Title
Mapping Highly Nonconvex Energy Landscapes in Clustering, Grammatical and Curriculum Learning

Author
Pavlovskaia, Maria

Publication Date
2014

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Mapping Highly Nonconvex Energy Landscapes in Grammar and Curriculum Learning

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Maria Pavlovskaia

2017

ABSTRACT OF THE DISSERTATION

Mapping Highly Nonconvex Energy Landscapes in Grammar and Curriculum Learning

by

Maria Pavlovskaia
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2017
Professor Song-Chun Zhu, Chair

We introduce *Energy Landscape Maps* (ELMs) as a new and powerful analysis tool of non-convex problems to the machine learning community. An ELM characterizes and visualizes an energy function with a tree structure, in which each leaf node represents a local minimum and each non-leaf node represents the barrier between adjacent energy basins. We construct ELMs using an advanced MCMC sampling method that dynamically reweights the energy function to facilitate efficient traversal of the hypothesis space. By providing an intuitive visualization of energy functions, ELMs could help researchers gain new insight into the non-convex problems and facilitate the design and analysis of non-convex optimization algorithms. We demonstrate this on two classic machine learning problems: clustering with Gaussian mixture models and biclustering.

The dissertation of Maria Pavlovskaia is approved.

Luminita Vesa

Qing Zhou

Yingnian Wu

Song-Chun Zhu, Committee Chair

University of California, Los Angeles

2017

To my mom and dad . . .

who—among so many other things—

always encouraged me to play around with math problems

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I'd like to acknowledge S.C Zhu, Kewei Tu, Adrian Barbu, and my committee members for all their help, guidance, and support

VITA

2009	B.S. (Mathematics), Harvey Mudd College.
2011	M.A. (Applied Mathematics), UCLA, Los Angeles, California.
2010–2011	Research Assistant, Mathematics Department, UCLA.
2011-2014	Research Assistant, Statistics Department, UCLA.

PUBLICATIONS

Unsupervised Structure Learning of Stochastic And-Or Grammars Advances in Neural Information Processing Systems 26, T. Kewei, M. Pavlovskaia, S.C. Zhu

Rates for Inductive Learning of Compositional Models AAAI Workshop Replearn 2013, A. Barbu, M. Pavlovskaia, S.C. Zhu.

CHAPTER 1

Introduction

In a typical machine learning problem, the goal is to identify a model in a hypothesis space that optimizes a pre-specified energy function. Many machine learning problems involve non-convex optimizations, such as the learning of multilayer neural networks, mixture models and probabilistic grammars. Typically, local search techniques such as gradient descent (e.g., backpropagation for training neural networks) and coordinate descent (e.g., expectation-maximization for learning mixture models) are employed to find a local minimum in the energy landscape. Analysis of the properties of such non-convex energy landscape is much less studied.

We introduce *Energy Landscape Maps* (ELMs) as a new and powerful analysis tool of non-convex problems to the machine learning community. ELMs are also known as *disconnectivity graphs* in physics [BK97, WMW98]). An ELM characterizes and visualizes the energy function with a tree structure, in which each leaf node represents a local minimum and each non-leaf node represents the barrier between adjacent energy basins. Figure 2.1 shows an example ELM and the energy landscape it represents. We adopt and extend the algorithms proposed in the literature [Zho11, AL10] to efficiently construct ELMs using an advanced MCMC sampling strategy. Specifically, we employ a MCMC sampler to traverse the hypothesis space; unlike traditional MCMC sampling, we divide the hypothesis space into subregions and reweight the energy function at each subregion in a way that the energy of a subregion frequently visited by the sampler would be increased. The reweighting scheme helps the sampler to frequently transit across energy barriers and efficiently traverse the hypothesis space. Based on the resulting Markov chain, we can discover local minima of the energy function as well as estimate the energy barriers between them, from which the

ELM can be constructed.

ELMs provide an intuitive visualization of energy functions, which could help researchers gain new insight into the non-convex problems and facilitate the design and analysis of non-convex optimization algorithms. We demonstrate the utility of ELMs in analyzing two classic machine learning problems: clustering with Gaussian mixture models and biclustering. We illustrate how the shape and complexity of ELM is influenced by (a) the input data generated from different ground-truth models, and (b) the hyperparameters of the energy function. We also visualize the behavior of a variety of learning algorithms using ELMs.

Many machine learning problems involve non-convex optimizations. A representative example that we study is unsupervised learning of dependency grammars. A dependency grammar models the dependency relations between the words of a sentence and is widely used in natural language syntactic parsing [Mel88, Col99, KMN09]. In unsupervised learning, one tries to recover a dependency grammar from a set of unannotated sentences that are generated from the grammar. The learning objective is typically the posterior probability of the dependency grammar. Since the parse of each training sentence is latent and the number of possible assignments of a parse is huge, the energy function of unsupervised dependency grammar learning is highly non-convex. Typically, local search techniques such as the expectation-maximization algorithm are employed to find a local optimum grammar. Significant progress has been made over the past ten years in designing new algorithms and regularization techniques to improve the learning accuracy [KM04, HJM09]. On the other hand, analysis of the properties of the highly non-convex energy landscape is much less studied, which would nonetheless facilitate the analysis and design of dependency grammar learning algorithms.

We demonstrate the utility of ELMs in analyzing unsupervised dependency grammar learning, in particular the curriculum learning technique which has been successfully applied in learning dependency grammars [BLC09, SAJ10]. Curriculum learning divides the learning process into multiple stages and each stage involves learning a more complex grammar than the previous stage which culminates in the target grammar in the final stage. By plotting the ELM of each curriculum stage, we show that the effectiveness of a curriculum is a result of

starting with a highly smoothed energy function and then gradually reducing the smoothness over the curriculum stages, as hypothesized by Bengio et al. [BLC09].

The rest of this thesis is organized as follows. In Chapter 2, we discuss background material on Energy Landscape Maps, And-Or graphs, and the Wang-Landau algorithm. We also present a general-purpose distance metric in the hypothesis space which is used in our ELM construction. Chapter 3 discusses the ELM construction algorithm for high-dimensional continuous spaces and various tests for convergence. In section 3.3, we present our experimental results on the Gaussian mixture model fitting problem and the biclustering problem respectively. Chapter 4 introduces the dependency grammar and presents our experimental results. Chapter 5 presents some theoretical results on PAC learning of And-Or Graph spaces. Chapter 6 concludes our work and present avenues for future research.

CHAPTER 2

Background

2.1 ELM

In [BK97], Becker proposed visualizing multi-dimensional energy surfaces with a "disconnectivity graph", which we refer to here as an ELM. The graphs are constructed from the local minima of an energy landscape; the local minima and displayed as leaves and their energies are displayed as the y-axis position of the leaves. The graphs are divided into distinct branches which connect at the minimum energy threshold of any path between the local minima as shown in Figure 2.1.

The ELM allows us to visualize the topological characteristics of high-dimensional energy spaces. It partitions the the energy space into "energy wells", the basins of attraction of the local minima as shown in Figure 2.11. The branches of the ELM reflect the connectivity of the energy wells, which characterize the shape of the high-dimensional energy function.

Figure 2.2 shows several examples of energy function topologies and the resulting ELMs. These allow us to view important characteristics of the space, including 1. the number of low-energy local minima, the energy differences between the local minima, the energy barriers between the global minima and nearby local minima. These characteristics determine the efficacy of learning algorithms in the space; for instance, if there are multiple low-energy local minima with high energy barriers separating their energy basins, most MCMC-style algorithms will become "stuck" in the local minima resulting in slow convergence speeds.

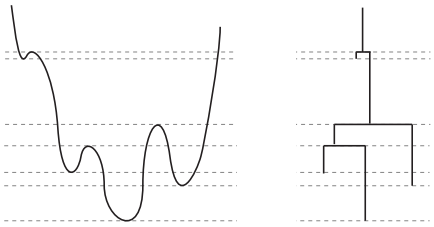


Figure 2.1: An energy function and the corresponding ELM. The y-axis of the ELM is the energy level, each leaf node is a local minimum and the leaf nodes are connected at the ridges of their energy basins.

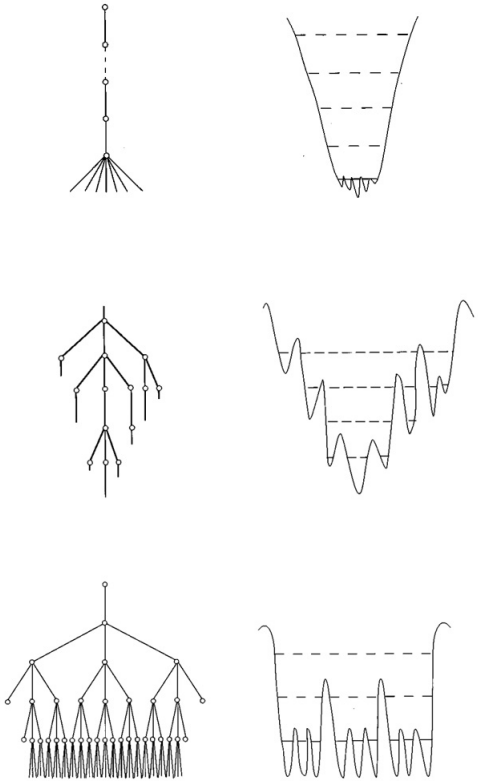


Figure 2.2: Sample landscape topologies and corresponding ELMs presented in increasing order of difficulty for learning the global optimum [BK97]. (top) Single minimum with weak noise (middle) funnel energy function with one global minimum (bottom) multiple global minima with large energy barriers.

2.2 AOG

A stochastic AND-OR graph (AOG) consists of a hierarchical structure with alternating layers of And-nodes and Or-nodes. An AND node represents a probabilistic composition of its child nodes. An OR node represents a stochastic choice among its child nodes and a leaf node represents an observable random variable. AOGs are an extension of constituency grammars in natural language parsing [MS99] and have been employed in computer vision to model objects [ZM06] and events [PJZ11].

The AOG is an alternative visual representation of grammar production rules. In the context of image grammars, It is used to represent the hierarchical decomposition for scene level, to object level, to part level and finally to pixel level. Figure **??** shows the AOG model for the hierarchical decomposition of a clock from the object level to the template level.

An AND node represents a decomposition of an entity into its parts. This corresponds to the grammar rule $A \rightarrow A_1 A_2 \dots A_n$. For example, a clock face can be decomposed into a frame, 12 numbers and two clock hands so it can be represented as an AND node as seen in Figure 2.3.

An OR node represents a decomposition into alternative possible sub-structures. This corresponds to the grammar rule $A \rightarrow A_1 | A_2 | \dots | A_n$. For example, this can include the views of a single object from different angles (such as the front and side views of a human face) or alternative parts that can make up an object (such as different types of clock faces as seen in Figure 2.3).

A terminal node is an instance of a base-level dictionary element. In the context of image grammars, this can for example be an image patch, a part template, or a primitive. The terminal nodes can occur at any level of the graph hierarchy and correspond to the production rule $A \rightarrow a$.

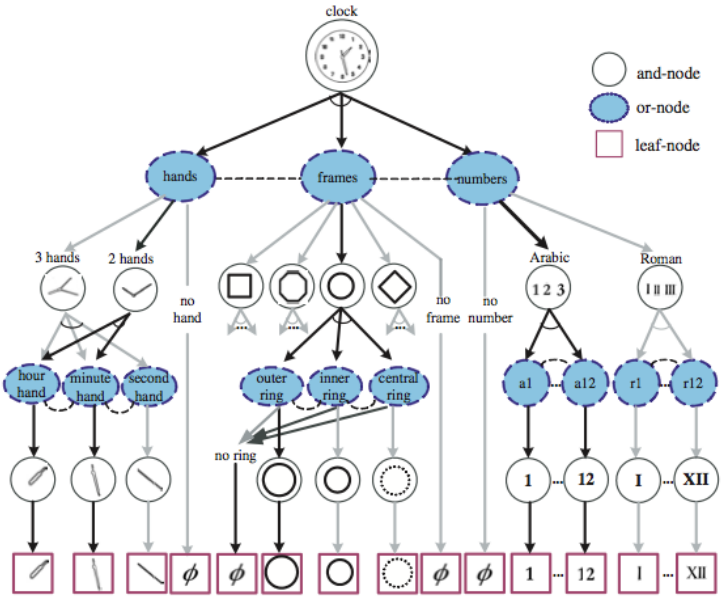


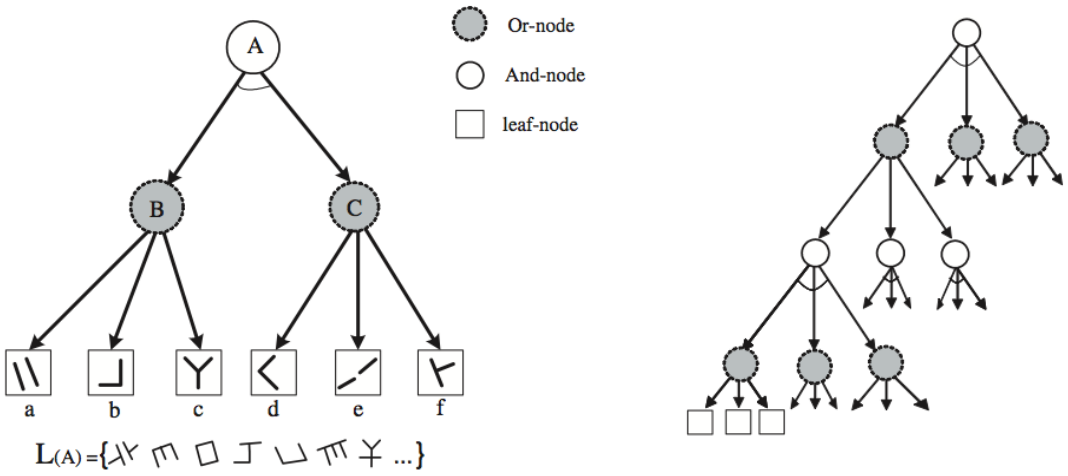
Figure 2.3: AOG model for the hierarchical decomposition of a clock from the object level to the template level.

2.2.1 Special cases of AOGs

1. A Gaussian mixture model can be seen as a special case of AOG. The root node of the AOG is an OR node that makes choice among the mixture components. Each child of the OR node is a AND node representing a Gaussian distribution. The child nodes of each AND node are leaf nodes representing the mean vector and covariance matrix.

2. A coherent bicluster [MO04] can be seen as a special case of AOG. The root node of the AOG is an AND node with two child nodes, representing the composition of the two dimensions of the bicluster. Each of the two child nodes is an OR node, whose child nodes represent the indexes of the rows or columns of the bicluster.

3. A dependency grammar (DG) is a recursive AOG with an infinite depth. It is a context-free grammar that requires its grammar rules to take the form of $ROOT \rightarrow A$, $A \rightarrow AB$, $A \rightarrow BA$ or $A \rightarrow a$, where $ROOT$ is the start symbol, A and B are nonterminals, and a is a terminal.



(a) (b)

Figure 2.4: (a) An AOG with root node A. Node A is an AND-node containing two child nodes B and C. The nodes B and C are OR-nodes each containing three children, which are terminal nodes. The terminal nodes $\{a, b, c, d, e, f\}$ are elements of the dictionary $L(A)$ for the AOG. (b) Recursive AOG structure with 2 layers of AND nodes and 2 layers of OR nodes. With a branching number of 3, this AOG contains 10 And-nodes, 30 Or-nodes, and 81 leaf nodes.

2.3 Distance Measure in the Hypothesis Space

The hypothesis space $\mathcal{H}$ is the set of all possible models that may be learned. We want to minimize an energy function that is defined on the hypothesis space: $E : \mathcal{H} \to \mathbb{R}$. In order to define local minima of the energy function, we must define a distance metric on the hypothesis space $\mathcal{H}$.

In this section, we define a general-purpose weak distance metric that we used in our experiments. Our metric is general-purpose in the sense that it is applicable to a large variety of probabilistic models, including the two types of models studied in this paper. An important feature of our metric is that it measures the distance between two models in terms of their *strong generative capacity*, i.e., the difference of the models in assigning values of both observable and latent variables. This is very useful for analyzing machine learning problems in which the distribution over latent variables is important (e.g., unsupervised learning of probabilistic grammars in which the parse tree is the latent variable). In comparison, many existing metrics either only consider the distribution of observable variables (e.g., the total variation distance), or directly compare the model structures and parameters and therefore distinguish models that are practically equivalent (e.g., the L_1 distance of model parameters). Our metric is also efficient to compute without the need to traverse all possible assignments of the observable and latent variables.

2.3.1 AND-OR Graph Space

We introduce the stochastic AND-OR graph (AOG) [ZM06] as an overarching model with a large variety of probabilistic models as special cases. The distance metric that is defined on the AOG space is then applicable to the hypothesis spaces of all the subclasses of AOG.

2.3.2 Distance Measure of AOGs

We recursively define the distance metric D between two N-level AOGs by defining the metric on each type of node. It can be shown that D is a weak distance metric in the sense

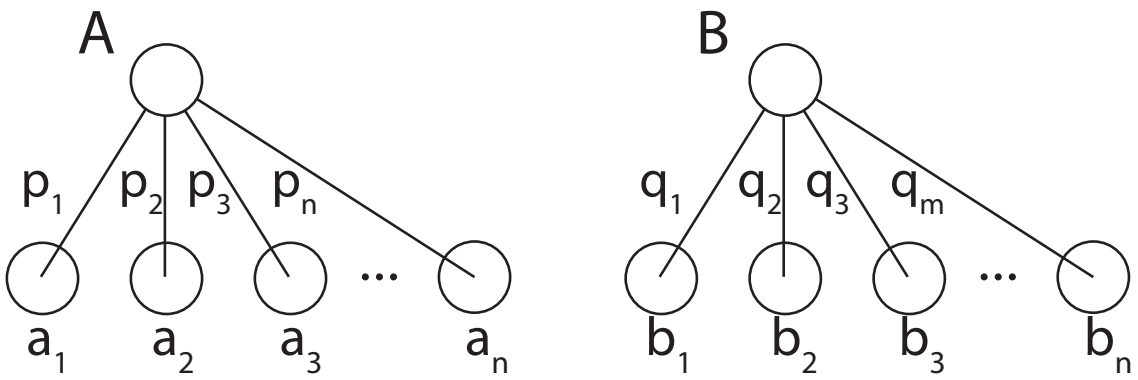


Figure 2.5: Two OR nodes A and B with children $\{a\}_I$ and $\{b\}_J$ and branch probabilities $\{p\}_I$ and $\{q\}_J$ where $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$.

that it satisfies non-negativity, positive-definiteness, and symmetry but does not satisfy the triangle inequality.

Metric for OR Nodes: The distance between two OR nodes is a function of the distances between pairs of their closest children.

Let A and B be OR nodes with children $\{a\}_I$ and $\{b\}_J$ and branch probabilities $\{p\}_I$ and $\{q\}_J$ where $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$ with $n \leq m$ as show in Figure 2.5. We iteratively construct K to be an injective mapping between the index sets I and J. Let

$$(i, j) = \arg\min_{(k,l)} \{D(a_k, b_l)\}, \tag{2.1}$$

that is a_i and b_j are the closest pair of child nodes. Then we define $K(i) = j$. Next we remove i and j from their index sets, find the next closest pair of child nodes and obtain $K(i') = j'$. We repeat this until the first index set is empty (the mapping will not be surjective if $n < m$). The distance between the parent nodes A and B is

$$D(A, B) := 1 - \sum_{i=1}^{n} \min(p_i, q_{K(i)}) \left(1 - D(a_i, b_{K(i)})\right).$$

Metric for AND Nodes: The distance between two AND nodes is a function of the distances between their children. Let A and B be AND nodes with children $\{a\}_{i=1}^{n}$ and $\{b\}_{j=1}^{m}$ as show in Figure 2.6.

