Estimation of Gaussian Mixture Networks

Author:
Andriy Dmytruk

Supervisor:
Ryan Browne

Associate Professor,
Dept. of Statistics and
Actuarial Science,
University of Waterloo

Research paper submitted in fulfillment of the requirements for the degree of Master of Mathematics (MMath) in
Computational Mathematics
University of Waterloo

Waterloo, Ontario, Canada, 2022

Abstract

Clustering is one of the most important topics in unsupervised learning. Generative models solve the problem of clustering by fitting to the distribution of the data. More flexible models allow better fitting to distributions, thus estimating better clusters. At the same time, overly unconstrained models often are prone to overfitting and require to be restricted with certain assumptions. In this work we propose a highly flexible Gaussian Mixture Network model and discuss fitting strategies to avoid overfitting of the model. We show improved performance compared to other mixture models on several real-world datasets. We provide the implementation in Python[1,2].

[1]GitHub repository: https://github.com/KangaroosInAntarcitica/mixes
[2]Python package: https://pypi.org/project/mixes/

Table of Contents

Chapter 1

Introduction

The problem of clustering is one of the most important topics in unsupervised learning. While it is impossible to meaningfully cluster any given dataset, certain reasonable assumptions are introduced such as compactness of data points in a cluster. Gaussian mixture models (GMM) described by Reynolds, 2009 use the assumption that each cluster is generated by samples of a normally-distributed variable centered around its mean. In many applications this however might not be a reasonable model, as real-world data often can have very complex distributions and require more flexible models. For example, skew Gaussian mixture models (Zhang and El-Shaarawi, 2010, Lin, 2009) extend GMMs and fit asymmetric distributions better. Franczak et al., 2013 propose using asymmetric Laplace distribution mixtures instead of GMMs and show improved performance with real-world data. A better fit to the data is able to produce more informative clusters and density estimates of the distribution (Tang et al., 2012), a highly important task for generative models (Harshvardhan et al., 2020).

While increasing the number of parameters of the model generally produces a better fit on a given dataset, such models become prone to overfitting (Tang et al., 2012). Therefore more flexible models with fewer parameters are preferred. Mixtures of factor analyzers (MFA) (Ghahramani, Hinton, et al., 1996) model concurrently performs clustering and dimensionality reduction, and as a result, fits Gaussian distribution with fewer parameters which reduces both computational time and overfitting for high-dimensional data.

An approach that was inspired by the success of deep neural networks in discriminative learning is to create networks of simpler distributions conditioned on each other in a layer-wise manner.

Deep mixtures of factor analyzers (DMFA) (Tang et al., 2012) propose to improve the factorization performance by stacking layers MFAs. The next layer MFAs are fitted conditionally on the previous layer. The proposed training is layer-wise where the next layer mixtures would be fitted on transformed data from the previous layer. Although this improves the resemblance of distributions with real data, the number of parameters is increased exponentially for deeper layers, and the proposed method does not fully benefit from the layered structure, as the first layer is fitted independently of the other.

Deep Gaussian mixture models (DGMM) create a network of Gaussian distributions where each layer is a Gaussian mixture (Van den Oord and Schrauwen, 2014; Viroli and McLachlan, 2017). This provides a more flexible model which was shown to be advantageous to other clustering techniques on many datasets. Van den Oord and Schrauwen, 2014 and Viroli and McLachlan, 2017 propose repeatedly utilizing the EM algorithm to fit the model. While the model is similar to DMFA and allows to perform dimensionality reduction, the authors propose to share the parameters of the higher layers among clusters of lower layers, which results in a smaller number of parameters.

Making more flexible models allows to better fit to a variety of distributions, but achieving a good fit is a challenging task itself. Therefore developing optimization techniques is an important task in unsupervised learning. Common methods for optimization of mixture models are Expectation Maximization (EM) (Dempster et al., 1977) and Minorization-Maximization (MM) (Sun et al., 2016). Stochastic Expectation-Maximization (SEM) and Monte-Carlo EM algorithms estimate the expectation by simulation and require much less computational time than EM to reach similar model performance (Nielsen, 2000). Gradient descent methods can be used inside the EM algorithm for optimization, or standalone as proposed by Gepperth and Pfülb, 2021, who efficiently fit GMM by stochastic gradient descent on streaming data.

Optimization is a complicated task since log-likelihood is often a non-concave function with multiple modes. It is a well-known practice to fit a model on multiple initial parameters to avoid termination in inferior modes. Zhou and Lange, 2010 propose a few methods of deterministic annealing that are shown to terminate the EM algorithm in dominant modes more frequently.

Chapter 2

Preliminaries

2.1 Clustering formulation

Let y be an observed variable with $\mathcal{D} = \{y_1, \dots, y_n\}$ being a dataset of points from the distribution of y. We will assume that the points can be grouped into k clusters represented by a discrete hidden variable $z \in \{1, \dots, k\}$. The task of hard clustering is to assign a value to z_i for each of the points $y_i \in \mathcal{D}$. The problem can be further extended by defining soft clustering, which assigns probabilities $p(z \mid y_i)$, $\forall z \in \{1, \dots, k\}$. A good clustering model is also able to generalize to the whole distribution of y and get good predictions for unseen points $y \notin \mathcal{D}$.

2.2 Gaussian distribution

Let variable y belong to a Gaussian distribution (also called normal distribution) with mean μ and covariance matrix Σ. Then it has the following probability density function:

$$p(y) = \mathcal{N}(y \mid \mu, \Sigma) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)\right).$$

We will also denote the fact that y has Gaussian distribution as $y \sim \mathcal{N}(\mu, \Sigma)$.

When $\Lambda\Lambda^T + \Psi = \Sigma$, the formulation can be alternatively rewritten in a stochastic form:

$$y = \mu + \Lambda z + u, \quad \text{where } z \sim \mathcal{N}(0, I),\ u \sim \mathcal{N}(0, \Psi)$$

where Λ is a square matrix, Ψ is a positive definite diagonal matrix, 0 is a zero vector and I is the identity matrix. Stochastic formulation shows how given samples of z we can generate samples of y. Conditional probability can easily be deduced from the formulation above: $p(y \mid z) = \mathcal{N}(\mu + \Lambda z, \Psi)$. Using this we can deduce the whole probability density function as

$$p(y) = \mathcal{N}(y \mid \mu, \Lambda\Lambda^T + \Psi).$$

2.3 Mixture Models

A mixture model with k components is a model with the following probability density function:

$$f(y \mid \tau, \Theta) = \sum_{j=1}^{k} \tau_j p(y \mid \Theta_j), \quad \text{where } \sum_{j=1}^{k} \tau_j = 1.$$

It assumes that the observed variable y is generated by component $j \in \{1, \dots, k\}$ with probability τ_j and that the component is distributed according to $p(y \mid \Theta_j)$ depending on its set of parameters Θ_j. If we create a hidden discrete variable s that corresponds to the component that generated a particular value of y, we can perform clustering by calculating the probability $p(s \mid y)$ using Bayes' rule.

A Gaussian Mixture Model (GMM) described by Reynolds, 2009 is a mixture model where each component j is assumed to be normally distributed: $p(y \mid \Theta_j) = \mathcal{N}(y \mid \mu_j, \Sigma_j)$.

A Mixture of Factor Analyzers (MFA) model proposed by Ghahramani, Hinton, et al., 1996 is a mixture model where each component is assumed to have the following stochastic form:

$$y = \mu_j + \Lambda z + u, \quad \text{where } z \sim \mathcal{N}(0, I), u \sim \mathcal{N}(0, \Psi).$$

Similarly to GMM, each component is normally distributed. Additionally, Λ is allowed to be rectangular, therefore the model has fewer parameters. This gives it the capability of performing dimensionality reduction by calculating $E(z \mid y)$ for the hidden variable z, since $\dim(z) < \dim(y)$.

Various other probability density functions can be used in mixture models, like the skew normal distribution described by Lin, 2009.

2.4 Expectation Maximization Algorithm

A measure of how well a model fits is its log-likelihood:

$$\log \mathcal{L}(\Theta \mid y) = \log p(y \mid \Theta).$$

For a model with hidden variable z, the log-likelihood is:

$$\log \mathcal{L}(\Theta \mid y) = \log \int_z p(y, z \mid \Theta) dz \log \int_z p(y \mid z, \Theta) p(z \mid \Theta) dz$$

If we fix the current value of model parameters Θ', the marginal log-likelihood function w.r.t. Θ is:

$$E_{z|y,\Theta'} \log p(y, z \mid \Theta).$$

The idea of the Expectation Maximization algorithm (EM) proposed by Dempster et al., 1977 is based on the fact that an increase in the value of the marginal likelihood is guaranteed to increase the value of the whole log-likelihood. Therefore EM repeatedly fixes Θ' and performs maximization on the marginal likelihood function w.r.t. Θ.

The algorithm can be summarised in these 2 steps:

E step: Calculate the probabilities $p(z \mid y, \Theta')$. One can extend this step by also calculating values directly dependent on these probabilities, like $E_{z|y,\Theta'}(zz^T)$.

M step: Perform a step of maximization of the marginal log-likelihood using values from the E step. Update parameters with the maximizer Θ.

The optimization is commonly performed by finding an extreme point of the function. Alternatives include the Minorize-maximization algorithm proposed by Sun et al., 2016 and gradient descent optimization methods.

EM is commonly used for maximizing mixture models with the observed variable y and hidden variable s representing the cluster association.

Chapter 3

Approach

3.1 Gaussian Mixture Networks

Gaussian Mixture Networks (GMN) proposed in this paper are based on the ideas presented by Tang et al., 2012, Van den Oord and Schrauwen, 2014 and Viroli and McLachlan, 2017. We will first explain the structure of a GMN, and then discuss the relation to current models.

GMN extends a GMM by creating a network of Gaussian distributions. Each layer of the network represents a mixture and each node is a component. This structure can be seen in Figure 3.1. Let L be the number of layers of the network. Let $k_l, \forall l \in \{1, \ldots, L\}$ represent the size of each layer. Also, let z_l represent the variable generated by layer l. Then z_{l+1} is the input variable of layer l. We will take $z_1 = y$ to be the output of the model.

