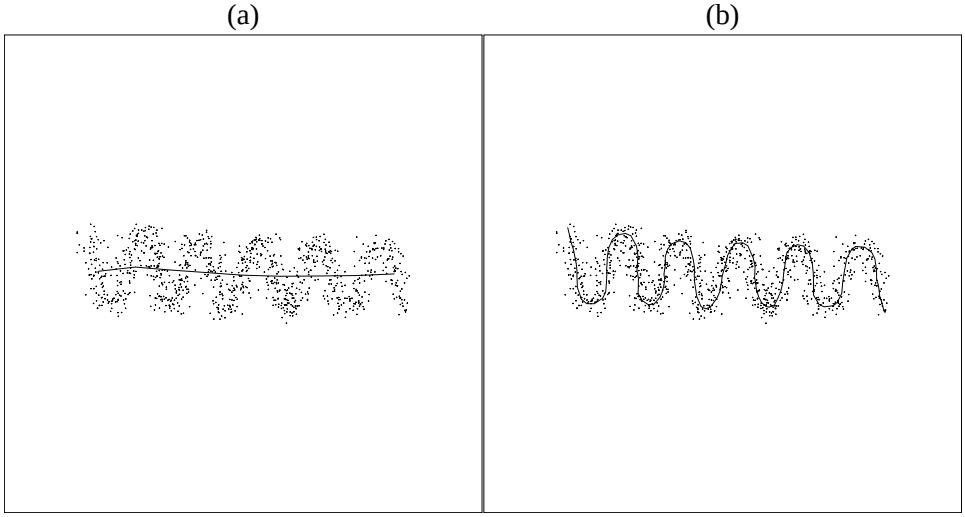

Figure 1: An ill-defined unsupervised learning problem. Which curve describes the data better, (a) a short curve that is "far" from the data, or a (b) long curve that follows the data more closely?

To clarify this ambiguity intrinsic to the unsupervised learning model, consider the simple example depicted by Figure 1. Both images show the same data set and two smooth curves intended

to represent the data set in a concise manner. Using the terminology introduced above, f is a function that maps every point in the plane to its projection point on the representing curve. Hence, in this case, the first objective of unsupervised learning means that the representing curve should go through the data cloud as close to the data points, on average, as possible. Obviously, if this is the only objective, then the solution is a "snake" that visits all the data points. For restricting the set of admissible curves, several regularity conditions can be considered. For instance, one can require that the curve be as smooth as possible, or one can enforce a length limit on the representing curve. If the length limit is hard, i.e., the length of the curve *must* be less or equal to a predefined threshold, the problem is well-defined in the sense that the curve that minimizes the average distance from the data cloud exists. In practice, however, it is hard to prescribe such a hard limit. Instead, the length constraint is specified as a soft limit, and the informal objective can be formulated as "find a curve which is as short as possible and which goes through the data as close to the data points, on average, as possible". This "soft" objective clearly makes the problem ill-defined in the sense that without another principle that decides the actual mixing proportion of the two competing objectives, one cannot choose the best of two given representing curve. In our example, we need an outside source that decides between a shorter curve that is farther form the data (Figure 1(a)), or a longer curve that follows the data more closely (Figure 1(b)).

The reason of placing this discussion even before the formal statement of the problem is that it determines our philosophy in developing general purpose unsupervised methods. Since the general problem of unsupervised learning is ill-defined, "turnkey" algorithms cannot be designed. Every unsupervised learning algorithm must come with a set of parameters that can be used to adjust the algorithm to a particular problem or according to a particular principle. From the point of view of the engineer who uses the algorithm, the number of such parameters should be as small as possible, and their effect on the behavior of the algorithm should be as clear as possible.

The intrinsic ambiguity of the unsupervised learning model also limits the possibilities of the theoretical analysis. On the one hand, without imposing some restrictive conditions on the model, it is hard to obtain any meaningful theoretical results. On the other hand, to allow theoretical analysis, these conditions may be so that the model does not exactly refer to any specific practical problem. Nevertheless, it is useful to obtain such results to deepen our insight to the model and also to inspire the development of theoretically well founded practical methods.