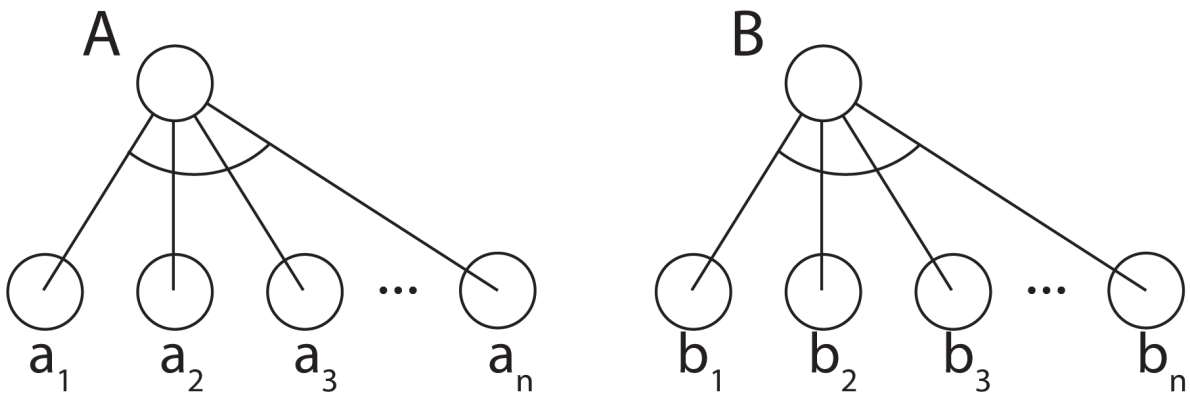


Figure 2.6: Two AND nodes A and B with children $\{a\}_{i=1}^{n}$ and $\{b\}_{i=1}^{m}$.

The distance between the parent nodes A and B is

$$D(A,B) := 1 - \prod_{i,j} \max(\epsilon, 1 - D(a_i, b_j))$$

for some fixed $0 < \epsilon < 1$.

Metric for Terminal Nodes: Any positive-definite, symmetrical distance measure D that satisfies $D(A,B) \in [0,1]\ \forall A, B$ can be used to define the distance between terminal nodes. In particular, if the terminal nodes are in a metric space with a metric d^*, then we may use the measure $D(A,B) = \frac{2}{\pi}\tan^{-1}(d^*(A,B))$.

2.3.3 Example Distance Calculation

Let G and G' be two AND-OR graph fragments with start nodes S and S' as shown in figure 2.7. We can assume that the hypothesis space $\mathcal{H}$ consists of two-level AND-OR graphs with terminal nodes A, B, C, D, E, F.

Step 1: We calculate the distance between the terminal nodes. Since in this case we are in a discrete space, $D(A,A) = D(B,B) = \cdots = D(F,F) = 1$ while $D(A,B) = D(A,C) = \cdots = D(E,F) = 0$.

Step 2: We calculate the distance between the parent nodes of the terminal nodes.

$$\begin{aligned}
D(N, N') &= 1 - 0 = 1 \\
D(N, M') &= 1 - (\min(0.3, 0.4)D(A, A) \\
&\quad + \min(0.2, 0.3)D(B, B) + \min(0.5, 0.3)D(C, C)) \\
&= 1 - (0.3 + 0.2 + 0.3) = 0.2 \\
D(M, M') &= 1 - 0 = 1 \\
D(M, N') &= 1 - (\min(0.3, 0.7)D(D, D) + \\
&\quad + \min(0.2, 0.3)D(E, E)) \\
&= 1 - (0.3 + 0.2) = 0.5
\end{aligned}$$

Step 3: Since (N, M') have the minimum distance between them, we match $N \rightarrow M'$. The next smallest distance is between M and N', so we match $M \rightarrow N'$.

Step 4: We calculate the distance between the start nodes S and S'.

$$\begin{aligned}
D(S, S') &= 1 - (\max(\epsilon, 1 - D(N, M'))) \, (\max(\epsilon, 1 - D(M, N'))) \\
&= 1 - (\max(\epsilon, 1 - 0.2)) \, (\max(\epsilon, 1 - 0.5)) \\
&= 1 - (\max(\epsilon, 0.8)) \, (\max(\epsilon, 0.5)) \\
&= 0.6
\end{aligned}$$

assuming that we chose a sufficiently small ϵ such that $\epsilon < 0.5$. Hence the distance between the two AND-OR graph fragments is 0.6.

Comparison with the symmetric KL divergence We calculate the model distance using KL divergence for comparison. First we generate all possible samples from each model and compute the frequency histograms of each sample as shown in Figure 2.8. Then we compute the symmetric KL divergence between S and S' using the equation

$$D(S, S') = \frac{\sum \left(p_i \log \frac{p_i}{q_i} \right) + \sum \left(q_i \log \frac{q_i}{p_i} \right)}{2}$$

where p_i is the probability of sample i from the first AoG fragment and q_i is the probability of sample i from the second AoG fragment. Note that since certain samples (FA, FB, FB) have probability 0 in the second AoG fragment, the equation evaluates to ∞. Consequently, we use a modified version

$$D(S, S') = \frac{\sum \left(p'_i \log \frac{p'_i}{q'_i} \right) + \sum \left(q'_i \log \frac{q'_i}{p'_i} \right)}{2}$$

where $p'_i = \max(p_i, \epsilon)$ and $q'_i = \max(q_i, \epsilon)$. For $\epsilon = 10^{-10}$, we obtain $D(S, S') = 5.58$. However, there are two issues calculating the model distance using the symmetric KL divergence approach.

- The calculated distance strongly depends on the chosen value of ϵ and approaches ∞ as $\epsilon \to 0$. For example, $D(S, S') = 0.93$ for $\epsilon = 0.01$ and $D(S, S') = 17.09$ for $\epsilon = 10^{-30}$.
- The distance does not reflect the structure of the model.

2.4 Wang-Landou

The objective of the Wang-Landau algorithm is to construct a Markov Chain that will efficiently traverse a large state space Ω by frequently visiting each energy basin. This is done by partitioning the state space into bins corresponding to energy basins and energy levels, then constructing a chain that can visit every bin with equal probability, ie. it will not become trapped at deep energy basins.

Let Ω be partitioned into disjoint subspaces $\Omega = \cup_{i=1}^{K} D_i$ corresponding to the energy wells of the target function as shown in Figure 2.9.

The intuition for this algorithm is the following: we divide the hypothesis space Ω into bins $\mathcal{B}_i$ as shown in Figure 2.11 (b) and attempt to draw samples from a target distribution $\pi(x)$ in such a way that there is equal number of samples in each bin. Let $\phi(x) = i$ for $i \in \mathcal{B}_i$ be the mapping between the hypothesis space and the bin indices and let $\beta(i)$ be the probability mass of the i-th bin, $\beta(i) = \int_{x \in \Omega_i} \pi(x)\, dx$. We define a new probability function $\pi'(x) = \pi(x)/\beta(\phi(x))$ so that MC samples from $\pi'(x)$ have equal density among all the bins.

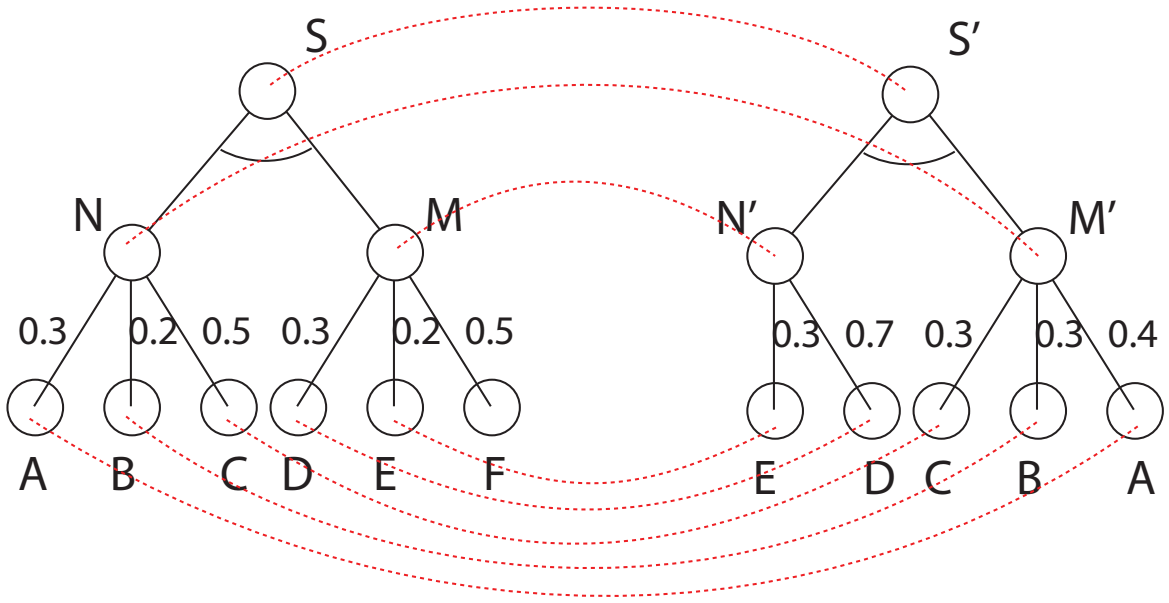


Figure 2.7: Shows the minimum-distance matching nodes in two AoG fragments. See section 2.3.3 for an example calculation of the distance between these fragments. In this example, $D(N, M') = 0.2$, $D(M, N') = 0.5$ and so the AoG distance is $D(S, S') = 0.6$

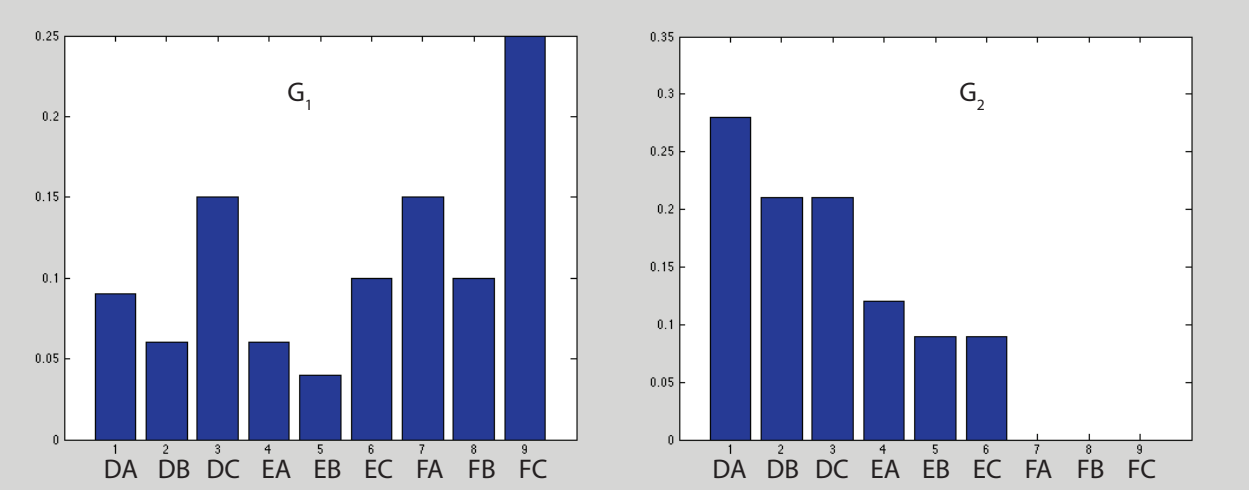


Figure 2.8: Histograms of the possible outputs sampled from the graphs G_1 and G_2 with start nodes S and S' shown in Figure 2.7. See section 2.3.3 for an example calculation of the distance between these fragments using the symmetric KL divergence as the distance measure. In this example, $\mathrm{KL}(S, S') = 5.5792$ for $\epsilon = 10^{-10}$.

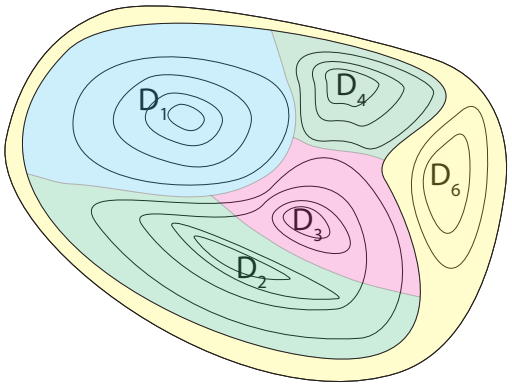


Figure 2.9: State space Ω partitioned into disjoint subspaces $\Omega = \cup_{i=1}^{K} D_i$ corresponding to the energy wells of the target function.

We want to sample from $\pi'(x)$, but need to iteratively refine our estimate of the unknown probability mass function $\beta(i)$ by stochastic gradient method [LLC07]. We use γ_i^t as the probability mass estimate of bin i by incrementing $\log \gamma_i^t$ by 1 for every new sample in D_i at iteration t. As $t \to \infty$, the weights γ_i^t converge to $\beta(i)$, and the samples become equally distributed among the bins.

We construct the Energy Landscape Map (ELM) using a modified version of the algorithm proposed by Zhou [Zho11]. Zhou's original algorithm is designed for discrete hypothesis spaces. The algorithm performs a random walk over the basins of attraction of the energy function $E : \mathcal{H} \to \mathbb{R}$. The random walk is implemented similarly to the generalized Wang-Landau (GWL) algorithm [LLC07, AL10] and has the advantage of rapidly traversing across multiple local minima in high-dimensional spaces.

The Hypothesis space is partitioned into energy basins and the energy is partitioned into unit intervals, which defines a natural partitioning of $\mathcal{H} \times \mathbb{R}$ into bins. The random walk is a MCMC chain of samples $(x_t, E(x_t))$ that is constructed in such a way that it has equal probability of visiting each bin in $\mathcal{H} \times \mathbb{R}$. The algorithm goes as follows:

1. Initialize a sample $x_0 \in \mathcal{H}$ and the bin weights γ_i^0 for the bins $\mathcal{B}_i \in \mathcal{H} \times \mathbb{R}$. Repeat step 2-6:

2. At step t, sample $y \sim Q(x_t, y)$ from some proposal distribution Q.

3. Perform steepest descent initialized with y to find the energy basin that y belongs to. Let $\phi(y)$ be the index of the bin containing $(y, E(y))$.

4. Accept proposal y with probability $\alpha(x_t, y)$ where

$$\alpha(x_t, y) = \min\left(1, e^{-b(E(x_t)-E(y))} \frac{\gamma_{\phi(x_t)}^t}{\gamma_{\phi(y)}^t}\right). \tag{2.2}$$

5. If the proposal is accepted, increase the weight $\gamma_{\phi(y)}^{t+1} = \gamma_{\phi(y)}^t * f$ for some constant f.

6. If x_t and y belong to different basins, then perform ridge descent and update the estimated upper-bound of the energy barrier between the two basins.

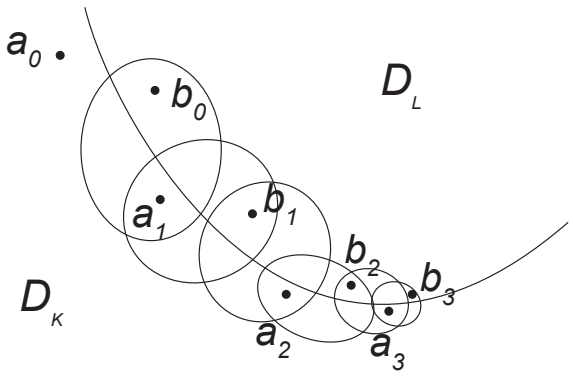


Figure 2.10: The ridge descent algorithm is used for estimating the energy boundary between basins D_k and D_l initialized at consecutive MCMC samples $a_0 = x_t, b_0 = x_{t+1}$ where $a_0 \in D_k$ and $b_0 \in D_l$.

Figure 2.12 (a) illustrates the Markov chain produced by the algorithm.

From the algorithm, we collect N samples $x_1, \ldots, x_N$. Next we find the energy barriers by ridge descent: We collect all consecutive pairs that move across two basins $\Omega_{kl} = \{(x_t, x_{t+1}) \text{ st. } x_t \in D_k, x_{t+1} \in D_l\}$. For each basin pair k, l we choose $(a_0, b_0) \in \Omega_{kl}$ with the lowest energy $(a_0, b_0) = \text{argmin}_{(a,b)\in\Omega_{kl}} [\min(E(a), E(b))]$. Next we iterate to find (a_t, b_t):

$$a_t = \text{argmin}_a \{E(a) : a \in \text{Neighborhood}(b_{t-1}) \cap D_k\}$$
$$b_t = \text{argmin}_b \{E(b) : b \in \text{Neighborhood}(a_{t-1}) \cap D_l\}$$

until $b_{t-1} = b_t$, as shown in Figure 2.10.

After enough samples are collected, we can construct the ELM based on the energy of the basins that have been discovered and the estimated energy barriers between them. The ELM tree is constructed recursively from the MCMC samples $x_0, \ldots, x_t$ in the following way: 1. the tree is initialized as a root node x_0 2. For $i = 1, \ldots, k$, if x_i is not contained in the tree, compute the barrier $B(x_i, x_{i-1})$, create a node at height $B(x_i, x_{i-1})$ on a parent branch of x_{i-1}, and add the leaf x_i as show in Figure 2.13 (a). Otherwise if x_i is contained in the tree, merge the parent branches of x_i and x_{i-1} at height $B(x_i, x_{i-1})$ as shown in 2.13 (b).

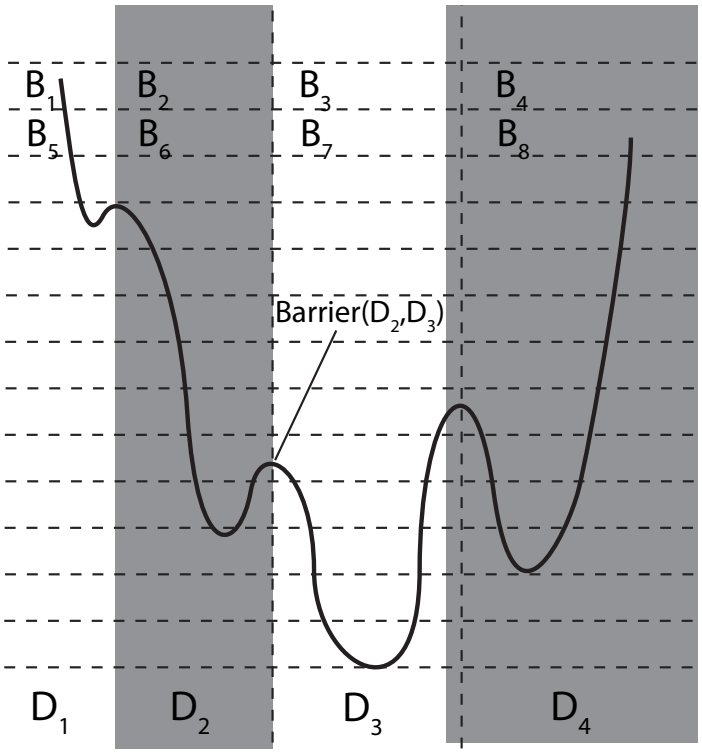


Figure 2.11: Bins $\mathcal{B}_i \in \mathcal{H} \times \mathbb{R}$. The energy space $\mathbb{R}$ (shown as the y-axis) is partitioned into uniform intervals. The hypothesis space $\mathcal{H}$ is partitioned into energy basins (shown as the partitions along the x-axis).

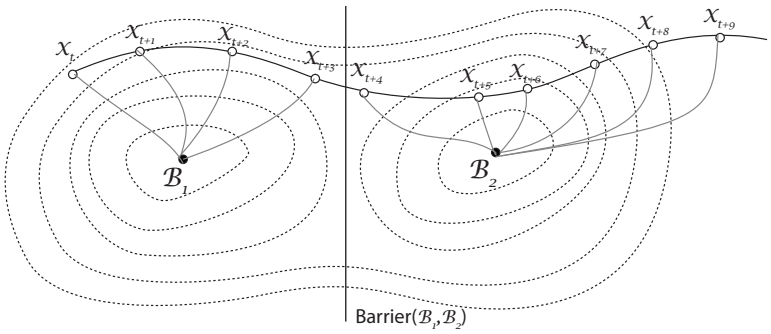


Figure 2.12: Sequential MCMC samples $x_t, x_{t+1}, \ldots, x_{t+9}$. For each sample, we perform gradient descent to determine which energy basin the sample belongs to. If two sequential samples fall into different basins (x_{t+3} and x_{t+4} in this example), we estimate or update the upper-bound of the energy barrier between their respective basins (B_1 and B_2 in this example).