Since each layer is considered a mixture, the output z_l that is generated from the layer is generated by each component of the layer with a certain probability. Let $s_l \in \{1, \ldots, k_l\}$ denote a discrete variable corresponding to the component on a particular layer l that generated z_l. We will denote the set of possible values of the variable s_l as $\mathcal{S}_l$. The output y is then generated by a path of components $s = (s_1, \ldots, s_L), s_l \in \mathcal{S}_l, \forall 1 \leq l \leq L$. An example of a path can be seen in figure 3.3. We will denote the set of all paths as $\mathcal{S} = \bigtimes_{l=1}^{L} \mathcal{S}_l$ where $\bigtimes$ is the Cartesian product.

We will now describe the exact distributions of each variable z_l. Let $z_{L+1} \sim \mathcal{N}(0, I)$ be generated by a multivariate standard normal. Each component s_l on layer l can be described in stochastic form as $z_l = \eta_{s_l}^{(l)} + \Lambda_{s_l}^{(l)} z_{l+1} + u_{s_l}^{(l)}$ where $u^{(l)} \sim \mathcal{N}(0, \Psi_{s_l}^{(l)})$ with diagonal matrix $\Psi_{s_l}^{(l)}$. The component performs transformation of its input while adding Gaussian noise. Each component has Gaussian distribution w.r.t. its input:

$$p(z_l|z_{l+1}, s_l) = \mathcal{N}\left(y \mid \eta_{s_l}^{(l)} + \Lambda_{s_l}^{(l)} z_{l+1}, \Psi_{s_l}^{(l)}\right) \tag{3.1}$$

The probability of each component s_l being used for generation on a given layer will be denoted $\tau_{(s_l, s_{l+1})}^{(l)} = p(s_l \mid s_{l+1})$ and is conditional on the selection of component of the previous layer. For the last layer, we can consider $s_{L+1} = 1$ to be constant but may omit it from paths and notations for simplicity. The set of all parameters of a node is $\Theta_{s_l}^{(l)} = \left\{\eta_{s_l}^{(l)}, \Lambda_{s_l}^{(l)}, \Psi_{s_l}^{(l)}, \tau_{(s_l, s_{l+1})}^{(l)}, \quad \forall s_{l+1} \in \mathcal{S}_{l+1}\right\}$.

The whole model can be described in stochastic form by the set of equations:

$$\begin{cases} y = z_1 = \eta_{s_1}^{(1)} + \Lambda_{s_1}^{(1)} z_2 + u_{s_1}^{(1)} & \text{with probability } \tau_{(s_1, s_2)}^{(1)}, \\ \quad z_2 = \eta_{s_2}^{(2)} + \Lambda_{s_2}^{(2)} z_3 + u_{s_2}^{(2)} & \text{with probability } \tau_{(s_2, s_3)}^{(2)}, \\ \quad \dots & \\ \quad z_L = \eta_{s_L}^{(L)} + \Lambda_{s_L}^{(L)} z_{L+1} + u_{s_L}^{(L)} & \text{with probability } \tau_{(s_L)}^{(L)}, \\ z_{L+1} \sim \mathcal{N}\left(0, I\right), & \\ u_{s_l} \sim \mathcal{N}\left(0, \Psi_{s_l}^{(l)}\right), \forall s_l, \forall l & \text{where } \Psi_{s_l}^{(l)} \text{ is diagonal .} \end{cases} \tag{3.2}$$

See figures 3.1 and 3.2 for visualization of the model and a particular node from the model. The variables z_l, $\forall l \in \{2, \dots, L+1\}$ are considered hidden variables, as are $s_l, \forall l \in \{1, \dots, L\}$ while y is the observed output.

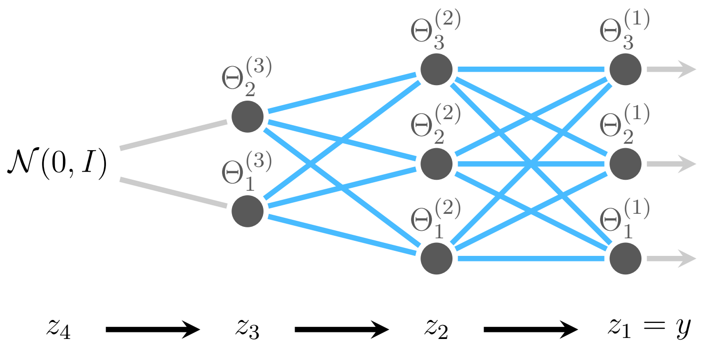
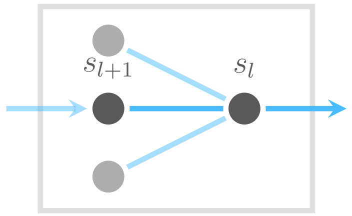


Figure 3.1: A schematic of a Gaussian Mixture Network. The network has $k = (3, 3, 2) = (k_1, k_2, k_3)$ and therefore the total number of paths is $k_1 \cdot k_2 \cdot k_3 = 18$. Each node represents a distribution dependent on nodes from the next layer with parameters written on top of it. Nodes are described in more detail in figure 3.2.



$$z_l = \eta_{s_l}^{(l)} + \Lambda_{s_l}^{(l)} z_{l+1} + u_{s_l}^{(l)}, \qquad u_{s_l}^{(l)} \sim \mathcal{N}(0, \Psi_{s_l}^{(l)})$$

With probability $\tau_{(s_l, s_{l+1})}^{(l)}$

Figure 3.2: A particular node of the network. If the value on the next layer is z_{l+1} and was generated by node s_{l+1}, this node has probability $\tau_{(s_l, s_{l+1})}^{(l)}$ of being used and will generate z_l by the formula above.

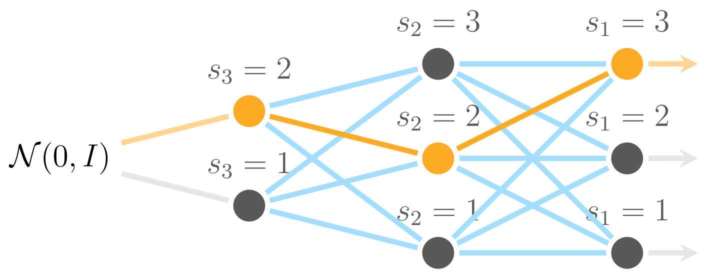


Figure 3.3: An example of a single path through GMN. Path is $s = (3, 2, 2) = (s_1, s_2, s_3)$. The path has probability $\pi_s = \tau_2^{(3)} \tau_{2,2}^{(2)} \tau_{3,2}^{(1)} = \tau_{(s_3)}^{(3)} \tau_{(s_2, s_3)}^{(2)} \tau_{(s_1, s_2)}^{(1)}$.

3.1.1 Dimensionality Reduction

The number of parameters can be greatly reduced by lowering the dimensions of variables on deeper layers. For the first layer $\dim(z_1) = \dim(y)$ is fixed and is equal to that of our observed data. We can reduce the dimensions of z_l, $\forall l \geq 2$ by reducing the dimensionality of parameters $\Psi_{s_l}^{(l)}, \eta_{s_l}^{(l)}$ and making $\Lambda_{s_{l-1}}^{(l-1)} \in \mathbb{R}^{\dim(z_{l-1}) \times \dim(z_l)}$. Note, that equations 3.2 and the model description above remain valid.

With dimensionality reduction, the layers become more similar to MFA models than GMMs, but we will retain the naming of Gaussian Mixture Networks due to the conditional normal distribution of each node in the network.

3.1.2 Path distributions in GMN

We can calculate the probability of a given path with

$$\pi_s = p(s) = \prod_{i=1}^{L} p(s_l | s_{l+1}) = \prod_{i=1}^{L} \tau_{(s_l, s_{l+1})}.$$

Given the probability of a path, the joint probability of variables $y, z = \{z_2, \ldots, z_{L+1}\}$

$$p(y, z \mid s) = \left(\prod_{i=1}^{L} p(z_l \mid z_{l+1}, s_l) \right) p(z_{L+1}).$$

Let $s_{(l:)} = (s_l, \ldots, s_L)$ denote a path through the network starting at layer l. It can be easily seen that the probability $p(y|s)$ has Gaussian distribution as does $p(z_l|s_{(l:)})$:

$$p(z_l|s_{(l:)}) = \mathcal{N}(z_l|\mu_{s_{(l:)}}, \Sigma_{s_{(l:)}}). \tag{3.3}$$

Let $\mu_{s_{(l:)}}$ and $\Sigma_{s_{(l:)}}$ denote the mean and covariance matrix of the probability distribution for path $s_{(l:)}$. These parameters can be easily calculated in a recursive manner:

$$\begin{cases} \mu_{s_{(l:)}} & = \eta_{s_l}^{(l)} + \Lambda_{s_l}^{(l)} \mu_{s_{(l+1:)}}, \\ \Sigma_{s_{(l:)}} & = \Psi_{s_l}^{(l)} + \Lambda_{s_l}^{(l)} \Sigma_{s_{(l+1:)}} (\Lambda_{s_l}^{(l)})^T, \\ \pi_{s_{(l:)}} & = \tau_{(s_l, s_{l+1}}^{(l)} \pi_{s_{(l+1:)}}, \\ \text{with} & \mu_{s_{(L+1:)}} = 0, \Sigma s_{(L+1:)} = I, \pi_{s_{(L+1:)}} = 1. \end{cases} \tag{3.4}$$

As an example, observe the model in figure 3.3. The path $s = (3, 2, 2)$, and $s_{(2:)} = (2, 2)$. We can calculate $\mu_{s_{(2:)}} = \eta_2^{(2)} + \Lambda_2^{(2)}\eta_3^{(3)}$, $\Sigma_{s_{(2:)}} = \Psi_2^{(2)} + \Lambda_2^{(2)}(\Psi_2^{(3)} + \Lambda_2^{(3)}\Lambda_2^{(3)T})\Lambda_2^{(2)T}$. Similarly $\mu_s = \eta_3^{(1)} + \Lambda_3^{(1)}\mu_{s_{(2:)}}$ and $\Sigma_s = \Psi_3^{(1)} + \Lambda_3^{(1)}(\Sigma_{s_{(2:)}})\Lambda_3^{(1)T}$. The probability of a given output for this path is then $p(y \mid s) = \mathcal{N}(y \mid \mu_s, \Sigma_s)$ and depends on the parameters $\{\Theta_3^1, \Theta_2^2, \Theta_2^3\}$.

