Towards disease progression sub-typing via responsibility sampling for robust expectation-maximisation learning

MATHIAS EDMAN
Towards disease progression sub-typing via responsibility sampling for robust expectation-maximisation learning

MATHIAS EDMAN

Degree Projects in Optimization and Systems Theory (30 ECTS credits)
Degree Programme in Applied and Computational Mathematics (120 credits)
KTH Royal Institute of Technology year 2019
Supervisor at KTH: Johan Karlsson
Examiner at KTH: Johan Karlsson
Abstract

Most diseases have different heterogeneous effects on patients. Broadly, one may conclude what manifested symptoms correspond to which diagnosis, but usually there is more than one disease progression pattern. Because there is more than one pattern, and because each pattern may require a bespoke (and personalised) therapeutic intervention, time-series clustering is one option by which disease sub-populations can be identified. Such patient sub-typing is difficult due to information heterogeneity, information sparsity (few longitudinal observations) and complex temporal governing disease dynamics. To deal with these problems, and seeking to gain a robust description of them, we introduce a generative clustering model by way of a mixture of hidden Markov models. Our model deals with non-ergodic temporal dynamics, has variable state cardinality for the mixtures components and initialises the mixture in a more structured way than current methods.

With the task of disease progression modelling in mind, we also take a broader perspective on parameter learning in finite mixture models (FFM). In many mixture models, obtaining optimal or near-optimal parameters is difficult with current learning methods, where the most common approach is to employ monotone learning algorithms e.g. the conventional expectation-maximisation algorithm. While effective, the success of any monotone algorithm is crucially dependant on good parameter initialisation. A common approach is to repeat the learning procedure multiple times starting from different points in the parameter space or to employ model specific initialisation schemes e.g. K-means initialisation for Gaussian mixture models. For other types of mixture models the path to good initialisation parameters is often unclear and may require a solution specific not only model, but also the data.

To this end, we propose a general heuristic learning algorithm that utilises Boltzmann exploration to assign each observation to a specific base distribution within the mixture model, which we call Boltzmann exploration expectation-maximisation (BEEM). With BEEM, hard assignments allow straightforward parameter learning for each base distribution by conditioning only on its assigned observations. Consequently it can be applied to mixtures of any base distribution where single component parameter learning is tractable. The stochastic learning procedure is able to escape local optima and explores the parameter space, thus mitigates sensitivity to parameter initialisation. We show competitive performance on a number of synthetic benchmark cases as well as on real-world datasets. Finally we employ BEEM for the disease progression sub-typing task and contrast it to a task specific initialisation procedure on synthetic data as well as on a real progression modelling task, where we identify clinical phenotypes in Parkinson’s disease.
Sammanfattning


1 Introduction

Clustering is a statistical classification technique in which observations (data) are divided into groups (or clusters) such that items in one cluster are similar according to some metric, and different to items in other clusters. Clustering is typically used to identify (or reveal) hidden associations, patterns, relationships and structures in observations, not obvious through visual (or other) inspection. A temporal sequence, or time-series, is like any other object, and so also amenable to clustering. That being said, there is one crucial difference with time-series clustering (TSC); namely that attributes in a time-series are ordered [1, 4]. Because attributes are ordered in the sequence, this implies that the ordering, unlike regular clustering, may in itself be used as a discriminatory feature by a clustering algorithm. Consequently TSC has invited a multitude of applications, in a variety of domains such as biomedical informatics, biology [1], genetics, multimedia [39] and finance. In these domains, a variety of tasks are performed such as anomaly detection, sub-sequence matching [1], classification, segmentation and disease progression modelling [11, 6]. It is the latter topic in which this work finds utility.

Disease progression modelling, similar to “patient subtyping” [6], seeks to find patient groups with “similar disease progression pathways” [6]. Taking our queue from Liu et al. [32]; the goal of disease progression modelling is to construct and train (optimise) a model for the temporal evolution of a disease, trained on sequences of clinical measurements obtained from a longitudinal sample of patients. The need for this type of modelling has been tendered by the medical community, where precision medicine allows for patients to receive treatment tailored to their unique personal needs [6]. By leveraging the information contained in electronic health records (EHR) or clinical trials, disease progression modelling can be posed as an unsupervised learning problem, which seeks to segment clinical trials data of patients, into interpretable and clinically useful groupings. This requires us to uncover and capture the hypothesised structure in the observations.

A particular type of structure which is common to a lot of diseases, takes main stage in this work. Therein an important and particular feature of e.g. neurodegenerative diseases of the central nervous system is that of non-ergodicity. Chronic disease (i.e. unidirectional) such as multiple sclerosis (MS), Parkinson’s disease (PD), and Alzheimer’s disease (AD) are all non-ergodic – see fig. 1. Practically this means that if we can segment disease progression into different stages, invariably that segmentation will be ordinal, and is usually quantised on a severity scale which goes from ‘good’ to ‘worse’ in terms of the disease’s clinical manifestation. More importantly, the state of medical treatment for these ailments [62], means that there is currently no medication available (for PD, AD and MS) which cures the symptoms. For this reason, these neurodegenerative diseases follow a stationary non-ergodic temporal evolution process; a patient cannot today transition from a clinical assigned state of ‘worse’ to ‘better’. Consequently by way of Markov chains, if it is fully connected, or ergodic, each state of the model can be reached from every other state any number of steps greater than or equal to some finite integer. By considering the transition matrix for the aforementioned Markov chain, we can conclude that it must be fully connected. But if it is of a left-to-right type transition matrix, it is not, and hence cannot be ergodic. By approximating these disease progressions according to a left-to-right style transition flow, we employ a more faithful modelling paradigm.

To identify different types of disease progression, we cluster the temporal processes themselves. In doing so we adopt a generative modelling framework by way of the hidden Markov model (HMM) [38, 8], and in particular using a mixture of HMMs – one for each temporal process. HMM mixtures have been used to great effect before in various domains: categorisation of webpages [60] as well as activity analysis in microtubule
videos [2] – to mention but a few. We, however, design a new model-based clustering method which is capable of dealing with non-ergodic temporal processes, of variable length for use in sub-type identification of neurodegenerative diseases.

Within this thesis we also explore broader perspectives on mixture models. Prominent examples include the Gaussian mixture model (GMM) [31, 8, 38], the mixture of hidden Markov models (MHMM) [44, 28, 52, 12, 25, 53, 40], mixture of experts (MoE) [45], mixture of Gaussian processes (MGP) [30, 45, 61], as well as more recent additions such as the mixture of generative adversarial networks (MGAN) [37]. Application domains are plentiful and include e.g. multi-object target-tracking [30]; speaker-identification [48] as well as document clustering [9] to mention but a few.

When fitting finite mixture models (FMM) [38], we are required to find the parameters that maximise the likelihood of the observations. To fit a GMM for example, it is common to adopt the expectation-maximisation (EM) algorithm [13]. Blömer & Bujna [10] explain that the EM algorithm alternates between computing a lower bound of the log-likelihood and improving the current model w.r.t. this lower bound. In concluding the learning, the procedure converges to a particular stationary point on the likelihood function. However; the likelihood function is generally non-convex, possesses many stationary points, includes small local maxima and minima as well as saddle points. Importantly, the convergence of the EM algorithm to either type of point, depends crucially on the initialisation [10].

Initialisation sensitivity is not particular to GMMs however, the same problems arise for e.g. MHMMs as well, and indeed many of the other aforementioned mixture models. In particular those which rely on EM as the main driver of inference. To deal with this problem we propose a general method which carries across mixture model modalities (i.e. can be used for clustering sequences as well as objects) and, compared to many of the current methods in the respective mixture model domains, is comparatively simple. Our EM strategy uses the principle of maximum entropy, where our main contributions are:

1. We derive a mixture of HMMs capable of dealing with non-ergodic dynamics.
2. We propose a task specific initialization procedure for disease progression modelling with HMMs.

![Figure 1: Ergodic and non-ergodic transition dynamics. Figure 1a depicts a fully-connected (ergodic) transition diagram. Figure 1b depicts a edge-pruned transition diagram which is non-ergodic. Which is to say that fig. 1b is a directed graph with self-loops. Numbers represent state identifiers.](image-url)
3. Boltzmann exploration (also known as softmax action selection) [54, §2.8], is used to select a cluster (arm) for each observation. This is done under the standard reinforcement learning aegis, where each arm is selected with a probability proportional to its average reward [29].

4. Conventional EM [13] seeks out zero-gradients in the log-likelihood landscape, under maximum-likelihood estimation, and is thus prone to get stuck. To counter this we follow the Boltzmann update with an exploration step which seeks to avoid local maxima, minima and saddle-points.

5. We apply a novel Boltzmann exploration based initialisation scheme to GMMs, MHMMs and MGPs, and therein demonstrates competitive performance on synthetic and real datasets.

The thesis is outlined as follows; We start with a brief introduction to Parkinson’s disease and the Parkinson’s progression markers initiative (PPMI) data which is the data source that inspired this work. Following we give a review of finite mixture models and the EM algorithm where we go in depth on GMMs and HMMs. We then briefly touch on the multi armed bandit (MAB) problem to introduce the exploration-exploitation mechanism that inspired our proposed heuristic algorithm for parameter learning before we describe it in detail. We validate the algorithms performance for a selection of mixture models on both real and synthetic data sets. Finally we showcase the utility of our work for the application of disease progression modeling on Parkinson’s disease via the PPMI [36] data set.

2 Parkinson’s progression markers initiative

Parkinson’s disease is a progressively disabling neurodegenerative disorder that is manifested clinically by bradykinesia (slowness of movement), tremor, rigidity, flexed posture, postural instability and freezing of gait [23]. Practically PD is characterised by the loss of pigmented dopaminergic neurons [23] in the basal ganglia structure located in the mid-brain [43]. The development of novel treatments for many slowly progressing diseases, such as PD, is dependent on the ability to monitor and detect changes in disease progression. A clinical study which investigates this is the Parkinson’s Progression Markers Initiative (PPMI) – a publicly available study [34]. The PPMI data set contains clinical and behavioural assessments, imaging data as well as biospecimens [6]. It is longitudinal, has “unstructured” [6] elapsed time and is very data sparse (patients drop-out, some features only have a small number of recorded values etc.). A summary of the data set is found in table 1.

In this thesis we rely on the Movement Disorder Society (MDS) sponsored revision of the unified Parkinson’s disease rating scale (UPDRS) to measure quantify disease progression. There are four parts to the rating scale [21]; Part I concerns nonmotor experiences of daily living, part II concerns motor experiences of daily living. These parts designed to capture the subjects perception of how they are affected by the disease and are in part assessed by the subject. Part III is a the motor examination, and Part IV concerns motor complications. Both latter parts are assessed by the investigator. Part III, from heron referred to as MDS-UPDRS III, ranges from 7 to 86 points, where a higher score means more severely impaired motor function. MDS-UPDRS III is often claimed to be the most reliable metric to assess disease progression, however it has limitations which limits the precision of measurement of motor symptoms and impact in early stages of Parkinson’s disease [47]. Additionally there is also the Hoehn and Yahr (H&Y) scale [20], a commonly used scale with five stages where stage one implies minimal or no disability while stage five is given to subjects confined to bed or wheel-chair due to their...
Table 1: PPMI data overview.

