Data Modeling with Gaussian Mixture Based Clustering: A Unifying View of Discriminative and Generative Clustering

Vered Tsedaka

January 12, 2006

Abstract

Unsupervised methods in machine learning are usually used when no classes are defined a-priori. Clustering methods are designed to split the data into groups of related data points, called clusters, based on measures of distance or affinity between points. There is generally no attempt to model the 'distribution' of these clusters. By contrast, generative methods aim at modeling the distribution of the data (or the class, if class labels are given). There is usually no attempt to split the data into meaningful groups in an unsupervised way. This report describes a scheme for data modeling which combines affinity-based clustering with generative GMM modeling of the data distribution. This scheme reduces the risk of fitting an oversimplified model to each class, while avoiding the loss of prediction value caused by merely clustering the data.

We first provide the theoretical foundation behind this approach, and show that the approximations sought after by our method exist, and that they can be obtained explicitly. We then describe an algorithm which approximates the data with a large number of "small" Gaussians, and then clusters these Gaussians with an agglomerative clustering algorithm. Finally, we demonstrate excellent clustering results on synthetic examples, and discuss the results of clustering the MNIST data base. We also demonstrate how new multi-class samples of the same data can be faithfully generated by the model we compute.

Acknowledgements

I wish to thank my advisor, Prof. Daphna Weinshall, who has been everything one could ever wish a great advisor would be: knowledged, patient, supportive, and above all, a very pleasant person to work with.
In addition, I would like to thank Asaf Zimmerman, for his wits, love and support.

Contents

Chapter 1

Introduction

Every learning process begins with the following questions:
"what knowledge do we wish to obtain?" and
"what do we already know?". When restricted to the field of Machine Learning input data, the learning task can be viewed as the most rigorous attempt to *drastically compress* data with a minimal loss of the inherent information [13].

The first question can be broken down into two separate goals:

- Supplying an adequate description or model of the existing data, and
- Supplying a mechanism that makes it possible to draw informative conclusions regarding data that have not yet been seen.

It is worth noting that there is often a distinct trade-off between these two goals: a very fine description of existing data may leave very little room for generalization, resulting in poor judgement regarding new data - and vice versa.

1.1 Supervised VS. Unsupervised Learning

The answer to the second question - what prior knowledge we already possess, obviously depends on the specific task at hand. Our main hypothesis regarding the data is that for each $x \in X$ there exists a labeling $t \in T$, such that the set (x_i, t_i) is drawn independently from an unknown probabilistic relationship $p(x, t)$. Even so, two different scenarios are easily distinguished:

- Supervised Learning
- Unsupervised Learning.

1.1.1 Supervised Learning

In the supervised scenario, the labeling of each data point $x \in X$ is known. Knowing the full labeling of the data narrows possible learning tasks to estimating $p(x,t)$ and providing a mechanism for labeling samples $\tilde{x} \notin X$ such that $(\tilde{x},\tilde{t})$ is assumed to be drawn out of $p(x,t)$.

The supervised scenario is in many respects an unlikely one: obtaining full classification of the data is usually very costly, and at times - impossible. It requires an "oracle", a knowledged factor outside of the learning scenario - and many learning problems have no such luxury.

For this reason, this paper will mainly address other scenarios, ones that require less prior knowledge to perform the learning task.

1.1.2 Unsupervised Learning

In the Unsupervised Learning Scenario, there is no knowledge of the labeling t. Furthermore, $|T|$ may not necessarily be known. Therefore, a learning task within this framework may face a triple challenge: construction of a labeling function $F : X \rightarrow T$, estimating $p(x,t)$ and determining the labeling of unseen data.

Unsupervised learning problems of the first type, i.e. the construction of F, are generally referred to as *clustering* problems: grouping a set of points into clusters such that points in the same cluster are more similar to each other than points in different clusters, under a particular similarity metric [4].

Traditionally, clustering problems are categorized as *generative* or *discriminative*.

generative clustering:

A clustering problem is said to be *generative* - if there exists a hypothesis that $p(x,t)$ - is of a certain parametric form (i.e. $p(x,t) = p_\theta(x,t)$), thus reducing the problem into finding the parameters best fitting the data. The resulting labeling function will then label using the likelihood score given by the estimated $p_{\tilde{\theta}}(x,t)$.

The vast majority of the work in the field of generative clustering (or mixture model based clustering) follows the following reasoning: we assume that each cluster $j = 1...K$ is generated by a distribution with a density function $p_j(x|\Theta_j)$ where Θ_j is a set of parameters. We also assume that the a-priori probability of a point belonging to cluster j is α_j (naturally $\sum_{j=1}^{K} \alpha_j = 1$). Thus we assume the data points $x_1, ...x_N$ are chosen independently from the distribution

$$\sum_{j=1}^{K} \alpha_j p_j(x|\Theta_j)$$

We have now reduced the target problem of estimating $p(x,t)$ (after assuming the general form of the distributions p_j), to finding the parameters $\{\Theta_j\}_{j=1}^K$ and $\{\alpha_j\}_{j=1}^K$ which maximize the likelihood

$$L(\Theta = \{\Theta_j\}_{j=1}^K, \{\alpha_j\}_{j=1}^K | x_1, ...x_N) = \times_{i=1}^N (\sum_{j=1}^{K} \alpha_j p_j(x_i|\Theta_j))$$

or equally, the log likelihood

$$log(L(\Theta|x_1, ...x_N)) = \sum_{i=1}^{N} log(\sum_{j=1}^{K} \alpha_j p_j(x_i|\Theta_j))$$

The problem is that this function is hard to analyze, mainly because the log contains a sum. The EM algorithm (see [5]) treats this problem. In its most general form, the EM assumes that the random variable X we see is a part of a random variable $Z = (X, Y)$. In our case Y represent the actual cluster from which a data point was drawn. In unsupervised learning, we do not see Y.

The EM algorithm uses two facts:

1. Given a set of parameters $\Theta = \{\Theta_j\}_{j=1}^K, \{\alpha_j\}_{j=1}^K$ It is easy to compute (using the Bayes formulæ) the probability that a sample x was drawn from the cluster j (i.e. $y = j$):

$$p(Y = j|X = x, \Theta) = \frac{p(X = x|y = j, \Theta)p(Y = j|\theta)}{\sum_{j'=1}^{K} p(X = x|y = j', \Theta)p(Y = j'|\theta)} = \frac{\alpha_j p_j(X = x|\Theta_j)}{\sum_{j=1}^{K} \alpha_{j'} p_{j'}(X = x|\Theta_{j'})}$$

2. Given a probability on Y the expectation of the log likelihood has the following form:

$$\begin{aligned} E_Y\left[log(L(\Theta|X = x, Y))\right] &= E_Y(log(p(X = x|Y = y, \Theta))) = E_Y(log(p_Y(x_i|\Theta_Y))) \\ &= \sum_{j=1}^{K} p(Y = j) log(p_j(x|\Theta_j)) \end{aligned}$$

This simplifies matters since this expression no longer contains a log of a sum.