3.1.3 Inverse distribution at a node in GMN

From equations 3.1 and 3.4 we have:

$$p(z_l|z_{l+1}, s_l) = \mathcal{N}\left(y \mid \eta_{s_l}^{(l)} + \Lambda_{s_l}^{(l)} z_{l+1}, \Psi_{s_l}^{(l)}\right),$$
$$p(z_{l+1}|s_{l+1}) = \mathcal{N}\left(z_{l+1}|\mu_{s_{(l+1:)}}, \Sigma_{s_{(l+1:)}}\right)$$

As was proposed by Viroli and McLachlan, 2017 given a path $s_{(l:)}$ the inverse probability of 3.1 can be calculated using the Bayes' rule:

$$p(z_{l+1} \mid z_l, s_l) = \frac{p(z_l \mid z_{l+1}, s_l)p(z_{l+1} \mid s_{(l+1:)})}{p(z_l \mid s_{(l:)})}$$

Making algebraic simplifications, we get the probability function:

$$\begin{aligned} p(z_{l+1}|z_l, s_l) &= \mathcal{N}\left(z_{l+1} \mid \rho_{s_l}(z_l), \xi_{s_l}\right), \qquad \text{where:} \\ \xi_{s_l} &= \left((\Sigma_{s_{(l+1:)}})^{-1} + \Lambda_{s_l}^{(l)T}(\Psi_{s_l}^{(l)})^{-1}\Lambda_{s_l}^{(l)}\right), \\ \rho_{s_l}(z_l) &= \xi_{s_l}\left(\Lambda_{s_l}^{(l)T}(\Psi_{s_l}^{(l)})^{-1}(z_l - \eta_{s_l}^{(l)}) + (\Sigma_{s_{(l+1:)}})^{-1}\mu_{s_{(l+1:)}}\right). \end{aligned} \tag{3.5}$$

A detailed derivation of this is outlined in Appendix A. This result will be further used inside the Expectation-Maximization approach.

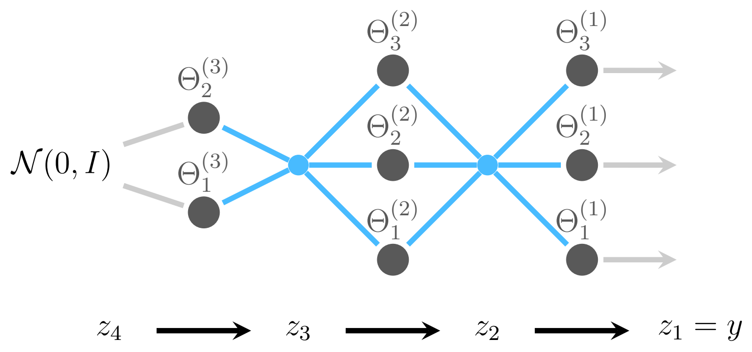


Figure 3.4: A Visualization of the structure of a Deep Gaussian Mixture Model (DGMM) with $k = (k_1, k_2, k_3) = (3, 3, 2)$.

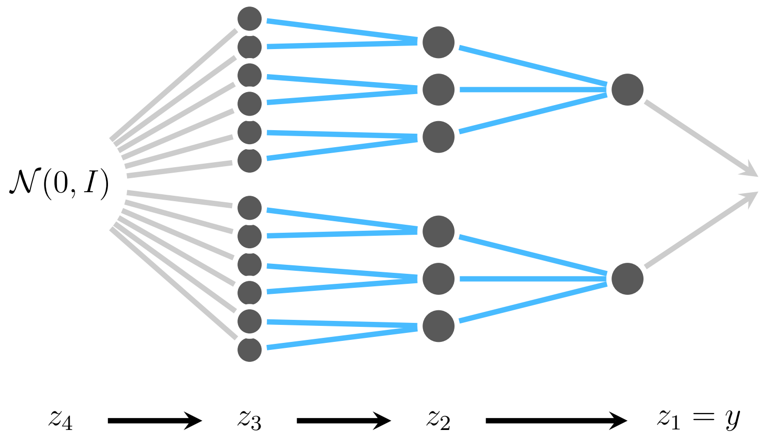


Figure 3.5: A Visualization of the structure of a Deep Mixture of Factor Analysers (DMFA) with $k_1 = 2, k_2 = 3, k_3 = 2$. Each next layer creates mixtures for each of the components of the previous layer. Therefore layer 3 has 12 components.

3.2 Comparison to existing models

3.2.1 Deep Mixtures of Factor Analyzers

Deep Mixtures of Factor Analyzers (DMFA) were described by Tang et al., 2012. The proposed model replaces the prior of a component s_l of layer l with a mixture:

$$p(z_l \mid s_l) = \mathrm{MFA}^{(l+1)}_{s_l}(z_l).$$

A separate mixture is trained for each of the components, therefore the number of components in layers grows exponentially, as can be seen in figure 3.5. It was shown by Tang et al., 2012 that such structure has significant advantages to simply increasing the number of components of an MFA model. Since deeper layers further reduce dimensions, the total number of parameters is reduced and the model is less prone to overfitting.

As can be seen, the model still has a lot of parameters, especially when the number of layers is large. Additionally, there is no parameter sharing between the components of deeper layers, which was shown to be extremely beneficial for discriminative networks. As described by Szegedy et al., 2015 deeper neural networks with more parameter tying on each layer, as in convolutional neural networks generally get better scores while avoiding overfitting better.

The structure of each node in DMFA is identical to the one of GMN, therefore it is not hard to verify that DMFA models are a subset of GMN models. By setting certain $\tau^{(l)}_{(s_l, s_{l+1})} = 0$ and increasing the number of components on subsequent layers, a GMN can be converted into a tree structure of a DMFA. GMN extends DMFA by allowing parameter sharing between different components, which allows to reduce the number of components on further layers. Additionally, we will update all the layer parameters on each EM step during the training of GMN instead of fitting each layer separately as proposed by Tang et al., 2012.

3.2.2 Deep Gaussian Mixture Models

Deep Gaussian Mixture Models (DGMM) described by Van den Oord and Schrauwen, 2014 and Viroli and McLachlan, 2017 are also a subset of GMN models with component probabilities not conditioned on the previous layer, or equivalently

$$\tau^{(l)}_{(s_l, s_{l+1})} = \tau^{(l)}_{(s_l, t_{l+1})}, \qquad \forall s_l \in \mathcal{S}_l, \quad \forall s_{l+1}, t_{l+1} \in \mathcal{S}_{l+1}.$$

However, unlike in GMN, this means that the prior of all the components at a given layer is identical:

$$\begin{aligned} p(z_{l+1} \mid s_l) &= \sum_{s_{l+1} \in \mathcal{S}_{l+1}} p(z_{l+1} \mid s_{l+1}) p(s_{l+1} \mid s_l) \\ &= \Big(\sum_{s_{l+1} \in \mathcal{S}_{l+1}} p(z_{l+1} \mid s_{l+1}) \Big) p(s_{l+1}), \end{aligned}$$

which provides less flexibility for the model by restricting the similarity between components of the same layer. Therefore DGMM graph can be visualized as in figure 3.4, where nodes of a layer l depend only on the input variable z_{l-1} and not all the nodes of ther previous layer.

We propose leveraging the advantages of both DMFA and DGMM models. Therefore GMN models are a generalized class of models containing both DMFA and DGMM. Note that Fuchs et al., 2022 and Viroli and Anderlucci, 2021 extend the DGMM model by using non-Gaussian distributions in nodes, which can also be applied to GMN.

We would like to note that Viroli and McLachlan, 2017 discuss the identifiability of the model, which is an important topic in model-based clustering. We will assume that by preventing overfitting, we will get more accurate clustering and less variation between trained models with different initialization. We will not restrict the number of components in layers, but similarly to Van den Oord and Schrauwen, 2014 assume that deeper models provide a better fit, while parameter-tying prevents overfitting.

3.3 Expectation-Maximization Approach

Let $\Theta^{(l)} = \{\Theta^{(l)}_{s_l} \ \forall s_l \in \mathcal{S}_l\} = \{\eta^{(l)}, \Lambda^{(l)}, \Psi^{(l)}, \tau^{(l)}\}$ be the set of all parameters of layer l and Θ the set of all the parameters of GMN.

Let $z = (z_2, \ldots, z_{L+1})$ denote all the continuous hidden variables of the model, and $s = (s_1, \ldots, s_L)$ denote the path variable with $\mathcal{S}$ being the set of all its possible values.

The Expectation Maximization algorithm when applied to a GMN will fix the current Θ' and will maximize the following objective function w.r.t. Θ:

$$\begin{aligned} &E_{z,s|y,\Theta'} \log L(\Theta; y, z, s) \\ =&E_{z,s|y,\Theta'} \left(\log p(y, z, s \mid \Theta)\right) \qquad (3.6) \\ =&E_{z,s|y,\Theta'} \big(\log p(y, s^{(1)} \mid (z,s)^{(2)}, \Theta^{(1)}) + \cdots + \log p((z,s)^{(L)} \mid (z,s)^{L+1}, \Theta^{(L)})\big) \end{aligned}$$

As can be seen, the log-likelihood function can be decoupled and minimized w.r.t. parameters $\Theta^{(l)}$ of each layer separately. For layer (l) it will be:

$$E_{z,s|y,\Theta'}\left(\log p((z,s)^{(l)} \mid (z,s)^{(l+1)}, \Theta^{(l)})\right) \tag{3.7}$$

Furthermore, the log-likelihood can be decoupled for each distribution in the layer separately. For a given distribution $\hat{s}_l$, we need to take the expectation only over the paths s passing through this distribution:

$$f_{\hat{s}_l}^{(l)} = E_{z,s|s_l=\hat{s}_l,y,\Theta'}\Big(\log \underbrace{p(z_l \mid \hat{s}_l, z_{l+1}, \Theta_{\hat{s}_l}^{(l)})}_{\mathcal{N}(z_l|\eta_{\hat{s}_l}^{(l)}+\Lambda_{\hat{s}_l}^{(l)} z_{l+1}, \Psi_{\hat{s}_l}^{(l)})} + \log \underbrace{p(s_l = \hat{s}_l)}_{\tau_{(\hat{s}_l, s_{l+1})}^{(l)}} \Big) \tag{3.8}$$

3.3.1 Minorization-Maximization variant

Using Jensen's inequality, we can get a lower bound at all points for the objective function by taking the log function out of the expected value in previous equation:

$$g_{\hat{s}_l}^{(l)} = \log\left(E_{z,s|s_l=\hat{s}_l,y,\Theta'}\left(p(z_l \mid \hat{s}_l, z_{l+1}, \Theta_{\hat{s}_l}^{(l)}) + p(s_l = \hat{s}_l)\right)\right) \leq f_{\hat{s}_l}^{(l)}.$$

The minorization-maximization (MM) algorithm proposes to maximize the lower bound function instead of $f^{(l)}$. At point Θ', we have $g_{\hat{s}_l}^{(l)}(\Theta') = f_{\hat{s}_l}^{(l)}(\Theta')$, and at all other points Θ we have $g_{\hat{s}_l}^{(l)}(\Theta) \leq f_{\hat{s}_l}^{(l)}(\Theta)$. Therefore, maximizing the lower bound guarantees to increase the original objective function. Since optimization w.r.t. $f_{\hat{s}_l}^{(l)}$ is infeasible for our complex model structure, we will utilize MM.