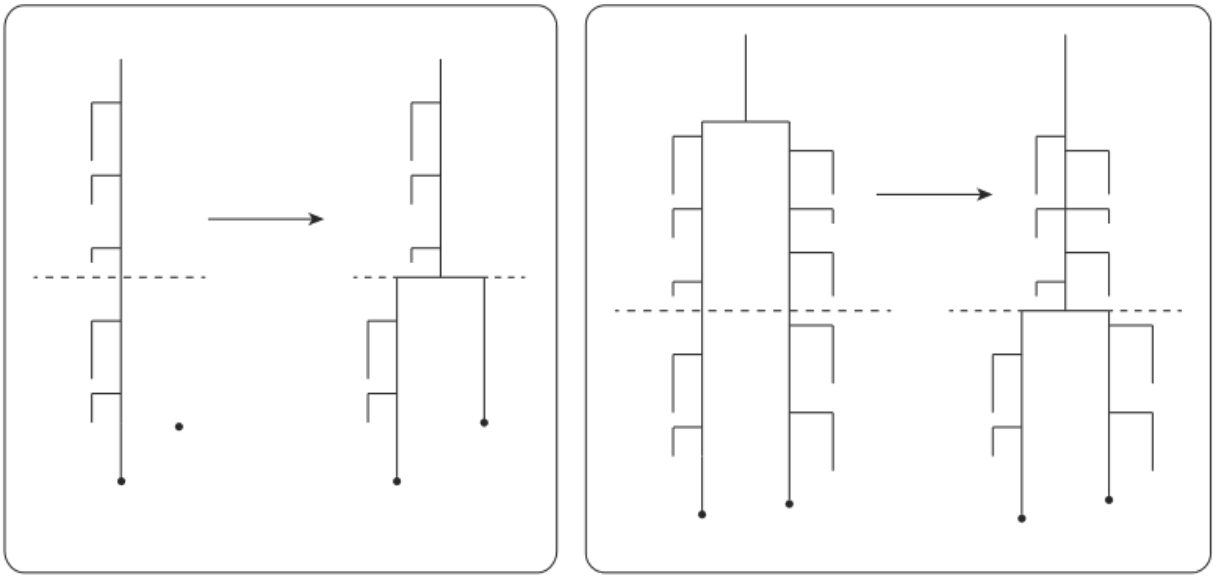


Figure 2.13: Recursively constructing the ELM from MCMC samples (a) left: adding a node (b) merging two branches.

2.5 Equal Domain Samplers

CHAPTER 3

ELMs for high-dimensional continuous spaces

3.1 ELM Construction

We construct the Energy Landscape Map (ELM) using a modified version of the algorithm proposed by Zhou [Zho11]. Zhou's original algorithm is designed for discrete hypothesis spaces. The algorithm performs a random walk over the basins of attraction of the energy function $E : \mathcal{H} \to \mathbb{R}$. The random walk is implemented similarly to the generalized Wang-Landau (GWL) algorithm [LLC07, AL10] and has the advantage of rapidly traversing across multiple local minima in high-dimensional spaces.

In addition to the local minima and the energy barriers between the energy wells, we can also estimate the probability mass and the volume of each well. One the chain has converged to a stationary distribution, we have $e^{w_i^k} \to \int_{\mathcal{B}_i} p(x)dx$ where $w_{i,j}^k$ is the number of samples in bin $\mathcal{B}_i$ at iteration k and energy level j. Hence the probability mass of $\mathcal{B}_i$ is approximated by $\sum_j e^{w_i^k}$. [Lia05a]. Similarly, given a sufficiently dense energy ladder (ie. the energy space $\mathbb{R}$ is partitioned into sufficiently small intervals $[u, u + du]$, then we have

$$w_j = \Omega(u)e^{-u}du.$$

As $du \to 0$, the volume of the energy basin k is approximated by $V_k = \sum_u \Omega_k(u)du = \sum_u w_{k,j}e^u$, as shown in Figure 3.1.

Many non-convex optimization problems are defined in a continuous space. We make the following extensions to the algorithm in the continuous case. First, we perform gradient descent or coordinate descent to identify the basin of each sample. Because the landscapes that we study are smooth only in small neighborhoods, we use some optimizations of gradient

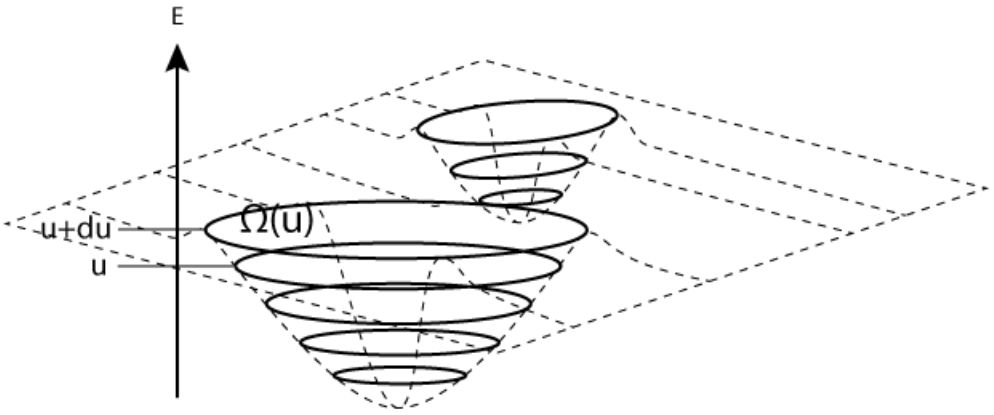


Figure 3.1: Assuming that du is sufficiently small, the volume of the left energy basin is approximated by $V = \sum_u \Omega_k(u)du = \sum_u w_{k,j}e^u$.

descent in order to better detect the local minima. In particular, we use coordinate descent, and additionally use Armijo line search to determine the step size as shown in Figure 3.2. If the hypothesis space $\mathcal{H}$ is a manifold in $\mathbb{R}^n$, we perform projected gradient descent.

Second, to avoid erroneously identifying multiple local minima within the same basin (especially when there is large flat regions), we merge local minima identified by gradient descent based on the following criteria: (1) the distance between two local minima is smaller than a constant ϵ; or (2) there is no barrier along the straight line between two local minima. Third, when estimating the energy barrier based on two consecutive samples that fall into different basins, since we cannot use ridge descent in the continuous space, we search for the local maximum along the straight line between the two samples and use its energy to update the upper-bound estimation of the energy barrier.

A significant portion of non-convex optimization problems involve latent variables. When constructing the ELM for such problems, we use data augmentation [TW87] to improve the efficiency of sampling. Specifically, in order to propose a new model x_{t+1}, we first sample the values of the latent variables Z_t based on $P(Z_t|x_t)$ and then sample the new model x_{t+1}

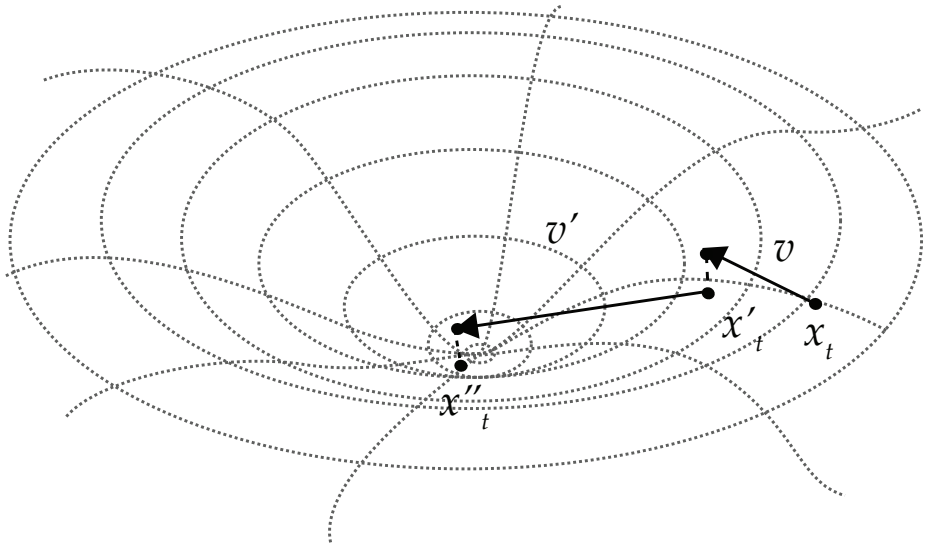


Figure 3.2: First two steps of projected gradient descent. The algorithm is initialized with MCMC sample x_t. v is the gradient of E at the point x_t. Armijo line search is used to determine the step size α along the vector v. x'_t is the projection $T(x_t + \alpha v)$ onto the subspace Γ. Then x''_t is the projection $T(x_t + \alpha' v')$, and so on.

based on $P(x_{t+1}|Z_t)$. The proposal x_{t+1} is then either accepted or rejected based on the same acceptance probability (Eq.2.2). Note that, however, our goal in ELM construction is to traverse the hypothesis space instead of sampling from the probability distribution. When enough samples are collected and therefore the weights in θ^t become large, the reweighted probability distribution would be significantly different from the original distribution and the rejection rate of the models proposed via data augmentation would become high. Therefore, we mix the proposal approach based on data augmentation with random proposal (i.e., randomly sample a new model in the neighborhood of the current model), and we increasingly rely on random proposal when the weights become large.

3.2 Tests for Convergence

3.2.1 MCMC convergence

The MCMC chain in the GWL algorithm converges to a stationary distribution over a long enough time period [Lia05b]. However, an issue with the implimentation is determining when the convergence occurs and how long of a burn-in period is necessary.

In the literature, there are several criteria (called convergence diagnistics) used to monitor the convergence of MCMC samples. In the univeriate case, the Geweke diagnostic can be used to test for convergence [Gew92] is done by splitting the samples into two batches and verifying that their means are equal using a modified z-test. Alternatively, the Gelman and Rubin criterion [GR92] uses independent parallel chains with different starting points to check for convergence; if the chains converged, they should appear similar to one another. The similarity is measured by calculating the between intra-chain and between-chain variances. This method is generalized for multivariate distributions in [**?**].

In our case, the convergence tests cannot be directly applied because the distance measure on our hypothesis space $\mathcal{H} \subset \mathbb{R}^n$ is not topologically equivalent to the Euclidean distance. Therefore, we use Multidimensional Scaling to project the hypothesis space $\mathcal{H}$ into $\mathbb{R}^d$ (where $d \leq n$) with the Euclidean metric. We then use the Brooks and Gelman's multivariate

extension of the Gelman and Rubin criterion on $m = 2$ chains initialized at random starting points to determine convergence within a small error tolerance.

3.2.2 ELM convergence

The convergence of the WL algorithm to a stationary distribution is a necessary but not sufficient condition for the convergence of the Energy Landscape Map; we need another test for the convergence of the ELM in structure.

An ELM constructed at time step k in the algorithm can be represented by:

- The set of local minima $M^k = \{m_1^k, \ldots, m_{n(k)}^k\}$ and
- The energy barrier matrix B^k where $B^k(i,j)$ is the energy barrier between m_i^k and m_j^k.

By construction of the algorithm, $M^k \subseteq M^{k+1}$ for all k and $|M^k|$ converges to the total number of local minima in the energy landscape as the MCMC converges to the stationary distribution with finitely many local minima. If all of the local minima are found at time $k = \tau$, the energy barrier matrices B^k have the same number of rows and columns for each $k \geq \tau$. Additionally, for every (i,j) pair, $B^k(i,j)$ is monotonically decreasing with time, ie. $B^k(i,j) \leq B^{k+1}(i,j)$ for all $k \geq \tau$. We assume that the energy barriers have converged if $\sum_{i,j} |B^k(i,j) - B^{k+N}(i,j)| < \epsilon * |M^k|$ for some small threshold ϵ and large number of iterations N. Consequently, we use the following algorithm to check for convergence:

1. Run $m = 2$ MCMC chains initialized with random starting values and discard the first 1,000 iterations.
2. Project the MCMC samples into $\mathbb{R}^d$ using MDS. Then check whether the chains have converged to a stationary distribution using the multivariate extension of the Gelman and Rubin criterion. If not, continue each chain for another 1,000 iterations and repeat step 2.
3. Run the chains for another N iterations. If any new local minima are found, repeat step 3.

4. If at the last time step k the sum $\sum_{i,j} |B^{k-N}(i,j) - B^k(i,j)|$ for both chains is less than ϵ, terminate. Otherwise, repeat step 3.

We can further verify convergence by comparing the two ELMs that have been generated. We define a distance measure DE between ELMs $E^{(1)}$ and $E^{(2)}$ to be

$$\begin{aligned} DE(E^{(1)}, E^{(2)}) &= \alpha_e(1/n) * \sum_i d(\theta_i^{(1)}, \theta_{f(i)}^{(2)}) \\ &+ \beta_e * (1/n^2) \sum_{i,j} |B^{(1)}(\theta_i^{(1)}, \theta_j^{(1)}) - B^{(2)}(\theta_{f(i)}^{(2)}, \theta_{f(j)}^{(2)})| \\ &+ \gamma_e * (1/n)||M^{(1)}| - |M^{(2)}||, \end{aligned}$$

where the number of local minima in $E^{(1)}$ is greater or equal to the number of local minima in $E^{(2)}$. $\theta_j^{(i)}$ is the jth local minimum, $B^{(i)}$ is the barrier matrix, and $M^{(i)}$ is the set of local minima in the ith ELM, $E^{(i)}$. The function $f(i)$ is a one-to-one mapping of the indices of local minima in the first ELM to their matches in the second ELM and is defined recursively as $f(1) = \text{argmin}_j\{d(\theta_j^{(2)}, \theta_1^{(1)}) \mid \theta^{(2)} \in M^{(2)}\}$, $f(i) = \text{argmin}_j\{d(\theta_j^{(2)}, \theta_i^{(1)}) \mid \theta_j^{(2)} \in M^{(2)} \setminus \{\theta_{f(1)}, \theta_{f(2)}, \ldots, \theta_{f(i-1)}\}\}$. The parameters $\alpha_e, \beta_e, \gamma_e$ are the weights corresponding respectively to the distances between the local minima, the differences between the energy barriers, and the differences between the number of local minima. Figures **??** (a), (b), (c) show the distance and number of local minima found for two convering MCMC chains and the resulting ELMs.

3.3 ELMs of Gaussian Mixture Models

An n-component Gaussian Mixture Model (GMM) G is a weighted mixture of n Gaussians. The energy function that we use is the negative log of the posterior, given by $E(G) = -\log P(G|z_i : i = 1 \ldots m) - \log P(G)$ for m samples $\{z_i\}$. We use a Dirichlet prior on the weights of the model and the Normal-inverse-Wishart prior on the means and variances of the model components.

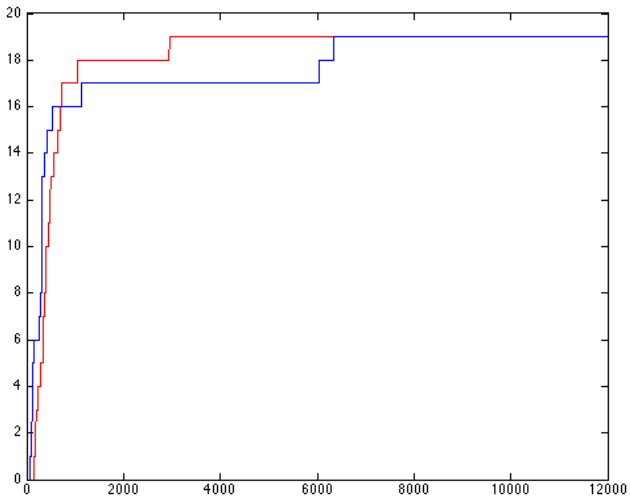


Figure 3.3: Convergence of ELMs generated from two MCMC chains C_1 and C_2 initialized at different starting points: number of local minima found vs number of iterations for C_1 and C_2

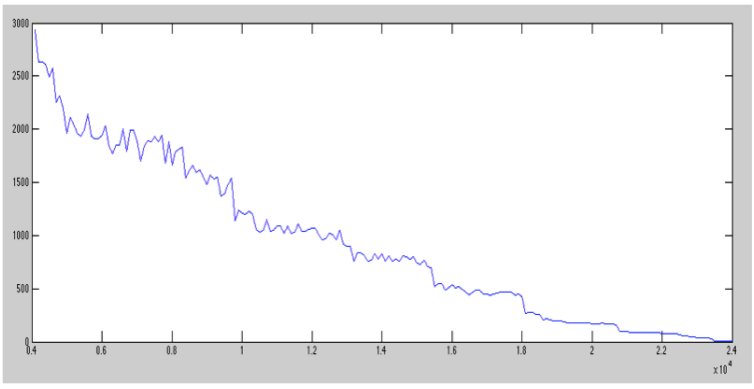


Figure 3.4: Convergence of ELMs generated from two MCMC chains C_1 and C_2 initialized at different starting points: distance $DE(C_1, C_2)$ between C_1 and C_2 vs. number of iterations.

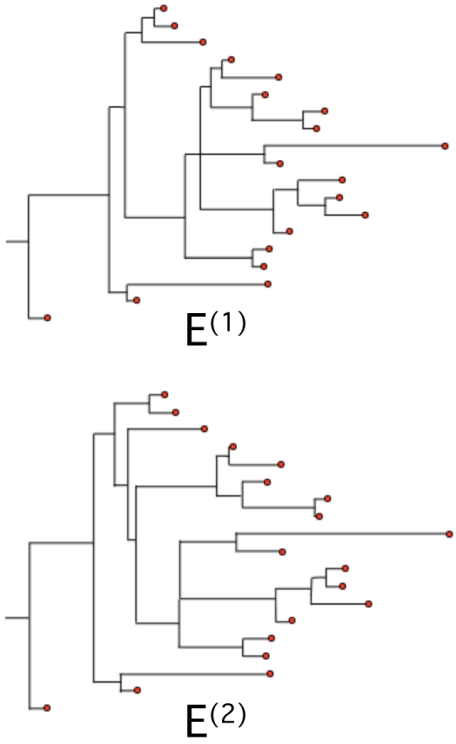


Figure 3.5: Convergence of ELMs generated from two MCMC chains C_1 and C_2 initialized at different starting points: The ELMs of C_1 and C_2 after convergence in 24,000 iterations.

3.3.1 Energy and Gradient Computations

In order to use the Armijo line search gradient descent method, we find need to find the gradient of the energy function. Suppose the mixture model has n components in d dimensions with means μ_i and covariance matrices Σ_i for $i = 1, \ldots, n$. Then energy function is $E(G)$ is

$$\begin{aligned} E(G) &= -\log P(G|z_i : i = 1 \ldots m) - \log P(G) \\ &= -\sum_{i=1}^{m} \log \left(\sum_{j=1}^{n} w_j \frac{1}{\sqrt{\det(2\pi\Sigma_j)}} \exp\left[-\frac{1}{2} (z_i - \mu_j)^T \Sigma_j^{-1} (z_i - \mu_j) \right] \right) - \log P(G). \end{aligned}$$

Since $P(G)$ is the sum of a Dirichlet prior and a NIW prior, its partial derivates are trivial to compute. It remains to compute the derivatives of $\log P(G|z_i : i = 1 \ldots m)$. Let $N_{z_i}(\Sigma_j, \mu_j) = \frac{1}{\sqrt{\det(2\pi\Sigma_j)}} \exp\left[-\frac{1}{2} (z_i - \mu_j)^T \Sigma_j^{-1} (z_i - \mu_j) \right]$ and

$$f_{\text{mm}}(z_i) = \sum_{j=1}^{n} w_j \frac{1}{\sqrt{\det(2\pi\Sigma_j)}} \exp\left[-\frac{1}{2} (z_i - \mu_j)^T \Sigma_j^{-1} (z_i - \mu_j) \right] = \sum_{j=1}^{n} w_j N_{z_i}(\Sigma_j, \mu_j).$$

Then, for one fixed sample z_i, we have the following partial derivatives.

Partial derivative with respect to each weight w_j:

$$\begin{aligned} \frac{\delta \log f_{\text{mm}}(z_i)}{\delta w_j} &= \frac{\delta \log \left[w_j N_{z_i}(\Sigma_j, \mu_j) \right]}{\delta w_j} \frac{w_j N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)} \\ &= \frac{N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)}. \end{aligned}$$

Partial derivative with respect to each mean μ_j:

$$\begin{aligned} \frac{\delta \log f_{\text{mm}}(z_i)}{\delta \mu_j} &= \frac{w_j N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)} \frac{\delta \log \left[w_j N_{z_i}(\Sigma_j, \mu_j) \right]}{\delta \mu_j} \\ &= \frac{w_j N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)} \Sigma_j^{-1} (\mu_j - z_i). \end{aligned}$$

Partial derivative with respect to each covariance Σ_j:

$$\begin{aligned} \frac{\delta \log f_{\text{mm}}(z_i)}{\delta \Sigma_j} &= \frac{w_j N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)} \frac{\delta \log \left[w_j N_{z_i}(\Sigma_j, \mu_j) \right]}{\delta \Sigma_j} \\ &= \frac{w_j N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)} \left[\frac{\delta}{\delta \Sigma_j} \log \frac{1}{\sqrt{\det(2\pi\Sigma_j)}} - \frac{\delta}{\delta \Sigma_j} \frac{1}{2} (z_i - \mu_j)^T \Sigma_j^{-1} (z_i - \mu_j) \right] \\ &= \frac{w_j N_{z_i}(\Sigma_j, \mu_j)}{\sum_{k=1}^{n} w_k N_{z_i}(\Sigma_k, \mu_k)} \frac{1}{2} \left[\frac{\delta}{\delta \Sigma_j} \log \frac{1}{\sqrt{\det(2\pi\Sigma_j)}} \Sigma_j^{-T} + \Sigma_j^{-T} (z_i - \mu_j) (z_i - \mu_j)^T \Sigma_j^{-T} \right]. \end{aligned}$$

Then we can compute the gradient of $-\log P(G|z_i : i = 1 \dots m)$ using the substitution

$$\begin{aligned} -\log P(G|z_i : i = 1 \dots m) &= -\sum_{i=1}^{m} \log \left(\sum_{j=1}^{n} w_j \frac{1}{\sqrt{\det(2\pi\Sigma_j)}} \exp\left[-\frac{1}{2} (z_i - \mu_j)^T \Sigma_j^{-1} (z_i - \mu_j) \right] \right) \\ &= -\sum_{i=1}^{m} \log f_{\text{mm}}(z_i), \end{aligned}$$

for which we have computed the partial derivatives above.