The EM heuristics is the following. We start with some initial parameters Θ^0, and perform the following iterations: in the l-th iteration we have already obtained a set of parameters Θ^{l-1}. We use these parameters to calculate the a-posteriori probability for each data point that the point has been taken from each cluster using the formulægiven in the first fact. We then use this distribution to write an expression for the expected log likelihood of new parameters Θ, as follows

$$\begin{aligned} Q(\Theta, \Theta^{l-1}) &= \sum_{j=1}^{K}\sum_{i=1}^{N} log(\alpha_j p_j(x_i|\Theta_j))p(j|x_i, \Theta^{l-1}) \\ &= \sum_{j=1}^{K}\sum_{i=1}^{N} log(\alpha_j)p(j|x_i, \Theta^{l-1}) + \sum_{j=1}^{K}\sum_{i=1}^{N} log(p_j(x_i|\Theta_j))p(j|x_i, \Theta^{l-1}) \end{aligned}$$

We now maximize this expression to get

$$\Theta^l = argmax_\Theta(Q(\Theta, \Theta^{l-1}))$$

To maximize this expression, we can maximize the term containing α_j and the term containing Θ_j independently since they are not related. To find the expression for α_j we use the Lagrange multiplier technique and obtain that

$$\alpha_j = \frac{1}{N}\sum_{i=1}^{N} p(j|x_i, \Theta^{l-1})$$

Maximizing Θ_j is dependent on the form of $p(x|\Theta)$ and cannot be calculated analytically for any such p. This is where we make use of our hypothesis that the kernel functions are Gaussians; $\Theta = (\mu, \Sigma)$ - μ is the Gaussian expectation and Σ is its covariance matrix.

p, therefore, is of the form

$$p_j(x|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)}$$

For this kernel function we can analytically express Θ^l:

$$\begin{aligned}\mu_j^l &= \frac{\sum_{i=1}^N x_i p(j|x_i,\Theta^{l-1})}{\sum_{i=1}^N p(j|x_i,\Theta^{l-1})} \\ and & \\ \Sigma_j^l &= \frac{\sum_{i=1}^N p(j|x_i,\Theta^{l-1})(x_i-\mu_j^l)(x_i-\mu_j^l)^T}{\sum_{i=1}^N p(j|x_i,\Theta^{l-1})}\end{aligned}$$

Obtaining an estimate for $p(x,\Theta)$ now enables us to perform our other tasks.

Constructing a labeling function F is now a matter of computing for each point the probability that it belongs to each of the clusters and assign it to the cluster that maximizes this probability .

Therefore, $F : X \to T$, $F(x) = j$ which maximizes $p(j|x,\Theta)$

Similarly, for any new sample $\tilde{x}$, we can now compute the probability of $\tilde{x}$ belonging to each of the clusters and assigning it to the one that maximizes this probability.

Discriminative Clustering:

A clustering problem is said to be discriminative if no assumption is made regarding the underlying parametric data-generation model. The goal in such a case is given a similarity metric to maximize within-cluster similarity, and minimize between-cluster similarity [6]. The resulting labeling will be the natural one under the clustering obtained.

Within the framework of Discriminative Clustering, one approach is that of Graph-Based (or Graph-Theoretic) Clustering. The data set is represented by a full graph $G(V,E)$, where $V = D \subseteq \mathbb{R}^n$, E is the set of edges connecting the points, and d is a weight function that assigns a weight to each edge representing the affinity between the data points.

The weight function d is derived from a distance or similarity function $\tilde{d}$ ($\tilde{d} : D \times D \to \mathbb{R}$), and therefore plays a key role in the Discriminative Clustering framework.

Much attention has been given to the problem of choosing an adequate distance function. Though the data set D is represented in $\mathbb{R}^n$, the natural Euclidean distance function

$$d(x,y) = \|x-y\|$$

can be ill-fitting for obvious reasons: the representation of the data set D in $\mathbb{R}^n$, for some n, is not unique. Often, a *feature selection* process has been applied, to extract important information and discard unnecessary traits. Therefore, the way one chooses to embed the data in $\mathbb{R}^n$ will have direct consequences on the Euclidean distance between the points. Moreover, it can be easily shown that even by embedding the data without any loss of information, the Euclidean distance function will not reflect an appropriate affinity between data points. A classic illustration of this feature is demonstrated by the embedding of a grayscale image $A_{m\times n}$ as a vector in $\mathbb{R}^{m\times n}$. If we blacken the very first column of image A, the resulting image will appear quite similar to the original image. However, the embedding of the altered image can be quite distant from the embedding of A, as all the coordinates corresponding to the first column have been switched to zero, and hence the Euclidian distance is $(\sum_{k=1}^{n} A_{1,k}^2)^{\frac{1}{2}}$ which can be large in magnitude.

Given a similarity or affinity matrix, different graph-based algorithms make different uses of this:

Agglomerative algorithms begin by placing each data point in a distinct cluster, and then successively merging clusters together until a halting criterion is satisfied. Examples of such agglomerative techniques can be either single-link clustering (merging of only two clusters in a single step) or complete-link (merging all clusters that are closer than a certain threshold in a single step).

Spectral Methods (or eigendecomposition algorithms) make use of the eigenvectors of the similarity matrix to perform segmentation, using the fact that eigenvectors of permutated matrices are the permutations of eigenvectors of the original matrix, coupled with analysis of the eigenvectors of block matrices. Several such methods are reviewed in [16] along with a proposed algorithm that combines them for enhanced results.

Squared Error clustering initially selects cluster centers out of the data set. Each step of the algorithm assigns the rest of the data, according to the similarity matrix, to one of the clusters. Then, for each cluster, using the current cluster membership, the centroid is computed, and the points are re-assigned. The steps are applied until a convergence criterion is met. By far, the most commonly-used squared-error clustering algorithm is the k-means algorithm. It has several variants, differing in either the strategy for the initial selection, the re-construction of clusters from step to step, or the convergence criteria ([10]).

Discriminative algorithms deal mainly with the task of constructing the labeling function F, and do not address the issue of approximating $p(x, \Theta)$ at all.

1.1.3 Semi-Supervised Learning

As noted, the distinction between the Supervised Learning and the Unsupervised Learning frameworks was whether or not there is full prior knowledge of the labeling of the data points. Naturally, it can be argued that an entire continuum exists between these two underlying assumptions in which there is partial knowledge of the labels, or further yet - information regarding relations between different data points and their corresponding labels. Clearly, this information is not useless: if one were to know that data points x_1 and x_2 belong to the same class, any information regarding the labeling of one of the points - would directly affect our knowledge of the labeling of the other.

A choice to work only within one of the two frameworks (Supervised or Unsupervised) will therefore imply that any additional, but partial information that we might have will need to be discarded.

This seems to run counter our intuitive understanding of the learning process, since surely knowing more is better than knowing nothing. An entire area of statistical Machine Learning addresses the issue of learning with partial knowledge, and is generally referred to as *Semi-Supervised Learning.*

Different learning tasks may have various degrees of prior knowledge, so an algorithm that depends heavily on prior knowledge may perform poorly when the information is not at hand. To guarantee a certain level of performance, most algorithms within the semi-supervised learning framework can be viewed as unsupervised learning algorithms, to that additional information can be incorporated into them. This applies to both generative and discriminative clustering methods. Throughout this paper we will focus on additional information in the form of equivalence constraints:

Along with the data set D, the input will consist of a set $C \subseteq D \times D$ of positive and negative constraints where
A *positive constraint* $(x, y) = c_p \in C$ indicates that x and y belong to the same cluster, and
A *negative constraint* $(x, y) = c_n \in C$ indicates that x and y belong to different clusters.

There have been quite a few references in the literature to the incorporation of such added information to various algorithms. In particular, [12] Hertz et al deal with incorporating equivalence constraints into the EM algorithm. This was done by viewing the data not as data *points*, but as data *chunklets* - a small subset of data points that are known to belong to the same class. Incorporating positive constraints therefore becomes an issue of unifying all data points with positive relations amongst themselves into chunklets - and performing the EM algorithm on the set of chunklets rather than on the original data set D. They also showed how to incorporate negative constraints, noting that this task is costly in running time, due to the use of Markov Networks.