3.3.2 M Step

We will take the derivatives w.r.t. each parameter. For simplicity of notation, we will omit the subscript in expectation and assume that all the parameters are for layer (l) and distribution $\hat{s}$. We will also perform a change of variables $v = z_l, w = z_{l+1}$ We get:

$$\begin{cases} \frac{\partial g_{\hat{s}}^{(l)}}{\partial \eta} & = E\left(\Psi^{-1}(v - \eta - \Lambda w)\right) = 0, \\ \frac{\partial g_{\hat{s}}^{(l)}}{\partial \Lambda} & = E\left(\Psi^{-1}(v - \eta - \Lambda w)\, w^T\right) = 0, \\ \frac{\partial g_{\hat{s}}^{(l)}}{\partial \Psi} & = -\frac{1}{2} E\left(\Psi^{-1} - \Psi^{-1}(v - \eta - \Lambda w)(v - \eta - \Lambda w)^T \Psi^{-1}\right) = 0, \\ & \text{with } v = z_l, w = z_{l+1}. \end{cases}$$

Unlike Viroli and McLachlan, 2017, we will solve the equations as a system instead of each separately. We will describe the whole solution in Appendix B. As a result, we get an elegant solution:

$$\begin{cases} \Lambda & = \mathrm{Cov}(v, w) \left(\mathrm{Var}(w)\right)^{-1}, \\ \eta & = E(v - \Lambda w), \\ \Psi & = \mathrm{Var}(v - \Lambda w). \end{cases} \tag{3.9}$$

Alternatively, we can formulate the solution in a way that is easier to implement as part of the EM algorithm:

$$\begin{cases} \Lambda & = \left(E(vw^T) - E(v)E(w)^T\right) \cdot \left(E(w)E(w)^T - E\left(ww^T\right)\right)^{-1}, \\ \eta & = E(v) - \Lambda E(w), \\ \Psi & = E(vv^T) - 2E(vw^T)\Lambda^T + \Lambda E(ww^T)\Lambda^T - \eta\eta^T. \end{cases} \tag{3.10}$$

This system 3.10 requires fewer values to be estimated. The only expectations that need to be taken during the E step are $E(v)$, $E(w)$, (vv^T), $E(vw^T)$, $E(ww^T)$.

Maximizing the lower bound objective w.r.t. the parameters $\tau^{(l)}$ we get:

$$\tau^{(l)}_{(\hat{s}_l, \hat{s}_{l+1})} = E([s_l = \hat{s}_l] \cdot [s_{l+1} = \hat{s}_{l+1}])$$

where $[\cdot]$ is the indicator function s.t. $[x] = \begin{cases} 1 & \text{if } x \text{ is true,} \\ 0 & \text{otherwise} \end{cases}$, or equivalently:

$$\tau^{(l)}_{(\hat{s}_l, \hat{s}_{l+1})} = p(\hat{s}_l, \hat{s}_{l+1} \mid y, \Theta') \tag{3.11}$$

3.3.3 E Step

In system 3.10 the expectation function $E(\cdot) = E_{z,s|s^{(l)}=\bar{s},y,\Theta'}(\cdot)$. Although the parameter estimates require only the values of v and w, we still need to integrate over all the latent variables to calculate the expectation. The exact calculation over all the continuous variables is infeasible, so we will utilize its stochastic batch variant.

Therefore we can rewrite each expectation as:

$$\begin{aligned} E(\cdot) &= E_{z,s|s_l=\hat{s}_l,y,\Theta'}(\cdot) \\ &= E_{s|y,\Theta'}\left(E_{z_{(1:l)}|s_{(1:l)},y,\Theta'}\left(E_{z_{l+1}|z_l,s_l}(\cdot)\right)\right) \\ &= \sum_{z_l\in\mathcal{Z}_l}\left(E_{s_{(l+1:)}|z_l,\Theta'}\left(\frac{E_{z_{l+1}|z_l,s_l,\Theta'}(\cdot)}{|\mathcal{Z}_l|}\right)\right), \end{aligned} \tag{3.12}$$

where $\mathcal{Z}_l$ is a set of samples from the probability distribution $p(z_l \mid z_{l-1}, s_l, \Theta')p(s_l \mid z_{l-1}, \Theta')$. Here, $z_{l-1} \in \mathcal{Z}_l$ or is the observed variable from dataset $z_1 \in \mathcal{Z}_1 = \{y_1, \dots, y_n\}$. Therefore, when performing the E step, we will use samples calculated for the previous layer, and on the end of the step we will sample from the current distribution. We will be calculating the expected values over all the $s_{(l+1:L)}$ exactly.

While we can take samples of both v and w, we are required to sample only v to make the equations feasible.

For example, using the equations 3.5, we can calculate the values required by M step in system 3.10 as:

$$\begin{aligned} E_{z_{l+1}|z_l,s_l,\Theta'}(z_{l+1}) &= \rho_{s_l}(z_l) \\ E_{z_{l+1}|z_l,s_l,\Theta'}(z_{l+1}\,(z_{l+1})^T) &= \xi_{s_l} + \rho_{s_l}(z_l)\,(\rho_{s_l}(z_l))^T \\ E_{z_{l+1}|z_l,s_l,\Theta'}(z_l\,(z_{l+1})^T) &= z_l\,(\rho_{s_l}(z_l))^T \end{aligned} \tag{3.13}$$

Finally, to calculate the expectation over paths and equation 3.11 in M step we will calculate the probability $p(s_{(l:)} \mid z_l, \Theta')$ using the Bayes' rule and results from equations 3.3, 3.4:

$$\begin{aligned} p(s_{(l:)} \mid z_l, \Theta') &= \frac{p(s_{(l:)})p(z_l \mid s_{(l:)})}{p(z_l)} \\ &= \frac{p(s_{(l:)})p(z_l \mid s_{(l:)})}{\sum_{\hat{s}_{(l:)}\in\mathcal{S}_{(l:)}} p(\hat{s}_{(l:)})p(z_l \mid \hat{s}_{(l:)})}, \end{aligned} \tag{3.14}$$

and since $\mathcal{S}_{(l+1:)}$ is a subset of $\mathcal{S}_{(l:)}$ we can easily get $p(s_{(l+1:)} \mid z_l) = \sum_{s_l\in\mathcal{S}_l} p(s_{(l:)} \mid z_l)$.

3.3.4 Algorithm

A summary of the whole Expectation-Maximization algorithm for GMN is presented in algorithm 1. Note that in implementation, certain variables do not need to be stored, like

$E_{w|\hat{s}_l,\mathcal{D}}(w)$ could accumulate the values of $E_{w|s^{(l:)},\mathcal{D}}(w)$ over iterations $s_{(l:)} \in \mathcal{S}_{(l:)}$ instead of computing the sum afterward, therefore simplifying the implementation.

Van den Oord and Schrauwen, 2014 propose a simplified procedure for the expectation maximization that makes computation on large image datasets computationally feasible while possibly resulting in a worse fit and inference. We won't cover these simplifications, and focus on data with smaller dimensionality to evaluate the capabilities of the proposed GMN model.

Implementation is provided in Python in a GitHub repository [1] and could be used as reference for implementation in other languages.

3.3.5 Convergence criterion

The EM algorithm should be terminated when it has converged. Let $\{l_t\}_{t=1}^{\infty}$ be a sequence of log-likelihoods calculated at each step of the EM algorithm. A common approach is to terminate when $l_{t+1} - l_t \leq \epsilon$ for some $t > 1$ and tolerance parameter ϵ. This approach assumes that when the steps are small enough, the algorithm should be close to convergence.

Another approach was proposed by Böhning et al., 1994 (and also used by Franczak et al., 2013). Let $\hat{l}$ denote the limit $\lim_{t\to\infty} = \hat{l}$. The approach assumes that EM steps have linear convergence: $\frac{|l_{t+1}-\hat{l}|}{|l_t-\hat{l}|} = c$, $\forall t > 1$ and estimates $\hat{l}$ with l_t^{∞} on each step:

$$l_{t+1}^{\infty} = l_t + \frac{l_{t+1} - l_t}{1 - a_t}, \quad \text{where}$$

$$a_t = \frac{l_{t+1} - l_t}{l_t - l_{t-1}} \quad \text{is the Aitken acceleration}$$

The proposed termination criterion is when for some tolerance parameter ϵ the current log-likelihood is close to the estimated converged log-likelihood:

$$|l_{t+1}^{\infty} - l_{t+1}| < \epsilon.$$

We will use Böhning's approach, due to its more profound theoretical justification, but for small enough ϵ both convergence criteria should perform well.