<table>
<thead>
<tr>
<th>Enrolment Category</th>
<th>PD</th>
<th>HC</th>
<th>SWEDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of participants (n)</td>
<td>423</td>
<td>196</td>
<td>64</td>
</tr>
<tr>
<td>Total variable count</td>
<td>339</td>
<td>339</td>
<td>339</td>
</tr>
<tr>
<td>Mean observation years</td>
<td>5.1</td>
<td>4.9</td>
<td>2.4</td>
</tr>
<tr>
<td>Mean MDS-UPDRS\textsubscript{II}</td>
<td>5.9</td>
<td>0.5</td>
<td>5.7</td>
</tr>
<tr>
<td>Mean MDS-UPDRS\textsubscript{III}</td>
<td>20.9</td>
<td>1.2</td>
<td>14.3</td>
</tr>
<tr>
<td>Mean H&amp;Y stage</td>
<td>1.6</td>
<td>0.0</td>
<td>1.4</td>
</tr>
</tbody>
</table>

disability. Nowadays H&Y is largely superseded by the more modern MDS-UPDRS due to its improved granularity. Both are rating and are evaluated by interview and clinical observation. Consequently, they are very noisy. The last three features in table 1 are taken at baseline i.e. when the patients enter the PPMI study. There is also growing interest for brain imaging bio-markers for the assessment of PD progression, specifically dopamine transporter (DAT) levels in the striatum [42, 50], which can be subdivided into the putamen and caudate. Some patients have undergone a DAT-scan but shown no evidence of dopaminergic deficit (SWEDD), others are healthy controls (HC) and the table 1 shows the count for the number of PD patients.

While disease progression modelling is the target application for this thesis, the majority of this body of work is constituted by the development and modification of existing algorithms for this task, analysis of any clinical implications is omitted. Nonetheless, in section §7.2 we demonstrate an attempt to find PD progression sub types with the help of MDS-UPDRS\textsubscript{III} and DAT-putamen levels as progression markers.

3 Mixture models

A mixture model is a latent variable model which generates observable emissions $x$ from discrete latent states $z_n \in \{1, \ldots, K\}$ – see fig. 2. Each state is in itself a generative model $M_k$ or base distribution [38], such that we are mixing $K$ models together:

$$
P(x) = \sum_{k=1}^{K} P(x | z) P(z).
$$

In this way, complex distributions can be constructed from simpler components, even describing arbitrary densities when $K \to \infty$. In this thesis we turn to mixture models not to model complex distributions, but to find modalities in our observed data by the way of clustering.

Definition 3.1. (Mixture model). The mixture model is so prescribed because we are mixing together $K$ base distributions [38] as a linear combination, yielding a joint likelihood of the form:

$$
P(x_1, \ldots, x_n, \ldots, x_N, z_1, \ldots, z_n, \ldots, z_N | \zeta, \eta) = \prod_{n=1}^{N} P(x_n | z_n, \zeta) \cdot P(z_n | \eta)
$$

Figure 2: Graphical representation of a mixture model as a directed acyclic graph (DAG) with a single hidden node in one of $K$ states, parameters are common for all data points.
where \( P(z_n \mid \eta) \) is the mixture weight for latent mixture index (component) \( z_n \in \{1, \ldots, K\} \) with parameters \( \eta \). Further, \( P(x_n \mid z_n, \zeta) \) is the likelihood model for observation \( x_n \) with parameters \( \zeta \). The most likely assignment of observations is given by

\[
\text{arg max}_{z_1, \ldots, z_N} P(z_1, \ldots, z_N | x_1, \ldots, x_N). \tag{3}
\]

Equation (2) shows that we can factorise the joint, consequently the expression in eq. (3) is equivalent \([5, \S 20.1]\) to computing \( \text{arg max}_{z_n} P(z_n | x_n) \forall n \in \{1, \ldots, N\} \). Because cluster assignments are typically a priori unknown (in unsupervised learning), they need to be located via inference. Consequently, the optimal cluster assignment

\[
P(x_1, \ldots, x_N | \zeta, \eta) = \text{arg max}_{\zeta, \eta} \prod_{n=1}^{N} \sum_{z_n} P(x_n | z_n, \zeta) \cdot P(z_n | \eta) \tag{4}
\]

can be inferred via an optimisation procedure. Numerically this can be achieved using e.g. a gradient based approach, or, when the cluster indices are explicitly latent, we can apply an EM algorithm \([5, 38]\).

We adopt the view of mixtures presented by Bishop \([8, \S 9]\), in discrete latent variables are interpreted as defining assignments of observations to particular mixture components. This interpretation will later allows us to dip our toes in to the field of reinforcement learning and further consider such assignments as actions.

### 3.1 Gaussian mixture models

The Gaussian mixture model (GMM) is one of the first model one encounters in the literature \([38, 8]\) on unsupervised learning and clustering. As the name states, it is a superposition of \( K \) multivariate Gaussians with mean \( \mu_k \in \mathbb{R}^d \) and covariance matrix \( \Sigma_k \in \mathbb{R}^{d \times d} \). As alluded to in §3 the mixing of these components is controlled by a \( K \)-dimensional binary random variable \( z \) with elements \( z_k \). It is important to emphasize that a single emission of a mixture model is not generated by all or multiple, base distributions in the mixture. We instead think of an observation \( x \), as being generated by a single base distribution whose identity \( z \), is unobserved, hence we call it a latent variable. The values of \( z_k \) therefore satisfy \( z_k \in \{0, 1\} \) and \( \sum_k z_k = 1 \) such that \( z \) is a latent state vector with only one nonzero element determining the state.

Our goal is now to arrive at eq. (1) for the GMM, by defining the joint distribution \( P(x, z) \) in terms of a marginal distribution \( P(z) \) and a conditional distribution \( P(x \mid z) \). The marginal distribution over \( z \) is defined as a categorical distribution in terms of mixing weights \( \pi_k \) such that,

\[
P(z_k = 1) = \pi_k, \tag{5}
\]

where the mixing weights satisfy, \( 0 \leq \pi_k \leq 1 \) and

\[
\sum_k \pi_k = 1, \tag{6}
\]

assuring that they yield a proper probability distribution. Referencing definition 3.1 we recognize the collection of \( \Sigma_k \), and \( \mu_k \) as \( \zeta \) and the collection of \( \pi_k \) as \( \eta \). Since \( z \) is a binary vector with only one nonzero element we can write this distribution in the form,

\[
P(z) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{7}
\]

In a similar fashion, the conditional Gaussian distribution of \( x \) given a fixed \( z \) is expressed as,
\( P(x \mid z_k = 1) = N(x \mid \mu_k, \Sigma_k), \) \hspace{1cm} (8)

which can be written on the form,

\[ P(x \mid z) = \prod_{k=1}^{K} N(x \mid \mu_k, \Sigma_k)^{z_k}. \] \hspace{1cm} (9)

Finally the marginal distribution of \( x \) can be obtained by summing the joint distribution \( P(z) P(x \mid z) \) over all possible states of \( z \),

\[ P(x) = \sum_z P(z) P(x \mid z). \] \hspace{1cm} (10)

making use of eq. (7) and eq. (9) to arrive at,

\[ P(x) = \sum_k \prod_{k=1}^{K} \pi_k^{z_k} N(x \mid \mu_k, \Sigma_k)^{z_k}, \] \hspace{1cm} (11)

\[ = \sum_{k=1}^{K} \pi_k \cdot N(x \mid \mu_k, \Sigma_k). \] \hspace{1cm} (12)

What we now have in eq. (12) is a formulation of the Gaussian mixture model with an explicit latent variable which allows us to work with the joint distribution \( P(x, z) \). This is of great utility in the aforementioned clustering task, where the model parameters of the data generating mixture is unknown, and have to be inferred from observations. In the following section we will introduce a powerful method that can help us find these parameters, namely the expectation-maximization (EM) [13] algorithm.

3.1.1 Expectation maximization for Gaussian mixture models

Before we venture into the details of finding optimal model parameters we should define precisely what we are trying to achieve. As mentioned in section §3, a mixture can describe arbitrary densities when \( K \to \infty \), this implies that we could obtain an arbitrarily good fit to any data set by simply increasing \( K \). However, as \( K \) approaches the cardinality of our dataset \( N = |X| \), the desired effect of grouping observations is lost. Therefore we define our task as, given a predefined number of base distributions, i.e. clusters, find the optimal mixture parameters \( \zeta^* \) and \( \eta^* \). For the Gaussian mixture model we can formalize this as maximizing the likelihood of observing \( X \),

\[ P(X \mid \zeta, \eta) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \Sigma_k), \] \hspace{1cm} (13)

for a fixed \( K \). There is more than one way to maximize this quantity, including gradient based [17] and Markov chain Monte Carlo methods. In this section we present and motivate the EM algorithm, a general algorithm used for computing the maximum likelihood (ML) or maximum a posteriori\(^1\) (MAP) parameter estimate in many machine learning models.

Before we begin, it is helpful to define a few quantities that will help us understand the motivation behind the EM algorithm. First we define the base distribution \( k \)'s

\(^1\) A MAP estimate of the model parameters is obtained by maximizing the likelihood of the observed data conditioned on a prior distribution over the model parameters. It can be used to incorporate prior information about model parameters or as a form of regularization to avoid over-fitting to the observed data.
responsibility for sample \(x_n\) as the conditional probability of \(z\) given \(x\), which can be found via Bayes’ theorem:

\[
r(z_k) \triangleq P(z_k = 1 \mid x) = \frac{P(z_k = 1) P(x \mid z_k = 1)}{\sum_{i=1}^{K} P(z_i = 1) P(x \mid z_i = 1)} = \frac{\pi_k \cdot N(x \mid \mu_k, \Sigma_k)}{\sum_{i=1}^{K} \pi_i \cdot N(x \mid \mu_i, \Sigma_i)}.
\]

If we view \(\pi_k\) as the prior probability of \(z_k = 1\), then \(r(z_k)\) is the posterior probability after we have observed \(x\). It follows that an observation is said to belong to the base model (i.e. cluster) with largest value of this quantity.

We can now make use of that by definition, the gradient of the likelihood function w.r.t. the model parameters is zero at the location of \(\zeta^*, \eta^*\) in the parameter space. Noting the product over the observations in eq. (13) we take the logarithm of the likelihood function to simplify our next step of finding the derivative at this location,

\[
\log\{P(X \mid \zeta, \eta)\} = \sum_{n=1}^{N} \log\left\{ \sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \Sigma_k) \right\}.
\]

By taking the derivative of eq. (15) with respect to the means \(\mu_k\) and setting it to zero, we have

\[
0 = \sum_{n=1}^{N} \pi_k N(x_n \mid \mu_k, \Sigma_k) \Sigma_k^{-1} (x_n - \mu_k).
\]

If we now recognize the fraction in eq. (16) as \(r_n(z_k)\) and let \(N_k = \sum_{n=1}^{N} r_n(z_k)\), we can multiply by \(\Sigma_k\) and rearrange to obtain,

\[
\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_n(z_k) x_n.
\]

Here \(N_k\) can be interpreted as the effective number of points assigned to base distribution \(k\), and eq. (17) as the responsibility weighted mean of all observations in \(X\). If we now instead take the derivative of the log-likelihood function with respect to \(\Sigma_k\), and repeat a similar procedure, we obtain,

\[
\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r_n(z_k)(x_n - \mu_k)(x_n - \mu_k)^T,
\]

which analogous to eq. (17) can be viewed as a responsibility weighted covariance matrix. Finally we repeat the procedure one last time with respect to the mixing weights \(\pi_k\), this time however, things are more complex since the condition in eq. (6) must hold. To achieve our goal of isolating \(\pi_k\) we need to follow the suggestion of Bishop [8] and make use of the Lagrange multipliers to maximize

\[
\log\{P(X \mid \zeta, \eta)\} + \lambda \left( \sum_{k=1}^{K} \pi_k - 1 \right)
\]

which gives,

\[
\sum_{n=1}^{N} \frac{N(x_n \mid \mu_k, \Sigma_k)}{\sum_{i=1}^{K} \pi_i N(x_n \mid \mu_i, \Sigma_i)} + \lambda = 0 \quad \forall k \in \{1, \ldots, K\}.
\]


We can now multiply eq. (20) by \( \pi_k \) to obtain,

\[
N_k = -\pi_k \lambda, \tag{21}
\]

with the help of eq. (14). Summing eq. (21) over \( K \), as allowed by eq. (20), we see that \( \lambda = -N \) and thus,

\[
\pi_k = \frac{N_k}{N}, \tag{22}
\]

which can be interpreted as the average responsibility of base distribution \( k \) w.r.t. all observations. To summarise, we now have solutions for the maximum likelihood w.r.t. all mixture parameters in results eqs. (17), (18) and (22), however they do not constitute a closed form solution since \( r_n(z_k) \), and in turn \( N_k \), depends on \( \pi, \mu \) and \( \Sigma \). Despite not providing an analytic solution they indicate an iterative method for finding the maximum likelihood parameters, namely the EM algorithm for GMMs. Given some initial values for the mixture parameters we can step towards the solution using our results. Each iteration involves two consecutive steps, first component responsibility \( r_n(z_k) \) is inferred from the current parametrisation through eq. (14), this is called the expectation step or the `E-step'. Following the E-step is the maximisation step or `M-step', the mixture parameters are re-estimated given the inferred component responsibility via eqs. (17), (18) and (22). Completion of the M-step marks the end of one EM iteration, the process is then repeated until

\[
\log\{p(X \mid \zeta, \eta)\} - \log\{p(X \mid \zeta_{\text{old}}, \eta_{\text{old}})\} < \varepsilon, \tag{23}
\]

where \( \varepsilon \) is some predefined tolerance. The pseudo-code for the EM-algorithm for GMMs is given in algorithm 1.