In the rest of the section we describe the formal model of unsupervised learning, outline some of the application areas, and briefly review the possible areas of theoretical analysis.

1.1.1 The Formal Model

For the formal description of the problem of unsupervised learning, let $\mathbb{D}$ be the domain of the data and let $\mathcal{F}$ be the set of functions of the form $f : \mathbb{D} \to \mathbb{R}^d$. For each $f \in \mathcal{F}$ we call the range of f the *manifold* generated by f, i.e.,

$$\mathcal{M}_f = f(\mathbb{D}) = \{ f(\mathbf{x}) : \mathbf{x} \in \mathbb{D} \}.$$

The set of all manifolds generated by all functions in $\mathcal{F}$ is denoted by $\mathbb{M}$, i.e., we define

$$\mathbb{M} = \{ \mathcal{M}_f : f \in \mathcal{F} \}.$$

To measure the distortion caused by the mapping of $\mathbf{x} \in \mathbb{D}$ into $\mathcal{M}_f$ by the function f, we assume that a *distance* $\Delta(\mathcal{M}, \mathbf{x})$ is defined for every $\mathcal{M} \in \mathbb{M}$ and $\mathbf{x} \in \mathbb{D}$. Now consider a random vector $\mathbf{X} \in \mathbb{D}$. The *distance function* or the *loss* of a manifold $\mathcal{M}$ is defined as the expected distance between $\mathbf{X}$ and $\mathcal{M}$, that is,

$$\Delta(\mathcal{M}) = E\left[\Delta(\mathbf{X}, \mathcal{M})\right].$$

The general objective of unsupervised learning is to find a manifold $\mathcal{M}$ such that $\Delta(\mathcal{M})$ is small and $\mathcal{M}$ has a low complexity relative to the complexity of $\mathbb{D}$. The first objective guarantees that the information stored in $\mathbf{X}$ is preserved by the projection whereas the second objective means that $\mathbf{M}$ is an efficient representation of $\mathbf{X}$.

1.1.2 Areas Of Applications

The general model of unsupervised learning has been defined, analyzed, and applied in many different areas under different names. Some of the most important application areas are the following.

- *Clustering* or *taxonomy* in multivariate data analysis [Har75, JD88]. The task is to find a usually hierarchical categorization of entities (for example, species of animals or plants) on the basis of their similarities. It is similar to supervised classification in the sense in that both methods aim to categorize $\mathbf{X}$ into a finite number of classes. The difference is that in a supervised model, the classes are predefined while here the categories are unknown so they must be created during the process.

- *Feature extraction* in pattern recognition [DK82, DGL96]. The objective is to find a relatively small number of features that represent $\mathbf{X}$ well in the sense that they preserve most of the variance of $\mathbf{X}$. Feature extraction is usually used as a pre-processing step before classification to accelerate the learning by reducing the dimension of the input data. Preserving the information stored in $\mathbf{X}$ is important to keep the Bayes error (the error that represents the confusion inherently present in the problem) low.

- *Lossy data compression* in information theory [GG92]. The task is to find an efficient representation of $\mathbf{X}$ for transmitting it through a communication channel or storing it on a storage device. The more efficient the compression, the less time is needed for transmission. Keeping the expected distortion low means that the recovered data at the receiving end resembles the original.

- *Noise reduction* in signal processing [VT68]. It is usually assumed here that $\mathbf{X}$ was generated by a latent additive model,

$$\mathbf{X} = \mathbf{M} + \varepsilon, \tag{1}$$

  where $\mathbf{M}$ is a random vector concentrated to the manifold $\mathcal{M}$, and ε is an independent multivariate random noise with zero mean. The task is to recover $\mathbf{M}$ based on the noisy observation $\mathbf{X}$.

- *Latent-variable models* [Eve84, Mac95, BSW96]. It is presumed that $\mathbf{X}$, although sitting in a high-dimensional space, has a low intrinsic dimension. This is a special case of (1) when the additive noise is zero or nearly zero. In practice, $\mathcal{M}$ is usually highly nonlinear otherwise the problem is trivial. When $\mathcal{M}$ is two-dimensional, using $\mathbf{M}$ for representing $\mathbf{X}$ can serve as an effective visualization tool [Sam69, KW78, BT98].