[1] github.com/KangaroosInAntarcitica/mixes

```
Input  : Dataset $\mathcal{D}$, initial values for parameters $\Theta$
Output: Fitted values for parameters $\Theta$

while not converged do
    $\mathcal{Z}_1 \leftarrow \mathcal{D}$
    Compute $\eta_{s_{(l:)}}, \Sigma_{s_{(l:)}}, \Psi_{s_{(l:)}},\ \forall s_{(l:)} \in \mathcal{S}_{(l:)}$ with equations 3.4

    for $l \in \{1, \dots, L\}$ do
        E Step
        for $s_{(l:)} \in \mathcal{S}_{(l:)}$ do
            Denoting $v = z_l, w = z_{l+1}$
            Compute $p(s_{(l:)} \mid v),\ \forall v \in \mathcal{Z}_l$
                using equation 3.14 and vars from line 3
            Compute $\xi_{s_l}$ with equation 3.5
            Compute $\rho_{s_l}(v),\ \forall v \in \mathcal{Z}_l$ with equation 3.5
            Compute $E_{w|v,s^{(l:)}}(w), E_{w|v,s^{(l:)}}(ww^T),\ \forall v \in \mathcal{Z}_l$
                with equations 3.13

            Estimate $p(s_{(l:)} \mid \mathcal{D}) = \frac{1}{|\mathcal{Z}_l|} \sum_{v \in \mathcal{Z}_l} p(s_{(l:)} \mid v)$
            Estimate $E_{w|s^{(l:)},\mathcal{D}}(w) = \frac{1}{|\mathcal{Z}_l|} \sum_{v \in \mathcal{Z}_l} (E_{w|v,s^{(l:)}}(w))$
                and similarly for $E_{w|s^{(l:)},\mathcal{D}}(ww^T)$
            Estimate $E_{w,v|s^{(l:)},\mathcal{D}}(vw^T) = \frac{1}{|\mathcal{Z}_l|} \sum_{v \in \mathcal{Z}_l} v(E_{w,v|v,s^{(l:)}}(w))^T$
                and similarly for $E_{v|s^{(l:)},\mathcal{D}}(vv^T)$
            Sample $\mathcal{Z}_{l+1}^{s_{(l:)}}$ of size $n$ from pdf in equation 3.5
        end
        M Step
        for $\hat{s}_l \in \mathcal{S}_l$ do
            Compute $p(\hat{s}_{l+1} \mid \hat{s}_l, \mathcal{D}) = \frac{1}{|\mathcal{S}_{(l+1:)}|} \sum_{s_{(l+1:)} \in \mathcal{S}_{(l+1:)}} \big(p(s_{(l:)} \mid \mathcal{D})[s_{l+1} = \hat{s}_{l+1}]\big)$
            Compute denominator $p(\hat{s}_l \mid \mathcal{D}) = \frac{1}{|\mathcal{S}_{(l:)}|} \sum_{s_{(l:)} \in \mathcal{S}_{(l:)}} \big(p(s_{(l:)} \mid \mathcal{D})[s_l = \hat{s}_l]\big)$
            Compute $E_{w|\hat{s}_l,\mathcal{D}}(w) = \frac{1}{|\mathcal{S}_{(l:)}| \cdot p(\hat{s}_l|\mathcal{D})} \sum_{s_{(l+1:)} \in \mathcal{S}_{(l+1:)}} (p(s_{(l:)} \mid \mathcal{D}) E_{w|s^{(l:)}}(w))$
              using vars from line 13
              and similarly for $E_{w|\hat{s}_l,\mathcal{D}}(ww^T),\ E_{w,v|\hat{s}_l,\mathcal{D}}(vw^T),\ E_{v|\hat{s}_l,\mathcal{D}}(v)$
              using vars from lines 18, 19

            Update $\tau^{(l)}_{(s_l,s_{l+1})}$ using vars from line 25
            Update $\Lambda^{(l)}_{s_l}, \eta^{(l)}_{s_l}, \Psi^{(l)}_{s_l}$ using equation 3.10 and vars from lines 27, 29
        end
        Sample $\mathcal{Z}_{l+1}$ of size $n$ from $\mathcal{Z}_{l+1}^{s_{(l:)}}$ using probabilities $p(s_{(l:)} \mid \mathcal{D})$
    end
end
```

Algorithm 1: GMN Expectation Maximization

3.3.6 Regularization

While the EM algorithm provides theoretical guarantees for improvements of log-likelihood on each step, certain problems appear in practice when performing computation on floating points.

One common computational problem can occur when we have singular or close to singular covariance matrices Σ. For such matrices, the condition number is very large which results in large relative errors during the computation of the inverse. Since inversion is required in the probability density function $\mathcal{N}(\cdot \mid \mu, \Sigma)$, we want to avoid such covariance matrices. For Gaussian mixture models Gepperth and Pfülb, 2021 propose directly optimizing w.r.t. the inverse Σ^{-1}, but this solution cannot be extended to GMN. Pedregosa et al., 2011 add a regularization on each step of the EM algorithm for GMM: $\bar{\Sigma} = \Sigma + I\delta$ where δ is a small number. We have observed in practice that small values of δ have a negligible effect on the total log-likelihood, but bring a large computational advantage.

We are going to implement this strategy in our EM algorithm. We will add regularization to both Ψ and Λ on each step of the EM algorithm:

$$\begin{aligned} \bar{\Psi} &= \Psi + I\delta, \\ \bar{\Lambda} &= U\sqrt{D}, \qquad \text{where } U, D, V = \mathrm{SVD}(\Lambda\Lambda^T + I\delta). \end{aligned}$$

Note, that such replacement also forces $\bar{\Lambda}$ to have orthogonal columns.

Another practical problem is overfitting. We will use the term overfitting for cases when a model is too flexible and is able to fit to all the variation and noise present in the dataset instead of generalizing, which results in bad clustering accuracy compared to real classes. As described in previous sections, one way of avoiding overfitting is to restrict the allowed distributions of the model, as it was done with parameter tying. We propose to further use the regularization parameter δ to avoid overfitting. Setting a larger value to δ will restrict covariances on each layer to have larger values.

3.3.7 Deterministic annealing

Log likelihood of mixture models is often a non-concave function which results in optimization algorithms terminating in inferior modes. Zhou and Lange, 2010 propose using deterministic annealing to flatten the surface of the log-likelihood function and increase the chance of terminating in a dominant mode. Because of the complex structure, multimodality should be an even larger problem for GMN. We can incorporate annealing in our implementation of the EM algorithm by replacing the probabilities in formula 3.14 with

$$p(s_{(l:)} \mid z_l, \Theta') = \frac{\left[p(s_{(l:)})p(z_l \mid s_{(l:)})\right]^v}{\sum_{\hat{s}_{(l:)} \in \mathcal{S}_{(l:)}} \left[p(\hat{s}_{(l:)})p(z_l \mid \hat{s}_{(l:)})\right]^v} \tag{3.15}$$

in the E step. The additional parameter $v \in (0, 1]$ determines the amount of annealing, where small values correspond to a more flattened surface and 1 makes no change.

As proposed by Zhou and Lange, 2010 we will start optimization with a small value v_{init} and gradually increase it to 1. In practical experiments, $v_{\text{init}} = 0.5$ showed the most predictable improvement for most datasets, while smaller value $v_{\text{init}} = 0.2$ highly improved performance on certain datasets.

3.3.8 Initialization

One way of initializing the GMN is using completely random values. We have to be careful with setting the means and covariances on different layers, as these are accumulated in the distribution of paths. Therefore we randomly sample the elements of Λ from a uniform distribution $(-1, 1)$, diagonal elements of Ψ from a uniform positive distribution $(0, \frac{1}{L^2})$, $\eta^{(1)}$ from $\mathcal{D}$, and elements of $\eta^{(2)}, \ldots, \eta_L$ from a uniform distribution $(-\frac{1}{L}, \frac{1}{L})$.

Another option for initialization is based on a simpler clustering algorithm. We will use the popular K-Means algorithm with components equal to nodes k_l on each layer, and then use factor analysis to estimate the parameters $\Psi^{(l)}_{s_l}, \Lambda^{(l)}_{s_l}, \eta^{(l)}_{s_l}$ for each node. We will use the same factor analyzer model to reduce the dimensionality of data clustered to this component and then we will use the lower-dimensional data for the next layer's initialization. This will be the preferred initialization approach in our experiments.

Chapter 4

Experiments

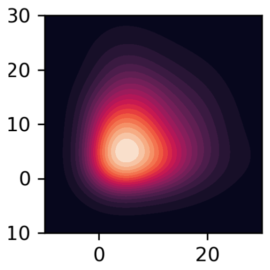
4.1 Distribution fitting

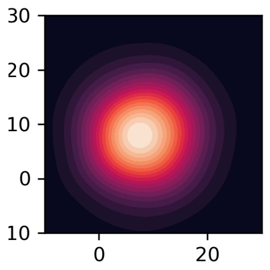
We will first evaluate the capability of GMN to fit different distributions. This is an essential problem in generative learning, that is useful for sampling from the distribution and inferring the probability of association with each cluster.

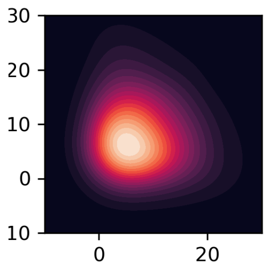
All the presented experiments can be found in our implementation mentioned in section 3.3.4.

	Dataset	
Algorithm	Skew Normal	Old Faithful
GMM	$-1.292 \cdot 10^4$	$-3.855 \cdot 10^2$
Skew GMM	$-1.248 \cdot 10^4$	$-3.843 \cdot 10^2$
GMN	$-1.250 \cdot 10^4$	$-3.676 \cdot 10^2$
DGMM	-	$-3.726 \cdot 10^2$

Table 4.1: Log-likelihood of fitted algorithms on two datasets


(a) Skew normal distribution that we will train models on


(b) Fitted GMM distribution


(c) Fitted Skew GMM distribution

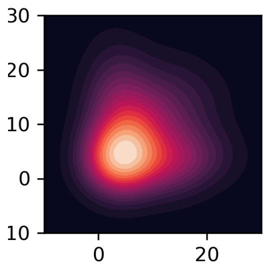
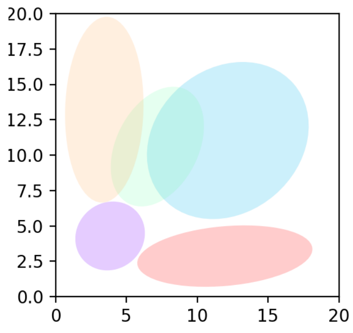
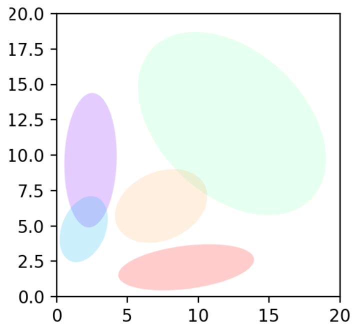

(d) Fitted GMN distribution

Figure 4.1: Kernel density estimate plot for data sampled from a distribution and models fitted with the data


(a) Paths of GMN from 4.1d


(b) Alternative distribution of paths after training

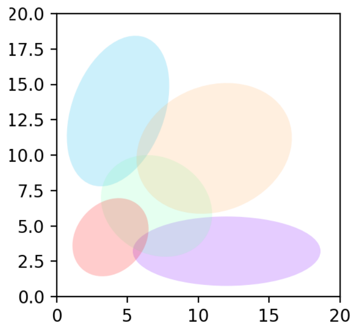

(c) Alternative distribution of paths with a better fit

Figure 4.2: Multiple fits of a GMN model on the data from Figure 4.1a. GMN has 1 cluster and 5 paths ($k_1 = 1, k_2 = 5$). The regions correspond to different paths and the region edges are at 1 variance (approx 40% for 2D Gaussians).