In the discriminative clustering scenario, the incorporation of equivalence constraints has been addressed as well. Both in the agglomerative and spectral algorithms, the key step is to alter the similarity or affinity matrix d in accordance with the constraints. For example, in the agglomerative case, a positive constraint $c_p = (x, y)$ will be incorporated into the algorithm by setting the distance between x and y to the value zero. A negative constraint will be incorporated by setting the distance between x and y to be infinity. For some algorithms these changes to the distance matrix suffice, but others may need adjusting to deal with the new values [1]. In a similar way, the problem of incorporating constraints into the K-means algorithm has been addressed in [10].

1.2 Our Approach

There are several things worth noting regarding the semi supervised learning framework. Though it does present a continuum with regards to prior knowledge (as opposed to the dichotomy of zero prior knowledge in the unsupervised scenario, or full knowledge of the labeling in the supervised scenario), it maintains a second just as rigid separation: the dichotomy between discriminative and model-based clustering. A key observation here is that whenever there is a choice between one or the other form - something is lost. Choosing a discriminative learning process will indeed lead to a certain understanding of a pointwise "distance function" - but with no grasp of the modeling of the data.

As noted earlier, many of these algorithms may perform well with regards to classifying the given data, yet the lack of the ability to generalize may result in poor performance concerning unseen data. Given a new data point, the only possibility is to try to estimate its distance from data points that have already been labeled - and deducing a labeling according to some minimization principle.

The distance function, as pointed out previously, can be skewed due to a number of factors: the choice of embedding of the data in $\mathbb{R}^n$ and the appearance of rare samples in misfit numbers (this is especially true when the data set is small in magnitude), to name a few.

There is something to be lost if we opt for the Model-Based Clustering path as well. We do gain the ability to generalize according to probabilities and density functions, but all under a severe constraint. The model is simple - each class is described by a function of the predetermined parametric form. In the majority of the cases this is a Gaussian function. If the classes within the data set roughly "behave well" to match this description, the results of the learning process will be good under both criteria: classifying points in the data set, and generalizing results to data points which have not been seen. However, if the model is ill-fitting, an algorithm will perform poorly in *both* aspects.

A common heuristic to overcome this obstacle is using large quantities of unlabeled data on top of the labeled data. The reasoning is that though the model is ill-fitting, the parameter estimation process will yield optimal results to fit the data by weighing it down with the unlabeled data. However, in [7], Cozman et al demonstrated that unlabeled data can in fact lead to an increase in classification error, due to the fact that the bias hypothesis of the parametric form is adversely affected by unlabeled data, having exactly the opposite effect than the one intended. They present a mathematical analysis of this fact and some examples illustrating such circumstances.

In light of this, the constraint of a convex model appears much too rigid, and attempts have been made to tackle this problem. Many such attempts have focused on exploiting the density traits of the data in more ways than for the parameter estimation. For example, in [3] Georgescu et al apply a gradient descent approximation technique to enhance the clustering. By evaluation of the density of the data, the center of a Gaussian is located where the data is dense. A different approach is the hierarchical view, as demonstrated by Vasconcelos and Lippman in [15]. By using a bottom-up approach, they model the data with small-variance Gaussians, followed by iteratively merging the Gaussians into larger Gaussians using only the parametric form of the Gaussians, with no further reference to the original data. While providing valuable insights into the parameter estimation process, such methods do not resolve the problem of fitting an adequate model to the data, since the final model inevitably consists of a Gaussian describing each class.

The goal of this work is to introduce a new method for modeling the data, in which the final model for each class is more intricate that a single Gaussian. A key observation here is that though a single Gaussian may not model well an entire class, it can serve as a very good *local* approximation. In a region D of $\mathbb{R}^n$ where the data is dense, a Gaussian with small variance centered on D will model the data very well. If the data are sparse in D however, there is little information about the density function that determined the distribution, and so one may opt to split D into smaller sub regions and approximate the density function locally in each of them separately.

This motivates us to suggest modeling the data with a large number of small-variance Gaussians, and then determining for each of them in which of the classes it lies. The resulting model therefore is a mixture of mixtures: each class is modeled by a mixture of Gaussians models.

In chapter 2 of this report we introduce a mathematical formulation of the problem, and within that framework set out to prove two results:

1. assuming F is a density function with $F = \alpha_1 f + \alpha_2 g$, with f, g both density functions and $\alpha_1 + \alpha_2 = 1$ (and under some assumptions on the separation of f and g), if F can be approximated well by a set of kernel functions h_i, then there exists a partition of $\{h_i\}$ into two disjoint subsets such that each yields a good approximation of $\alpha_1 f$ and $\alpha_2 g$ respectively.

2. For each kernel function h_i it is possible to determine which of the two functions is being locally approximated by it.

The mathematical analysis is the motivation for presenting the GMBC algorithmic scheme in chapter 3, and it provides its theoretical justification. We discuss the theoretical implications of the scheme and describe simulations on several data sets.The empirical results are discussed, highlighting the scheme's strengths and weaknesses. We conclude by proposing several possible directions for further research.

Chapter 2

Theoretical Analysis

Let F be a density function of the form $F = \alpha_1 f + \alpha_2 g$, where both f and g are density functions and $\alpha_1 + \alpha_2 = 1$.[1] Suppose additionally that F can be approximated by a weighted sum of M kernel functions h_i, all density functions as well, such that

$$\left\| F - \sum_{i=1}^{M} \beta_i h_i \right\|_2 , \left\| F - \sum_{i=1}^{M} \beta_i h_i \right\|_1$$

are small. We will show that if the variances of the kernel functions are small, and if F is approximated well by the set of kernel functions, then we can partition this set into two subsets, such that the sum over the first subset provides a good approximation to the function f, and the second subset provides a good approximation to the function g. We will then present a way to determine for each kernel function whether it approximates f or g.

The outline of the proof is as follows: First, we observe that by bounding the sum of the variances of the kernel functions, we can easily partition them into two disjoint sets, each approximating one of the two functions f or g. The approximation error depends on several parameters describing the behavior of the functions f,g,h_i. Given the existence of such a partition, we show the following: suppose that in such partitioning h_1 is among the functions approximating f, and h_2 is among the functions approximating g. Then $\|\sqrt{h_1} - \sqrt{h_2}\|$ cannot be too small, and has a positive lower bound. From this we deduce that if the distance between a pair of kernel functions is smaller than this bound, they both approximate the same function and are therefore related by a similarity relation. We then apply transitive closure over this relation, in order to obtain the desired partition.

[1] All the reported results can be readily extended to any finite sum of weighted density functions.

We begin by introducing some notations and stating basic results:

Definition 2.1. *For a density function h, let $V_\Sigma(h)$ denote the sum $V_\Sigma(h) = \sum_{i=1}^{n} Var(h^j)$, where h^j is the projection of h along the j-th axis.*

Lemma 2.2.

$$V_\Sigma(h) = \frac{1}{2}\int\int \|x-y\|^2 h(x)h(y)dxdy \tag{2.1}$$

Proof.

$$\begin{aligned}
\int\int \|x-y\|^2 h(x)h(y)dxdy &= \int\int \sum_{i=1}^{n}(x_i-y_i)^2 h(x)h(y)dxdy \\
= \quad & \sum_i \int\int x_i^2 h(x)h(y)dxdy + \sum_i \int\int y_i^2 h(x)h(y) - 2\sum_i \left(\int x_i h(x)dx\right)\left(\int y_i h(y)dy\right) \\
= \quad & \sum_i E_h(x_i^2) + \sum_i E_h(x_i^2) - 2\sum_i E_h(x_i)E_h(x_i) \\
= \quad & 2\left[\sum_i E_h(x_i^2) - E_h(x_i)^2\right] = 2V_\Sigma(h)
\end{aligned}$$

□

We shall now define what we mean by saying that F is approximated well by a set of kernel functions.