**Algorithm 1:** The EM algorithm for Gaussian mixture models

| Input: \{\{x_1, \ldots, x_N\}, \{\pi_1, \ldots, \pi_K\}, \{\mu_1, \ldots, \mu_K\}, \{\Sigma_1, \ldots, \Sigma_K\}, \varepsilon\|

| for \( k \in \{1, \ldots, K\} \) do |
| for \( n \in \{1, \ldots, N\} \) do |
| \( r_n(z_k) = \frac{\pi_k \cdot N(x_n \mid \mu_k, \Sigma_k)}{\sum_{i=1}^{K} \pi_i \cdot N(x \mid \mu_i, \Sigma_i)} \) \( \triangleright \) Update responsibilities |

| for \( k \in \{1, \ldots, L\} \) do |
| \( \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_n(z_k) x_n \) \( \triangleright \) Update mean |
| \( \Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r_n(z_k) (x_n - \mu_k)(x_n - \mu_k)^T \) \( \triangleright \) Update covariance |
| \( \pi_k = \frac{N_k}{N} \) \( \triangleright \) Update mixing weights |

| Output: Model parameters \( \zeta, \eta \) which in the limit segments the observations into \( K \) bins. |

Now that we have motivated and defined the steps of the EM-algorithm we will proceed by discussing what one can expect from the obtained solution. As a consequence of the results in in eqs. (17), (18) and (22), we see that stepping trough any zero gradient location will result in \( \zeta_{\text{new}} = \zeta_{\text{old}} \) and \( \eta_{\text{new}} = \eta_{\text{old}} \). Further, as elegantly shown by Murphy [38, § 11.4.7.1-2], the EM algorithm is guaranteed to monotonically increase
the observed data log likelihood. This means that EM is effectively a hill climbing algorithm, that always improves the solution until a critical point is found (given a sufficiently small $\varepsilon$). Equivalently, EM can neither escape a local maximum or saddle point.

It can be shown [38, § 11.3.2] that computing the ML estimate of a mixture model is non-convex. As such there are no general guarantees for the quality of an estimate produced via EM. This has inspired research into various alternative methods able to avoid local maximaus or providing guarantees for the quality of the solution. A common approach is to repeat the EM algorithm multiple times from different initial parameters or to use K-means initialisation [8, § 9.2.2] to find good initial parameters. In section §5 we present Boltzmann exploration exploration maximization (BEEM) as a more general alternative, that can be applied for parameter learning in a multitude of FMMs including mixtures of Hidden Markov models that will be discussed in the following section.

3.2 Hidden Markov model review

In contrast to the GMM the Hidden Markov model is capable of generating sequences of observations emitted one at the time through the evolution of a Markov process. The graphical model of the HMM is shown in fig. 3. Seen is a Markov chain with latent variables given by $z_{1:L}$ and observed of measured variables given by $x_{1:L}$, where $L$ is the length of an observation sequence. The first-order Markovian nature of the model manifests itself through the conditional relationships found in the latent and observed structure of the model. An emission model arises from the conditional relationship $P(x_t | z_t)$ and a transition model through $P(z_t | z_{t-1})$ [5]. A formal definition [14] is provided in definition 3.2.

**Definition 3.2.** (Hidden Markov model). A hidden Markov model (HMM) is a doubly-stochastic Markov chain in which the latent state-sequence $z_{1:L}$ is drawn according to a Markov chain on a discrete state-space defined on $Z$ [56, §3]. The size of $Z$ is given as $|Z| = m$. In parallel, there evolves an observation sequence $x_{1:L}$ which is step-wise conditionally dependent on $z_{1:L}$. Each observation $x_t$ is conditionally independent of all the other observations given $z_t$. The state-specific transition distribution $A_k$ for state $k$, allows latent states to evolve as $z_t | z_{t-1} \sim A_{z_{t-1}}$, yielding the generative process for the HMM:

- Initial state dist.: $z_1 \sim \pi$, $\pi \in \mathbb{R}^{1 \times m}$
- Transition dist.: $z_t | z_{t-1} \sim A_{z_{t-1}}$, $A \in \mathbb{R}^{m \times m}$
- Emission dist.: $x_t | z_t \sim P(\cdot | \lambda_{z_t})$, $\lambda \in \lambda$

![Figure 3: A hidden Markov model. We can model sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variable [8].](image)
where $\lambda_k$ are the state-specific emission parameters s.t. $\lambda \triangleq \{\lambda_k\}_{k=1}^m$. The transition functions (rows of $A$) are implicitly normalised by virtue of the modelling density employed in the model construction, where we either perform element-wise division by the sum of the transition function or use e.g. a Dirichlet distribution as our density. Finally, $\zeta \triangleq \{\pi, \lambda, A\}$ denotes the set of parameters which govern the model. Parameter estimation in an HMM is by way of the Baum-Welch algorithm – an expectation-maximisation method [44].

In definition 3.2 we have assumed that the evolving Markov chain is ergodic, which is to say that $A$ is fully connected. For many temporal processes, this assumption is too weak to be interpretable. Recalling the discussion in §1, this is true in many disease domains where there often is no ready cure, so the severity of a patient’s disease cannot reverse but is unidirectional. That being said, many treatments are available to help relieve the symptoms and maintain the patient’s quality of life [51].

3.2.1 Ergodicity

Bishop[8] explains that a sufficient condition for ergodicity is that none of the conditional distributions (rows) in the transition matrix be anywhere zero. If this is the case, then any point in $\mathcal{Z}$ space can be reached from any other point in $\mathcal{Z}$ space, in “a finite number of steps” – a formal definition is provided in definition 3.3.

**Definition 3.3. (Ergodic Markov chain).** A Markov chain is said to be **ergodic** if it asymptotically converges to its (unique) invariant distribution. Or, in other words, it visits every part of the state-space $\mathcal{Z}$.

If definition 3.3 does not hold true, and the transition matrix instead has restricted movement options, e.g:

$$
A = \begin{bmatrix}
a_{1,1} & a_{1,2} & 0 & \cdots & 0 \\
0 & a_{2,2} & a_{2,3} & \cdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & 0 \\
0 & 0 & \cdots & a_{m-1,m-1} & a_{m-1,m} \\
0 & 0 & \cdots & 0 & a_{m,m}
\end{bmatrix}
$$

then an ergodic Markov chain cannot be formed. Equation (24) is a special case of a non-ergodic left-to-right HMM transition matrix, typically used to model transient processes (an example of which is Parkinson’s disease). In such a model, state $m$ is necessarily absorbing and is naturally the final state [12]. Like the matrix shown in eq. (24), we restrict the model topology to only allow self-transitions and one-step forward transitions as shown in eq. (24). Such a restriction enforces a hierarchical relationship between the states and severely limits the dynamics any one component in the mixture can model. For a descriptive clustering algorithm both these properties are desirable since we want the clusters to contain only similar dynamics and the hierarchy allows us to interpret and compare clusters.

Parameter estimation (via expectation-maximisation [44]) in a non-ergodic HMM is a more interesting problem, than the standard ergodic setting. Consistent parameter estimates are difficult to achieve from a single, long, observation sequence because as soon as the HMM reaches the final absorbing state $m$, the observed part of the HMM is emitted from the absorbing states distribution. Consequently the remainder of the sequence provides no additional information [12], from whence to adjust the parameters of the transition matrix. A partial solution to this problem is to use multiple observation sequences, to estimate the model parameters associated with the transient states. We say ‘partial’ because this solution in conditional on the sequence lengths $\{L_n \mid n = 1, \ldots, N\}$ being large enough. Precisely how large that is, depends on how many states exist in
the HMM, and if a complete chain can be formed in any observed sequence. If they are short, as is common in clinical trials data, then estimating parameters becomes difficult for the aforementioned reasons. Consequently, we require a method for clustering short, non-ergodic, time-series.

3.3 Time-series clustering with HMMs

In our problem setting of heterogeneous sequences and short heterogeneous sequence lengths, we seek to discover natural groupings in \( D \). Standard finite mixture models employ linear combinations of distributions to model variables. The grouping task becomes more complex and non–trivial when we are dealing with sequences since there is no obvious distance metric and sequences might not be of equal lengths (which most clustering algorithms require). Consequently, instead of using a distribution as mixture component we can use a generative time-series model i.e. the HMM. And so we furnish eq. (25) with sequence modelling capabilities – an idea first proposed by Van de Pol & Langeheine [58]. This results in a mixture of HMMs (MHMM), a paradigm previously employed in e.g. [44, 28, 52, 12, 25, 53, 40] – though none of which deal with short non-ergodic sequences. Because of this, it is difficult to compare how their methods would perform in our proposed application domain. Whilst single HMMs are a ubiquitous tool for modelling sequences, the expressiveness of one HMM may not be enough to model complex interactions in an observation stream.

To investigate this idea let \( M_k(S | \zeta_k) \) be the density for sequence data \( S \), with mixture component \( M_k \) i.e. the \( k^{th} \) HMM – with parameters \( \zeta_k \). The objective is then to find \( K \) clusters described by \( K \) distinct HMMs. Consequently we assume that the data is generated by such a mixture in the form

\[
P(S) \triangleq \sum_{k=1}^{K} \text{P}(S | M_k) \cdot \text{P}(M_k),
\]

which can be recognised as eq. (1) with observed sequences and hidden states represented as \( M_k \) to avoid confusion with the existing hidden states of the HMM base distributions. In fact, such a mixture can be described by a single composite HMM with a block diagonal transition matrix \( A \) composed of diagonal blocks \( \{A_1, \ldots, A_k, \ldots, A_K\} \):

\[
A = \\
\begin{bmatrix}
A_1 & 0 & \ldots & 0 \\
0 & A_2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & A_K
\end{bmatrix}
\]

(26)

where \( A_k \) is the transition matrix of the \( k^{th} \) HMM. The objective of the algorithm is thus to find the parameters of the (hypothesised) composite HMM. We say ‘hypothesised’ because it is important, again, to remember that the underlying assumption of this model is that the data was generated by a model of this topology and type. Further, we extend the original MHMM by allowing each mixture component to have a component-specific number of states i.e. \( m_k \forall k \in \{1, \ldots, K\} \).