4.1.1 Skew Gaussian distribution fitting

We will first try fitting to a skew Gaussian distribution, which extends the Gaussian distribution by allowing asymmetry.

We will use the formulation by Lin, 2009. If y is following a skew Gaussian distribution, the variable has the following stochastic representation:

$$y = \eta + \Lambda|x| + u,$$

where $u \sim \mathcal{N}(0, \Psi), x \sim \mathcal{N}(0, I)$. Here $| \cdot |$ denotes the piece-wise absolute value and then $|x| \sim \mathcal{HN}(0, I)$ has standard half-normal distribution.

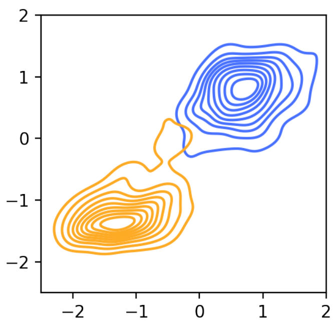
We generated samples from a 2D skew Gaussian distribution with parameters $\eta = 0, \Psi = I, \Lambda = \text{diag}\{10, 10\}$ which can be seen in figure 4.1a. We fitted GMM, GMN and a Skew Gaussian mixture model (SGMM) to the data points. We used the Expectation Maximization algorithm to fit all the models with a single cluster. For SGMM we implemented EM as proposed by Lin, 2009. We will fit a GMN with $k_1 = 1, k_2 = 5$. We will not fit DGMM, as it has the same structure in the case of a single cluster and 2 layers.

The resulting log-likelihoods directly measure how well different models fit the distribution and are shown in table 4.1. As can also be seen in figure 4.1 the GMM is unable to capture certain variations present in the distribution. The SGMM has the best fit, which is an expected outcome, as its initial assumption of the distribution is correct. However, the GMN fitted to the distribution almost as good (log-likelihood has a small difference), which shows the flexibility of the model.

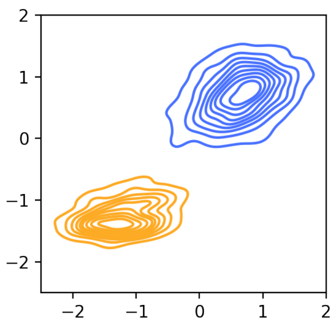
We can study how the model achieved this result by inspecting figure 4.2a. The plot visualizes the distributions of each path of the model, which as mentioned in section 3.1.2 are Gaussian distributions. As can be seen, different paths attribute to parts of the distribution and their composition forms a similar distribution to the initial skew Gaussian. As can be seen in figure 4.2 there can be a lot of variation between paths in a GMN for different initializations and fits. However, this has little effect on the whole distribution of the model.

4.1.2 Old Faithful geyser data

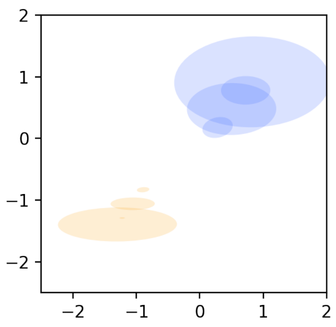
We will test the performance of models on the data from eruptions of the Old Faithful geyser in the Yellowstone national park. The data was first published by Härdle et al., 1991 and contains 2 observed variables: the duration of geyser eruption and time to next eruption. Data has 2 clusters, which correspond to short and long eruptions.



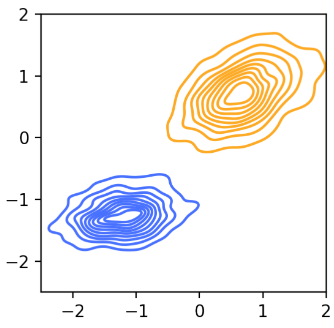
(a) Old faithful geyser data distribution



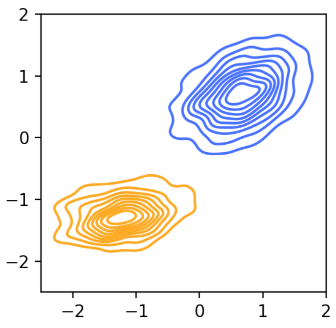
(b) Fitted GMN distribution



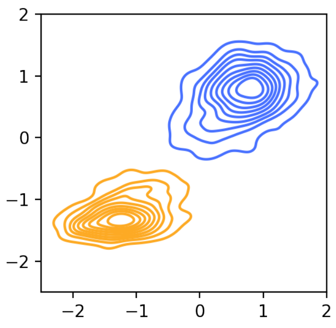
(c) Path distributions plot similar to figure 4.2, but scaled according to the path probabilities π_s



(d) Fitted GMM distribution



(e) Fitted Skew GMM distribution



(f) Fitted DGMM distribution

Figure 4.3: The distributions of the Old Faithful geyser data and models fitted on the dataset. The distributions are calculated with sampling therefore may not be exact.

All the models are trained to fit 2 clusters. GMN and DGMM are fit with layer sizes $k_1 = 2, k_2 = 5$ and $\dim(z_2) = 1$. The results are shown in table 4.1 and figure 4.3. As can be seen, GMN achieved the best log-likelihood of all the presented mixture models. The original data doesn't have the distributions centered around their modes, which cannot be achieved by a GMM. In this case GMN reaches a significant improvement compared to both GMM and SGMM.

Figure 4.3c visualizes the distributions of each path of the GMN while scaling them by the path probability τ_s. As can be seen, components s_2 of the second layer have varying probabilities in each cluster, while the yellow cluster treats some of them as completely redundant. This allows the model to have more flexible distributions between clusters.

4.2 Clustering

4.2.1 Practical performance

We will compare the performance of different algorithms on real-world data. We will use the Wine dataset[1] by Forina et al., 1986, Ecoli dataset[2] by Nakai and Kanehisa, 1991, Vehicle dataset[3] by Siebert, 1987, Satellite dataset[4] and the Digits dataset[5] by Xu et al., 1992. All of the datasets are available from the UCI machine learning repository (Dua and Graff, 2017).

We give a short description of each of the datasets in table 4.3. The datasets have a varying number of features, complexity and size.

The results of fitting different algorithms are presented in table 4.4. We will fit the K-Means model implemented by Pedregosa et al., 2011, GMM, DGMM and GMN. The experiment will focus on the practical capabilities of the model. Therefore, for each model we will perform 10 runs and show the average of all metrics among these runs instead of the best score. We will use models of increasing depth depending on the complexity of dataset. We used the parameters $k = (3, 1), \dim(z) = (13, 3, 2)$ for the Wine dataset, $k = (8, 4, 1), \dim(z) = (7, 6, 5, 4)$ for the Ecoli dataset, $\dim(z) = (18, 7, 3, 3)$ for the Vehicle dataset, $k = (6, 5, 1), \dim(z) = (36, 13, 5, 1)$ for the Satellite dataset, $k = (4, 3, 2)$ and $k = (10, 5, 2)$, $\dim(z) = (64, 10, 6, 2)$ for the Digits dataset.

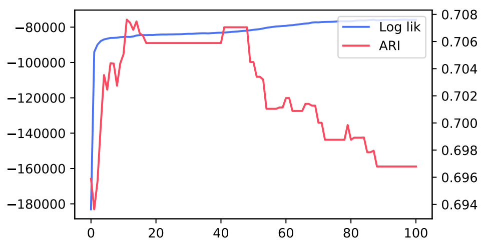
[1]https://archive.ics.uci.edu/ml/datasets/wine
[2]https://archive.ics.uci.edu/ml/datasets/ecoli
[3]https://archive.ics.uci.edu/ml/datasets/Statlog+(Vehicle+Silhouettes)
[4]https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)
[5]https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

Dataset	Classes		Features		Size
Wine	3	Types of wines from the Piedmont region of Italy	13	Physical and chemical properties of the wine	178
Ecoli	8	Protein celluar localization sites	7	Protein features based on sequence recognition methods	336
Vehicle	4	Types of vehicles	18	Features extracted from the silhouette of a vehicle	846
Satellite	6	Types of soil	36	Attributes from a 3×3 RGB satellite image of the soil	6435
Digits	10	Digits from 0 to 9	64	Attributes from a 8×8 greyscale image of a written digit	1797

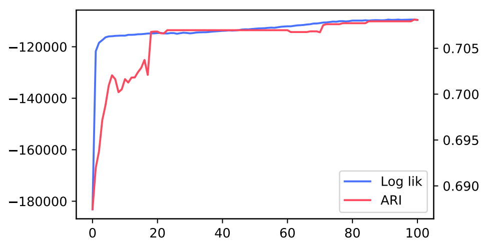
Table 4.3: Descriptions of datasets used. For each dataset, we show the true classes present in data, features we will perform clustering on and the total number of data points in the dataset

Datasets	Wine		Ecoli		Vehicle		Satellite		Digits	
	ARI	m.r.	ARI	m.r.	ARI	m.r.	ARI	m.r.	ARI	m.r.
KMeans	0.873	0.041	0.529	0.355	0.069	0.638	0.530	0.321	0.632	0.257
GMM	0.945	0.016	0.653	0.239	0.095	0.619	0.468	0.413	0.640	0.245
DGMM	0.975	0.008	0.750	0.181	–	–	0.571	0.282	0.698	0.175
DGMM ann.	0.975	0.008	0.734	0.190	0.109	0.599	**0.578**	**0.280**	0.670	0.175
GMN	0.962	0.011	**0.764**	**0.173**	–	–	0.535	0.301	0.702	0.172
GMN ann.	**0.983**	**0.006**	0.737	0.187	**0.130**	**0.585**	0.524	0.305	**0.704**	**0.172**

Table 4.4: Average adjusted rand index (ARI) and miss-classification rate (m.r.) over 10 fits of different algorithms on given datasets (ann. corresponds to algorithms using deterministic annealing)



(a) Fitting of GMN with regularization $\delta = 10^{-4}$. Final silhouette score is 0.169



(b) Fitting of GMN with regularization $\delta = 10^{-3}$. Final silhouette score is 0.171

Figure 4.4: Comparison of metrics during the fitting of a DGMM with $k = \{10, 3, 2, 2\}$, $\dim(z) = \{64, 10, 6, 4, 2\}$ and different values of regularization hyperparameter δ

A version of the GMN model got the best performance on all the datasets except for the Satellite dataset, where DGMM performed better. This could be because for some distributions it is more useful to have the same prior for all components. In other cases, we suppose that GMN performed better thanks to its increased flexibility.