Note that we need to restrict the Σ_j matrices so that the each inverse Σ_j^{-1} exists in order to have a defined gradient. As the Σ_j are covariance matrices, they are guaranteed to be semi-positive definite, so each eigenvalue is great or equal to zero. Consequently we only need the minor restriction that for each eigenvalue λ_i of Σ_j, $\lambda_i > \epsilon$ for some $\epsilon > 0$. However, it is possible that after one gradient descent step, the new GMM parameters will be outside of the valid GMM space, ie. the new Σ_j^{t+1} matrices will not be symmetric positive definite. In order to project back into the GMM space, we project each Σ_j^{t+1} into the SPD space with the projection

$$P_{\text{symm}}(P_{\text{pos}}(\Sigma_j^{t+1})).$$

The function $P_{\text{symm}}(\Sigma)$ projects the matrix into the space of symmetric matrices by

$$P_{\text{symm}}(\Sigma) = \frac{1}{2}(\Sigma + (\Sigma)^T).$$

Assuming that Σ is symmetric, the function $P_{\text{pos}}(\Sigma)$ projects Σ into the space of symmetric matrices with eigenvalues greater than ϵ. Because Σ is symmetric, then it can be decomposed into $\Sigma = Q\Lambda Q^T$ where Λ is the diagonal eigenvalue matrix $\Lambda = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$ and Q is an orthonormal eigenvector matrix. Then

$$P_{\text{pos}}(\Sigma) = Q \begin{pmatrix} \max(\lambda_1, \epsilon) & 0 & \dots & 0 \\ 0 & \max(\lambda_2, \epsilon) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \max(\lambda_n, \epsilon) \end{pmatrix} Q^T$$

ensures that $P_{\text{pos}}(\Sigma)$ is symmetric positive definite and therefore both a valid covariance matrix and invertible, which is required for gradient computations.

3.3.2 Bounding the GMM space

Suppose that we are given m samples z_i for $i = 1, \ldots, m$ from the Gaussian mixture model G with weights $\{w_j\}$, means μ_j and covariance matrices Σ_i. Assuming that m is sufficiently large, we can estimate a boundary on the space of possible parameter for G. In the 1-dimensional case, each mean μ_j is bounded by $\max(z_1, \ldots, z_m) + \epsilon_m$ on the upper end and $\min(z_1, \ldots, z_m) - \epsilon_m$ on the lower end. By the central limit theorem, the parameter ϵ_m can be set to 0 if m is sufficiently large. Each variance σ_j^2 can similarly be bounded by $\sigma_j^2 \leq \text{Var}(\{z_i\}) = \frac{1}{m}\sum(z_i - \frac{1}{m}\sum(z_i))^2 + \epsilon_m$.

We generalize the bound to d-dimensional spaces in the following way: Let

$$c = \frac{1}{m}\sum_{i=1}^{m} z_i$$

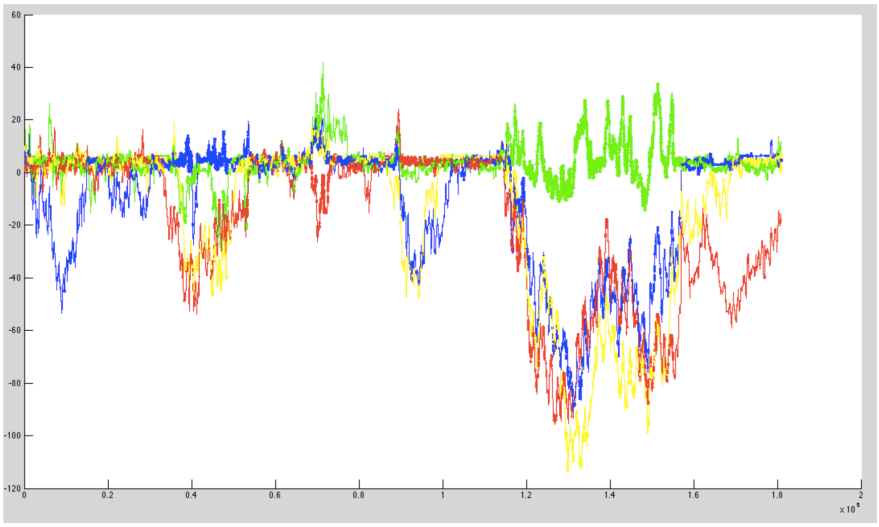
be the mean of the samples $\{z_i\}$ and let

$$r = \max_i(||z_i - c||_{L_2})$$

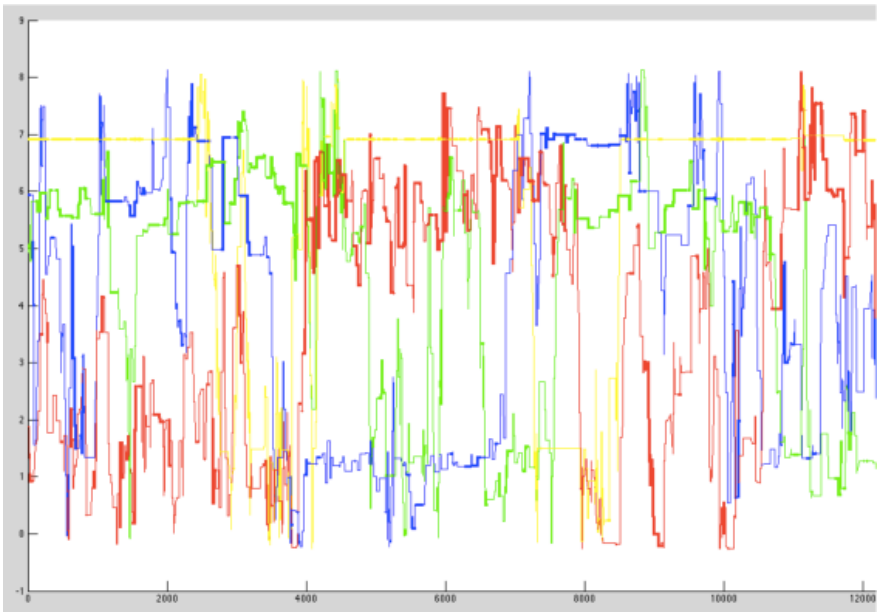
be the L2 distance between the center c and the furthest sample. Then for a sufficiently large sample size m, each mean in the GMM is bounded within a radius $r + \epsilon_m$ of the center c:

$$||\mu_j - c||_{L_2} < r + \epsilon_m.$$

To bound the covariance matrices Σ_j, let $\Sigma = \text{Cov}(\{z_i\})$ and let $\Sigma = Q\Lambda Q^T$ be the eigenvalue decomposition of Σ where $\Lambda = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{pmatrix}$ and $L = \max(\lambda_1, \ldots, \lambda_n) + \epsilon_m$. Then each eigenvalue of Σ_j is bounded by the maximum eigenvalue of Σ. That is, if Σ_j has



(a) unbounded GMM space



(b) bounded GMM space

Figure 3.6: We sampled 70 data points from a 1-dimensional, 4-component GMM and ran the MCMC random walk for ELM construction algorithm in the (a) unbounded (b) bounded GMM space. The plots show the evolution of the location of the centers of each of the 4 components over time. The width of the line represents the weight of the corresponding component.

eigen-decomposition $\Sigma = Q_j \Lambda_j Q_j^T$ and $\Lambda_j = \begin{pmatrix} \lambda_1^j & 0 & \dots & 0 \\ 0 & \lambda_2^j & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^j \end{pmatrix}$, then $\lambda_i^j \leq L$.

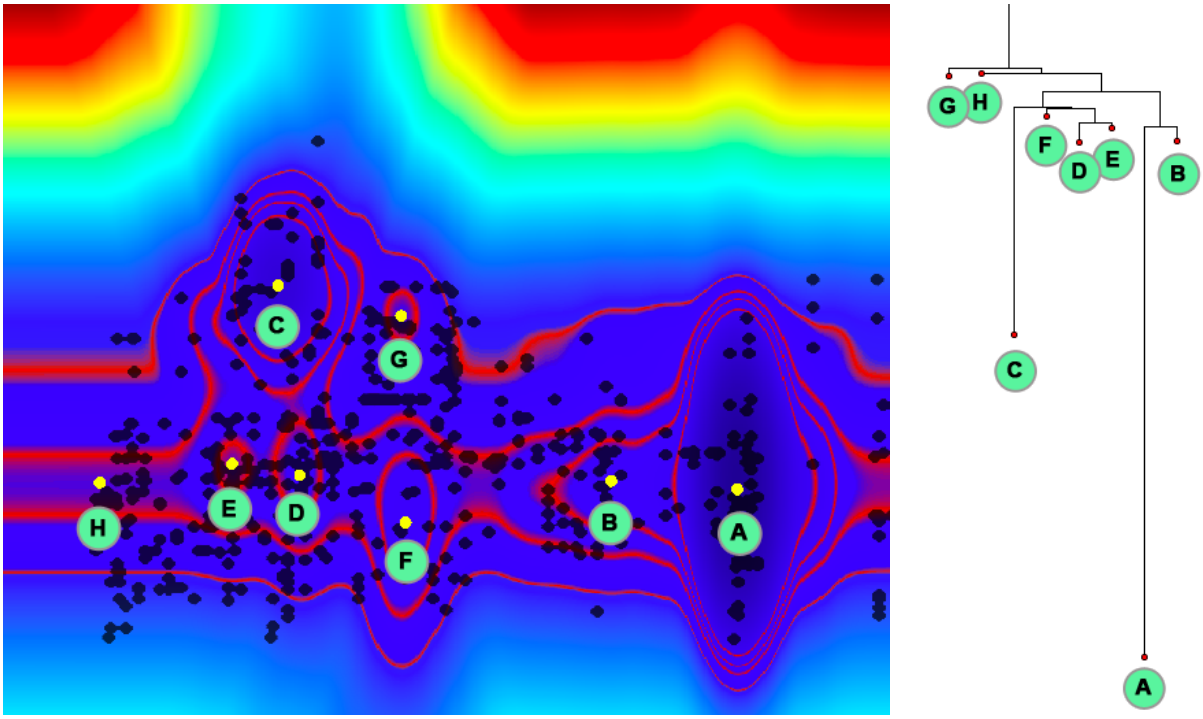
3.3.3 A 2D Example

We computed the Energy Landscape Map of a highly separable ($c = 2.4$) 1-dimensional 4-component GMM in order to show the convergence of the ELM construction algorithm to the correct solution. For this experiment, we varied two parameters: the means (μ_1 and μ_2) of the first and second components. The remaining parameters (the weights and the standard deviations, as well as the means of the 3rd and 4th components) were fixed. Figure 3.7 (a) shows the heat map of the ground truth energy landscape obtained by calculating the energy on a grid with $0 \leq \mu_1, \mu_2 \leq 5$. There are two low-energy local minima (A) and (C) that correspond to the values of the means μ_1 and μ_2 in the true model and there are 6 shallow local minima. The asymmetry in the landscape is caused by the fact that that the true model has different weights between first and second component.

We sampled 70 data points from the GMM and ran the ELM construction algorithm. Figure 3.7 (a) shows that all local minima were identified. Additionally, it shows the first 200 MCMC samples that were accepted. The samples are clustered around the local minima, and cover all energy basins. They are not present in the high energy areas away from the local minima, as would be desired. Figure 3.7 (b) shows the resulting ELM and the correspondence between the leaves and the local minima in the energy landscape. Figures 3.8 (a) and (b) show the mass and the volume of the energy basins.

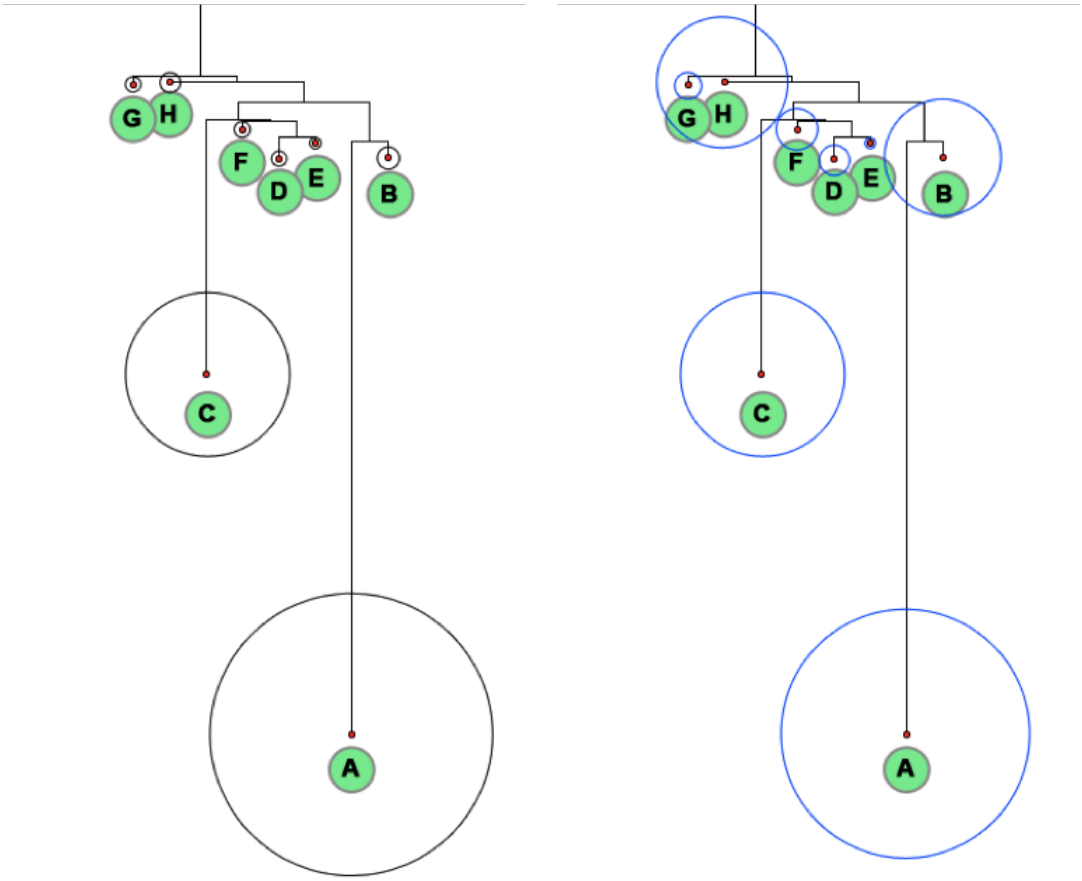
3.3.4 Experiments on Synthetic Data

We synthesize a 2-dimensional, 3-component GMM, draw n samples from it, and run our algorithm to plot the ELM. We want to analyze how the separability and dimension of the true model and the number of samples drawn from it affect the energy landscape. In order



(a) (b)

Figure 3.7: (a) Energy Landscape for a 4 component 1-d GMM with all parameters fixed except two means. Level sets are highlighted in red. The local minima are shown yellow and the first 200 MCMC samples are shown in black (b) Learned ELM and corresponding local minima from the energy landscape.



(a) probability mass (b) volume

Figure 3.8: The probability mass and volume of the energy basins for the 2-d landscape shown in Figure 3.7.

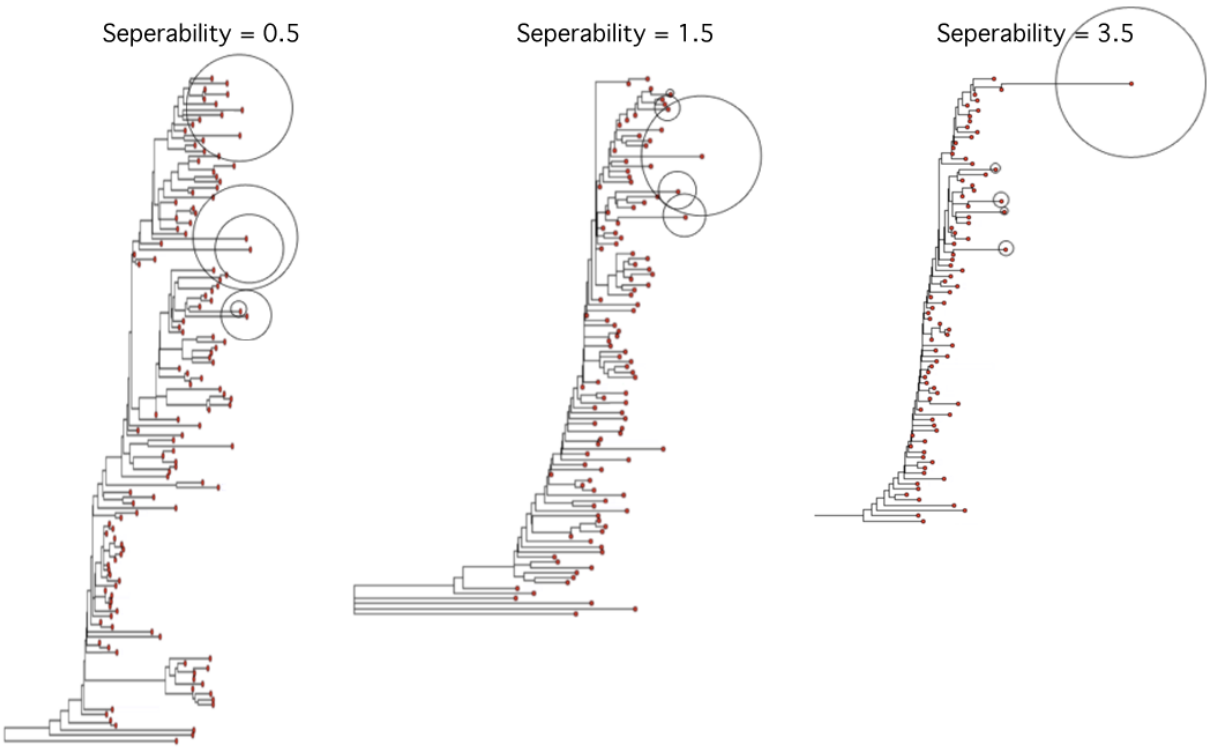
to do this, we repeat the process with true models with varying dimensions, sample sizes and separabilities. The separability of the GMM represents the overlap between separate components of the model and is defined as $c = \min\left(\frac{||\mu_i - \mu_j||}{\sqrt{n}\max(\sigma_1,\sigma_2)}\right)$ [DS00]. We also look at the effect of partial supervision on the energy landscape by assigning ground truth labels to a fraction of the samples in the sample space.