Finally, the log-likelihood of a composite HMM is given [25] by

\[
\log \mathcal{L} = \sum_{i=1}^{N} \log \left[ \sum_{k=1}^{K} \text{P}(M_k) \sum_{z, M_k} \text{P}(S | z, M_k) \text{P}(z | M_k) \right]
\]

(27)

Our model is from hereon referred to as a disease progression HMM or DP-HMM, to distinguish it from the baseline MHMM used in this exposition, as well as in the experiments.
3.3.1 Parameter learning in HMM mixtures

Now, Smyth [52] notes that since the Baum-Welch algorithm is hill-climbing the likelihood surface, the final solution is heavily influenced by the initial conditions. Combined with the non-ergodic nature, and short-sequence domain, in which our application area resides; the initialisation is crucial for our model to run at all.

We approach the parameter estimation from two fronts. First in §3.3.2, we explore a bespoke alternative initialization methodology to that proposed by Smyth [52]. Finally in §5, we introduce BEEM as defined by algorithm 3 in the context of HMM mixtures.

3.3.2 Cluster initialisation

To aid the learning of the DP-HMM parameters we want to initialise the clusters such that similar sequences belong to the same cluster more often than not. Smyth [52] suggests that the inter-sequence similarity can be estimated by fitting a model $M_i$ to each individual sequence $S_i$, and computing the log-likelihood of each of the $N$ sequences given model $M_i$, i.e. calculating $\log L(S_j | M_i), 1 \leq i, j \leq N$. However, given that observation sequences from e.g. the PPMI dataset [34] are very short $L_i \leq 11$ – see fig. 4) and that the latent temporal dynamics can be assumed to be non-ergodic i.e. a subject will not recover from Parkinson’s disease – the dynamical system used for modelling this evolution, is unidirectional. Consequently, any model estimated from a single sequence will be very noisy, for the reasons discussed in §3.2.1. We propose an alternative initial clustering scheme, which uses the linearised slope of each emission variable, for each sequence, and use that as a proxy for the transition dynamics. The cluster initialisation part of our algorithm is summarised as follows:

1. For each $S_i$, fit a linear regression model $S_i = m^T t + c$, for each emission variable where $m$ is the slope vector and $c$ is the intercept vector. We let the time-indices occupy the vector $t \in \mathbb{Z}_d \times L_i$ where $d$ is the dimensionality of the feature vectors used in $S_i$.

2. The sequences are separated into $K$ groups by applying $k$-means clustering to the set of extracted slope vectors $\{m_1, \ldots, m_i, \ldots, m_N\}$.

---

Figure 4: Histogram over preprocessed sequence lengths calculated from the longitudinal PPMI dataset, using one feature as reference.

---

2 Dopamine transporter single-photon emission computed tomography is used to measure the dopaminergic deficit in the putamen of the brain, this univariate feature was used to construct the histogram over sequence lengths.
Algorithm 2: DP–HMM sequence segmentation via slope initialisation

**Input:** \{S_1, \ldots, S_N\}, \{m_1, \ldots, m_K\}, \varepsilon

for \( i \in \{1, \ldots, N\} \) do
\[ S_i = m_i^T t + c_i \quad \triangleright \text{Fit a linear regression model to each sequence} \]
\{\bar{S}_1, \ldots, \bar{S}_K\} ← k\text{-means cluster} \{m_1, \ldots, m_i, \ldots, m_N\}

while \( \theta - \theta_{\text{old}} > \varepsilon \) do
  for \( k \in \{1, \ldots, K\} \) do
    Condition model \( \mathcal{M}_k \) on sequence cluster \( \bar{S}_k \)
    Let \( A_k \in \mathbb{R}^{m \times m} \quad \triangleright \text{Store the transition matrix for model } \mathcal{M}_k \)
    \[ \pi \leftarrow \frac{|S_k|}{N} \pi_1, \ldots, \frac{|S_k|}{N} \pi_K \quad \triangleright \text{Initial state distribution is weighted by cluster assignments} \]
    \[ A = \text{diag}[A_1, \ldots, A_K] \quad \triangleright \text{Compose block diagonal transition matrix} \]
    \[ \lambda \leftarrow \{\lambda_{S_1}, \ldots, \lambda_{S_K}\} \quad \triangleright \text{Emission parameters initialised to learned values} \]
  end for
  Condition composite model \( \mathcal{M} \) on sequence \( S_i \) \quad \triangleright \text{Fit composite model to all sequences}
  \{\bar{S}_1, \ldots, S_i, \ldots, S_N\} ← \text{argmax}_k \log L(\mathcal{M}_k | S_1, \ldots, S_i, \ldots, S_N) \quad \triangleright \text{Use equation eq. (27)}
end while

**Output:** Composite model parameters \( \theta \) which in the limit segments the sequences into \( K \) bins.

**Model initialisation and learning** Following the cluster initialisation, \( K \) HMMs are fitted to the sequences in each cluster. The resulting \( K \) sets of model parameters are then used to initialise the composite HMM. The block diagonal transition matrix \( A \) (see eq. (26)) is built from the components \( \{A_k \mid k = 1, \ldots, K\} \). The initial state distributions \( \pi_k \) are normalised by \( |S_k|/N \) where \( |S_k| \) is the number of sequences that belong to cluster \( k \). Finally the emission distribution parameters are initialised to their learned values.

Once the composite model parameters have been initialised, model learning proceeds via normal HMM expectation-maximisation through the Baum-Welch algorithm [8], and the DP-HMM model requires no modification of the standard algorithm to facilitate parameter estimation. Note that the initial clusters are noisy estimates and that sequences may be reclassified during the final training procedure. The initialisation procedure described within in this section will from hereon be referred to as slope initialisation or SI. Finally, the full algorithmic details for training the DP-HMM via SI, can be found in algorithm 2.

Throughout we have not discussed how to set \( m \) or \( K \). The reason being that there has been considerable prior work on this problem. Smyth [52] suggests using the posterior probability of \( K \) as estimated by cross-validation. Extensive work has also been done within the space of Bayesian nonparametric (BNP) statistics using e.g. infinite mixture models [45], or BNP HMMs [14, 7]. Certainly there is ample scope to expand our method to incorporate any or many of these powerful ideas, but that is left for future work. Because model evaluation and parameter learning is fast, a simple
search over \( m \) and \( K \) suffices for the purposes of this model. This concludes the review of mixture models and we will now proceed to present ideas for improved robustness of parameter learning via a reinforcement-learning inspired initialisation scheme.

4 Bandit optimisation review

Before presenting Boltzmann exploration expectation maximization in the next section we will give a brief introduction to the topic that inspired the proposed heuristic, namely bandit optimisation. The bandit problem was introduced in a paper [57] by William R. Thompson in 1933 and has since become a classic problem in operations research. Despite being simple to understand even for amateur mathematicians it was by some considered unsolvable. The MAB problem is possibly best explained in terms of a gambler playing multiple slot machines at a casino. Consider the analogy presented in the following paragraph.

The \( N \)-armed bandit problem: Consider a gambler entering a casino for the first time. She is faced with \( N \) different one arm bandit slot machines, see fig. 5. Our gambler obviously wants to win as much money as possible before it is time to go home, however she does not know beforehand which machine is the best to play. In layman’s terms, she is faced with the problem of figuring out which slot machine gives the highest reward while same time maximising her profit. To pose this as an optimisation problem we can formalise the setting as follows.

A one armed bandit or a bandit process, is a special type of Markov decision process with only two possible actions.

- **Continue**: The process evolves to state \( z_{t+1} \) according to the Markov transition probability \( P(z_t, z_{t+1}) \) and reward \( r(z_t) \) is observed.

- **Freeze**: The state does not change and no reward is observed.

In turn a collection of \( N \) such bandit processes is a \( N \)-armed bandit or a simple family of alternative bandit processes [27]. At each time \( t \in \{1, 2, \ldots\} \) an agent has select one out of \( N \) actions by evolving one bandit process \( i_t \in \{1, \ldots, N\} \). An action consumes one time unit and gives a reward \( r_{i_t} \). The rewards for each action is governed by a discrete or continuous distribution with unknown parameters. These distributions are independent for the different arms. The objective that arises from this setting is to take actions in a way that maximizes the expected accumulated reward at some time horizon \( T \). That is maximising the expression,

\[
\mathbb{E} \left[ \sum_{t=0}^{T} \beta^t r_{i_t}(z_i(t)) \right],
\]

where \( z_i(t) \) is the state of the of arm (process) \( k \) at time \( t \). Note that we have included a discount parameter \( \beta \in (0, 1) \) to accommodate scenarios where short term rewards are of higher value, the undiscounted case is achieved by letting \( \beta \) go to 1. Maximising this quantity introduces the problem of exploration and exploitation trade-off that is central to reinforcement learning for which the MAB is an important corner stone. One has to
choose between exploiting arms that seem to have good reward distributions given our past history of actions and explore less played arms that could turn out to be better.

The MAB problem has been thoroughly researched since its introduction and was notably solved by Gittins [19] in 1979, when he presented an optimal dynamic programming solution for the discounted MAB, today known as Gittin’s index. Approximate solution nevertheless still receive a lot of attention since they can offer near optimal performance at a fraction of the computational cost. The assortment of approximate algorithms is wide, ranging from Bayesian ideas such as upper confidence bound (UCB) [41, 3] algorithms and Thompson sampling [57] to simple semi-uniform strategies where the best performing arm is selected with higher probability. In this section we restrict ourselves to the introduction of a middle ground approach known as Boltzmann exploration or softmax action selection [54]. It is a common technique used to introduce an exploration-exploitation mechanism in reinforcement learning settings, even beyond the MAB problem. We will later in §5 show how to re-purpose it for parameter learning in mixture models.

4.1 Boltzmann exploration for bandits

The most primitive bandit strategy would be to always chose the arm with the highest empirical mean reward. It is easy to see that this greedy strategy fails to find the best arm with positive probability. A simple improvement is the ε-greedy algorithm [55] where the arm with currently highest empirical mean reward is selected with probability $1 - \varepsilon$ and a random arm is selected with probability $\varepsilon$. While this type of algorithm always finds the best arm given a sufficient amount of actions due to its constant exploration level, it is also prone to over explore, for the same reason. Another variation of this type of strategy is Boltzmann exploration [55] in which the next arm to pull is drawn from a multinomial distribution.

**Definition 4.1.** (Boltzmann exploration). Softmax action selection methods are based on Luce’s axiom of choice [33] and selects each arm with a probability given by the Boltzmann distribution, that is proportional to its average reward [29]. As such, one version of Softmax action selection, selects arm $z_n$ on the $n^{th}$ play, using the Boltzmann distribution

$$P(z_n | \tau) = \frac{\exp(\hat\mu_{z_n} / \tau)}{\sum_{n=1}^{N} \exp(\hat\mu_{z_n} / \tau)} \quad (29)$$

where $\hat\mu_{z_n}$ is the empirical average of the rewards obtained from arm $z_n$ up until round $n$. High temperatures $\tau$ cause the actions to be all (nearly) equiprobable, whereas low temperatures cause a greater difference in selection probability for actions that differ in their value estimates. In the limit as $\tau \to 0$, softmax action selection becomes the same as greedy action selection [29, 54].

Correspondingly for both Boltzmann exploration and ε-greedy there exists no rule to determine the parameters $\varepsilon$ and $\tau$ in a way that ensures good performance. While Boltzmann exploration takes the relative performance of an arm in to account, both algorithms lack a measure of confidence. This results in that a suboptimal arm that has been played enough times to give a good estimate of its underlying distribution will continue to be played indefinitely although with lower probability. In the light of the previously mentioned algorithms Boltzmann-exploration may seem primitive, but with adequately tuned parameter decay over time it can be competitive with more sophisticated algorithms.