Furthermore, the results show that deterministic annealing is a very useful technique for GMN optimization. For the Wine, Satellite and Digits datasets the highest scores were achieved by models with annealing, although annealing degraded the performance on the Ecoli dataset. We were unable to fit DGMM and GMN models without annealing on the Vehicle dataset at all. We would like to note that we only used annealing with the starting value of 0.5, and it is likely that better performance could be achieved by tuning the v_{init} hyperparameter.

Overall, the experiments showed that GMN is a useful extension of DGMM models, as is the deterministic annealing, while the best results could be achieved by choosing among variations of these models.

4.2.2 Effect of Hyperparameters on the GMN fitting

We will use the Digits dataset to further study the performance of GMN.

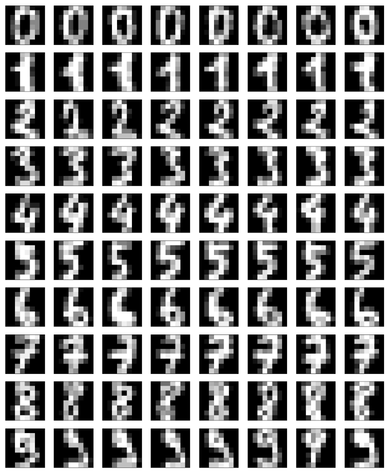
We will first inspect the effect of regularization δ mentioned in section 3.3.6. The comparison of optimization with different values of the hyperparamter can be seen in figure 4.4. The 4-layer network with regularization $\delta = 10^{-4}$ reached a higher log-likelihood but degraded its accuracy in the process due to overfitting. On the other hand, the same network with increased regularization $\delta = 10^{-3}$ monotonously improved both log-likelihood and ARI. The silhouette score is also better for the more regularized model in this case. In practice, we observed similar behavior for other datasets and therefore consider δ to be a useful hyperparamter for model fine-tuning.

We will generate samples from the better-performing GMN model. As can be seen in figure 4.5, some samples exactly match the digits. However, digits seven and nine seem to be confused and certain samples are similar to a combination of digits. Since the model is trained in a completely unsupervised way, this result can be explained by the high similarity between certain handwriting of different digits.

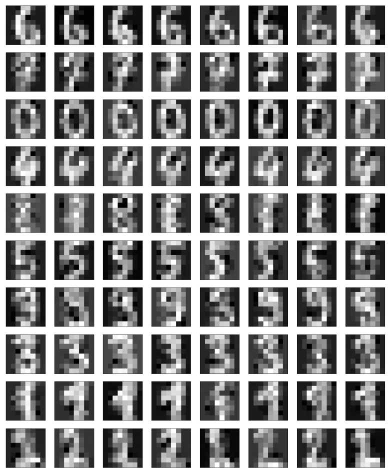
Finally, we will study the effect of the annealing hyperparameter v_{init} on GMN fitting. The results of fitting a GMN on the Digits dataset for different values of the v_{init} hyperparameter can be seen in figure 4.6. Note, that we don't use a convergence criterion in this case, as for values of the annealing parameter $v \neq 1$, the model might be in a sub-optimal state due to the changed surface of log-likelihood. For a fair practical comparison, we perform the same number of 100 EM steps for all the values of the v_{init} hyperparameter. As we can observe, annealing produces a more predictable behavior for the optimization, but optimization doesn't terminate in the dominant mode. This could be because of the complex structure of a GMN model and therefore a complex surface of its log-likelihood. Nevertheless, we can observe that the best average ARI was reached by a model with annealing. Notice, that a very small value of $v_{\text{init}} = 0.1$ results in an inferior behavior compared to a model without annealing ($v_{\text{init}} = 1$). This is likely also caused by the complex surface of the log-likelihood, and possibly could be improved by performing a larger amount of optimization steps.

4.2.3 Discussion

In practice, we observed that fine-tuning of hyperparameters in GMN performs similarly to discriminative deep learning, where reducing layer sizes k_l or dimensionality $\dim(z_l)$ helps to avoid overfitting, and making deeper networks seems to provide a greater learning capacity. At some point, increasing the depth stops improving accuracy, which could be the result of the data limitations or incorrect assumptions in the model. Similar to supervised learning, distributions have an overlap in the probability density functions which results in a theoretical lower bound of the error, called the Bayes error (Antos et al., 1999). However, due to the lack of labels during fitting, unsupervised learning must also rely on the correctness of certain assumptions, like the compactness of clusters, which imposes further bounds on the theoretically best error. As a result the best accuracy rarely corresponds to the best silhouette score, as can easily be seen for the geyser data in figure 4.3a.

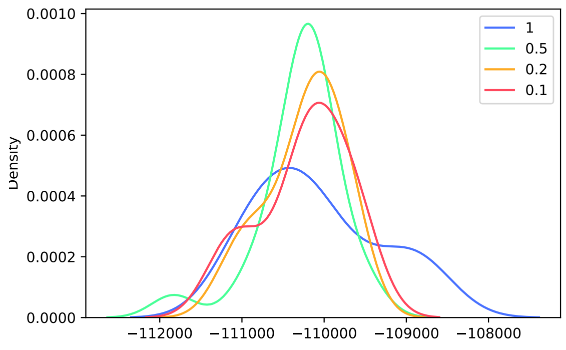


(a) Samples from the dataset



(b) Samples from the fitted GMN

Figure 4.5: Comparison of data and distribution samples of a fitted GMN on the Digits dataset. Each row of the figure corresponds to a different class or component



(a) The distribution of final log-likelihood

Annealing	ARI	log. lik
$v_{\text{init}} = 1.0$	0.702	$\mathbf{-1.101 \times 10^5}$
$v_{\text{init}} = 0.5$	**0.705**	-1.103×10^5
$v_{\text{init}} = 0.2$	0.703	-1.102×10^5
$v_{\text{init}} = 0.1$	0.693	-1.102×10^5

(b) Average metrics

Figure 4.6: Log-likelihood distribution and average metrics after fitting with different values of initial annealing v_{init}. The metrics were collected over 40 fits of GMN with $k = \{10, 5, 2\}$, $\dim(z) = \{36, 10, 6, 2\}$

Chapter 5

Conclusion

Based on the ideas of the DGMM and DMFA models, we propose the Gaussian Mixture Network. GMN provides the ability to tune the required model flexibility to fit more complex distributions or avoid overfitting. We develop an EM algorithm that directly solves a system of equations to find extreme points and further discuss improvements to the optimization scheme. We utilize deterministic annealing to reach the dominant mode with higher probabilities and use regularization for computational stability. We compare GMN to other models on five datasets and show that it reaches better results on most of them.

References

Antos, A., Devroye, L., & Gyorfi, L. (1999). Lower bounds for bayes error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(7), 643–645.

Böhning, D., Dietz, E., Schaub, R., Schlattmann, P., & Lindsay, B. G. (1994). The distribution of the likelihood ratio for mixtures of densities from the one-parameter exponential family. *Annals of the Institute of Statistical Mathematics*, *46*(2), 373–388.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1–22.

Dua, D., & Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml

Forina, M., Drava, G., & De La Pezuela, C. (1986). Sixth chemometrics in analytical chemistry conference (cac). *Tarragona, June*, 25–29.

Franczak, B. C., Browne, R. P., & McNicholas, P. D. (2013). Mixtures of shifted asymmetric laplace distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(6), 1149–1157.

Fuchs, R., Pommeret, D., & Viroli, C. (2022). Mixed deep gaussian mixture model: A clustering model for mixed datasets. *Advances in Data Analysis and Classification*, *16*(1), 31–53.

Gepperth, A., & Pfülb, B. (2021). Gradient-based training of gaussian mixture models for high-dimensional streaming data. *Neural Processing Letters*, *53*(6), 4331–4348.

Ghahramani, Z., Hinton, G. E. et al. (1996). *The em algorithm for mixtures of factor analyzers* (tech. rep.). Technical Report CRG-TR-96-1, University of Toronto.

Härdle, W. K. et al. (1991). *Smoothing techniques: With implementation in s*. Springer Science & Business Media.

Harshvardhan, G., Gourisaria, M. K., Pandey, M., & Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, *38*, 100285.

Lin, T. I. (2009). Maximum likelihood estimation for multivariate skew normal mixture models. *Journal of Multivariate Analysis*, *100*(2), 257–265.

Nakai, K., & Kanehisa, M. (1991). Expert system for predicting protein localization sites in gram-negative bacteria. *Proteins: Structure, Function, and Bioinformatics*, *11*(2), 95–110.

Nielsen, S. F. (2000). The stochastic em algorithm: Estimation and asymptotic results. *Bernoulli*, 457–489.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Reynolds, D. A. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, *741*(659-663).

Siebert, J. P. (1987). Vehicle recognition using rule based methods.

Sun, Y., Babu, P., & Palomar, D. P. (2016). Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, *65*(3), 794–816.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tang, Y., Salakhutdinov, R., & Hinton, G. (2012). Deep mixtures of factor analysers. *arXiv preprint arXiv:1206.4635*.

Van den Oord, A., & Schrauwen, B. (2014). Factoring variations in natural images with deep gaussian mixture models. *Advances in neural information processing systems*, *27*.

Viroli, C., & Anderlucci, L. (2021). Deep mixtures of unigrams for uncovering topics in textual data. *Statistics and computing*, *31*(3).

Viroli, C., & McLachlan, G. J. (2017). Deep gaussian mixture models. *Statistics and computing*, *29*(1), 43–51.

Xu, L., Krzyzak, A., & Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, *22*(3), 418–435.

Zhang, H., & El-Shaarawi, A. (2010). On spatial skew-gaussian processes and applications. *Environmetrics: The official journal of the International Environmetrics Society*, *21*(1), 33–47.

Zhou, H., & Lange, K. L. (2010). On the bumpy road to the dominant mode. *Scandinavian Journal of Statistics*, *37*(4), 612–631.