3.3.4.1 Comparing Different Ground-truth Models

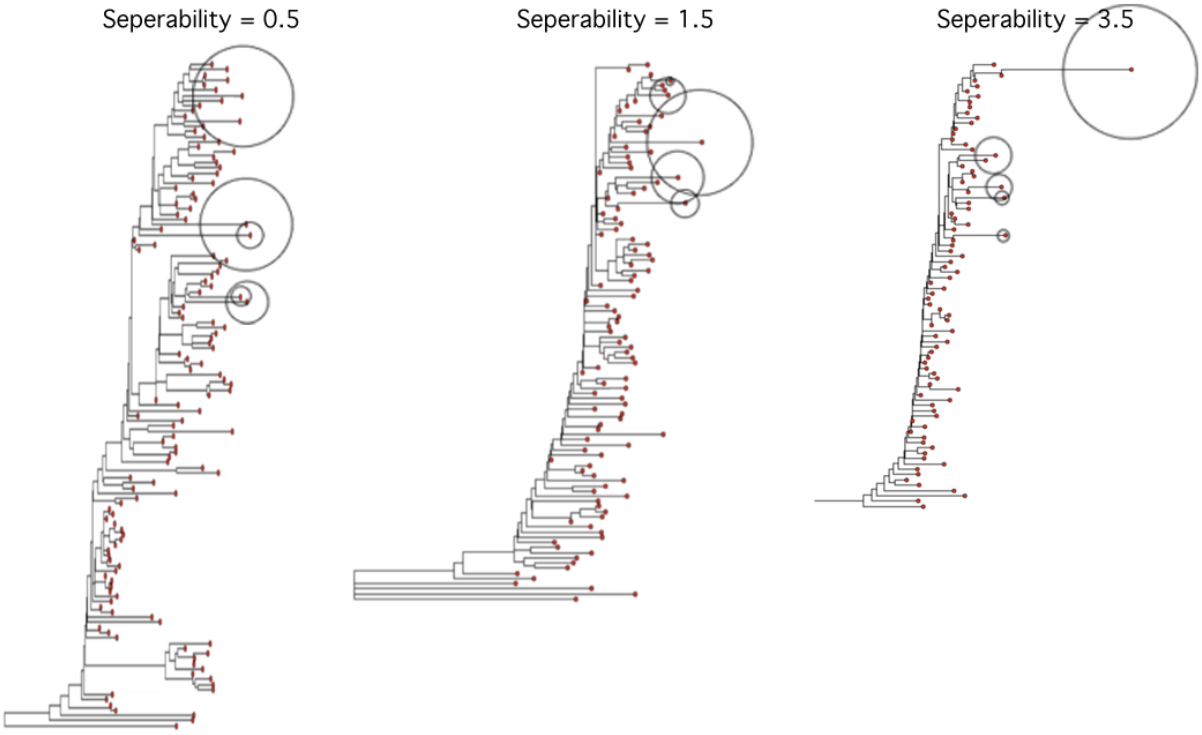
Figure 3.9 shows some of the ELMs with the separability being $\{0.5, 1.5, 3.5\}$ for $n = 100$ samples. The energy landscape becomes increasingly simple as the separability increases: The number of local minima decreases as the separability increases. The landscape for the high separability ($c = 3.5$) case has relatively small energy barriers between the high-energy local minima and a pronounced low-energy global minimum. Conversely, the landscape for the low separability has a structure with high energy barriers between local minima and multiple local minima with similar energy to the global minimum. This indicates that the complexity of learning the mixture model should increase as the separability decreases, as we would expect.

The probability mass and volume of the 5 energy basins corresponding to the lowest-energy local minima are shown in Figures 3.9 (a) and (b): the volume of the circle represents the volume and mass values. The ratio of both the mass and the volume of the lowest energy basin to the mass and volume of the remaining energy basins increases with separability. This is also consistent with the intuition that high-separability landscapes high lower complexity, as it is more likely that the global optimal solution can be found by gradient descent from a randomly sampled starting point.

We examine the affects of partial supervision by assigning ground truth labels to a portion of samples. Figure 3.10 shows the ELMs of a synthesized GMM (dimension = 2, number of components = 3,separability $c = 1.0$, number of samples = 100) with $\{0\%, 5\%, 10\%, 50\%, 90\%, 100\%\}$ labelled data points. Figure 3.11 shows the number of local minima in the ELM for the labeling of $1, \ldots, 100$ samples. This shows a significant decrease in landscape complexity for



(a) Probability mass



(b) Volume

Figure 3.9: ELMs for 100 samples drawn from GMMs with low, medium and high separability

the first 10 labels, and diminishing returns from supervised input after the initial 10%.

3.3.4.2 Behavior of Learning Algorithms

Expectation-maximization (EM) is one of the most popular algorithms for learning a GMM from data. K-means is another popular learning algorithm of GMM which can be seen as a degraded variant of EM with hard assignments in the E-step and the assumption of identical spherical Gaussian components. Two-step EM is a variant of EM proposed in [DS00] that has proved performance guarantee under certain conditions. We study the behavior of these algorithms and the Swedson-Wang cut algorithm by analyzing the distributions of their learning results on the ELMs.

The Swedsen - Wang Cut (SW-cut) algorithm [SW87] [BZ05] is a MCMC method that has much faster convergence rates than classic Markov Chain Monte Carlo methods such as the Gibbs sampler [**?**] in cases when model states are strongly coupled (such as the Ising-Potts model) [Pot52]. The SW algorithm iterates on graphs in two steps:

1. Stochastically group the graph vertices into 'connected component' clusters.

2. Jointly change the labels of all the vertices in each connected component.

To apply the SW-cute algorithm to the Gaussian the Mixture Model, we construct a graph in the sample space by adding edges between all samples $s_1, \ldots, s_n$ that are within a threshold distance of one another. The vertices are each initialized with a label $l(s_i) \in \{1, \ldots, m\}$, where m is the number of components in the GMM. The Mixture Model corresponding to a given labelled state has weights $w_j = |\{l(s_i) = j\}|/n$, means $\mu_j = n/w_j \sum_i s_i * 1(l(s_i) = j)$ for $j = 1, \ldots, m$ and variances $\Sigma_j = \mathrm{Var}(\{s_i | l(s_i) = j\})$.

In the clustering step of the algorithm, connected components are formed by probabilistically removing graph edges between each node pair s_i, s_j with probability $q(s_i, s_j)$:

$$\begin{aligned} q(s_i, s_j) &= (d(s_i, s_j)/M)^{1.5} \text{ if } s_i \text{ and } s_j \text{ have the same label} \\ &= 1 \qquad\qquad\qquad\qquad \text{otherwise} \end{aligned}$$

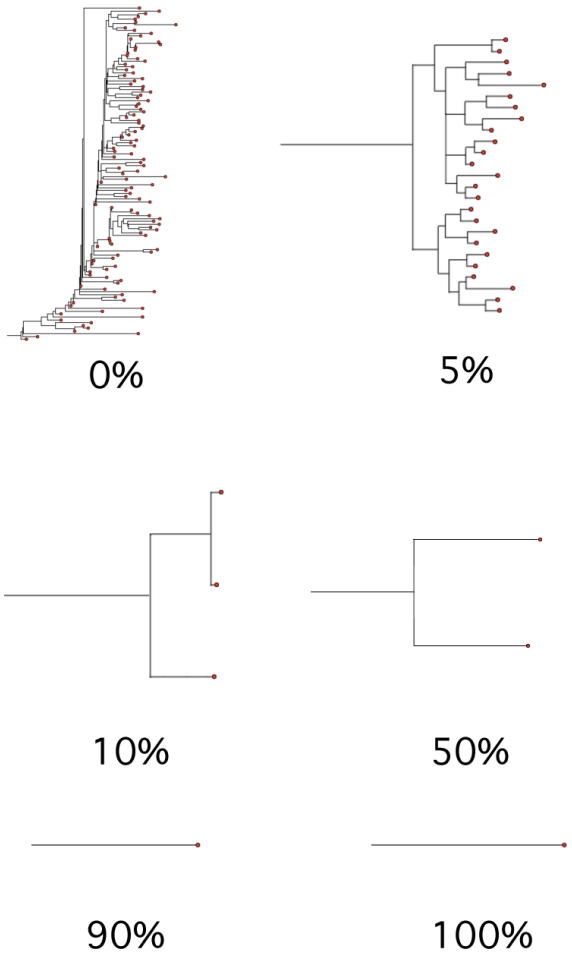


Figure 3.10: ELMs with of synthesized GMMs (separability $c = 1.0$, nSamples $= 100$) with $\{0\%, 5\%, 10\%, 50\%, 90\%, 100\%\}$ labelled data points.

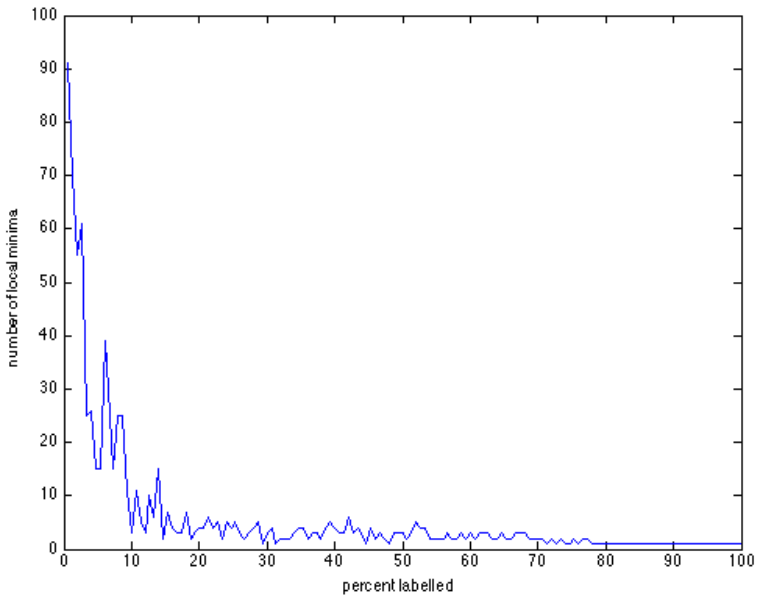


Figure 3.11: Number of local minima versus the percentage of labelled data points for a GMM with separability $c = 1.0$.

where M is the maximum distance between points in the sample space. Next, in the reassignment step, each connected component is flipped to a new label in the following manner: For the connected component C^k, the m relabelings $C_i^k, i = 1, \ldots, m$ are computed where $l(s_j) = i$ if $s_j \in C_i^k$. Next, we find the posterior probabilities of the GMMs corresponding to each relabeling $P(C_i^k)$, and sample the new label i from the discrete distribution $e^{-\beta P(C_i^k)}$. The temperature parameter β allows us to do simulated annealing; we begin with a small β and increase it over time it over time. We use the annealing pattern of $\beta = 0.5, 0.8, 1.3, 2.5, 5.0, 9.0$ for the following experiments.

For each synthetic dataset, we ran the three algorithms for 200 times and found the energy basins of the ELM that the learned models belong to. Hence we obtain a histogram of the learned models on the leaf nodes of the ELM for each learning algorithm as shown in Figures 3.13 and 3.12.

Figure 3.13 shows a comparison between the k-means, EM and two-step EM algorithms for $n = 10$ samples drawn from a low ($c = 0.5$) separability GMM. The results are scat-

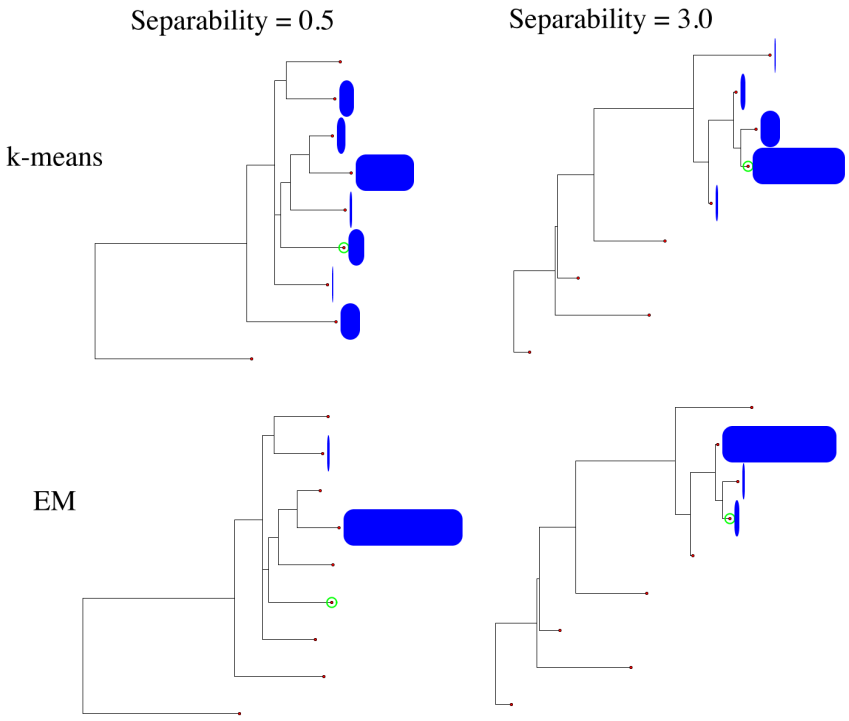


Figure 3.12: The distribution of samples from the k-means and EM algorithms on the ELMs for low ($c = 0.5$) and high ($c = 3.0$) separability 2-dimensional 3-component GMMs.

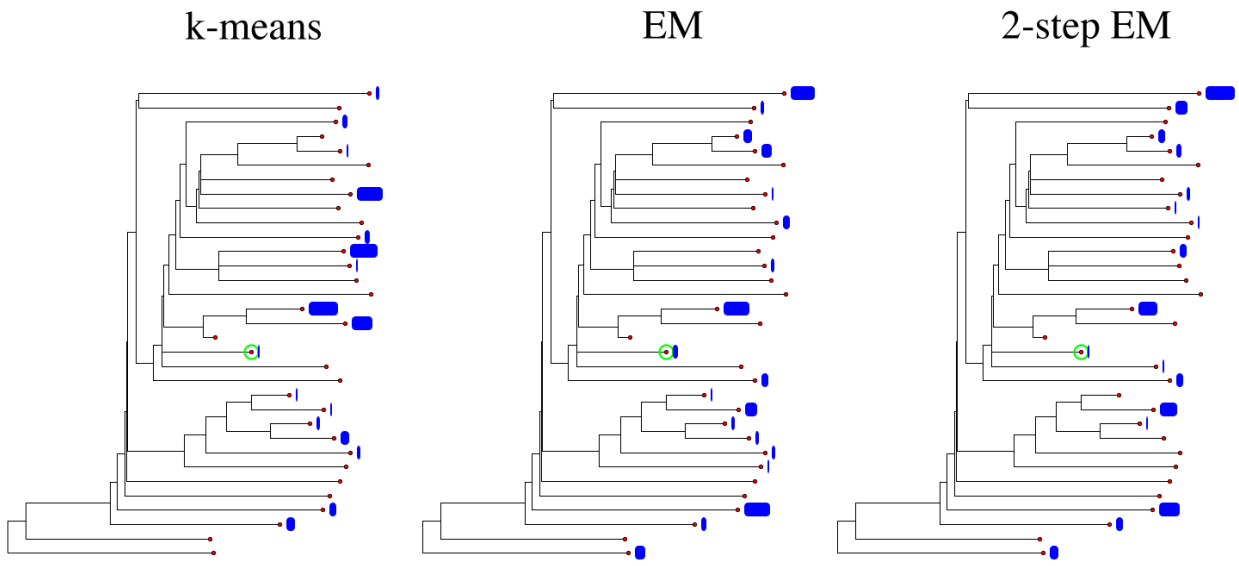


Figure 3.13: The performance of k-means, EM and 2-step EM algorithms on the energy landscape generated by drawing 10 samples from a GMM with low separability ($c = 0.5$)

tered across different local minima regardless of the algorithm. This illustrates the difficulty in learning a model from a landscape with many local minima separated by large energy barriers.

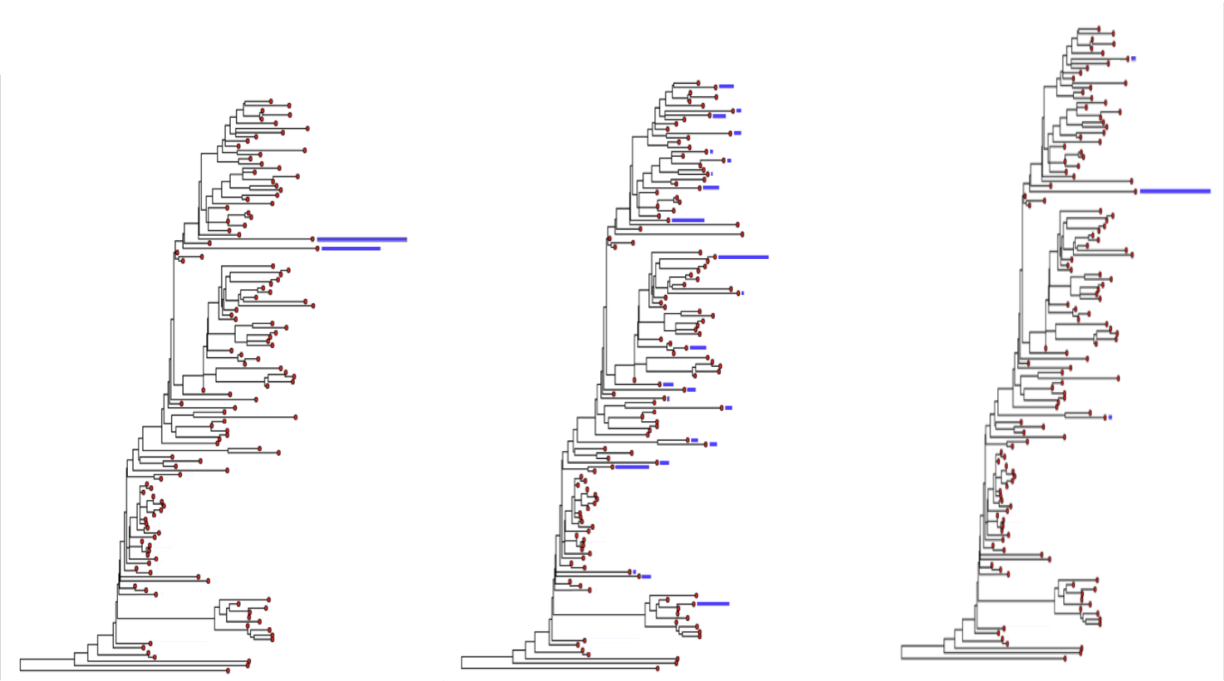
Figures 3.14, 3.15, and 3.16 show a comparison of the EM, k-means, and SW-cut algorithms for $n = 100$ samples drawn from low ($c = 0.5$), medium ($c = 1.5$) and high ($c = 3.5$) separability GMMs. The SW-cut algorithm performs best in each situation, always converging to the global optimal solution. In the low separability case, the k-means algorithm converges to one of the seven local minima, with a higher probability of converging to those with lower energy. The EM algorithm almost always finds the global minimum and thus outperforms k-means. This can be explained by the fact that k-means is a degraded variant of EM with extra assumptions that may not hold. However, in the high separability case, the k-means algorithm converges to the true model the majority of the time, while the EM almost always converges to a local minimum with higher energy than the true model. This can be explained by a recent theoretical result showing that the objective function of hard-EM (with k-means as a special case) is the summation of the standard energy function of GMM with an inductive bias in favor of high-separability models [TH12, SCR12].

3.3.5 Experiments on Real Data

We ran our algorithm to plot the ELM for the well-known Iris data set from the from the UCI repository [BM98]. The data set contains 150 points in 4 dimensions and can be modeled as a 3-components 4-dimensional GMM. The three components each represent a type of iris plant and the true component labels are known. The points corresponding to the first component are linearly separable from the others, but the points corresponding to the remaining two components are not linearly separable.

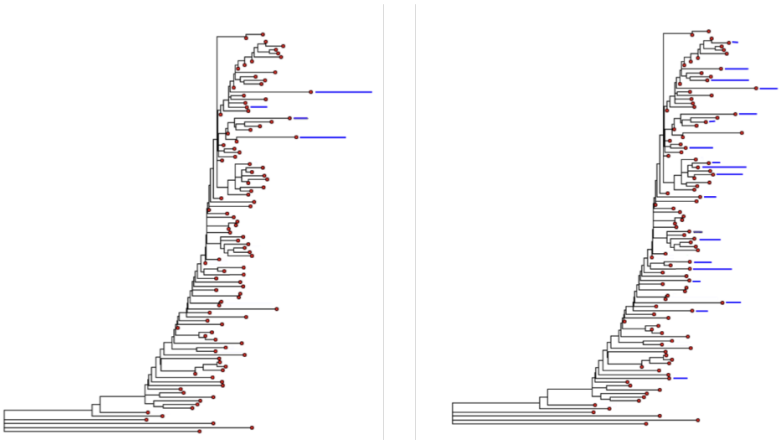
Figure 3.17 shows the ELM of the Iris dataset. We visualize the local minima by plotting the ellipsoids of the covariance matrices centered at the means of each component in 2 of the 4 dimensions.

The 6 lowest energy local minima are shown on the right and the 6 highest energy local



(a) EM (b) k-means (c) SW-cut

Figure 3.14: Low separability $c = 0.5$:histogram of EM, k-means, and SW-cut algorithm results on the ELM.