- *Factor analysis* [Eve84, Bar87] is another special case of (1) when $\mathbf{M}$ is assumed to be a Gaussian random variable concentrated on a linear subspace of $\mathbb{R}^d$, and ε is a Gaussian noise with diagonal covariance matrix.

1.1.3 The Simplest Case

In simple unsupervised models the set of admissible functions $\mathcal{F}$ or the corresponding set of manifolds $\mathbb{M}$ is given independently of the distribution of $\mathbf{X}$. $\mathcal{F}$ is a set of simple functions in the sense that any $f \in \mathcal{F}$ or the corresponding $\mathcal{M}_f \in \mathbb{M}$ can be represented by a few parameters. It is also assumed that any two manifolds in $\mathbb{M}$ have the same intrinsic dimension, so the only objective in this model is to minimize $\Delta(\mathcal{M})$ over $\mathbb{M}$, i.e., to find

$$\mathcal{M}^* = \arg\min_{\mathcal{M} \in \mathbb{M}} E\left[\Delta(\mathbf{X}, \mathcal{M})\right].$$

Similarly to supervised learning, the distribution of $\mathbf{X}$ is usually unknown in practice. Instead, we are given $\mathcal{X}_n = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \subset \mathbb{R}^d$, a sample of n independent and identical copies of $\mathbf{X}$, and the task is to find a function $\hat{f}_n(\mathbf{X}) = \hat{f}(\mathbf{X}, \mathcal{X}_n)$ *based on the data set* $\mathcal{X}_n$ that minimizes the distance function. Since the the distribution of $\mathbf{X}$ is unknown, we estimate $\Delta(\mathcal{M})$ by the *empirical distance*

function or *empirical loss* of $\mathcal{M}$ defined by

$$\Delta_n(\mathcal{M}) = \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{X}_i, \mathcal{M}). \tag{2}$$

The problem is well-defined in the sense that the performance of a projection function $\hat{f}_n$ can be quantified by the empirical loss of $\hat{f}_n$ measured on an independent test set $x'_m = \{\mathbf{X}'_1, \ldots, \mathbf{X}'_m\}$. As a consequence, the best of two given projection functions $\hat{f}_1$ and $\hat{f}_2$ can be chosen objectively by comparing $\Delta_n(\mathcal{M}_{\hat{f}_1})$ and $\Delta_n(\mathcal{M}_{\hat{f}_2})$ on a sufficiently large test sample.

In the theoretical analysis of a particular unsupervised model, the first question to ask is

$$\textit{“Does } \mathcal{M}^* \textit{ exist in general?”} \tag{Q1}$$

Clearly, if $\mathcal{M}^*$ does not exist, or it only exists under severe restrictions, the theoretical analysis of any estimation scheme based on finite data is difficult. If $\mathcal{M}^*$ does exist, the next two obvious questions are

$$\textit{“Is } \mathcal{M}^* \textit{ unique?”} \tag{Q2}$$

and

$$\textit{“Can we show a concrete example of } \mathcal{M}^*\textit{?”} \tag{Q3}$$

Interestingly, even for some of the simplest unsupervised learning models, the answer to Question 3 is no for even the most common multivariate densities. Note, however, that this fact does not make the theoretical analysis of an estimating scheme impossible, and does not make it unreasonable to aim for the optimal loss $\Delta(\mathcal{M}^*)$ in practical estimator design.