---

3Note that in thermodynamics it is typical to negate the exponent’s argument, as this corresponds to a lowest energy state having the highest probability.
5 Boltzmann exploration expectation-maximisation

We will now proceed to transfer the ideas found in bandit optimization to the domain of parameter learning. Using the material provided in §4 herein we describe a method that uses a modified version of Softmax action selection, to learn mixtures of generative models with a view of overcoming complex initialisation schemes by way of exploration-exploitation.

As alluded to in the above paragraph, we seek a principled way to maximise the expression in eq. (4) w.r.t the model parameters. What follows is a novel and simple method for achieving this goal using a modified gradient bandit algorithm for learning mixture-model component assignments. Unlike the EM procedure for e.g. GMMs or MHMMs, we do not seek to compute the responsibility weighted parameter updates, but rather assign each sample to a mixture component straight away and subsequently update each component as a likelihood model conditioned on the assigned samples. We will annotate the method exposition analogous to the EM procedure to promote ease of comparison.

Component assignment (expectation step) In the classical EM setting, the expectation step (E-step) for mixture models [5, §20.2] utilises the update

$$ p_{\text{new}}(z_n = k \mid x_n, \z_t, \eta_t) \propto P(x_n \mid z_n, \z_{t-1}) \cdot P(z_n \mid \eta_{t-1}) \tag{30} $$

where the right-hand side is also called the responsibility [38, §11.4], which cluster $$ z_n = k $$ takes for observation $$ x_n $$. We use index $$ t $$ to keep track of the iteration count. The E-step takes the following simple form, which is the same for any mixture model:

$$ r_{n, z_n} \triangleq p_{\text{new}}(z_n = k \mid x_n, \z_t, \eta_t) = \frac{P(x_n \mid z_n, \z_{t-1}) \cdot P(z_n \mid \eta_{t-1})}{\sum_{z_n} P(x_n \mid z_n, \z_{t-1}) \cdot P(z_n \mid \eta_{t-1})}. \tag{31} $$

Our first contribution modifies the standard E-step in eq. (31) by fixing uniform mixing weights $$ P(z_n \mid \eta) = \frac{1}{K} $$ and instead determines the cluster responsibility via the Boltzmann distribution vis-à-vis definition 4.1, by substituting eq. (30) into eq. (29):

$$ r_{n, z_n} = \frac{\exp \left[ \log P(x_n \mid z_n, \z_{t-1})/\tau + \log P(z_n \mid \eta_{t-1})/\tau \right]}{\sum_{z_n} \exp \left[ \log P(x_n \mid z_n, \z_{t-1})/\tau + \log P(z_n \mid \eta_{t-1})/\tau \right]} \tag{32} $$

$$ = \frac{P(x_n \mid z_n, \z_{t-1}) \cdot p(z_n \mid \eta_{t-1})}{\sum_{z_n} P(x_n \mid z_n, \z_{t-1}) \cdot p(z_n \mid \eta_{t-1})} \tag{33} $$

$$ = \frac{P(x_n \mid z_n, \z_{t-1}) \cdot P(z_n \mid \eta_{t-1})}{\sum_{z_n} P(x_n \mid z_n, \z_{t-1}) \cdot P(z_n \mid \eta_{t-1})}^{1/K} \tag{34} $$

Having asserted uniform mixing weights, we introduce a modified responsibility:

$$ r'_{n, z_n} \triangleq p_{\text{new}}(z_n = k \mid x_n, \z_t) = \frac{P(x_n \mid z_n, \z_{t-1})}{\sum_{z_n} P(x_n \mid z_n, \z_{t-1})} \tag{35} $$

where we have taken the logarithm of the exponential function’s arguments.

To understand the rationale behind the algorithm design, it is helpful to view the task of finding optimal cluster assignments roughly as a bandit problem. We pose

---

4 Assignment rewards are not independent of past decisions, thus not conforming to the Markov property.
the problem as follows; an agent is faced with the task of assigning samples to the components in such a way that maximises the accumulated maximum data likelihood \( \sum_t \sum_n \max_z \{ \mathbb{P}(x_n \mid z_n, \zeta_t, z_t) \} \) at horizon \( T \). Since each assignment is made in the context of \( x_n \) and \( \zeta_t \) are updated at each iteration, our task is similar to that of a non stationary contextual bandit. While contextual bandits explicitly model the expected reward for each action, we simply let \( \mathbb{P}(x_n \mid z_n, \zeta_{t-1}, z_t) \), be a noisy estimate of the reward at iteration \( t \). In order to introduce an exploration mechanism, we sample a hard component assignment from the responsibility distribution, rather than computing responsibility weighted parameter updates as in the EM algorithm. The motivation for fixing uniform mixing weights is to avoid the probability of assignments being amplified for large clusters, thus resulting in smaller clusters being assigned less samples and eventually vanishing. Consequently, we have constructed an algorithm which at each iteration learns a better similarity metric and tries to group similar samples together. We concede that our problem does not qualify under the definition of a bandit, however it is a apt analogy.

Now, the EM procedure monotonically increases the observed data log-likelihood until it reaches a local maxima, minima or saddle-point [38, §11.4.7]. But there is nothing which prevents the method from getting stuck at either of these points, as EM only seeks out zero-gradients in the log-likelihood landscape. Consequently, to prevent this, we introduce the aforementioned exploration mechanism as part of the component assignment. A new component assignment for observation \( x_n \), is sampled as:

\[
z_n \sim \text{Categorical}(K, \{r_{n,1}, \ldots, r_{n,K}\}) \tag{36}
\]

using the responsibilities found in eq. (35) when \( z_n \in \{1, \ldots, K\} \). One full pass of this update results in a new or the same assignment for each observation. Where the temperature in eq. (35) is allowed to decrease with each EM update, to reduce the level of exploration. To achieve this we introduce an exponential cooling schedule \( g(\tau, t, \alpha) \triangleq \tau \alpha^{t-1} \) which takes as input the current temperature \( \tau \), the step count \( t \) and decay factor \( 0 < \alpha < 1 \). The full BEEM procedure is presented in algorithm 3.

**Algorithm 3: BEEM**

```
Input : \( \{x_1, \ldots, x_N\}, \{M_1, \ldots, M_K\}, \varepsilon, \tau, g(\tau, t) \)
Let \( X = \{x_1, \ldots, x_N\} \) \( \triangleright \) Set of all observations
Let \( V \in \mathbb{R}^{N \times K} \) \( \triangleright \) Value matrix for \( \{M_1, \ldots, M_K\} \)
Let \( \{S_1, \ldots, S_k, \ldots, S_K\} \leftarrow \text{random split} \{X\} \) \( \triangleright \) Observation subsets
\( t = 0 \)

while \( \log(\mathbb{P}(X \mid \zeta, \eta)) - \log(\mathbb{P}(X \mid \zeta_{old}, \eta_{old})) > \varepsilon \) do
    for \( k \in \{1, \ldots, K\} \) do
        Condition model \( M_k \) on observations in \( S_k \)
        for \( i \in \{1, \ldots, N\} \) do
            \( v_{i,k} = \log(\mathbb{P}(x_i \mid M_k)) \) \( \triangleright \) Update value matrix \( V \)
        \( \{S_1, \ldots, S_k, \ldots, S_K\} = \emptyset \) \( \triangleright \) Reset observation subsets
    for \( i \in \{1, \ldots, N\} \) do
        \( a_i \sim \text{Boltzmann}(x_i | v_{i,1}, \ldots, v_{i,K}, \tau) \) \( \triangleright \) Sample observation assignment
        \( S_{a_i} \leftarrow x_i \) \( \triangleright \) Assign observation to subset \( a_i \)
        \( \tau \leftarrow g(\tau, t) \) \( \triangleright \) Update temperature
        \( t \leftarrow t + 1 \)

Output: Composite model parameters \( \zeta \) which in the limit segments the sequences into \( K \) bins.
```

Because of the way BEEM is constructed it means that once the component assignment has been updated, parameter inference continues as usual, depending on the
mixture base-class we are currently working with. This means that if we are considering e.g. a GMM then the parameter update is achieved through the maximisation step of the usual EM algorithm – no additional change of this step is required, and it is fully congruent on the component assignment. Or if we are using a mixture of HMMs, then we would employ the Baum-Welch algorithm [44] for the parameter update.

6 Experiments

In this section we investigate a GMMs and HMMs, applied to real and synthetic datasets. We compare inference in these models using BEEM alongside other state-of-the-art methods as well as standard approaches. For additional results and experiments for mixtures of Gaussian processes see appendix A. Reported metrics are: normalized mutual information (NMI), adjusted Rand index (ARI) [26], clustering purity (ACC), and homogeneity score [49] (Homo). The BEEM hyper parameters are set to $\tau = 1.5$ and $\alpha = 0.97$ for all experiments. The optimisation was terminated after the maximum complete data log-likelihood $\sum_{n} \max_{z_{n}} \{ p(x_{n} \mid z_{n}, \zeta_{t, z_{n}}) \}$ failed to improve for 10 EM-steps.

Unless otherwise stated, each experiment to was repeated 100 times. Error bounds are found within brackets in each results table.

Finally, we consider two different types of mixing weights for BEEM and two initialisation methods for GMMs described in table 2 and table 3 respectively.

<table>
<thead>
<tr>
<th>Mixing weight type</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed uniform</td>
<td>F</td>
<td>Mixing weights are fixed to be uniform throughout the optimisation.</td>
</tr>
<tr>
<td>Standard weights</td>
<td>S</td>
<td>Mixing weights are updated at every M-step as in the standard EM algorithm.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initialisation</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>R</td>
<td>The initial cluster means are randomly drawn from the set of observations.</td>
</tr>
<tr>
<td>$K$-means</td>
<td>K</td>
<td>The initial cluster means are computed via the $K$-means algorithm.</td>
</tr>
</tbody>
</table>

6.1 Gaussian mixture models

In this section we use GMMs with BEEM on a number of synthetic and real datasets. We include a comparison to EM with 100 re-initialisation (EM 100\(^5\)) to gauge the difficulty of each dataset. For the rainbow dataset (described overleaf), we conduct a larger comparison study, including recently published methods which includes: power $k$—means clustering (power) [59]; hierarchical density-based spatial clustering of applications with noise (HDBSCAN) [35] and affinity propagation (AP) [18]. For power $k$-means, we

\(^5\) Results reported for EM 100 is that of the solution with highest complete data log-likelihood.
explored \( s_0 \in \{-1, -3\} \) as it is not clear from the original publication what constitutes an appropriate initial value (note that \( s_0 \) can take any value in the domain \(-\infty < s_0 < 0\)). We got the best results when \( s_0 = -1 \) as the original paper suggested (see §4 and tables 1 and 2 of the original publication).

**Unbalanced square simulation.** A clustering task consisting of four two dimensional normal distribution located at the corners of a square with side length 10, centred at the origin – see fig. 6. All distributions have covariance matrix \( \Sigma = 0.3I_2 \). To create an unbalanced clustering scenario the number of samples drawn from each cluster is set to \([100, 50, 50, 10]\). As can be seen in fig. 6, this constitutes a simple clustering task for the human eye. However it is used here to demonstrate the vanishing cluster artefact that arises from including mixing weights in the BEEM algorithm. Results are shown in table 4.

![Figure 6: Unbalanced square simulation. From the top left corner in the clockwise direction the clusters contain 100, 50, 50, and 10 samples.](image)

**Rainbow simulation.** The rainbow clustering task consists of 8 two dimensional normal distributions with identity covariance matrices and distribution means along a half circle with radius \( r = 9 \). The means are homogeneously distributed by letting the angle, \( \omega = \{0, \frac{\pi}{8}, \frac{2\pi}{8}, \ldots, \pi\} \) and calculating their \( x \) and \( y \) mean location as, \( \mu_i = [\text{Re}\{r \exp(j\omega_i)\}, \text{Im}\{r \exp(j\omega_i)\}] \), where \( j \) is the complex unit. Samples are drawn with uniform probability from each distribution for a total of 1000 samples – see fig. 7. Results are shown in table 5.