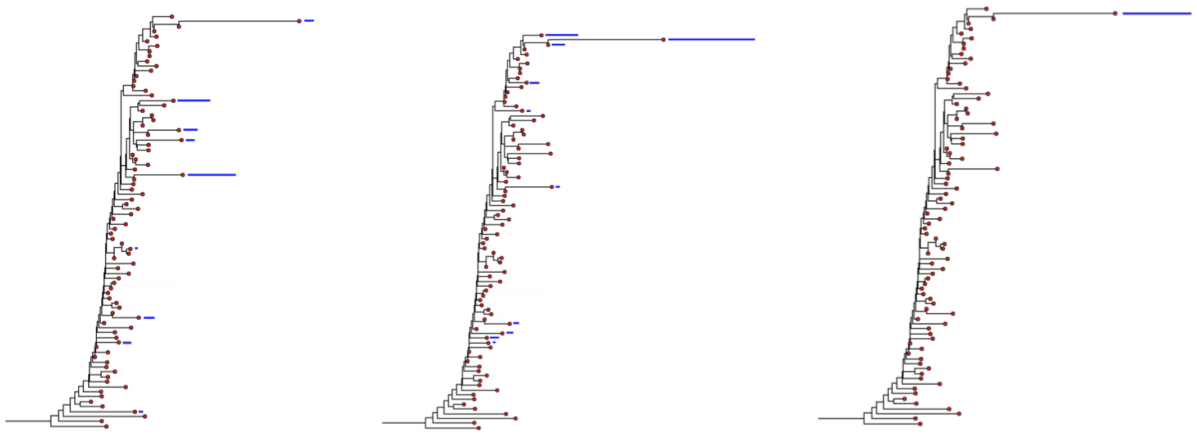


(a) EM

(b) k-means

(c) SW-cut

Figure 3.15: Medium separability $c = 1.5$: histogram of EM, k-means, and SW-cut algorithm results on the ELM.



(a) EM (b) k-means (c) SW-cut

Figure 3.16: High separability $c = 3.5$: histogram of EM, k-means, and SW-cut algorithm results on the ELM.

minima are shown on the left. The high energy local minima are less accurate models than the low energy local minima. The local minima (E) (B) and (D), have the first component is split into two and the remaining two (non-separable) components merged into one. The local minima (A) and (F) have significant overlap between the 2nd and 3rd components and (C) has the components overlapping completely. The low-energy local minima (G-L) all have the same 1st components and slightly different positions of the 2nd and 3rd components.

The same experiment was performed labeling 0, 5, 10, 50, 90, and 100 percent of the Iris data with the ground truth values. Figure 3.18 shows the global minimum of the energy landscape for each experiment.

3.4 Mixtures of Bernoulli Templates

A Bernoulli template $P \in \{0,1\}^n$ is an n-dimensional binary vector. A sample x generated from P is an n-dimensional vector whose i-th coordinate x_i is equal to P_i with a fixed probability p and equal to $1 - P_i$ with probability $1 - p$. That is, the sample drawn from P is a noisy version of P. An m-component Mixture of Bernoulli Templates (MBT) B is a weighted mixture of m Bernoulli templates defined by the set of templates $\{P_i\}$ and weights

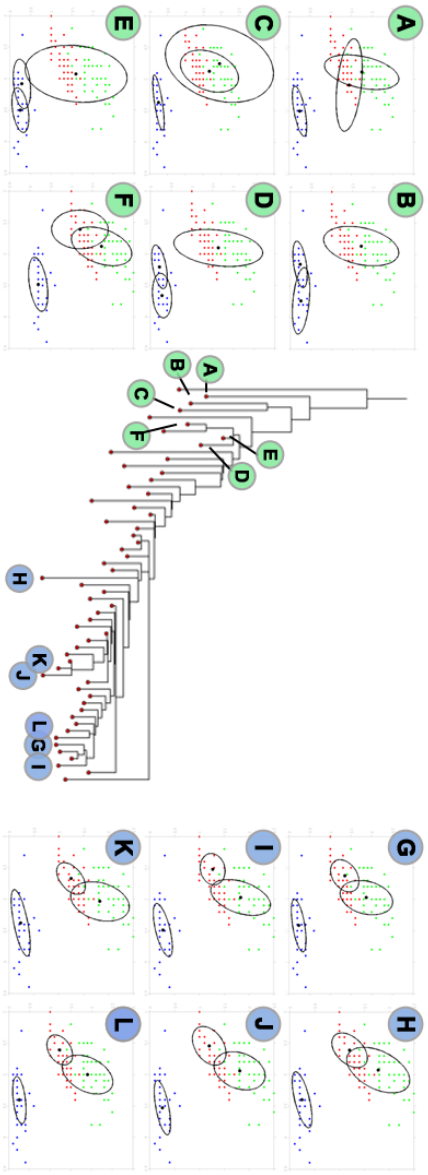


Figure 3.17: ELM of the Iris dataset and corresponding local minima.

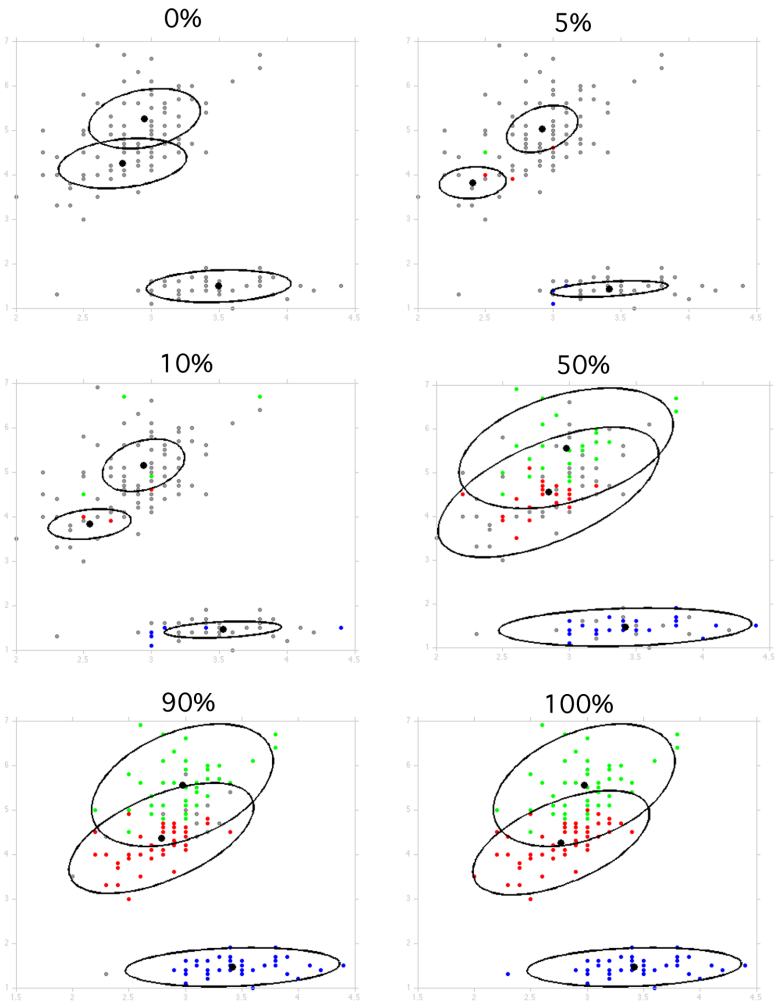


Figure 3.18: Global minima for learning the Iris Mixture Model with 0, 5, 10, 50, 90, and 100% of the data labelled with the ground truth values. Unlabeled points are drawn in grey and the labelled points are colorized in red, green or blue.

$\{w_i | w_i \in [0,1]\}$ for $i \in \{0,\ldots,m\}$ with $\sum w_i = 1$. Samples s_j are drawn from B by first sampling a component P_i from the discrete distribution of weights $\{w_i\}$, then sampling from the template P_i as outlined above. We wish to compute the Energy Landscape Map of the space of MBTs with a fixed noise level p. The energy function that we use is the negative log of the posterior, given by $E(B) = -\log P(B|z_i : i = 1 \ldots M)$ for M samples $\{z_i\}$, and

$$P(B|z_i) = \sum_{i=1}^{m} w_i p^{\sum_{j=1}^{n} I(z_i(j)=P_i(j))}(1-p)^{\sum_{j=1}^{n} I(z_i(j)\neq P_i(j))},$$

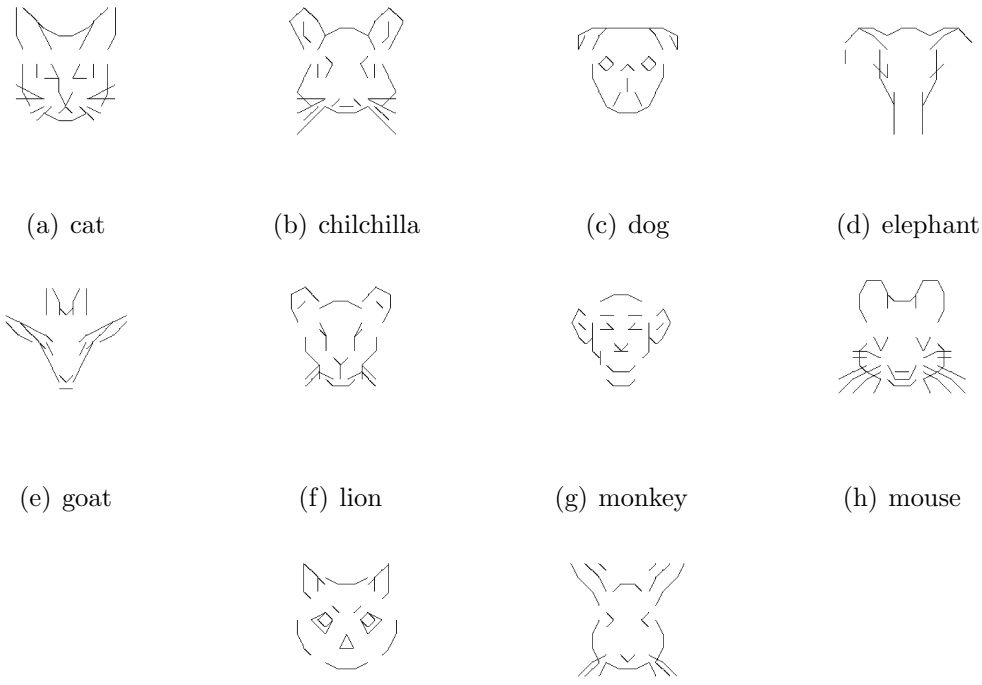
where $P_i(j)$ is the j-th component of the i-th Bernoulli template in B, and $z_i(j)$ is the j-th component of the i-th sample.

We discretize the hypothesis space by allowing the weights to take values $w_i \in \{0, 0.1, \ldots, 1.0\}$. In order to adapt the GWL algorithm to the discrete space, we use the coordinate descent algorithm in lieu of gradient descent.

In [**?**], Barbu proposes a Two-Round EM algorithm for learning MBTs with a performance bound that is dependent on the number of components m, the dimension Beronouilli template dimension n, and and noise probability p. We examine how the ELM of the hypothesis space changes with these factors.

Experiment on synthetic data We synthesized Bernouilli templates which represent animal faces as show in Figure 3.19. Each animal face is a 9x9 grid with each cell containing up to 3 sketches. The dictionary of sketches contain 18 elements, each of which is a straight line containing the midpoints of vertices of a square of show in Figure 3.20. The Bernouilli template can therefore be represented as a $18 * 9 * 9 * 3 = 4374$ dimensional binary vector. There are 10 animals total, so we have a Bernoulli mixture model with the number of component $M = 10$, dimension $d = 4374$ and a variable number of samples.

We construct the energy landscape maps of the Bernouilli mixture model for varying numbers of samples $n = 100, 300, \ldots, 7000$ and varying noise level $p = 0, 0.05, \ldots, 0.5, 0.55$. The number of local minima in each energy landscape is tabulated in Figure 3.21 (b) and drawn as a heat map in Figure 3.21 (b). As expected, the number of local minima increases as the noise level p increases, and decreases as the number of samples decreases. In particular, with no noise, the landscape is convex and with noise $p > 0.45$, there are too many local



(a) cat (b) chilchilla (c) dog (d) elephant

(e) goat (f) lion (g) monkey (h) mouse

(i) owl (j) rabbit

Figure 3.19: Animal face templates [low overlap]

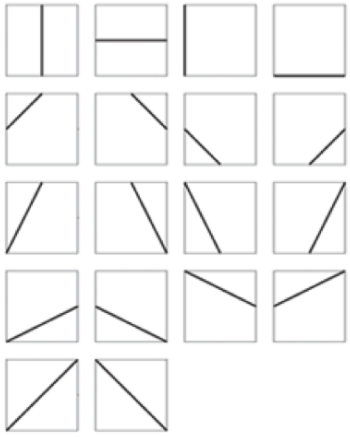
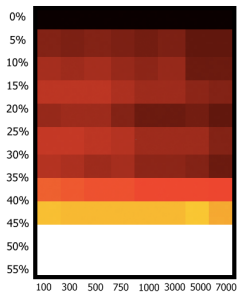


Figure 3.20: Synthetic animal face picture dictionary consisting of 18 sketches.



(a) landscape map

1	1	1	1	1	1	1	1
13	12	13	12	11	12	9	9
17	16	17	15	14	15	10	10
20	20	20	18	16	16	14	13
15	16	16	13	10	10	11	9
21	21	20	19	16	16	16	13
19	18	16	17	14	14	13	10
39	38	37	37	36	36	36	33
51	50	50	50	50	50	52	48
1591	1034	4945	5000	4566	4524	4103	4083
3123	2011	9833	9943	9075	8991	8147	8111

(b) number of local minima

Figure 3.21: Energy Landscape for varying values of p and number of samples in the Bernouilli Templates model.

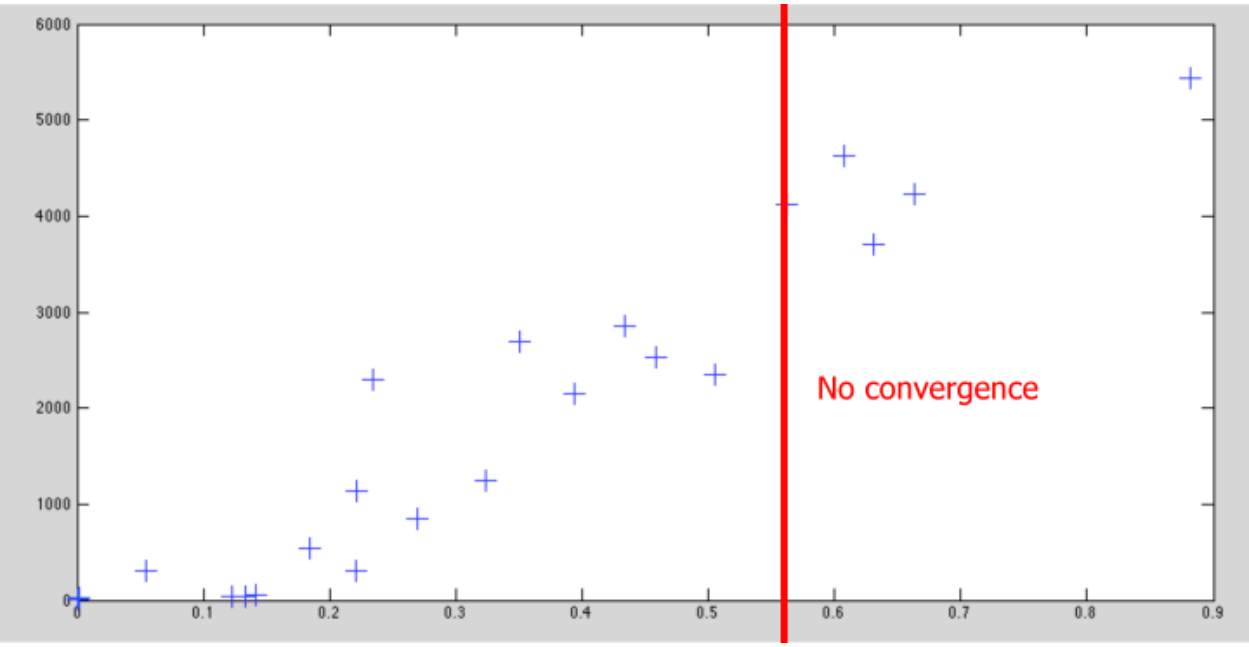


Figure 3.22: Number of local minima found for varying values of separation in the Bernouilli Templates model.

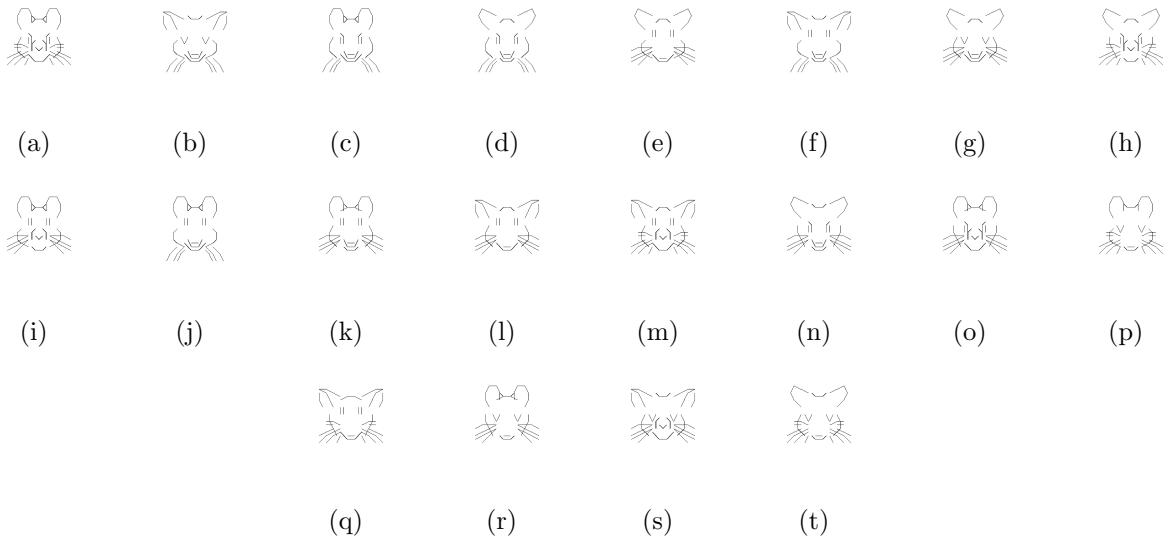


Figure 3.23: Mouse face templates [high overlap]

minima and the algorithm does not converge.

We repeat the same experiment using variants of a mouse face as shown in Figure 28. We swap out components of the mouse face (the eyes, ears, whiskers, nose, mouth, head top and head sides) for three different variants. We thereby generate 20 Bernouilli templates which have relatively high degrees of overlap. We generate the ELMs of various Bernouilli Mixture Models containing three of the generated templates and noise level $p = 0$. We calculate the overlap of the 3-component mixture models and generate their ELMs. Then we plot the number of local minima in the ELMs versus the degree of overlap as show in Figure 27. As expected, the number of local minima increases with the degree of overlap, and there are too many local minima for the chains to converge past overlap $c = 0.5$.

Experiment on real data

We use a data set consisting of binarized features in animal face pictures as shown in figure 3.25. These are generated by partitioning the image into n kxk square cells, convolving 8 Gabor filters (at 45 degree increments) with each subimage. The filter response is banarized by thresholding. This produces an $8n$ dimensional binary sample. We chose m different animals with N images of each animal represented in the sample set. We vary the noise level p by changing the binarization threshold.

Figure 3.24: Sample from the dog animal face template with noise $p = 0.1$

Figure 3.25: Animal face images and corresponding binary sketches indicates the existence of a Gabor filter response above a fixed threshold.