Definition 2.3. *Let F,$\{h_i\}_{i=1}^{M}$ be density functions. We say that $\{h_i\}$ approximate F with an error at most δ if $\|F - \sum_{i=1}^{M}\beta_i h_i\|_1 < \delta$ and $\|F - \sum_{i=1}^{M}\beta_i h_i\|_2 < \delta$.*

Definition 2.4. *Let f, g be a pair of density functions. f, g are said to be (ϵ, d) separated if there exists two disjoint sets $\mathbb{F}, \mathbb{G}$ in $\mathbb{R}^n$ such that:*

- $Dist(\mathbb{F}, \mathbb{G}) \geq d$
- $\|f_{\mathbb{F}^c}\|_\infty < \epsilon, \quad \|g_{\mathbb{G}^c}\|_\infty < \epsilon.$
- $\|f_{\mathbb{F}^c}\|_1 < \epsilon, \quad \|g_{\mathbb{G}^c}\|_1 < \epsilon.$

where $f_{\mathbb{F}}$ denotes the function f restricted to set $\mathbb{F}$, and $\mathbb{F}^c$ denotes the set $\mathbb{R}^n \setminus \mathbb{F}$.

With $F = \alpha_1 f + \alpha_2 g$ and assuming f, g are (ϵ, d) separated, we state the following lemma:

Lemma 2.5. *Let h be a kernel function used to approximate F; then*

1. $\|h_{\mathbb{F}}\|_1 \geq \frac{\langle f,h\rangle - \epsilon}{\|f\|_\infty}$
2. $\|h_{\mathbb{G}}\|_1 \geq \frac{\langle g,h\rangle - \epsilon}{\|g\|_\infty}$

Proof. We will demonstrate the proof of 1 (2 follows in the exact same way):

$$\langle f,h\rangle = \int fh = \int f_{\mathbb{F}}h_{\mathbb{F}} + \int f_{\mathbb{F}^c}h_{\mathbb{F}^c} \leq \|f\|_\infty \int h_{\mathbb{F}} + \|f_{\mathbb{F}^c}\|_\infty \int h = \|f\|_\infty \|h_{\mathbb{F}}\|_1 + \epsilon$$

. □

We can now claim the following:

Theorem 2.6. *If $Dist(\mathbb{F},\mathbb{G}) \geq d$ and h is a density function then*

$$V_\Sigma(h) \geq \frac{d^2}{2}\|h_{\mathbb{F}}\|_1\|h_{\mathbb{G}}\|_1$$

Proof.

$$\begin{aligned} V_\Sigma(h) &= \frac{1}{2}\int\int \|x-y\|^2 h(x)h(y)dxdy \\ &\geq \frac{1}{2}\int_{\mathbb{F}}\int_{\mathbb{G}} \|x-y\|^2 h(x)h(y)dxdy \\ &\geq \frac{d^2}{2}\int_{\mathbb{F}}\int_{\mathbb{G}} h(x)h(y)dxdy \\ &= \frac{d^2}{2}\|h_{\mathbb{F}}\|_1\|h_{\mathbb{G}}\|_1 \end{aligned}$$

□

Corollary 2.7. *If $V_\Sigma(h)$ is bounded by a parameter σ and f,g are (ϵ,d) separated, one of the following holds*

$$\|h_{\mathbb{F}}\|_1 \leq \frac{\sqrt{2\sigma}}{d}$$

or

$$\|h_{\mathbb{G}}\|_1 \leq \frac{\sqrt{2\sigma}}{d}$$

If $\|h_{\mathbb{F}}\|_1 \leq \frac{\sqrt{2\sigma}}{d}$ then

$$\langle f,h\rangle \leq \frac{\sqrt{2\sigma}\cdot\|f\|_\infty}{d} + \epsilon$$

and if $\|h_{\mathbb{G}}\|_1 \leq \frac{\sqrt{2\sigma}}{d}$ *then*

$$\langle g, h\rangle \leq \frac{\sqrt{2\sigma}\cdot\|g\|_\infty}{d} + \epsilon$$

Proof. Immediate from Theorem 2.6 and Lemma 2.5 □

With these results we can now state that:

Theorem 2.8. *If* $\|F - \sum_{i=1}^{M}\beta_i h_i\|_2^2 < \delta$, f, g *are* (ϵ, d) *separated, and* $V_\Sigma(h_i)$ *is bounded by* σ *for each* i, *then there exists a partition of the set* $\{h_i\}$ *into two disjoint subsets that provide good approximations of the functions* $\alpha_1 f$ *and* $\alpha_2 g$ *respectively, with error smaller than*

$$\delta + 2\left[\frac{\sqrt{2\sigma}\cdot max\left\{\|f\|_\infty, \|g\|_\infty\right\}}{d} + \epsilon\right]$$

Proof. For each i we observe the quantities $\|(h_i)_{\mathbb{F}}\|_1, \|(h_i)_{\mathbb{G}}\|_1$. By Corollary 2.7 one of them is smaller than $\frac{\sqrt{2\sigma}}{d}$. If it is the first we associate h_i with g and if it is the second we associate h_i with f. Let us denote by I_f the indices of the kernel functions associated with f, and by I_g the indices of the kernel functions associated with g. Then

$$\begin{aligned}
\delta &> \|F - \sum_{i=1}^{M}\beta_i h_i\|_2^2 = \|\alpha_1 f + \alpha_2 g - \sum_{i=1}^{M}\beta_i h_i\|_2^2 \\
&= \|\alpha_1 f - \sum_{i\in I_f}\beta_i h_i\|_2^2 + \|\alpha_2 g - \sum_{i\in I_g}\beta_i h_i\|_2^2 + 2\left\langle \alpha_1 f - \sum_{i\in I_f}\beta_i h_i, \alpha_2 g - \sum_{i\in I_g}\beta_i h_i\right\rangle
\end{aligned}$$

Rearranging terms, and using the fact that all the functions are positive density functions, we get

$$\begin{aligned}
&\|\alpha_1 f - \sum_{i\in I_f}\beta_i h_i\|_2^2 + \|\alpha_2 g - \sum_{i\in I_g}\beta_i h_i\|_2^2 < \delta - 2\left\langle \alpha_1 f - \sum_{i\in I_f}\beta_i h_i, \alpha_2 g - \sum_{i\in I_g}\beta_i h_i\right\rangle \\
&\leq \delta + 2\left[\left\langle \alpha_1 f, \sum_{i\in I_g}\beta_i h_i\right\rangle + \left\langle \alpha_2 g, \sum_{i\in I_f}\beta_i h_i\right\rangle\right] = \delta + 2\left[\alpha_1\sum_{i\in I_g}\beta_i\left\langle f, h_i\right\rangle + \alpha_2\sum_{i\in I_f}\beta_i\left\langle g, h_i\right\rangle\right] \\
&\leq \delta + 2\left[\alpha_1\sum_{i\in I_g}\beta_i\left(\frac{\sqrt{2\sigma}\cdot\|f\|_\infty}{d} + \epsilon\right) + \alpha_2\sum_{i\in I_f}\beta_i\left(\frac{\sqrt{2\sigma}\cdot\|g\|_\infty}{d} + \epsilon\right)\right]
\end{aligned}$$

$$\leq \quad \delta + 2 \cdot \left[\frac{\sqrt{2\sigma} \cdot max\left\{\|f\|_\infty, \|g\|_\infty\right\}}{d} + \epsilon \right] \cdot \left[\alpha_1 \sum_{i \in I_g} \beta_i + \alpha_2 \sum_{i \in I_f} \beta_i \right]$$

$$\leq \quad \delta + 2 \cdot \left[\frac{\sqrt{2\sigma} \cdot max\left\{\|f\|_\infty, \|g\|_\infty\right\}}{d} + \epsilon \right]$$