**Fisher’s Iris data set.** The dataset [16, 15] contains three classes of 50 instances each, where each class refers to a type of Iris flower. Each sample is represented by an attribute vector (sepal length, sepal width, petal length, petal width). Results are shown in table 6.

The results reported in tables table 4, table 5, table 6 show that BEEM is competitive on all selected datasets. Further, under the ‘method’ header in each table we have also
Table 4: Unbalanced square simulation §6.1 results – mean and (std.). For this experiment we include a version of BEEM where the mixing weights are not fixed to be uniform, but updated at every M-step as in the conventional EM algorithm. Only random initialisation was used for the results in this table.

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC</th>
<th>Homo</th>
<th>NMI</th>
<th>ARI</th>
<th>EM steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEEM (R/F)</td>
<td>0.99 (0.02)</td>
<td>0.96 (0.05)</td>
<td>0.96 (0.07)</td>
<td>0.94 (0.11)</td>
<td>39.84 (12.48)</td>
</tr>
<tr>
<td>BEEM (R/S)</td>
<td>0.85 (0.11)</td>
<td>0.76 (0.18)</td>
<td>0.85 (0.12)</td>
<td>0.72 (0.22)</td>
<td>–</td>
</tr>
<tr>
<td>EM 100 (R)</td>
<td>1.00 0.00</td>
<td>1.00 0.00</td>
<td>1.00 0.00</td>
<td>1.00 0.00</td>
<td>20.89 3.98</td>
</tr>
<tr>
<td>EM (R)</td>
<td>0.86 (0.11)</td>
<td>0.76 (0.18)</td>
<td>0.86 (0.11)</td>
<td>0.75 (0.21)</td>
<td>12.02 (9.45)</td>
</tr>
<tr>
<td>DAEM (R)</td>
<td>0.85 (0.12)</td>
<td>0.76 (0.19)</td>
<td>0.84 0.12</td>
<td>0.73 (0.22)</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 5: Rainbow simulation results – mean and (std.).

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC</th>
<th>Homo</th>
<th>NMI</th>
<th>ARI</th>
<th>EM steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM (K)</td>
<td>0.93 (0.05)</td>
<td>0.89 (0.04)</td>
<td>0.89 (0.04)</td>
<td>0.87 (0.08)</td>
<td>3.60 (1.83)</td>
</tr>
<tr>
<td>BEEM (K/F)</td>
<td>0.96 (0.00)</td>
<td>0.91 (0.01)</td>
<td>0.91 (0.01)</td>
<td>0.91 (0.01)</td>
<td>11.00 (0.00)</td>
</tr>
<tr>
<td>EM 100 (R)</td>
<td>0.49 (0.14)</td>
<td>0.51 (0.14)</td>
<td>0.62 (0.10)</td>
<td>0.41 (0.14)</td>
<td>13.09 (16.92)</td>
</tr>
<tr>
<td>EM (R)</td>
<td>0.43 (0.05)</td>
<td>0.46 (0.06)</td>
<td>0.61 (0.05)</td>
<td>0.33 (0.07)</td>
<td>2.68 (4.85)</td>
</tr>
<tr>
<td>BEEM (R/F)</td>
<td>0.93 (0.05)</td>
<td>0.89 (0.04)</td>
<td>0.89 (0.04)</td>
<td>0.87 (0.06)</td>
<td>76.88 (17.18)</td>
</tr>
<tr>
<td>DAEM (R)</td>
<td>0.75 (0.08)</td>
<td>0.75 (0.06)</td>
<td>0.79 (0.06)</td>
<td>0.66 (0.09)</td>
<td>–</td>
</tr>
<tr>
<td>AP</td>
<td>0.93 (0.01)</td>
<td>0.91 (0.01)</td>
<td>0.78 (0.03)</td>
<td>0.59 (0.06)</td>
<td>–</td>
</tr>
<tr>
<td>HDBSCAN</td>
<td>0.70 (0.09)</td>
<td>0.66 (0.07)</td>
<td>0.67 (0.04)</td>
<td>0.46 (0.08)</td>
<td>–</td>
</tr>
<tr>
<td>power ($s_0 = -1$)</td>
<td>0.94 (0.05)</td>
<td>0.90 (0.04)</td>
<td>0.90 (0.03)</td>
<td>0.88 (0.07)</td>
<td>–</td>
</tr>
<tr>
<td>power ($s_0 = -3$)</td>
<td>0.91 (0.06)</td>
<td>0.88 (0.05)</td>
<td>0.88 (0.04)</td>
<td>0.84 (0.09)</td>
<td>–</td>
</tr>
<tr>
<td>power ($s_0 = -9$)</td>
<td>0.87 (0.07)</td>
<td>0.85 (0.05)</td>
<td>0.85 (0.05)</td>
<td>0.78 (0.09)</td>
<td>–</td>
</tr>
<tr>
<td>power ($s_0 = -18$)</td>
<td>0.85 (0.07)</td>
<td>0.83 (0.05)</td>
<td>0.84 (0.05)</td>
<td>0.75 (0.09)</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6: Iris dataset §6.1 results – mean and (std.).

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC</th>
<th>Homo</th>
<th>NMI</th>
<th>ARI</th>
<th>EM steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM (K/F)</td>
<td>0.97 (0.02)</td>
<td>0.90 (0.03)</td>
<td>0.90 (0.03)</td>
<td>0.91 (0.04)</td>
<td>16.86 (0.98)</td>
</tr>
<tr>
<td>BEEM (K/F)</td>
<td>0.97 (0.03)</td>
<td>0.90 (0.01)</td>
<td>0.90 (0.02)</td>
<td>0.90 (0.04)</td>
<td>27.67 (5.53)</td>
</tr>
<tr>
<td>EM 100 (R)</td>
<td>0.81 (0.13)</td>
<td>0.72 (0.14)</td>
<td>0.76 (0.10)</td>
<td>0.68 (0.16)</td>
<td>29.68 (8.21)</td>
</tr>
<tr>
<td>EM (R)</td>
<td>0.76 (0.05)</td>
<td>0.61 (0.06)</td>
<td>0.62 (0.06)</td>
<td>0.55 (0.06)</td>
<td>22.31 (6.05)</td>
</tr>
<tr>
<td>BEEM (R/F)</td>
<td>0.87 (0.07)</td>
<td>0.72 (0.10)</td>
<td>0.73 (0.10)</td>
<td>0.69 (0.11)</td>
<td>52.09 (0.09)</td>
</tr>
<tr>
<td>DAEM (R)</td>
<td>0.78 (0.02)</td>
<td>0.61 (0.01)</td>
<td>0.62 (0.01)</td>
<td>0.55 (0.01)</td>
<td>–</td>
</tr>
</tbody>
</table>
included different initialisation methods for each conventional EM algorithm. In addition
DAEM proved robust in table 4 but less so in the other two experiments. Comparing
the number of EM steps\(^6\), BEEM maintains competitive efficiency to EM, especially
considering that the BEEM M-step only requires each base model to be updated w.r.t.
a subset of the dataset while for EM each base model is updated on the complete dataset.
Our contribution outperforms EM 100, both in terms of EM steps and cluster metrics,
indicating that the proposed exploration mechanism is sound. Finally we see that BEEM
with mixing weights performs worse than the suggested uniform mixing weights on the
unbalanced square simulations. This is due to vanishing cluster artefact that arises
when the probability of assignment to large clusters is amplified by the mixing weights.
Initially this artefact was suspected to arise more frequently for unbalanced clustering
tasks, however results from the balanced square simulation described below indicates
that it is a general problem when mixing weights are employed.

Balanced square We continue with a simpler clustering task still, wherein each cluster
is of the same size. This clustering task consists of four two dimensional Gaussian
distribution located at the corners of a square with side length 10, centred at the origin
– see fig. 6. All distributions have covariance matrix \( \Sigma = 0.3I_2 \). Results are shown in
table 7.

Table 7: Balanced square results \([K = 4, x_n \in \mathbb{R}^2, N = 200]\) – mean and (std.).

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC (std.)</th>
<th>Homo (std.)</th>
<th>NMI (std.)</th>
<th>ARI (std.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEEM (R/F)</td>
<td>0.91 (0.12)</td>
<td>0.90 (0.13)</td>
<td>0.93 (0.10)</td>
<td>0.87 (0.18)</td>
</tr>
<tr>
<td>BEEM (R/S)</td>
<td>0.75 (0.09)</td>
<td>0.74 (0.09)</td>
<td>0.85 (0.07)</td>
<td>0.71 (0.09)</td>
</tr>
</tbody>
</table>

\(^6\)The EM steps for DAEM were omitted given that the result varies greatly with the choice of hyper
parameters.
In this simple example we see that employing a Bayesian treatment of the cluster assignments, is not necessarily a recipe for success. Recall that the posterior for each base distribution will be informed by the entire history of sampled observations and so, with each iteration, becomes increasingly difficult to modulate. This means that a bad initialisation will penalise the whole learning process. Hence, even in this trivial synthetic example, does not suggest that employing a prior is, in fact, beneficial. Indeed, consider the visualised learning processes in fig. 8.

![Figure 8: Visualisation of number of samples $|S_k|$ allocated to each base distribution during learning on the balanced square dataset. For completeness, BEEM seeks to allocate an equal number of samples to each of the four clusters.](image)

In this instantiation of the learning process (repeated 100 times to yield the results found in table 7) we observe that the learning process has indeed been penalised by a poor initialisation, which it cannot readily escape from. This phenomena is seen too for BEEM with mixing weights in fig. 8b where the algorithm fails to recover when the number of samples in cluster 0 (blue) and 1 (orange) become to small. On the other hand for the main version of BEEM in fig. 8a we see that cluster 3 (red) recovers despite almost vanishing at the eleventh iteration. Note that these figures are handpicked to highlight the difference in performance found in the statistics of table 7 and table 4.

6.2 Mixtures of hidden Markov models

Graduating from GMM parameter estimation we consider the more challenging mixtures of hidden Markov models. As revealed by the name, the base distribution of a MHMM (i.e. a single HMM) is in itself a latent variable model and as such, can be considered as a mixture model. A MHMM is simply a special case of an ordinary HMM in which the transition matrix is restricted in order to partition the state space, e.g if we consider a mixture of $K$ HMMs with transition matrices $A_k$, then the resulting MHMM transition matrix is a block diagonal matrix with elements, $\{A_1, \ldots, A_K\}$.

A common approach for HMM parameter initialisation is to set uniform transition probabilities and set the means and covariances using the K-means algorithm. Though sensible for a single HMM, it does not generalise to the mixture case. Consider for example the task of segregating sequences drawn from HMMs with identical emission distributions but distinct dynamics i.e. transition matrices. To extend $K$-means initialisation procedure, Smyth [52] suggests that the inter-sequence similarity can be estimated by fitting a HMM to each individual sequence in the data set, and constructing a $N \times N$ similarity matrix by computing the log-likelihood of each of the $N$ sequences w.r.t. the individual models. $K$-means clustering then proceeds on the similarity matrix such that $K$ sets of parameters can be initialised from the resulting segmentation. In this section
we compare BEEM parameter estimation to the Baum-Welch algorithm with random and Smyth [52] initialisation.