The most widely used principle in designing nonparametric estimation schemes is the *empirical loss minimization principle*. In unsupervised learning this means that based on the data set x_n, we pick the manifold $\mathcal{M}_n^* \in \mathbb{M}$ that minimizes the empirical distance function (2), i.e., we choose

$$\mathcal{M}_n^* = \arg\min_{\mathcal{M}\in\mathbb{M}} \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{X}_i, \mathcal{M}). \tag{3}$$

The first property of $\mathcal{M}_n^*$ to analyze is its *consistency*, i.e. the first question is

$$\textit{“Is } \lim_{n\to\infty}\Delta(\mathcal{M}_n^*) = \Delta(\mathcal{M}^*) \textit{ in probability?”} \tag{Q4}$$

Consistency guarantees that by increasing the amount of data, the expected loss of $\mathcal{M}_n^*$ gets arbitrarily close to the best achievable loss. Once consistency is established, the next natural question is

$$\textit{“What is the convergence rate of } \Delta(\mathcal{M}_n^*) \to \Delta(\mathcal{M}^*)\textit{?”} \tag{Q5}$$

A good convergence rate is important to establish upper bounds for the probability of error for a given data size. From a practical point of view, the next question is

"Is there an efficient algorithm to find $\mathcal{M}_n^*$ given a data set $\mathcal{X}_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$?" (Q6)

To illustrate this general analysis scheme, we turn to the simplest possible unsupervised learning method. Let the admissible manifolds $\mathcal{M}$ be arbitrary points of the d-dimensional space ($\mathbb{M} = \mathbb{R}^d$), and assume that $\Delta(\mathcal{M}, \mathbf{X})$ is the squared Euclidean distance of $\mathcal{M}$ and $\mathbf{X}$, i.e.,

$$\Delta(\mathcal{M}, \mathbf{X}) = \|\mathcal{M} - \mathbf{X}\|^2.$$

To find $\mathcal{M}_n^*$, we have to minimize $E[\Delta(\mathcal{M}, \mathbf{X})]$ over all $\mathcal{M} \in \mathbb{R}^d$. It is a well known fact that $E[\|\mathcal{M} - \mathbf{X}\|^2]$ is minimized by $E[\mathbf{X}]$ so we have

$$\mathcal{M}^* = E[\mathbf{X}].$$

The answer to all the first three questions is, therefore, yes. According to the empirical loss minimization principle, given $\mathcal{X}_n = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \subset \mathbb{R}^d$, a sample of n independent and identical copies of $\mathbf{X}$, the estimator $\mathcal{M}_n^* \in \mathbb{M}$ is the vector that minimizes the empirical loss, i.e.,

$$\mathcal{M}_n^* = \arg\min_{\mathcal{M} \in \mathbb{M}} \frac{1}{n} \sum_{i=1}^{n} \|\mathcal{M} - \mathbf{X}_i\|^2.$$

It is easy to see that the minimizing vector is the *sample mean* or *center of gravity* of $\mathcal{X}_n$, i.e.,

$$\mathcal{M}_n^* = \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}_i.$$

Consistency of $\mathcal{M}_n^*$ follows from the law of large numbers. For the convergence rate note that, if $\mathbf{X}$ has finite second moments,

$$\begin{aligned} \Delta(\mathcal{M}_n^*) - \Delta(\mathcal{M}^*) &= E\left[\|\mathbf{X} - \mathcal{M}_n^*\|^2\right] - E\left[\|\mathbf{X} - \mathcal{M}^*\|^2\right] \\ &= \left(1 + \frac{1}{n}\right)\sigma_{\mathbf{X}}^2 - \sigma_{\mathbf{X}}^2 \\ &= \frac{1}{n}\sigma_{\mathbf{X}}^2 \end{aligned} \tag{4}$$

where $\sigma_{\mathbf{X}}^2$ is the sum of the variances of the components of $\mathbf{X}$. Hence, if $\mathbf{X}$ has finite second moments, the rate of convergence is $\frac{1}{n}$.

Simple unsupervised learning methods of this type are Vector Quantization (admissible manifolds are finite sets of d-dimensional vectors), and Principal Component Analysis (admissible manifolds are linear subspaces of $\mathbb{R}^d$). We analyze these methods in Chapter 2. Theoretical analysis of principal curves with a length constraint in Chapter 4 also follows these lines.