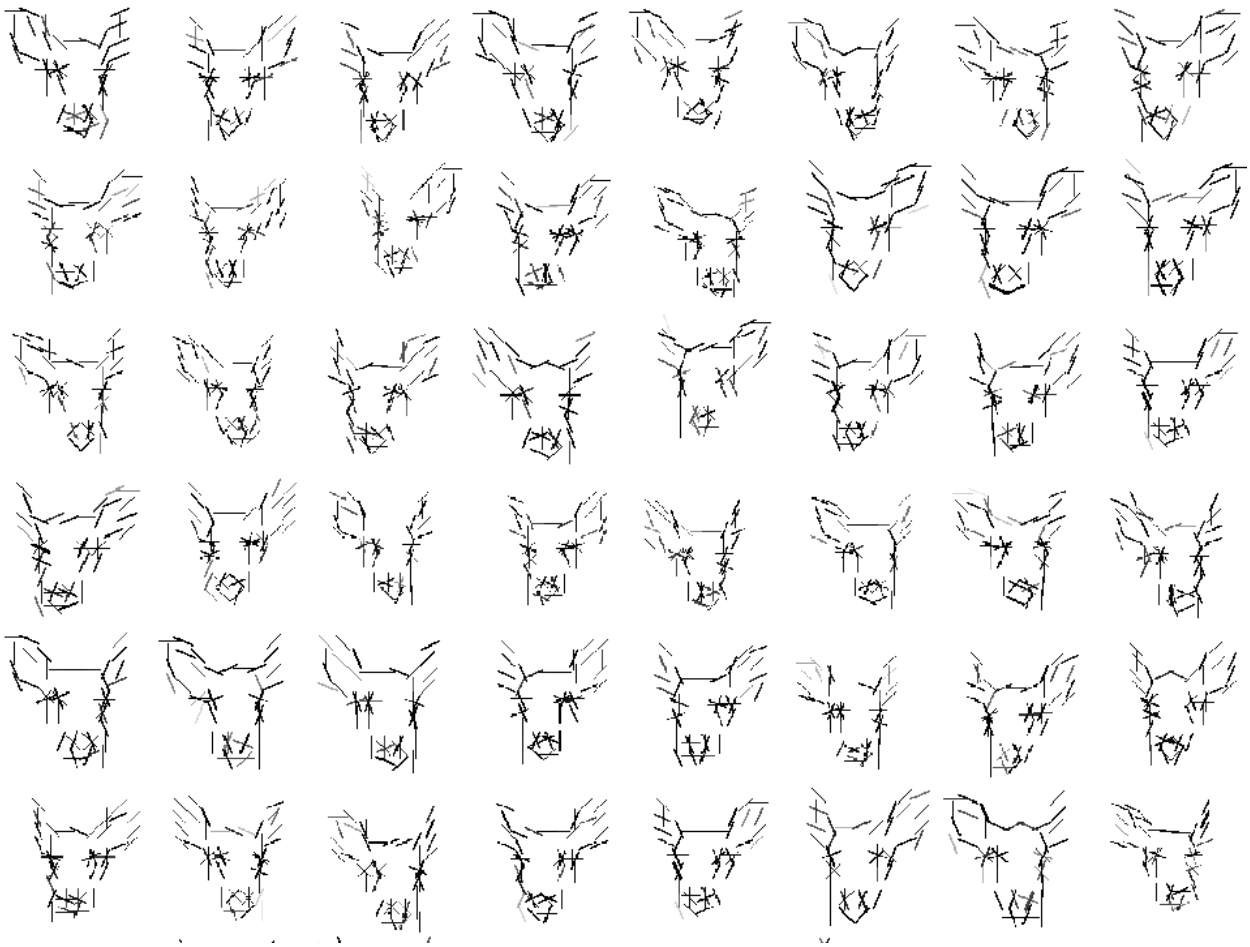


Figure 3.26: Deer face sketches

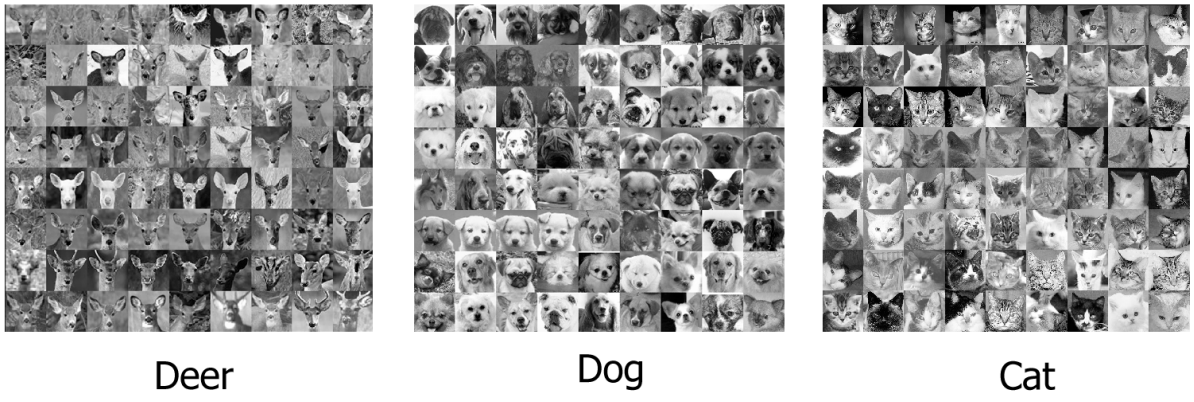


Figure 3.27: All faces

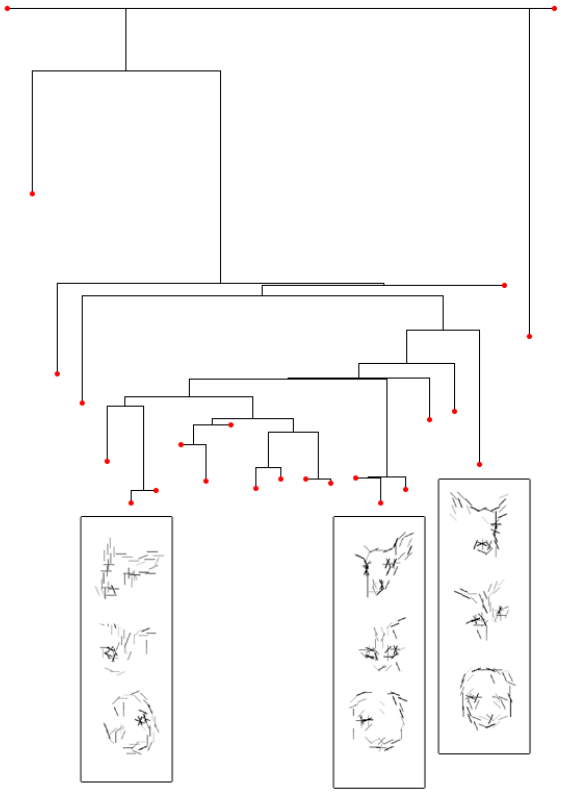
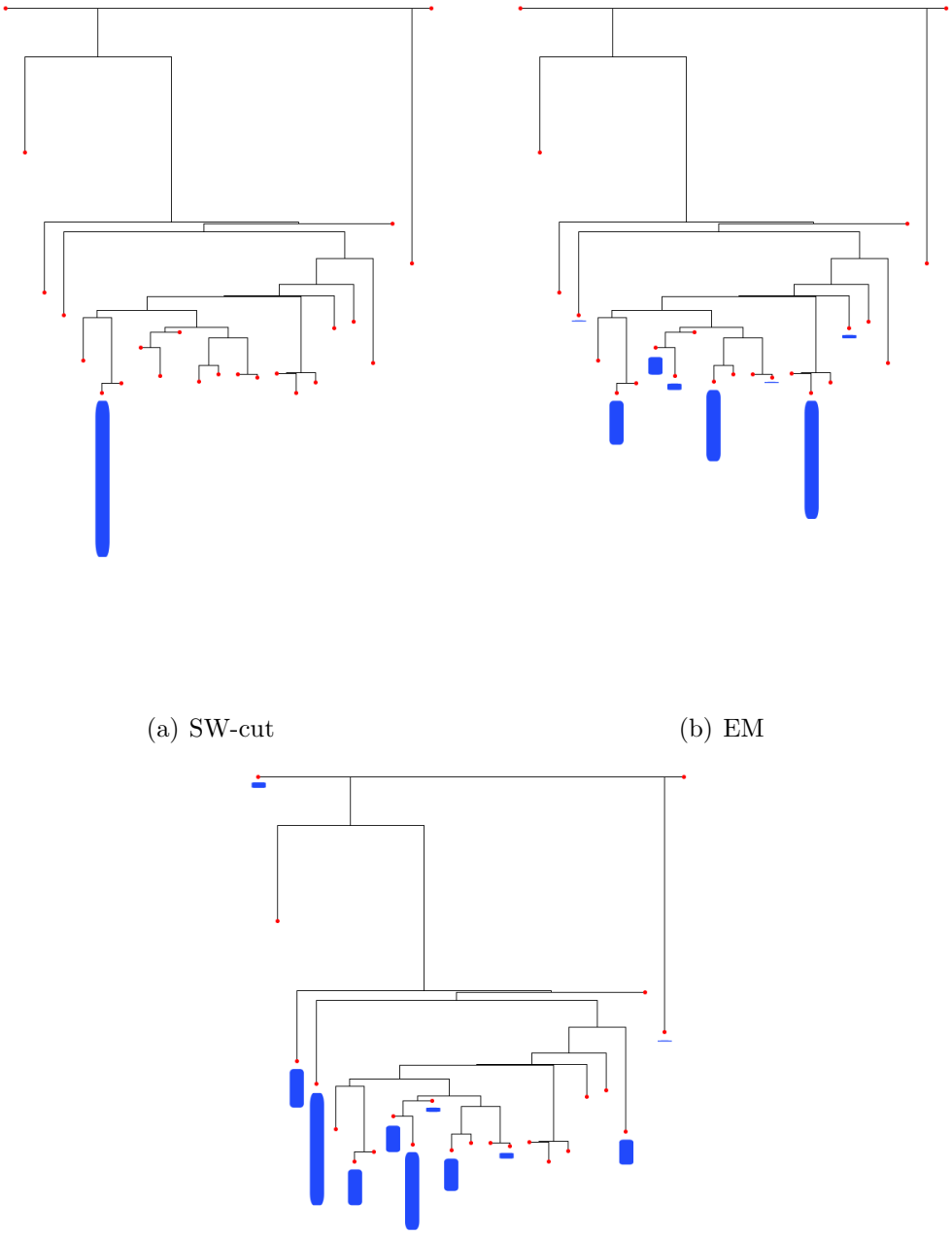


Figure 3.28: ELM of three animal faces (dog, cat, and deer). We show the Bernouilli templates corresponding to three local minima with large energy barriers.



(a) SW-cut

(b) EM

(c) k-means

Figure 3.29: Comparison of SW-cut, k-means, and EM algorithm performance on the ELM of animal face Bernouilli Mixture Model.

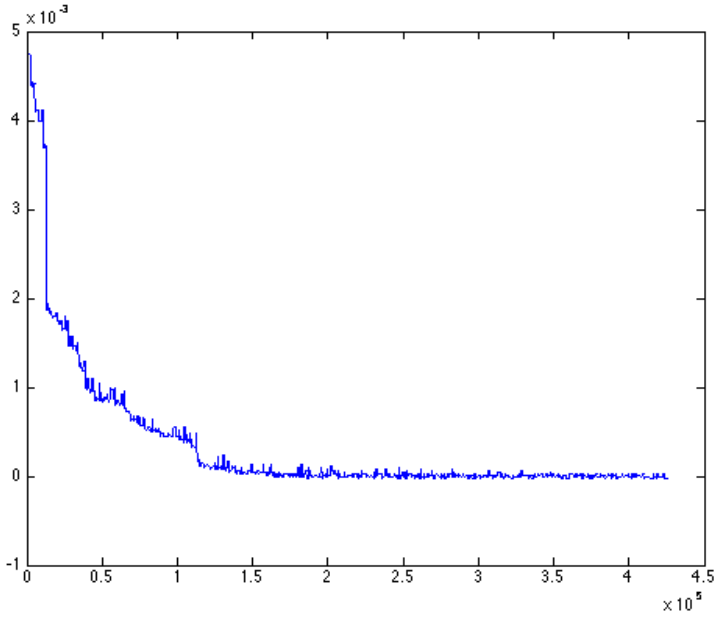


Figure 3.30: Intra-chain distance post burn-in period when no new local minima are found. As the energy barriers converge, the intra chain distance approaches 0.

3.5 ELMs of Biclustering

Biclustering is the process of learning biclusters (i.e., sub-matrices) from a data matrix [MO04]. It finds applications in a variety of domains, e.g., finding genes with similar expression patterns under subsets of conditions [CC00, GLD00, CDG04], finding people who enjoy similar movies [YWW02], and finding correlations between words and phrases to learn grammar rules [TH08].

We study a simple version of biclustering: identifying a single multiplicatively coherent bicluster from a possibly noisy data matrix that only contains the target bicluster and its transposition. There may be overlap between the bicluster and its transposition, and the data matrix elements in the overlapped part are the summations of the corresponding elements from the two biclusters. This biclustering problem becomes more difficult when there is more noise in the data and when the two biclusters have more overlap.

Energy: We use the energy function from [TH08], which contains two terms.

$$E(b) = -\text{Coh}(b) - \alpha \times \text{Size}(b)$$

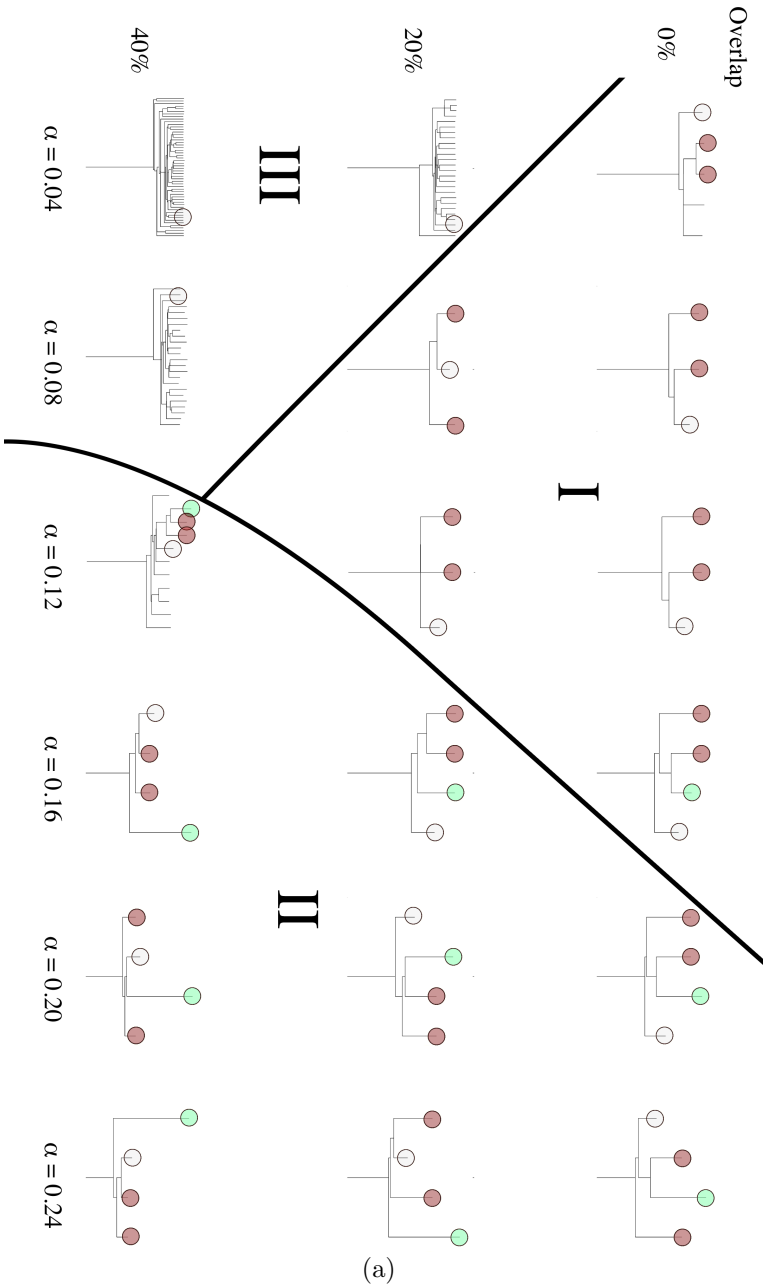
$\text{Coh}(b)$ measures the multiplicative coherence of the bicluster b, which corresponds to the likelihood term; $\text{Size}(b)$ measures the size of the bicluster b, which corresponds to the prior term that favors a large bicluster; the hyper parameter α is the strength of the prior.

Experiment: We synthesized biclusters of size 10×10 with 11 levels of overlap between the bicluster and its transposition. We then generated the data matrices based on the biclusters. Random noise of level p was added into the data matrix. For each data matrix, we ran our algorithm to plot the ELMs with different values of the hyperparameter α.

Figure 3.31 shows some of the ELMs with the overlap being $\{0\%, 20\%, 40\%\}$, the hyperparameter $\alpha = \{0.02, 0.06, \ldots, 0.22\}$ and the noise level $p = 0.02$. The local maxima corresponding to the correct biclusters (either the target bicluster or its transposition) are marked with solid red circles; the empty bicluster is marked with a gray circle; and the maximal bicluster containing the whole data matrix is marked with a solid green circle. These ELMs can be divided into three categories. Category I: the true model is easily learnable; the global maxima correspond to the correct biclusters and there are fewer than 6 local minima. Category II: the prior dominates the energy function because of a very large α; the true model cannot be learned. Category III: the landscape contains many local minima at similar energy levels which correspond to sub-matrices of the correct biclusters because α is too small; the true model cannot be learned, although it is possible (but difficult) to obtain approximately correct solutions. Figure 3.32(a) shows the difficulty map (the division of the three categories) with a larger range of overlap; Figure 3.32(b) shows the difficulty map with no noise. It can be seen that as the overlap increases, the acceptable region (i.e., category I) of hyperparameter α decreases. Hence it becomes increasingly more important to select an appropriate value of α when the biclusters are expected to overlap. Further, if the biclusters overlap greater than 50%, they become hard to learn for any selection of α. If the expected amount of bicluster overlap is know, constructing this type of difficulty map for a simplified case may be useful in determining the appropriate α for the full problem. By

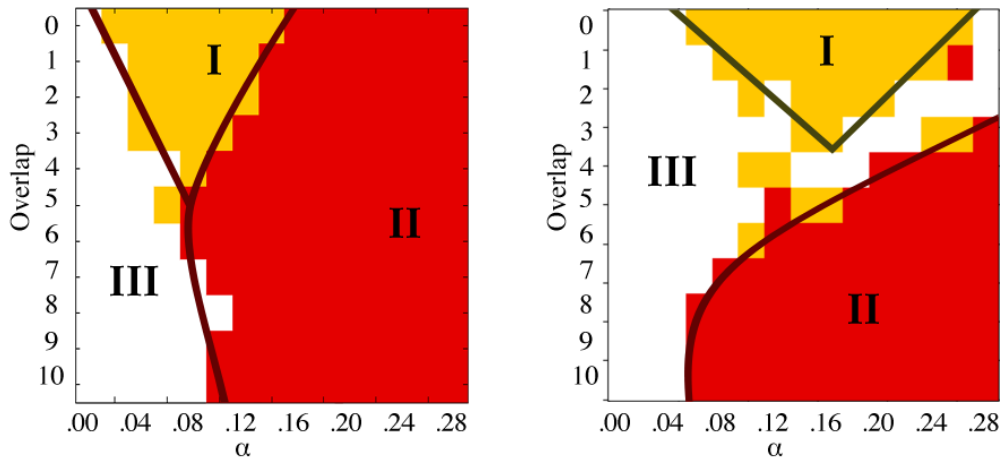
comparing figure 3.32(a) and 3.32(b), we can see that with less noise, the learnable region becomes wider and hence the learning algorithms would be less sensitive to the choice of α.

Experiment 2: We repeat the experiment with 3 biclusters of size 10×10 with 20% pairwise overlap, and $p = 0.1$ noise. Figures 3.5 (a) and (b) show the energy landscape with $\alpha = 0.1$ and $\alpha = 0.15$. The phase transition from region 1 (easily learnable landscape) and region 2 (impossible to learn) happens between those values, indicating an increased sensitivity to the α parameter for greater numbers of biclusters.



(a)

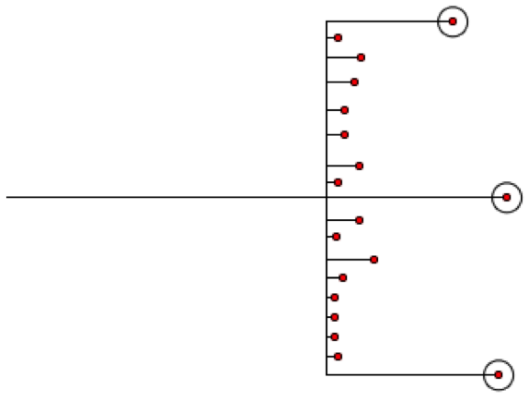
Figure 3.31: Energy Landscape Maps for learning two biclusters with 0%,20%, 40% overlap and hyperparameter α. Red: correct bicluster; Grey: empty bicluster; Green: maximal bicluster.