Random HMM simulation We simulate a clustering task by at each iteration initialising three distinct HMMs \((K = 3)\) with random transition matrices and initial distributions, each with four hidden states. The state distributions are fixed and equal for all clusters with state means \(\mu = [-2, -1, 0, 1]\) and standard deviation \(\sigma = 0.1\). At every iteration, 20 new sequences are drawn, with random sequence length \(L \sim U(a, b)\), from each HMM. The experiment is run for two different sequence length settings, \((a, b) = (5, 10)\) and \((a, b) = (20, 50)\).

Character trajectories This data set [15] contains three dimensional \((x, y, \text{pressure})\) pen tip trajectories for handwritten characters. We utilise a subset of the characters to create two clustering tasks. Separation of characters ‘A’ and ‘B’ \((K = 2)\), and separation of characters ‘A’ to ‘E’ \((K = 5)\). The average sequence length is 117. For each character the following instance-count holds: \{'A': 171, 'B': 141, 'C': 142, 'D': 157, 'E': 186\}.

Table 8: Character trajectories data set \(\S 6.2\) – best result for each algorithm after three initialisations.

<table>
<thead>
<tr>
<th>(K)</th>
<th>Method</th>
<th>ACC</th>
<th>Homo</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (&quot;\text{A'}-\text{B}')</td>
<td>EM (random)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>EM (Smyth [52])</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>BEEM</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>5 (&quot;\text{A'}-\text{E}')</td>
<td>EM (random)</td>
<td>0.96</td>
<td>0.90</td>
<td>0.91</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>EM (Smyth [52])</td>
<td>0.96</td>
<td>0.90</td>
<td>0.91</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>BEEM</td>
<td>0.98</td>
<td>0.95</td>
<td>0.95</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 9: Random HMM simulation \(\S 6.2\) results – mean and (std.).

<table>
<thead>
<tr>
<th>(L)</th>
<th>Method</th>
<th>ACC (std.)</th>
<th>Homo (std.)</th>
<th>NMI (std.)</th>
<th>ARI (std.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 – 10</td>
<td>EM (random)</td>
<td>0.51 (0.07)</td>
<td>0.13 (0.07)</td>
<td>0.13 (0.08)</td>
<td>0.09 (0.07)</td>
</tr>
<tr>
<td></td>
<td>EM (Smyth [52])</td>
<td>0.47 (0.07)</td>
<td>0.09 (0.07)</td>
<td>0.11 (0.08)</td>
<td>0.05 (0.07)</td>
</tr>
<tr>
<td></td>
<td>BEEM</td>
<td>0.49 (0.06)</td>
<td>0.09 (0.06)</td>
<td>0.10 (0.7)</td>
<td>0.07 (0.07)</td>
</tr>
<tr>
<td>20 – 50</td>
<td>EM (random)</td>
<td>0.82 (0.15)</td>
<td>0.64 (0.22)</td>
<td>0.67 (0.20)</td>
<td>0.63 (0.24)</td>
</tr>
<tr>
<td></td>
<td>EM (Smyth [52])</td>
<td>0.71 (0.13)</td>
<td>0.51 (0.18)</td>
<td>0.56 (0.18)</td>
<td>0.49 (0.20)</td>
</tr>
<tr>
<td></td>
<td>BEEM</td>
<td>0.87 (0.11)</td>
<td>0.68 (0.19)</td>
<td>0.69 (0.19)</td>
<td>0.68 (0.21)</td>
</tr>
</tbody>
</table>

The results in table 8 and table 9 demonstrate the utility of using BEEM with a mixture of HMMs. In table 8 we see that conventional methods as well as Smyth [52]’s sequence clustering algorithm, perform well on that dataset when the number of clusters is small. However, when the number of clusters increases their performance decreases, as too does BEEM, but not as much. Similarly in table 9 BEEM outperforms both methods, when it comes clustering sequences of random length 20 – 50. When the sequences are of length 5 – 10, there is no distinguishable difference between the methods, which is to be expected given that this is a very difficult clustering task. Concluding our experiments on mixtures of unconstrained HMMs we will now proceed with our investigation on the constrained DP-HMM for the application of disease progression modelling in the following section.
Finally we arrive at the intended application for our proposed algorithms. Our main objective here is to compare DP-HMM to the baseline unconstrained MHMM in the setting of disease progression sub-type discovery. We also investigate the performance of BEEM in relation to baseline EM learning as well as the task specific slope initialisation proposed in §3.3.2. Due to the lack of consensus regarding disease progression sub-types for Parkinson’s disease the quantitative assessment is performed on a synthetic dataset §7.1. We do however include results from an analysis performed on the PPMI dataset in §7.2 to demonstrate how our methodology can be used in a real world setting.

7.1 Synthetic observations

To simulate disease progression, we apply our method to sequences taken from two sigmoid functions – chosen to represent two non-ergodic transition dynamics seen in many diseases. The form of the sigmoid is given as

\[ f(t) = \frac{1}{1 + e^{a(b-t)}}. \]  

Observations were simulated according to the parameter settings found in table 10.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Slow progression</th>
<th>Fast progression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>(b)</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>(L)</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>(N)</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(a) Synthetic disease progression trajectories (b) Sample observation sequences drawn from generated from a sigmoid function given by synthetic clinical trial – i.e. eq. (37). The goal eq. (37). The blue line simulates a slower disease progression with higher disease severity at baseline. The red line simulates a faster progression with a lower severity at baseline.

Figure 9: Visualisation of the simulated clustering task. Note that the clearly separable generative processes in fig. 9a emits observation sequences depicted in fig. 9b, who’s origins are much harder to distinguish with the naked eye.

Samples were generated for each ‘synthetic subject’ (sigmoid – see fig. 9b for a reference) by first simulating the entry point into the hypothetical clinical study. In real
clinical trials, patients naturally do not join the instant they contract a disease. More often than not, several years can pass (see the results in the next section). Consequently, to faithfully replicate observations found in real clinical trials, we sample an entry point from a uniform distribution $U(0, 0.5)$. In fig. 10a, it can be seen that this on the $x$-axis represents a potential entry point at the very start of the disease, to a point which is half-way through the latent disease progression. This is evidenced by the red and blue boxes shown in fig. 10a which represent the synthetic clinical window for each subject. From thereon, observations are obtained by evaluating eq. (37) at 11 equidistant (where $\Delta t = 0.1$) points, to which white noise is added of magnitude $\mathcal{N}(0, 0.1)$. This noise magnitude is commensurate with similar noise levels seen in e.g. the PPMI dataset [34]. The standard deviation of the white noise is 10% of the severity range, found on the $y$-axis in fig. 10a. As patients progress in their disease, under this framework, the severity of their condition only increases.

![Figure 10](image.png)

**Figure 10:** Depiction of a synthetic latent disease progression and windows to represent clinical studies. Figure 10a shows a common black sigmoid representing a common disease progression, the coloured windows depict patients entering a study, with that part of their personal progression. Figure 10b shows the observations as seen from the point of view of the dataset: same time-axis but in actuality their progression is different.

On the model side the experimental setup is as follows, we set $K = 2$ and both the MHMM and the DP-HMM. For the MHMM parameter learning is compared over baseline EM, EM with Smyth [52] initialization and BEEM. In turn, for the DP-HMM we contrast is the task specific slope initialisation and to the more general BEEM algorithm. Results from this exercise can be found in table 11. Optimal parameter searches were conducted for all methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Homo</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHMM (EM)</td>
<td>0.71 (0.05)</td>
<td>0.016 (0.08)</td>
<td>0.17 (0.08)</td>
<td>0.18 (0.07)</td>
</tr>
<tr>
<td>MHMM (Smyth)</td>
<td>0.66 (0.06)</td>
<td>0.09 (0.05)</td>
<td>0.10 (0.06)</td>
<td>0.11 (0.07)</td>
</tr>
<tr>
<td>MHMM (BEEM)</td>
<td>0.87 (0.08)</td>
<td>0.50 (0.18)</td>
<td>0.50 (0.18)</td>
<td>0.57 (0.19)</td>
</tr>
<tr>
<td>DP-HMM (EM)</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
</tr>
<tr>
<td>DP-HMM (SI)</td>
<td>0.80 (0.18)</td>
<td>0.44 (0.31)</td>
<td>0.45 (0.31)</td>
<td>0.47 (0.36)</td>
</tr>
<tr>
<td>DP-HMM (BEEM)</td>
<td>0.74 (0.17)</td>
<td>0.27 (0.31)</td>
<td>0.27 (0.31)</td>
<td>0.30 (0.35)</td>
</tr>
</tbody>
</table>

Consider the results in table 11, first note that the MHMM by Smyth [52] performed worse on than baseline EM. To some degree this is to be expected. In his own toy example
[52, §2.2] he used $K = 2$ and $m = 2$ with a sequence length of 200, to demonstrate that model – which operates under a fully connected transition matrix. In our settings, there is not enough information for the MHMM to yield a useful result.

Second, MHMM with BEEM parameter learning shows the improved performance over baseline EM learning which is in agreement with the results presented in §6.2. Further it performs better than DP-HMM independent of learning algorithm. This may at first glance seem as a negative result for the DP-HMM, however it is not surprising given that the unconstrained MHMM has degrees of freedom. Reminding ourselves that ground truth won’t be available in a real world exploratory analysis, the DP-HMM can still be a more adequate option given that the resulting segmentation is highly interpretable, – see fig. 11.

Finally, parameter learning via SI outperforms BEEM for the DP-HMM. Given that SI is a task specific initialisation this is also an expected result. BEEM however shows strong performance despite the learning procedure being agnostic to how potential disease progression sub-types may differ.

![Figure 11: Visualisation of inferred clusters characteristics on the synthetic disease progression modeling task. The states of each HMM mixture component are depicted as dots. The dots are positioned at the state emission distribution mean on the vertical axis. The spacing between the states on the horizontal axis is the expected number of steps between any two consecutive steps in the Markov chain. The shaded areas show ±$\sigma$ standard deviation, around the emission distribution’s mean.](image)

7.2 Parkinson’s progression markers initiative experiments

Similar to §7.1 we apply the DP-HMM to two features of the PPMI dataset, in order to ascertain if we can evidence of clusters, in both domains. First we employ the DAT putamen feature, discussed in footnote 2 and a histogram of that sequence length, for all PD patients is shown in fig. 4. Clearly, we encounter much the same analysis regimen as in §7.1. Results for this experiment are shown in fig. 12a.

Using the parametrised DP-HMM the a clear time-series clustering can be found.
(a) Application of the DP-HMM for clustering the univariate feature DAT putamen from the PPMI dataset. Parameters set were $K = 2$, $m = [5, 7]$.

(b) Application of the DP-HMM for clustering the univariate feature MDS-UPDRSIII from the PPMI dataset. Parameters set were $K = 2$, $m = [10, 5]$.

**Figure 12:** Visualisation of inferred clusters characteristics on the PPMI dataset. The states of each HMM mixture component are depicted as dots. The dots are positioned at the state emission distribution mean on the vertical axis. The spacing between the states on the horizontal axis is the expected number of steps between any two consecutive steps in the Markov chain. The shaded areas show $\pm \sigma$ standard deviation, around the emission distribution's mean.

Moreover, we can evaluate this clustering further by considering the distribution of males and females in each cluster, as shown in table 12.

**Table 12:** Cluster gender distribution for the application of the DP-HMM to DAT putamen.

<table>
<thead>
<tr>
<th>Subset</th>
<th>Male</th>
<th>Female</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>All subjects</td>
<td>232</td>
<td>104</td>
<td>2.2</td>
</tr>
<tr>
<td>Cluster 0</td>
<td>125</td>
<td>51</td>
<td>2.4</td>
</tr>
<tr>
<td>Cluster 1</td>
<td>107</td>
<td>53</td>
<td>2.0</td>
</tr>
</tbody>
</table>

From table 12 we can see that there is no clear distinction in the gender distribution when using DAT putamen as a cluster feature. Moreover, from visualisation shown in fig. 12a we can see that the difference between clusters, appears to be more skewed towards the difference at baseline, rather than different progression dynamics. Given that the purpose of DP-HMM is to identify different dynamics, this can be seen as a failed application of the model. We now apply a similar analysis to the MDS-UPDRSIII feature. Samples from the PPMI dataset, and this feature, are shown in fig. 13.