□

Corollary 2.9. *If $\|F - \sum_{i=1}^M \beta_i h_i\|_2^2 < \delta$, f, g are (ϵ, d) separated, and $V_\Sigma(h_i)$ is bounded by σ for each i, then there exists a partition of the set $\{h_i\}$ into two disjoint subsets that provide good approximations of the functions $\alpha_1 f$ and $\alpha_2 g$ respectively with an error of the order $O(\frac{\sqrt{\sigma}}{d})$.*

The results we have seen so far show that if f and g are well separated, and if F is approximated well by the kernel functions h_i, then there exists a partition of the set $\{h_i\}$ in a way that provides a good approximation of both f and g. However, these results give only a proof of existence, since f, g are usually unknown. In order to provide a way to construct these approximations, we need the following result:

Theorem 2.10. *Assume that $\{h_i\}_{i=1}^M$ approximate F with an error at most δ, f, g are (ϵ, d) separated, and $V_\Sigma(h_i)$ is bounded by σ for each i. Suppose that in such a partition h_1 is among the functions approximating $\alpha_1 f$ and h_2 is one of those approximating $\alpha_2 g$, and let β_1, β_2 denote their respective weights. Then the following inequality holds:*

$$\|\sqrt{h_1} - \sqrt{h_2}\|_2^2 \geq 2 - \left[2\sqrt[4]{\frac{2\sigma}{d^2}} + \frac{\epsilon + \delta}{\sqrt{\beta_1 \beta_2}} \right]$$

Proof.

$$\|\sqrt{h_1} - \sqrt{h_2}\|_2^2 = 2 - 2\left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle$$

where

$$\left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle = \left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{F}} + \left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{G}} + \left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{R}^n \setminus (\mathbb{F} \cup \mathbb{G})}$$

Denote $\mathbb{D} = \mathbb{R}^n \setminus (\mathbb{F} \cup \mathbb{G})$.
By the Cauchy-Schwartz inequality, the first term can be bounded by:

$$\left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{F}} \leq \|\sqrt{(h_1)_\mathbb{F}}\|_2 \|\sqrt{(h_2)_\mathbb{F}}\|_2 = \sqrt{\|(h_1)_\mathbb{F}\|_1 \|(h_2)_\mathbb{F}\|_1} \leq \sqrt{\|(h_2)_\mathbb{F}\|_1}$$

It follows from Theorem 2.6 that for a kernel function h, the following inequality holds:

$$\|h_{\mathbb{F}}\|_1\|h_{\mathbb{G}}\|_1 \leq \frac{2\sigma}{d^2}$$

and if h is associated with g then by definition $\|h_{\mathbb{F}}\|_1 \leq \sqrt{\frac{2\sigma}{d^2}}$. Therefore in our case $\|(h_2)_{\mathbb{F}}\|_1 \leq \sqrt{\frac{2\sigma}{d^2}}$, and it therefore follows that

$$\left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{F}} \leq \sqrt[4]{\frac{2\sigma}{d^2}}$$

The same argument applies to the second term replacing the roles of h_1 with h_2. As for the third term, we know that

$$\left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{D}} \leq \sqrt{\|(h_1)_{\mathbb{D}}\|_1\|(h_2)_{\mathbb{D}}\|_1}$$

by using the fact that

$$\|\alpha_1 f + \alpha_2 g - \sum \beta_i h_i\|_1 \leq \delta$$

and

$$\|(\alpha_1 f + \alpha_2 g)_{\mathbb{D}}\|_1 \leq \epsilon$$

We can deduce, using the triangle inequality, that $\|(\sum \beta_i h_i)_{\mathbb{D}}\|_1 \leq \delta + \epsilon$ and in particular for every i $\|(\beta_i h_i)_{\mathbb{D}}\|_1 \leq \delta + \epsilon$. Thus

$$\left\langle \sqrt{h_1}, \sqrt{h_2} \right\rangle_{\mathbb{D}} \leq \frac{\delta + \epsilon}{\sqrt{\beta_1 \beta_2}}$$

and the theorem immediately follows. □

The latter theorem states that if two kernel functions approximate different functions, there is a bound on how close they can be. It follows that by looking at pairs of kernel functions, we can readily determine for some pairs that they *must* approximate the same functions. By applying transitive closure to this "must" relation, we end up with a partition of the set of kernel functions, where each subset is guaranteed to include only kernel functions which all approximate either $\alpha_1 f$ or $\alpha_2 g$. Note that this by itself does not guarantee a complete partition of the kernel functions into two sets, as $\mathbb{F}$ and $\mathbb{G}$ were not assumed to be connected sets. Further assumptions can be made regarding f and g to assure that indeed exactly two disjoint subsets of the kernel functions are formed. However, one can overcome this obstacle without assuming anything further about the density functions, by incorporating constraints into our scheme during the clustering of the kernel functions [12].

Chapter 3

The GMBC Algorithm

3.1 The Algorithmic Scheme

The results described in the previous section motivate us to propose the Gaussian Mixture Based Clustering algorithm for data modeling. We assume our data set is composed of a finite number of classes T (T is not necessarily known), each drawn from a density function f_j, $j = 1...T$.

The algorithm first views the entire data set as a sampling of a single density function F (and hence, under our assumption, F is a weighted sum $F = \sum_{j=1}^{T} \alpha_j f_j$ with $\sum_{j=1}^{T} \alpha_j = 1$), and models it using a large number of small-variance Gaussians.

After such modeling has been obtained (i.e. F is approximated by $\sum_{i=1}^{M} \beta_i h_i$ with h_i Gaussian functions and $\sum_{i=1}^{M} \beta_i = 1$), the algorithm constructs for each density function f_j an approximation using a partial weighted sum of our Gaussian kernel functions $\{h_i\}_{i=1}^{M}$. The set of M Gaussians is therefore divided into T disjoint sets, resulting in a separate model for each of the density functions $\{f_j\}$, $j = 1...T$. Specifically:

Algorithm 1 The GMBC Algorithm

Given a data set generated from several density functions.

1. Approximate the function generating the data by a large number of small kernels (as though it is one density functions).
2. Cluster the kernel functions obtained in step 1 using an agglomerative (nearest-neighbor) clustering method.

Our choice for step 1 is a Gaussian Mixture Model (GMM) computed with the

EM algorithm initialized by K-Means. Each Gaussian model is assumed to have a diagonal covariance matrix. This assumption is made due to the otherwise vast number of parameters needed to be estimated, but it is well compensated for by allowing many models. The clustering in step 2 is done using the agglomerative single linkage algorithm, which clusters the means of the Gaussians.

The choice in step 1 guarantees that as the number of models in the GMM increases, the variance of each model decreases, since for any given data distribution, k narrow non-overlapping Gaussians provide a higher likelihood than k wide overlapping Gaussians. Thus we are assured that our kernel functions indeed have small variances.

3.2 Simulations

We now apply the GMBC algorithm to several data sets, and compare it to other clustering and data modeling algorithms. The quality of clustering the data is measured, as well as the ability to generalize the results by clustering additional test data after the learning process is complete. In addition, there is a visual demonstration of the quality of the model constructed by generating new data in accordance with the density function estimated, and comparing its resemblance to the original data.