APPENDICES

Appendix A

Inverse distribution at a node in GMN derivation

Assume that we are looking for the inverse probability at the node s_l of layer l given path $s_{l:}$. For simplicity of notation, let $\eta = \eta^{(l)}_{s_l}$, $\Lambda = \Lambda^{(l)}_{s_l}$, $\Psi = \Psi^{(l)}_{s_l}$, $\mu = \mu_{s_{(l+1:)}}$ and $\Sigma = \Sigma_{s_{(l+1:)}}$.

Let also $v = z_l$ and $w = z_{l+1}$.

Then from equations 3.1 and 3.4 we have:

$$\begin{aligned} p(v \mid w, s_l) &= \mathcal{N}\left(v \mid \eta + \Lambda w, \Psi\right), \\ p(w \mid s_{(l+1:)}) &= \mathcal{N}\left(w \mid \mu, \Sigma\right) \end{aligned}$$

Using the Bayes' rule we get:

$$\begin{aligned} & p(w \mid v, s_{(l:)}) \\ =& \frac{p(v \mid w, s_l) p(w \mid s_{(l+1:)})}{p(v \mid s_{(l:)})} \\ =& c_1 \cdot \mathcal{N}\left(v \mid \eta + \Lambda w, \Psi\right) \mathcal{N}\left(w \mid \mu, \Sigma\right) \\ =& c_1 \cdot \exp\left(-\frac{1}{2}(v - \eta - \Lambda w)^T \Psi^{-1} (v - \eta - \Lambda w)\right) \exp\left(-\frac{1}{2}(v-\eta)^T \Sigma^{-1}(v-\eta)\right) \\ & \qquad \text{(Using properties of the exponent)} \\ =& c_1 \cdot \exp\left((v - \eta - \Lambda w)^T \Psi^{-1} (v - \eta - \Lambda w) + (v-\eta)^T \Sigma^{-1}(v-\eta)\right)^{-\frac{1}{2}} \\ & \qquad \text{(Expanding and taking the constant terms w.r.t } w \text{ out)} \end{aligned}$$

$$=c_2 \cdot \exp\Big(w^T\left(\Lambda^T\Psi^{-1}\Lambda\right)w - (\Lambda w)^T\Psi^{-1}(v-\eta) - (v-\eta)^T\Psi^{-1}(\Lambda w)$$
$$+ w^T\Sigma^{-1}w - w^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}w\Big)^{-\frac{1}{2}}$$

(Combining the expressions by color)

$$=c_2 \cdot \exp\Big(w^T\underbrace{\left(\Lambda^T\Psi^{-1}\Lambda + \Sigma^{-1}\right)}_{\xi^{-1}}w - w^T\left(\Lambda^T\Psi^{-1}(v-\eta) + \Sigma^{-1}\mu\right)$$
$$-\left((v-\eta)^T\Psi^{-1}\Lambda^T + \mu^T\Sigma^{-1}\right)w\Big)^{-\frac{1}{2}}$$

(Using the definition of ξ^{-1} from above)

$$=c_2 \cdot \exp\Big(w^T\xi^{-1}w - w^T\xi^{-1}\underbrace{\xi\left(\Lambda^T\Psi^{-1}(v-\eta) + \Sigma^{-1}\mu\right)}_{\rho}$$
$$-\underbrace{\left((v-\eta)^T\Psi^{-1}\Lambda^T + \mu^T\Sigma^{-1}\right)\xi}_{\rho^T}\xi^{-1}w\Big)^{-\frac{1}{2}}$$

$$=c_2 \cdot \exp\left((w-\rho)^T\xi^{-1}(w-\rho) - \rho^T\xi^{-1}\rho\right)^{-\frac{1}{2}}$$

(Taking constant out of the exponent)

$$=c_3 \cdot \exp\left((w-\rho)^T\xi^{-1}(w-\rho)\right)^{-\frac{1}{2}}$$

Since we know that $p(w \mid v, s_{(l:)})$ must be a valid probability distribution, we can ignore the scaling constant c_3 in front. The result is a gaussian distribution:

Therefore, given a path $s_{(l:)}$ the pdf is of the form:

$$p(w|v, s_{(l:)}) = \mathcal{N}\left(w \mid \rho(v), \xi\right), \qquad \text{where:}$$
$$\xi = \left((\Sigma^{-1} + \Lambda^T\Psi^{-1}\Lambda\right),$$
$$\rho(v) = \xi\left(\Lambda^T\Psi^{-1}(v-\eta) + \Sigma^{-1}\right),$$

where the parameters and variables are:

$$\eta = \eta_{s_l}^{(l)}, \Lambda = \Lambda_{s_l}^{(l)}, \Psi = \Psi_{s_l}^{(l)},$$
$$\mu = \mu_{s_{(l+1:)}}, \Sigma = \Sigma_{s_{(l+1:)}},$$
$$v = z_l, w = z_{l+1}.$$

Appendix B

Expectation Maximization of GMN derivation

For layer l and component $\hat{s}_l$ we want to minimize the following objective function:

$$g = E_{z,s|s_l=\hat{s}_l,y,\Theta'}\Big(\log \underbrace{p(z_l \mid \hat{s}_l, z_{l+1}, \Theta^{(l)}_{\hat{s}_l})}_{\mathcal{N}(z_l|\eta^{(l)}_{\hat{s}_l}+\Lambda^{(l)}_{\hat{s}_l}z_{l+1},\Psi^{(l)}_{\hat{s}_l})} + \log \underbrace{p(s_l = \hat{s}_l)}_{\tau^{(l)}_{(\hat{s}_l,s_{l+1})}}\Big)$$

For simplicity, we will omit the subscript in expectation $E(\cdot) = E_{z,s|s_l=\hat{s}_l,y,\Theta'}(\cdot)$. We will also assume that all the parameters are for layer (l) and distribution $\hat{s}$: $\Lambda = \Lambda^{(l)}_{\hat{s}_l}$, $\eta = \eta^{(l)}_{\hat{s}_l}$, $\Psi = \Psi^{(l)}_{\hat{s}_l}$. We will also perform a change of variables $v = z_l, w = z_{l+1}$ We get:

$$g = E\left(\det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left((v - \eta - \Lambda w)^T \Sigma^{-1}(v - \eta - \Lambda w)\right)\right)$$

Note that Ψ is diagonal. Therefore taking the derivatives w.r.t. to parameters η, Λ, Ψ, we get:

$$\begin{cases} \frac{\partial g}{\partial \eta} & = E\left(\Psi^{-1}\left(v - \eta - \Lambda w\right)\right) = 0, \\ \frac{\partial g}{\partial \Lambda} & = E\left(\Psi^{-1}\left(v - \eta - \Lambda w\right) w^T\right) = 0, \\ \frac{\partial g}{\partial \Psi} & = -\frac{1}{2}E\left(\Psi^{-1} - \Psi^{-1}\left(v - \eta - \Lambda w\right)\left(v - \eta - \Lambda w\right)^T \Psi^{-1}\right) = 0. \end{cases}$$

We will solve these equations as a system. We will first solve the system from the first 2 equations, as they are independent of Ψ. Since Ψ is assumed to be non-singular, we can multiply by it and get:

$$\begin{cases} E\left(v-\eta-\Lambda w\right)=0 \\ E(\left(\left(v-\eta-\Lambda w\right)w^T\right)=0 \end{cases}$$

(Taking out constant term from expectation)

$$\begin{cases} \eta = E\left(v-\Lambda w\right) \\ \quad E\left(vw^T-\eta w^T-\Lambda ww^T\right)=0 \end{cases}$$

(Performing substitution)

$$\begin{cases} \eta = E(v)-\Lambda E(w) \\ \quad E\left(vw^T-E(v)w^T+\Lambda E(w)w^T-\Lambda ww^T\right)=0 \end{cases}$$

(Taking out constant terms)

$$\begin{cases} \eta = E(v)-\Lambda E(w) \\ \quad E(vw^T)-E(v)E(w)^T+\Lambda\left(E(w)E(w)^T-E(ww^T)\right)=0 \end{cases}$$

(Moving Λ to the left side)

$$\begin{cases} \eta = E(v)-\Lambda E(w) \\ \Lambda = \left(E(vw^T)-E(v)E(w)^T\right)\left(E(ww^T)-E(w)E(w)^T\right)^{-1} \end{cases}$$

To solve the last equation we will first multiply it by Ψ:

$$E\left(I-\left(v-\eta-\Lambda w\right)\left(v-\eta-\Lambda w\right)^T\Psi^{-1}\right)=0,$$

(Multiplying by Ψ and rearanging)

$$\begin{aligned}
\Psi &= E\left(\left(v-\eta-\Lambda w\right)\left(v-\eta-\Lambda w\right)^T\right) \\
&= E\left((v-\Lambda w)(v-\Lambda w)^T\right)-\eta E\left(v-\Lambda w\right)^T-E\left(v-\Lambda w\right)\eta^T+\eta\eta^T \\
&\qquad \text{(Subtracting and adding } 2\eta\eta^T \text{ and using the equation for } \eta\text{)} \\
&= E\left((v-\Lambda w)(v-\Lambda w)^T\right)-\eta\underbrace{E\left(v-\eta-\Lambda w\right)^T}_{=0}-\underbrace{E\left(v-\eta-\Lambda w\right)}_{=0}\eta^T-\eta\eta^T \\
&= E(vv^T)-E(vw^T)\Lambda^T-\Lambda E(wv^T)+\Lambda E(ww^T)\Lambda^T-\eta\eta^T \\
&\qquad \text{(Using the fact that we are only solving for the entries on the diagonal of } \Psi\text{)} \\
&= \operatorname{diag}\{E(vv^T)-2E(vw^T)\Lambda^T+\Lambda E(ww^T)\Lambda^T-\eta\eta^T\}.
\end{aligned}$$

As a result, we get the following solutions for the system:

$$\begin{cases} \Lambda &= \left(E(vw^T) - E(v)E(w)^T\right) \cdot \left(E\left(ww^T\right) - E(w)E(w)^T\right)^{-1}, \\ \eta &= E(v) - \Lambda E(w), \\ \Psi &= E(vv^T) - 2E(vw^T)\Lambda^T + \Lambda E(ww^T)\Lambda^T - \eta\eta^T. \end{cases}$$

Notice that we can reformulate as

$$\begin{cases} \Lambda &= \mathrm{Cov}(v,w)\left(\mathrm{Var}(w)\right)^{-1}, \\ \eta &= E(v - \Lambda w), \\ \Psi &= \mathrm{Var}(v - \Lambda w). \end{cases}$$