1.1.4 A More Realistic Model

Although amenable for theoretical analysis, simple methods described above are often impractical. In one group of methods the strict restrictions imposed on the admissible manifolds result in that manifolds of $\mathbb{M}$ are not able to describe highly nonlinear data. Unfortunately, there is a problem even if admissible manifolds are rich enough to capture complex data. The problem is that in the simple model described above, the set of admissible manifolds must be specified *independently of the particular application*. Given a set of manifolds $\mathbb{M}$, it is possible that in a certain application $\mathbb{M}$ is too rich, while in another problem manifolds in $\mathbb{M}$ are too simple to capture the data. This problem can be solved by allowing the practitioner to choose from several classes of manifolds of different complexity. Assume, therefore, that a nested sequence of manifold model classes $\mathbb{M}^{(1)} \subset \mathbb{M}^{(2)} \subset \ldots$ is given, such that for a given $j = 1, \ldots$, the intrinsic dimensions of all manifolds in $\mathbb{M}^{(j)}$ are the same. Let c_j be a real number that measures the intrinsic dimension of manifolds in $\mathbb{M}^{(j)}$, such that $c_1 < c_2 < \ldots$. We can also say that c_j measures the *complexity* of $\mathbb{M}^{(j)}$. To define the theoretically best manifold, one can follow the following strategy. Find the optimal manifold in each class to obtain the sequence of manifolds $\mathcal{M}^{(1)^*}, \mathcal{M}^{(2)^*}, \ldots$. Then using a principle corresponding to the particular problem, select the manifold $\mathcal{M}^{(j^*)^*}$ from the j^*th model class that represents the data the best[1].

The same questions can be asked in this complex model as in the simple model described in the previous section. The existence of $\mathcal{M}^{(j^*)^*}$ depends on two conditions. First, optimal manifolds $\mathcal{M}^{(1)^*}, \mathcal{M}^{(2)^*}, \ldots$ must exist for all model classes. Second, the principle that governs the selection of $\mathcal{M}^{(j^*)^*}$ (by choosing the model class j^*) must be well-defined in the sense that it gives a total order over the set of optimal manifolds $\mathcal{M}^{(1)^*}, \mathcal{M}^{(2)^*}, \ldots$.

Estimating $\mathcal{M}^{(j^*)^*}$ can be done by combining the empirical loss minimization principle with the model selection technique described above. Accordingly, one can choose the empirically best manifold for each model class to obtain the sequence $\mathcal{M}_n^{(1)^*}, \mathcal{M}_n^{(2)^*}, \ldots$, and then use the principle corresponding to the particular problem to select the best manifold $\mathcal{M}_n^{(j_n^*)^*}$. Consistency analysis of the model is usually rather hard as one has to not only establish consistency in the model classes, but also to show that when the data size is large, the model class j^* selected in the theoretical model is the same as the model class j_n^* selected in the estimation.

To further complicate the situation, it is usually impractical to follow this scheme since it requires to find the empirically best manifold in several model classes. Instead, practical algorithms usually optimize the two criteria *at the same time*. In most of the algorithms, although not in all of

[1] Note that this approach resembles the method of *complexity regularization* [DGL96] or *structural risk minimization* [Vap98] used in supervised learning. There is a fundamental difference, however. While in supervised learning, complexity regularization is used in the *estimation* phase, here, we use it to define the *theoretically* best manifold. The reason, again, is that the general unsupervised learning problem is inherently ill-posed.

them, the two criteria are combined in one "energy function" of the form

$$G_n(\mathcal{M}) = \Delta_n(\mathcal{M}) + \lambda P(\mathcal{M})$$

where $\Delta_n(\mathcal{M})$ is the empirical distance function of $\mathcal{M}$, as usual, $P(\mathcal{M})$ is a *penalty* or *regularizer* term which penalizes the complexity of the manifold, and λ is a penalty coefficient that determines the trade-off between the accuracy of the approximation and the smoothness of the manifold. The algorithm proceeds by minimizing $G_n(\mathcal{M})$ over all admissible manifolds. In Chapter 3, we present several methods that follow this scheme.