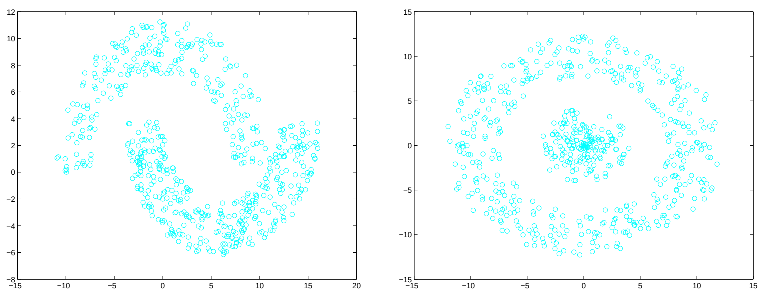
The choice for the agglomerative clustering algorithm is the single-linkage algorithm. The algorithms being used for comparison are:

1. The EM clustering and data modeling algorithm.

2. The single-linkage agglomerative algorithm.

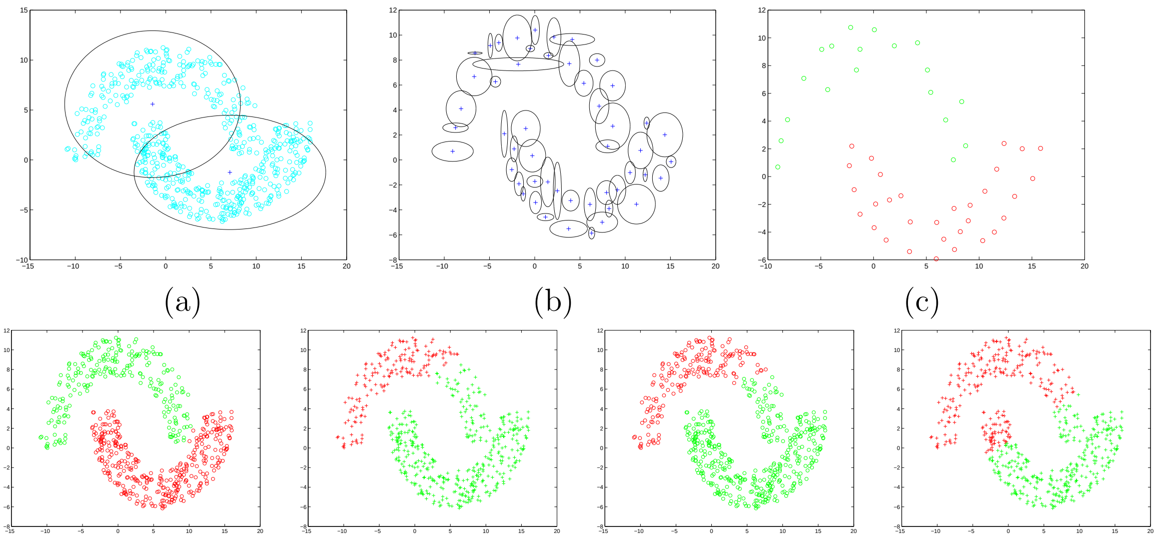
3. The Normalized Cut algorithm [14].

Both the "Ring" and "Bananas" synthetic data-sets are composed of two clusters which are visually easily distinguished. Since the spacial layout of the data in both samples is not concentrated in a convex region of $\mathbb{R}^n$ (see Fig 3.1), any attempt to model the data using two Gaussians will generate a misfitting model, and classification of new data will accordingly perform poorly. Figs. 3.2 and 3.4 show clustering results with the different algorithms. Figs. 3.3 and 3.6 show new points generated from the model learned.



(a) (b)

Figure 3.1: The data sets: 600 points in two clusters, generated by a density function $F = \alpha_1 f + \alpha_2 g$ with f, g both evenly distributed with $\alpha_1 = 0.4, \alpha_2 = 0.6$.
(a) The "Bananas" data set (b) The "Ring" data set.



(a) (b) (c)

GMBC Single Linkage Normalized Cut EM with 2 GMM

Figure 3.2: Clustering results for the Bananas dataset described in Fig 3.1a.
(a) The 2-source GMM model as computed by the EM algorithm. (b) Results of step 1 of the GMBC algorithm, approximating F with 50 Gaussians. (c) Clustering the Gaussian function means (showed in red and green).
Bottom row: clustering results with four different algorithms.

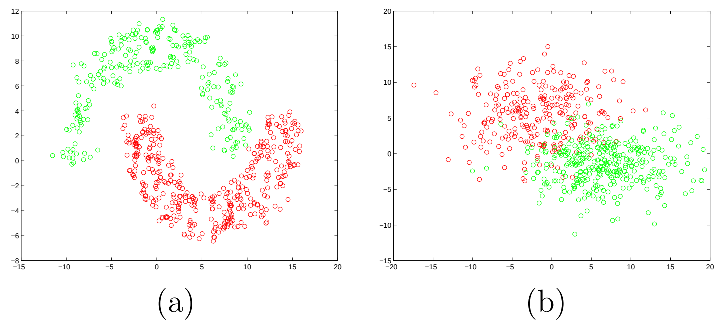


Figure 3.3: New data generated by the two generative methods:
(a) Points generated by the GMBC model. (b) Points generated by the computed 2-source GMM model.

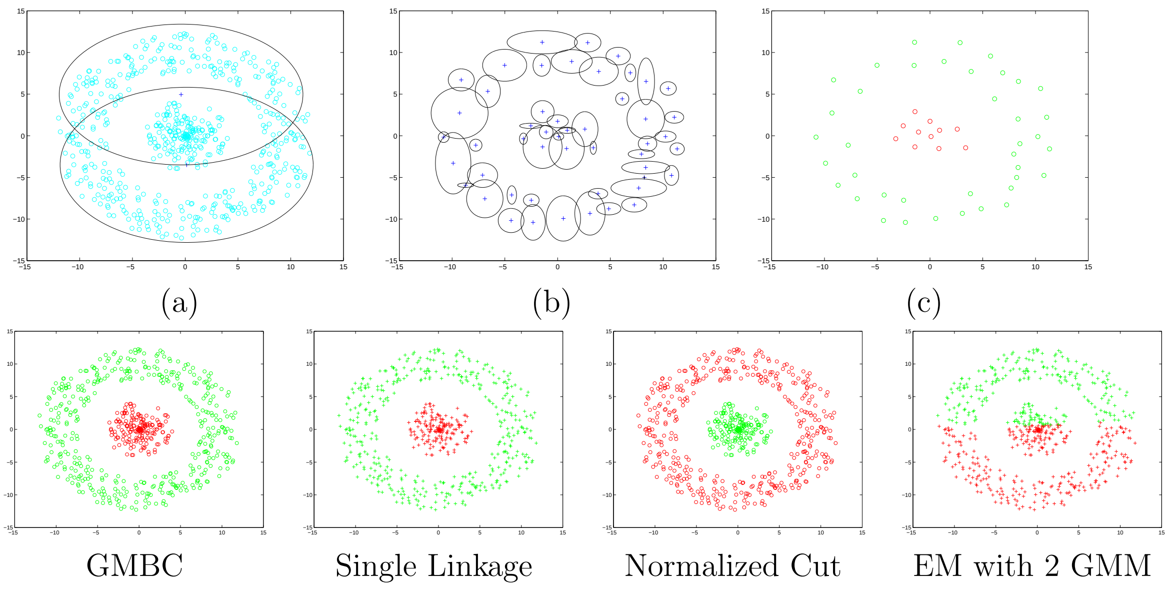
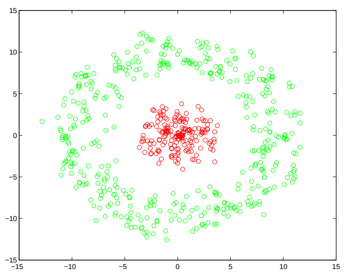


Figure 3.4: Clustering results for the Ring dataset described in Fig 3.1b.
(a) The 2-source GMM model as computed by the EM algorithm. (b) Results of step 1 of the GMBC algorithm, approximating F with 50 Gaussians. (c) Clustering the Gaussian function means (shown in red and green).
Bottom row: clustering results with four different algorithms.