Using samples such as those shown in fig. 13, the DP-HMM is applied. Results are shown in fig. 12b. From fig. 12b we can also see a clear clustering behaviour in the feature. Demonstrating that there are two progression modes; one fast and one slow. We again now consider the distribution of gender in each cluster, in table 13.

**Table 13:** Cluster gender distribution for the application of the DP-HMM to MDS-UPDRSIII.

<table>
<thead>
<tr>
<th>Subset</th>
<th>Male</th>
<th>Female</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>All subjects</td>
<td>248</td>
<td>109</td>
<td>2.3</td>
</tr>
<tr>
<td>Cluster 0</td>
<td>70</td>
<td>42</td>
<td>1.7</td>
</tr>
<tr>
<td>Cluster 1</td>
<td>178</td>
<td>66</td>
<td>2.8</td>
</tr>
</tbody>
</table>
we see a much more distinguished distribution of males and females. Specifically, males are much more prone to fall into the fast-progressing PD category. Related findings have been shown by Haaxma et al. [24], therein they conclude that “taken together, [their] findings suggest a more benign phenotype in women with PD”. The trends in fig. 12b are suggestive of this too (though, we stress; this is not evidence). This may also suggest that the MDS-UPDRS\textsubscript{III} feature is more informative with respect to this delineation in gender.

We have demonstrated the DP-HMM on the complex PPMI dataset, wherein sequences are of variable length, non-ergodic and noisy. We have shown that the DP-HMM is successful at deconstructing, estimating and identifying some underlying latent temporal processes, which show evidence of different disease progressions in PD, for men and women. This gender imbalance has previously been touched upon by Haaxma et al. [24] and heterogeneous disease progression in PD has been extensively studied by, amongst others, Marek et al. [34], wherein the authors also indentify different PD subtypes.

8 Conclusion and Discussion

The objective of this thesis was to explore a new approach to disease progression sub-type discovery via mixtures of hidden Markov models. First we take a step back, to tackle issues that arise from finding optimal parameters in the non-convex landscape of the likelihood function. For this we have presented the Boltzmann exploration expectation-maximisation (BEEM) algorithm for maximum likelihood estimation. BEEM overcomes many of the problem associated with the conventional EM algorithm [13], as well as more modern alternatives such as DAEM and EM with $K$-means initialisation. While an elegant theoretical motivation for BEEM has been omitted from this paper, we have empirically shown that it manages to effectively search the parameter-space while
avoiding local maxima. In addition to strong performance the algorithm is very simple to implement for finite mixtures of any base distribution, given that their parameters are updated independently at each iteration. This means, as demonstrated, that model-specific parameter inference can be used, off-the-shelf, whilst the component assignment can be effectively done using BEEM, without any major change in the overall model fitting procedure.

In this work we control the exploration level via exponentially decreasing temperature. The algorithm is insensitive to hyper-parameter settings given that the same parameter set was used successfully for all experiments. However, it is possible that the used parameter set caused excessive exploration leading to an unnecessarily large number of EM-steps.

After validating the performance of BEEM for standard mixture models, we propose a novel mixture of constrained hidden Markov models, which uses a new type of cluster initialisation to more accurately initialise the mixture. Because the mixture is very sensitive to initial conditions, the DP-HMM is able to handle more complex data when aided by custom initialisation procedures. Further, because our interest lies in clinical trials data, where sequences are short, and very heterogeneous, the DP-HMM was designed to learn with short sequences. In addition, because many neurodegenerative disease are non-ergodic, we imposed this paradigm on the model, and ensured that the initialisation scheme, could handle this type of dynamics. We also extended initial MHMM model by allowing a variable number of states for each cluster.

Finally, we applied the DP-HMM to synthetic data sampled from a sigmoid function, chosen to replicate progression curves for neurodegenerative diseases. We find that the task specific SI procedure out performs BEEM for the simulated application. BEEM however still outperforms standard Baum-Welch learning with random initialisation and offers an agnostic approach in contrast to SI. We also applied the DP-HMM with SI to the PPMI dataset to find PD subtypes. In both regards, the DP-HMM identified and built a generative and interpretable model, of the underlying latent temporal process. Further, there is also a need for experimental validation on chronic diseases which have known and validated progression subtypes, such that the effectiveness of the DP-HMM can be properly validated and explored further. Finally, we have already noted that our model does not at present learn $K$ and $m$ from the observations, but are set by the user. There are a multitude of ways to incorporate this functionality, and at present we are looking towards going down the Bayesian nonparametric route, to achieve this.

References


[57] Thompson, William R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. 1933.


A Additional results for mixture of Gaussian processes experiment

In this appendix we include two experiments where BEEM is applied to mixtures of Gaussian processes for data-association tasks.

A.1 A simple data-association task

In this section we investigate a simple data association (DA) problem. Data association seeks to map a source-model to each observation in the dataset, see fig. 14, in which the true number of clusters is \( K = 2 \) and \( N = 250 \). A prominent model which deals in DA, is the overlapping mixtures of Gaussian processes (OMGP) model by Lázaro-Gredilla et al. [30].

Lázaro-Gredilla et al. [30] use a “non-standard variational Bayesian algorithm to efficiently recover sample labels and learn the hyperparameters”. To this we compare a simple mixture of Gaussian processes (MGP) using BEEM inference. As with the other mixtures we use a off-the-shelf inference for learning the likelihood model, which in the case of the GP means we minimise negative log marginal-likelihood of each base-model w.r.t. the hyperparameters and noise level. We use the same radial basis function kernel [46] for each model, with noise variance set to 0.01 and \( K = 2 \) for both models. Each experiment, for each model, was repeated 50 times. The OMGP’s marginalised variational bound was minimised over 150 iterations for each experiment and the MGPs with BEEM, used 15 Boltzmann updates and ten GP updates each, per component, at each exploration. The cluster purity results are shown in fig. 15.

The results in fig. 15 demonstrate that a standard mixture of GPs using BEEM inference, is capable of achieving similar purity results as the more complex OMGP model, when applied to the toy dataset in fig. 14.

(a) Measured cluster purity for MGP w. BEEM.  
(b) Measured cluster purity for the OMGP.

Figure 15: Measured purity (ACC) for each method (higher is better). The thick line in each plot shows the mean trend \( \mu \pm 2\sigma \). Note that the horizontal axis is not the same for both methods.
Figure 16: Receiver-operating curves (ROC) for the mixtures of Gaussian process models. These ROC curves result from the experiments conducted in appendix A.1. The mixture of Gaussian processes with BEEM inference is shown in fig. 16a and the OMGP model’s ROC curve is depicted in fig. 16b. As noted in appendix A.1 each model was applied to the same dataset (see fig. 14) 50 times, consequently the above mean ROC curve is shown plus/minus two standard deviations.

The dataset for the simple data-association experiment is shown in fig. 14, wherein the true number of clusters is $K = 2$ and in total there are $N = 250$ observations, each of which needs to be associated with a source generative process. From those experiments the receiver-operating curves are plotted in fig. 16.

From fig. 16 a number of inferences can be drawn. First, the area-under-the ROC curve (AUROC) has been calculated and included with standard deviations over all experimental results. The AUROC of the data-association models corresponds to the probability that the model (MGPs w. BEEM or OMGP) will rank a randomly chosen positive example higher than a randomly chosen negative example. From this definition we see that the OMGP is, on average, better than the MGP w. BEEM at associating observations with the correct source. At the same time, we see from fig. 16a, that taking the uncertainty bounds into account, yields comparable performance alongside OMGP.

Now of course, the toy-dataset from whence the AUROCs were generated is simplistic, and there are many parameters to tune (see the next section). Certainly, it is possible to push the AUROC envelope further, for both models, by performing a hyper-parameter search using e.g. Bayesian optimisation, though that is outside the scope of this paper. This example demonstrates that MGP w. BEEM is capable of producing comparable performance to a much more complex model.
A.2 More complex data-association experiment

In this section we conduct a more complex data-association experiment. It is inspired by the [30, §4.1.1] but sacrifices dimensionality for a more complex noise regime, as well as allowing for an irregularly sampled observation space. Unlike the experiment conducted in appendix A.1 we provide a walk-through of the toy-data generation process in fig. 17, this is to provide insight into what we want the MGP w. BEEM and the OMGP, to reproduce, as well as understand what layers of complexity need to be overcome, in order to approximate the original sources in fig. 17a. The original sources are chosen as a positive and a negative sinusoidal curve, each over one full cycle, and both have multiple regions of interaction (i.e. where they overlap).

As before, our implementation for the MGP used GPy [22], where each model was furnished with the model parameters found in table 14. Both models are privy to the true number of clusters and were provided with the same periodic kernel – for details on this kernel see [46]. The output variance $\sigma^2$ determines the average distance of the function away from its mean, we note this because we performed two experiments with the OMGP where this parameter was varied (see table 14). In total the OMGP’s variational inference procedure was run for 150 iterations, and the GP base-models in BEEM were updated ten times each, for every Boltzmann update. We again measure cluster quality based on purity which is the percent of the total number of observations that were correctly classified. Results are shown in fig. 18.

Table 14: Parameters used for both data-association models applied to the dataset described in fig. 17.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MGP w. BEEM</th>
<th>OMGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>0.1</td>
<td>0.1 &amp; 0.01</td>
</tr>
<tr>
<td>Kernel</td>
<td>$\sigma^2 \exp \left(-\frac{2\sin^2(\pi</td>
<td>x-x'</td>
</tr>
<tr>
<td>Base model (GP) iterations</td>
<td>10</td>
<td>150</td>
</tr>
<tr>
<td>Boltzmann updates</td>
<td>15</td>
<td>–</td>
</tr>
<tr>
<td>$\tau$</td>
<td>1.1</td>
<td>–</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.97</td>
<td>–</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>–</td>
</tr>
</tbody>
</table>

The results for the MGP w. BEEM are shown in fig. 18a. Contrast this purity trend with those in fig. 18b and fig. 18c. The result in fig. 18a displays an exploration-exploitation behaviour on part BEEM where the method initially has a large spread of cluster assignments, but as the temperature is reduced, it is also clear that a good point has been found on the likelihood surface. The OMGP on the other hand, has much lower purity spread initially, taken across all experimental runs, but increases the variance as the variational bounds are optimised [30].
(a) Two functions representing the ‘true’ underlying data generation processes.

(b) To create a more realistic dataset, data points are removed at random, resulting in sequences of $N = 75$ and $N = 60$.

(c) Two different additive white-noise processes are included: $N(0, \sigma^2 = 0.3)$ and $N(0, \sigma^2 = 0.2)$.

(d) The final dataset as passed to the respective mixture models.

Figure 17: Synthetic data-generation process. The sub-figures within demonstrate the data-generation process, from top-left to bottom-right. This example is more complex than the one used in the body of this paper and has multiple overlapping areas. Colour-coded items in the sub-captions indicate a relation with the red and blue curves in the data-generating functions in fig. 17a. Moreover, the observation sets have been designed to have different noise-models fig. 17c and irregularly sampled fig. 17b. The normalised dataset, as seen by the data-association models is depicted in fig. 17d.
Figure 18: Measured purity for each method. Purity is an external evaluation criterion of cluster quality, the percent of the total number of observations that were classified correctly, in the unit range $[0, 1]$. Each method was run 10 times, the thick line in each plot shows the mean trend $\mu \pm 2\sigma$. Note that the horizontal axis is not the same for both methods.