1.2 Principal Curves

The main subject of this thesis is the analysis and applications of principal curves. Principal curves were originally defined by Hastie [Has84] and Hastie and Stuetzle [HS89] (hereafter HS) to formally capture the notion of a smooth curve passing through the "middle" of a d-dimensional probability distribution or data cloud (to form an intuitive image, see Figure 1 on page 2). The original HS definition of principal curves is based on the concept of *self-consistency*. Intuitively, self-consistency means that each point of the curve is the average of all points that project there. Based on the self-consistency property, HS developed a theoretical and a practical algorithm for constructing principal curves of distributions and data sets, respectively.

The field has been very active since Hastie and Stuetzle's groundbreaking work. Numerous alternative definitions and methods for estimating principal curves have been proposed, and principal curves were further analyzed and compared with other unsupervised learning techniques. Several applications in various areas including image analysis, feature extraction, and speech processing demonstrated that principal curves are not only of theoretical interest, but they also have a legitimate place in the family of practical unsupervised learning techniques.

Although the concept of principal curves as considered by HS has several appealing characteristics, complete theoretical analysis of the model seems to be rather hard. This motivated us to redefine principal curves in a manner that allowed us to carry out extensive theoretical analysis while preserving the informal notion of principal curves. Our first contribution to the area is, hence, a new *theoretical model* that can be analyzed along the lines of the general unsupervised learning model described in the previous section. Our main result here is the first known consistency proof of a principal curve estimation scheme.

The theoretical model proved to be too restrictive to be practical. However, it inspired the design of a new *practical algorithm* to estimate principal curves based on data. The polygonal line algorithm, which compares favorably with previous methods both in terms of performance and computational complexity, is our second contribution to the area of principal curves. To complete

the picture, in the last part of the thesis we consider an *application* of the polygonal line algorithm to hand-written character skeletonization. We note here that parts of our results have been previously published in [KKLZ], [KKLZ99], and [KKLZ00].

1.3 Outline of the Thesis

Most of the unsupervised learning algorithms originate from one of the two basic unsupervised learning models, vector quantization and principal component analysis. In Chapter 2 we describe these two models. In Chapter 3, we present the formal definition of the HS principal curves, describe the subsequent extensions and analysis, and discuss the relationship between principal curves and other unsupervised learning techniques.

An unfortunate property of the HS definition is that, in general, it is not known if principal curves exist for a given distribution. This also makes it difficult to theoretically analyze any estimation scheme for principal curves. In Chapter 4 we propose a new definition of principal curves and prove the existence of principal curves in the new sense for a large class of distributions. Based on the new definition, we consider the problem of learning principal curves based on training data. We introduce and analyze an estimation scheme using a common model in statistical learning theory. The main result of this chapter is a proof of consistency and analysis of rate of convergence following the general scheme described in Section 1.1.

Although amenable to analysis, our theoretical algorithm is computationally burdensome for implementation. In Chapter 5 we develop a suboptimal algorithm for learning principal curves. The polygonal line algorithm produces piecewise linear approximations to the principal curve, just as the theoretical method does, but global optimization is replaced by a less complex gradient-based method. We give simulation results and compare our algorithm with previous work. In general, on examples considered by HS, the performance of the new algorithm is comparable with the HS algorithm while it proves to be more robust to changes in the data generating model.

Chapter 6 starts with an overview of existing principal curve applications. The main subject of this chapter is an application of an extended version of the principal curve algorithm to hand-written character skeletonization. The development of the method was inspired by the apparent similarity between the definition of principal curves and the medial axis of a character. A principal curve is a smooth curve that goes through the "middle" of a data set, whereas the medial axis is a set of smooth curves that go equidistantly from the contours of a character. Since the medial axis can be a *set* of connected curves rather then only *one* curve, in Chapter 6 we extend the polygonal line algorithm to find a principal graph of a data set. The extended algorithm also contains two elements specific to the task of skeletonization, an initialization method to capture the approximate topology of the character, and a collection of restructuring operations to improve the structural quality of the

skeleton produced by the initialization method. Test results on isolated hand-written digits indicate that the algorithm finds a smooth medial axis of the great majority of a wide variety of character templates. Experiments with images of continuous handwriting demonstrate that the skeleton graph produced by the algorithm can be used for representing hand-written text efficiently.