(a)

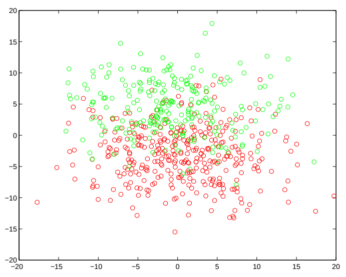

(b)

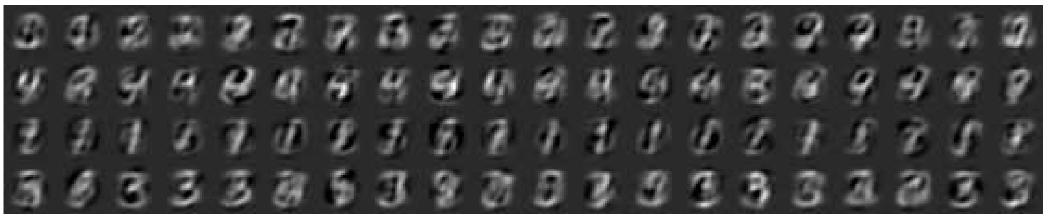
Figure 3.5: New data generated by the two generative methods:
(a) Points generated by the GMBC model. (b) Points generated by the computed 2-source GMM model.

3.3 Generating Digit Images

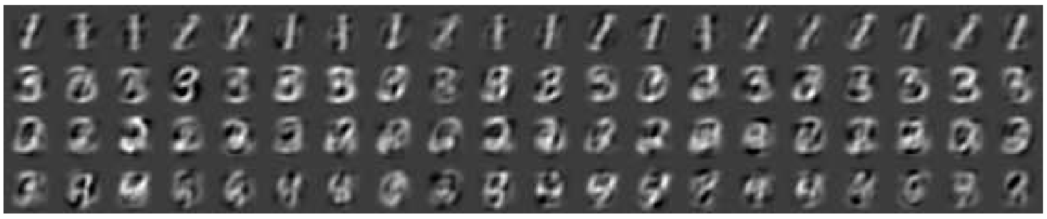
The MNIST data set is composed of images of digits. Each image size is 28×28 pixels, and there are 6000 different images for each digit. The aim in this case was to generate new images of digits, based on the model learned by the GMBC scheme. The results are compared to samples generated by the model learned by the EM algorithm, modeling each class with a single Gaussian.

Applying the scheme to the original data is not practical because of the high dimension, so the PCA dimensionality reduction method was used as preprocessing. The results, shown in Figs. 3.6,3.7, demonstrate that most of the important information was fully retained. What may have been lost is a certain degree of separation between classes, though for some digits it is likely that there was little separation to begin with; 1's and 7's are easily mixed up, as are 0's and 9's. What is also worth noting is that the algorithm does not need the entire set of samples for each class to generate a good model, and for each class only 1000 samples were used during learning.

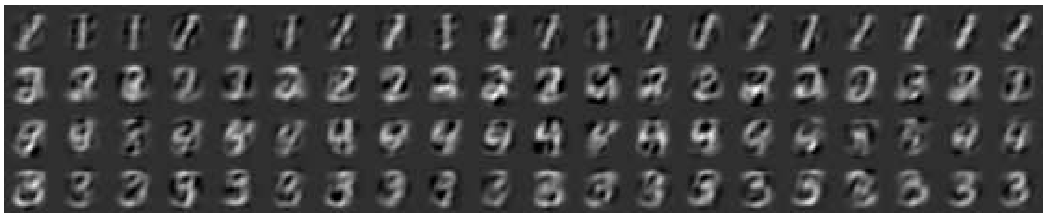
The images illustrate the quality of the model learned. When using only a single Gaussian to model each class, the new points generated do not reflect the true nature of the data, and digits are not clearly visible. As the number of Gaussians increases the model becomes more refined and the generated images are indeed images of digits. What may happen, something that can be seen as well, is that when the clustering is not precise, images generated out of the class model may resemble a different digit - but a digit nonetheless.



(a)

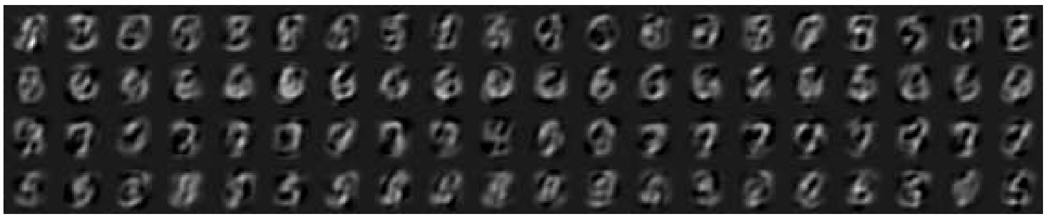


(b)

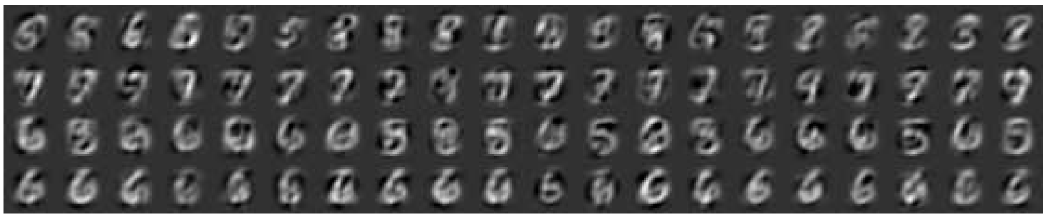


(c)

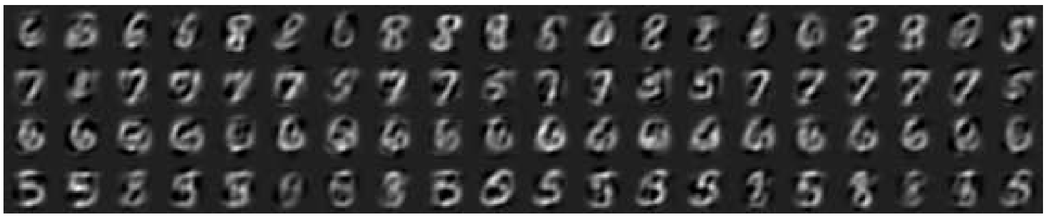
Figure 3.6: 1000 samples of each of the digits 1-4 reduced to dimension 20. (a) Modeling the data using 4 Gaussians (1 per class) (b) Using GMBC with 20 Gaussians (c) Using GMBC with 40 Gaussians



(a)



(b)



(c)

Figure 3.7: 1000 samples of each of the digits 5-8 reduced to dimension 20. (a) Modeling the data using 4 Gaussians (1 per class) (b) Using GMBC with 20 Gaussians (c) Using GMBC with 40 Gaussians

3.4 Discussion

The scheme described above presents an algorithmic continuum between affinity-based clustering and Gaussian Mixture Modeling in the following way: when observing a data set of size N, the finest approximation we can obtain for F is the weighted sum $F = \sum_{i=1}^{N} \alpha_i h_i$, with each function concentrated on a single data point. In a coarser approximation with $q < N$ kernels, finding the set of kernel functions that best approximates F is a problem of parameter estimation.

Since every model consisting of q kernel functions can be represented as a model consisting of $q' > q$ kernel functions, in terms of maximizing the likelihood score, there is a monotonic improvement in the approximation of F as we take more kernel functions. Hence once the number of kernel functions reaches and exceeds the number of data points, the best choice of parameters will be the one coinciding with the representation of F as $\sum_{i=1}^{N} \alpha_i h_i$. In this situation there is no longer any difference between the original data and the kernel functions, and the scheme reduces to simple clustering. On the other hand, if the number of kernel functions, in our case Gaussian functions, is the same as the number of classes, our scheme reduces to regular Gaussian Mixture Modeling.

It is important to point out that in practice the EM will not necessarily generate a model for each point, since when the number of Gaussians is nearly as high as the number of data points, the results of the EM are skewed due to erroneous parameter estimation. For the purpose of this discussion, however, we are only interested in the theoretical implications of this view, since what is the key feature here is what happens when we opt for a middle path, rather than choosing the algorithmic approaches of the extremities.

The discussion above suggests that there is a continuous transition between discriminative and generative methods, as opposed to the traditional view of them as two separate and completely different approaches. The parameter governing that transition in our model is the number of Gaussian functions in the approximation model. The drawbacks of using either clustering or GMM alone are well known. By choosing to cluster the data, one does not obtain a concise description of the underlying density functions generating the data. This results in poor handling of unseen data and inability to generate new samples. This corresponds to applying our scheme using a large number of Gaussian functions with very small variances. When viewing the data this way, we lose all prediction value; the model can only explain the observed points, and these are the only points it can generate.

On the other hand, by choosing to fit the data with a GMM consisting of a single

Gaussian function per class, the resulting model is often crude and misleading. This is because only rarely is a class generating function a simple Gaussian function. This point was illustrated in [7] within the framework of semi-supervised learning. It showed that when using a large amount of unlabeled data for learning based upon an incorrect model, one may get poorer results that those obtained by using just the labeled data set.

The optimal choice of the number of Gaussians is clearly dependent on the data. The natural question that arises, therefore, is what parameters of the data determine these dependencies - and in what way. Though this question is beyond the scope of this work, the results so far do suggest that the **density of the data** has very much to do with the choice of the number of Gaussians. If the data points are dense in a certain region D of $\mathbb{R}^n$, we can assume that this region lies in the support of the density function f which generated the data. Therefore a Gaussian with relatively large variance (relative to the volume of D) will present a good local approximation of f. On the other hand, when the data points are sparse in that region, there is very little information as to the true behavior of f in D. Approximating f with a large Gaussian in therefore "a shot in the dark" and is likely to be erroneous. In such a case, it is a better strategy to split D into smaller sub-regions - and approximate f on each separately - thus reducing the size of the Gaussian variances - and increasing the total number of functions used for the approximation. This strategy can be used repeatedly only to a certain extent, since the EM algorithm produces poor results when the variances are very small and there is little information. Therefore, an optimal choice of the number of Gaussians will, in essence, follow a min-max principle: the "largest" small variance Gaussians, taking into account the dimensionality of the data and their density.

A second key factor is the dimension n. Though the magnitude of n has practically no theoretical implications , it has substential impact when the scheme is put to the test with real data. The reason, again, has to do with the density of the data. If we observe a portion of the data embedded in an n-dimensional hull D, we see that the volume of D is exponentially dependent on n. Therefore, the larger the n the sparser the data become. As noted above, the strategy would therefore be to attempt to approximate the data with smaller Gaussians, but with so little information, parameter estimation becomes less stable. For these reasons, the GMBC algorithmic scheme will perform better when the data are embedded in a lower-dimensional space. Often, this need not be a deterring factor: dimensionality reduction is commonly used, and methods like the PCA algorithm, attempt to reduce the dimension while retaining most of the information.

The choice of nearest neighbor based clustering for the scheme is also worth addressing. As noted in the introduction, the matter of approximating an adequate distance function may be an intricate problem even when the elements involved are the embedding of our data in $\mathbb{R}^n$, and even more so when our elements now lay in the function space.

Our key assumption is that the data give us a good local reflection of the density function f. Under this assumption, the analysis presented above shows that as long as the bound on the variances of the kernel functions is significantly smaller than the distance between the support of the functions we approximate, any "reasonable" distance function in $\mathbb{R}^n$ will suffice. Our choice for a distance function was the distance between Gaussian means as a way of reducing the problem back to measuring the distance in $\mathbb{R}^n$ (as opposed to the function space), yet there are other possibilities that may be worth exploring in the future, such as the Jensen-Shannon distance between functions.

For some data sets, however, the noted assumption does not necessarily apply to the entire support of f, so sparse data on certain regions may not fully reflect the behavior of the function, and distant samples may be classified erroneously. In addition, the region of the support is not necessarily connected, and so one may want to add additional information that will link the connected components. Both these problems can be resolved by adding constraints to the learning process. Though in its general form the scheme is an unsupervised learning process, due to its double layer it can be readily extended to incorporate additional data in the form of constraints. Incorporating both negative and positive constraints is significantly easier in the discriminative framework, and usually involves tackling a distance or similarity matrix. By transforming constraints between data points to constraints between the Gaussians they are assigned to, incorporating the constraints has little impact on the running time and introduces a way to locally generalize the constraints.

Chapter 4

Summary

The objective of this work was to offer a unifying view of generative and discriminative methods, viewing the classic approaches as the extremities of a continuous spectrum. By laying the theoretical foundation for this view, insights were gained as to what may be the form of an algorithm in the continuum. Consequently we have described a scheme of data modeling that allows us to reduce the risk of fitting an over-simplified model on the one hand, while avoiding the loss of prediction power caused by merely clustering the data on the other.

Applying the scheme to various data sets presented the strength of a middle path in comparison to algorithms that are solely generative or discriminative, both in clustering the data, and classifying new data points. In addition, a graphic illustration of the model learned was presented by generating new data points out of the constructed model - and comparing the resemblance of the new data to the original data set.

The scheme was also applied to the MNIST data set. As noted above, this scheme works well when the data points are relatively dense, and this therefore requires the dimension to be considerably smaller that the full dimension of the data. Reduction of dimensions in this particular case, however, drastically affects the learning process, since the separation between the clusters is lost. Having said that, when we generate new data points and construct the corresponding images, it is clearly visible that the model constructed using the scheme generates results that have greater resemblance to the original data - i.e. that look like digits. This suggests that though some fine tuning may need to be applied for enhanced results, the presented scheme does provide a fresh and useful approach to data modeling.

Bibliography

[1] N. Shental A. Bar-Hiller, T. Hertz and D. Weinshall. Learning distance functions using equivalence relations. *CVPR*, pages 570–577, 2004.

[2] M. N. Murty A. K. Jain and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, pages 264–323, 1999.

[3] I. Shimshoni B. Georgescu and P. Meer. Mean shift based clustering in high dementions: A texture classification example. *In proc. Conf. Comp. Vision*, pages 456–463, 2003.

[4] S. Basu. Semi-supervised clustering: Learning with limited user feedback.

[5] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *Tech. Rep ICSI-TR-97-021*, 1998.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience Publication, 2000.

[7] I. Cohen F.G Cozman and M.C. Cirelo. Semi-supervised learning of mixture models. *In Proc of ICML*, pages 99–106, 2003.

[8] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, San Diego, 2nd edition, 1990.

[9] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[10] S. Rogers K. Wagstaff, C. Cardie and S. Schroedl. Constrained K-means clustering with background knowledge. *In Proc. of ICML*, pages 577–584, 2001.

[11] D. Miller and S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. *Advances In Neural Information Proccessing Systems*, 9:571–578, 2003.

[12] T. Hertz N. Shental, A.Bar-Hilel and D. Weinshall. Computing gaussian mixture models with EM using equivalence constrains. *Advances In Neural Information Proccessing Systems*, 15, 2003.

[13] M. Seeger. Learning with labeled and unlabeled data. 2001.

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000.

[15] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. *Advances In Neural Information Proccessing Systems*, 11:606–612, 1998.

[16] Y. Weiss. Segmentation using eigenvectors: a unifying view. *Proceedings of IEEE*, pages 975–982, 1999.

[17] D. Weinshall Y. Gdalyahu and M. Werman. Self organization in vision. *IEEE Trans.*, pages 1053–1074, 2001.