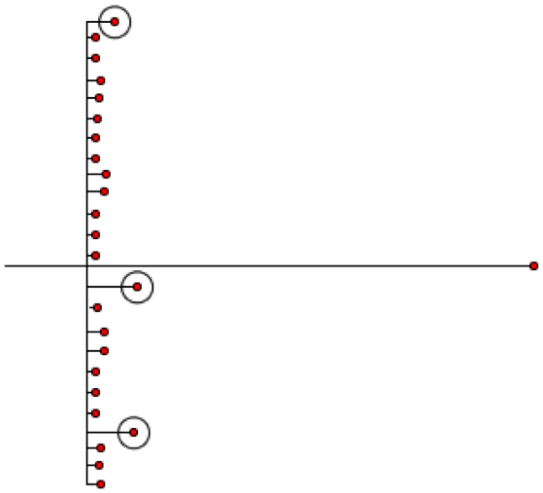
(a) Noise $p = 0.02$

(b) Noise $p = 0.00$

Figure 3.32: Difficulty map for biclustering (a) with noise (b) without noise. Region I: the true model is easily learnable. Region II: the true model cannot be learned. Region III: approximations to the true model may be learned with some difficulty.



(a) $\alpha = 0.1$



(b) $\alpha = 0.15$

Figure 3.33: ELMs for 3 biclusters of size 10×10 with 20% pairwise overlap, and $p = 0.1$ noise.

CHAPTER 4

Dependency grammar sampling

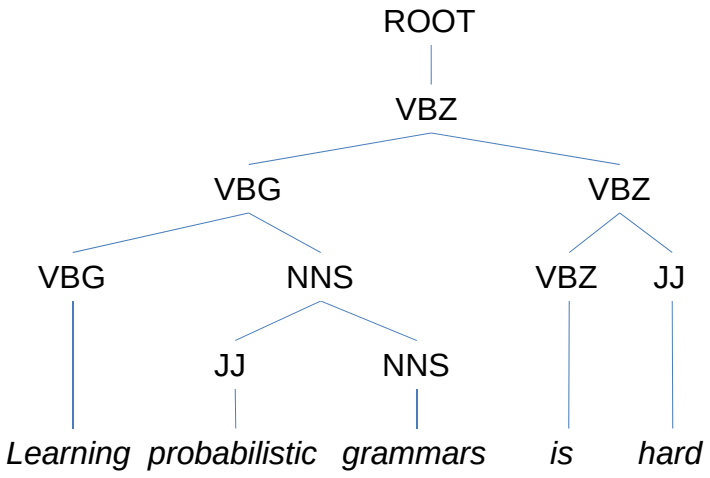
4.1 Dependency Grammar

Here we study a subclass of probabilistic context-free grammars called dependency grammars. The space of dependency grammars is significantly smaller than that of probabilistic context-free grammars and therefore is easier to explore. On the other hand, dependency grammars are still expressive enough to represent many natural language phenomena and have been widely used in natural language processing [Mel88, Col99, KMN09]. There has been increasing interest in learning dependency grammars from data, in either a supervised way (e.g. [Col99, Cha01]) or an unsupervised way (e.g., [KM04, HJM09]). By studying the energy landscape of dependency grammars, we hope to gain some insight that could benefit the research of dependency grammar learning.

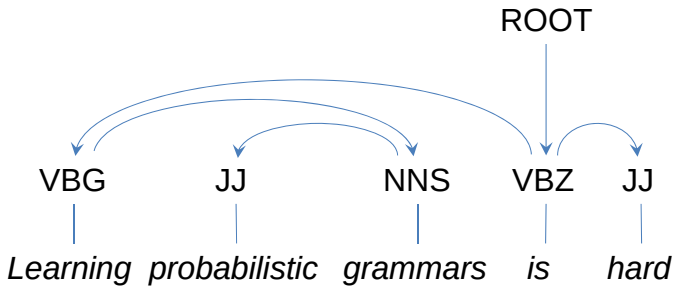
A dependency grammar is a context-free grammar that requires its grammar rules to take the form of $ROOT \to A$, $A \to AB$, $A \to BA$ or $A \to a$, where $ROOT$ is the start symbol, A and B are nonterminals, and a is a terminal. Figure 4.1(a) shows the grammatical structure of the example sentence specified by a dependency grammar. Equivalently, we can represent the grammatical structure specified by a dependency grammar using a set of *dependencies*, as shown in Figure 4.1(b). Each dependency is a directed arc between two nonterminals.

4.1.1 Space size

Formally a dependency grammar is a tuple $< S, N, R >$ where S is the start symbol for the rewrite rules, N is the set of terminal symbols (such as words in the english langue) and R is the set of rewrite rules $\{S \to n | n \in N\} \cup \{n \to \alpha n | \alpha \in N\} \cup \{n \to n\beta | \beta \in N\}$. The



(a) The parse tree representation



(b) The dependency representation

Figure 4.1: The grammatical structure generated by a dependency grammar.

dependency grammar is uniquely defined by: 1. the transition probability matrix between the terminal nodes 2. the vector of transition probabilities from the start node to each terminal node 3. the left and right direction stop probabilities of each terminal node. Hence the space of dependency grammars with n nodes has $n^2 + n + 2 * 2 * n$ dimensions. Since the probability vectors are constrained to sum to 1, the valid dependency grammars form a subspace of dimensionality $n * (n - 1) + n - 1 + 2 * n$.

4.2 Curriculum Learning of Dependency Grammar

Most of the existing automatic approaches to learning dependency grammar start with all the sentences of a training corpus and try to learn the whole grammar. On the other hand, humans learn the grammar of their native language in a very different manner: they are exposed to very simple sentences as infants and then to increasingly more complex sentences as they grow up. Such a learning strategy has been termed *curriculum learning* [BLC09]. Earlier research into curriculum learning of grammars produced both positive [Elm93] and negative results [RP99]. More recently, Spitkovsky et al. [SAJ10] empirically showed that curricula are helpful in unsupervised dependency grammar learning.

To explain the benefits of curricula, Tu and Honavar [TH11] suggested that an ideal curriculum gradually emphasizes data samples that help the learner to successively discover new grammar rules of the target grammar, which facilitates the learning. As an alternative explanation, Bengio et al. [BLC09] hypothesized that a good curriculum corresponds to learning starting with a smoothed objective function and gradually reducing the degree of smoothing over the curriculum stages, thus guiding the learner to better local minima of the energy function. In this section, we try to verify this second explanation by plotting and analyze the series of ELMs from different curriculum stages of learning dependency grammars.

4.2.1 Natural language representation

We chose to study a Dependency Grammar learned from the natural English language. We simplify this grammar to n nodes (corresponding to parts of speech such as Noun, Verb, Adjective and Adverb) by removing the nodes that appear with least frequency. This is done by drawing a large number of samples from the grammar and calculating the frequencies of each terminal node, then removing the node that occurs the least frequently and rescaling the remaining probability vectors to sum to 1. Thus we obtain a grammar with $n^2 + n + 2 * 2 * n$ parameters which represents a simplified version of the English language grammar.

4.2.2 Energy Function and Gradient

The energy function for the dependency grammar specified by the parameter vector θ is

$$E(\theta) = \sum_{x \in D} \log P(\theta|x)$$

where D is the set of samples $x_i, i = 1, \ldots, m$ and

$$\sum \log P(\theta|x) = \sum \log P(x|\theta) + \log P(\theta)$$

where $\log P(\theta)$ is the Dirichlet prior. The probability $P(x|\theta)$ is the sum of the probabilities of $P(PT, x|\theta)$ for each possible parse PT of the input sample x. The probability of every parse is the product of all the grammar rules θ_r is invoked; if a grammar rule θ_r appears $k(r, PT)$ times in the parse, then

$$P(x|\theta) = \sum P(PT, x|\theta) = \sum_{PT} \prod_{r} \theta_r^{k(r,PT)}.$$

The gradient of this function can be computed analytically for gradient descent. The partial derivative of $\log P(\theta|x)$ with respect to the i-th parameter θ_i is

$$\begin{aligned} \frac{\delta}{\delta\theta_i} \log P(\theta|x) &= \frac{\delta}{\delta\theta_i} \left(\sum \log P(x|\theta) + \log P(\theta) \right) \\ &= \frac{1}{P(x|\theta)} \frac{\delta}{\delta\theta_i} P(x|\theta) + \frac{1}{P(\theta)} \frac{\delta}{\delta\theta_i} P(\theta) \end{aligned}$$

and

$$\frac{\delta}{\delta\theta_i}P(x|\theta) = \sum_{PT}\frac{\delta}{\delta\theta_i}P(PT,x|\theta) = \sum_{PT}\frac{\delta}{\delta\theta_i}\prod_r \theta_r^{k(r,PT)}$$
$$= \sum_{PT}\left(\prod_{r\neq i}\theta_r^{k(r,PT)}\right)k(i,PT)\theta_i^{k(i,PT)-1} = \sum_{PT}\frac{k(i,PT)}{\theta_i}\prod_r \theta_r^{k(r,PT)}.$$

Note that after each gradient descent step, the new parameter vector $\theta^{t+1} = \theta^t + \alpha_t\vec{\nabla}E(g(\theta_t))$ will likely be outside of the valid dependency grammar parameter space which has the constraints $0 \leq \theta_i \leq 1$ for all i and $\sum_{i\in I_k}\theta_i = 1$ for disjoint index sets I_k covering the set $\{i = 1, 2, 3, \ldots, n^2 + n + 2 * 2 * n\}$ where n is the number of nodes in the dependency grammar.

In order to project the new parameter vector θ^{t+1} back into the dependency grammar space, we define the projection $P_{\text{sim}}(\theta) = \sum_k P_{\text{s}}(\theta_{k_1}, \ldots, \theta_{k_l})$ for simplex index sets I_k and the simplex projection $P_s(x)$. To efficiently compute $P_s(x)$, we a modified version of Wang's fast simplex projection algorithm outlined in Algorithm [1].

4.2.3 Convergence issues

We found that the space generated by this a dependency grammar with $n = 3$ or more nodes ($n^2 + n + 4n$ parameters) cannot be efficiently traversed by the WL algorithm for the following reasons:

- The number of local minima in the space is too large (number of local minima still growing linearly after 100,000 iterations as shown in Figure 4.2.3)
- The complexity of the gradient computations grows exponentially with the sentence length of each sample. As the gradient is typically computed over 100 times per iteration and only approximately 10 percent of proposed MCMC moves are accepted, it forms a bottle neck for the algorithm. This forces us to limit the maximum sentence length of each sample.

Algorithm 1 Probability Simplex Projection Algorithm

Input: $\vec{x} \in \mathbb{R}^d$

Output: $\vec{y} = P_s(\vec{x})$ st. $\sum y_i = 1$ and $\epsilon \leq y_i \leq 1$

{**Step 1**} set $[z_1, \ldots, z_d]$ = SortInDescendingOrder($[x_1, \ldots, x_d]$)

{**Initialize**}

sum = 0

prevLambda = 0

currLambda = 0

loop $i = 1, \ldots, d$

    sum += z_i

    currLambda = (sum - 1) / $(i + 1)$

    if $z_i \leq$currLambda **then**

        break

    end if

    prevLambda = currLambda

end loop

set $y_i = \max(x_i - \text{prevLambda}, \epsilon)$ for $i = 1, \ldots, d$

Output: $\vec{y} = [y_1, \ldots, y_d]$

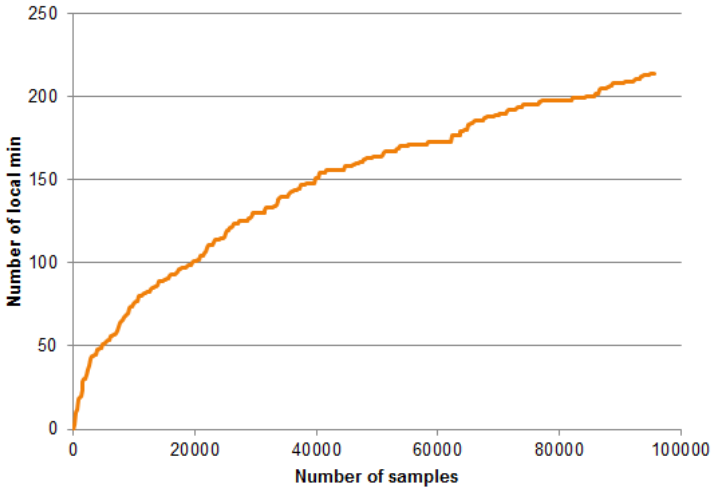


Figure 4.2: Dependency grammar not fully converging in the continuous space: 1. too many local minima 2. performing gradient descent at each iteration is slow

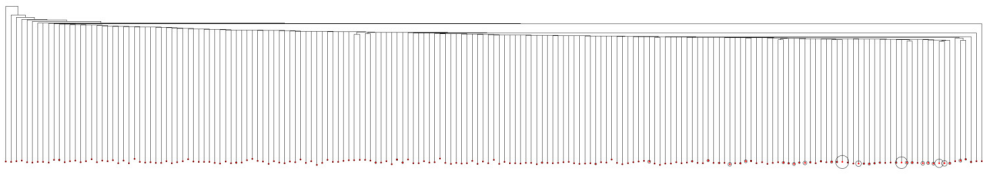


Figure 4.3: Dependency grammar not fully converging in discrete space: 1. energy barriers: the energy barrier convergence is very slow 2. extra local minima: multiple local minima in the discrete space correspond to the same energy basin in the continuous space.

4.2.4 Discretization of the hypothesis space

In order to address the issue of slow algorithm convergence, we discretize the parameter space and use Zhou's discrete version of the modified WL algorithm[Zho11]. Let Ω_r be the discretized parameter space with discretization $r > 4$:

$$\Omega_r = \{\vec{\theta} = [\theta_1, \ldots, \theta_k] | \theta_i = q/r \text{ for } q \in \{1, \ldots, r\} \text{ and } \sum_{j \in I_l} \theta_j = 1\}.$$

For example, for $r = 10$, and $n = 4$ nodes, the size of the discretized space is $|\Omega_r| \approx 10^{n^2+n+4n} = 10^{36}$.

In the discrete space, we perform coordinate descent (in lieu of gradient descent) to find the local minima. Given $\theta_t = [\theta_1, \ldots, \theta_k] \in \Omega$, let $\theta_t^{(i,j)} = [\theta_1, \ldots, \theta_i - \frac{1}{r}, \ldots, \theta_j + \frac{1}{r}, \ldots, \theta_k]$ for every ordered pair (i, j) in the same probability simplex $i, j \in I_a$. One coordinate descent step $s(\theta_t)$ is given by

$$s(\theta_t) = \text{argmin}_{(i,j)} \left(E\left(\theta_t^{(i,j)}\right) | i, j \in I_a \text{ for some } a \right) = \theta_{t+1}.$$

The coordinate descent algorithm terminates when $\theta_t = \theta_{t+1}$ for some t, indicating that θ_t is a local minimum in the discrete space. We set the proposal probability $Q(x_t, y)$ to be

$$\begin{aligned} Q(x_t, y) &= 0 \text{ if } |x_t - y|_{L1} \neq \frac{2}{r} \\ &= 1/Z \text{ if } |x_t - y|_{L1} = \frac{2}{r} \text{ and } x_t, y \text{ differ by two coordinates in the same probability simplex} \end{aligned}$$

for normalization constant $Z = |\{(i, j) \text{ st. } i, j \in I_a,\ a = 1, 2, \ldots\}|$.

When we attempt to run the naive implementation of the discretized algorithm, we find that (1) there are multiple discrete local minima found that belong to the same energy basin in the continuous space (2) the energies of a local minimum in the discrete space may be a poor approximations to the energy of the corresponding local minimum in the continuous space if the gradient is steep at the discrete local minimum. The resulting after 10^6 iterations is shown in Figure 4.2.3.

To achieve faster convergence and a better approximation to the continuous space, we employ a hybrid discrete-continuous approach. We run the main algorithm loop in discrete

($r = 10$) space with the following modifications. After each sample θ_t is accepted, we 1. perform coordinate descent in the discretized space initialized with θ_t to find local minimum θ_t^* 2. perform gradient descent in the continuous space initialized with θ_t^* to find the more accurate local minimum θ_t'. The use of the discrete space limits the number of local minima and the number of gradient descent computations and the subsequent use of the continuous space merges the discrete local minima belonging to the same continuous energy basin. To improve the energy boundary estimations, we iteratively perform ridge descent on the discrete mesh, then refine the discretization by a factor of 2 and repeat until convergence.

4.2.5 Curriculum Construction

We examine learning curriculums with the goal of increasing convergence speed to aid in cases with constrained computational resources (such as running time). We design the general curriculum as follows: the problem is split into curriculum stages, which are sub-problems of increasing complexity, with the last stage consisting of the full problem. Next we define the curriculum, which is the proportional allocation of computational resources to each stage. For example, the 'average' curriculum allocates equal computational resources between each stage, and the 'linearly increasing' curriculum allocates increasing resources to later stages as seen in Table 4.1.

Table 4.1: Curriculum resources allocation by stage

stage	1	2	3	4	5	6	7
none	0	0	0	0	0	0	1
average	1	1	1	1	1	1	1
linearly increasing	1	2	3	4	5	6	7
linearly decreasing	7	6	5	4	3	2	1
exponentially increasing	1	2	4	8	16	32	64
exponentially decreasing	64	32	16	8	4	2	1

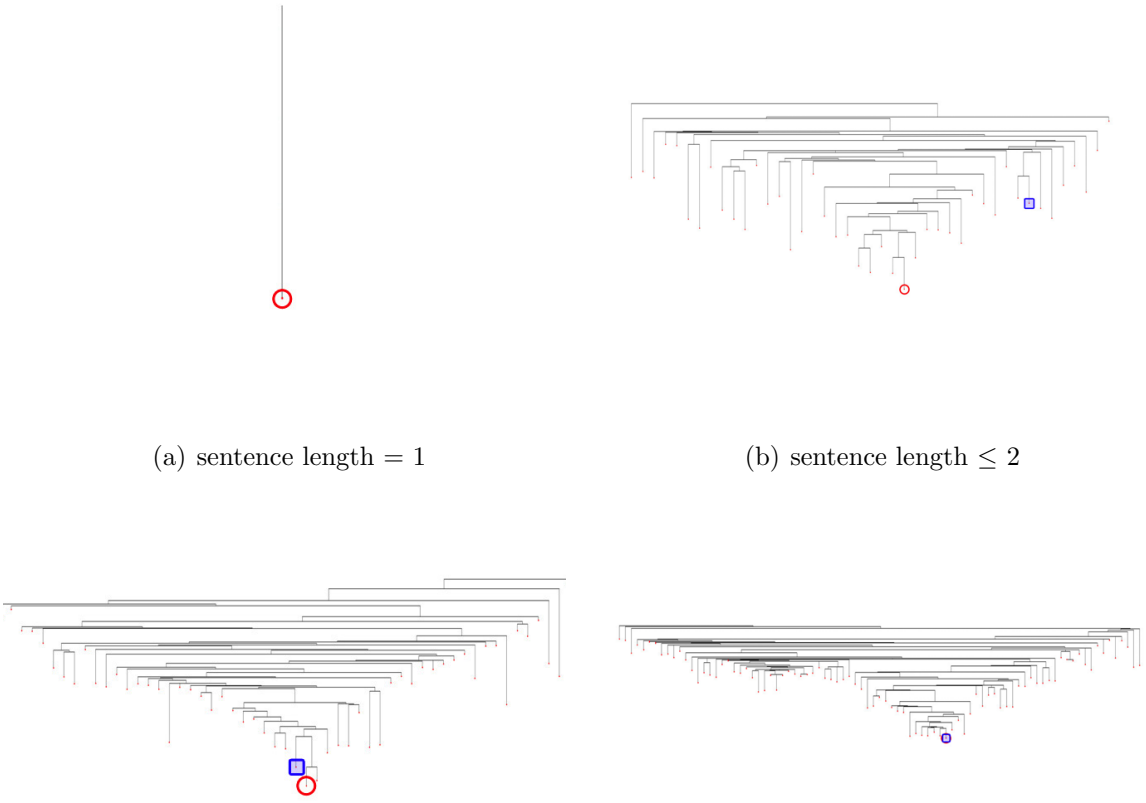
We first explore a curriculum based on sample sentence length. We construct a 3-node dependency grammar and discretize it using discretization factor $r = 10$. We call this

grammar θ_e. Next we sample $m = 200$ sentences $D = \{x_i\}$ from θ_e. We define $D_i \subset D$ to be the set of all sentences x_j containing i words or less. Let $w(x_j)$ be the word count of the sentence x_j, then $D_i = \{x_j | w(x_j) \leq i$. The sets D_i are nested ($D_i \subset D_{i+1}$) and $\cup_i^\infty D_i = D$. The first stage of the curriculum is trained on the set D_1, the second stage is trained on D_2 and the k-th stage is trained on D_k. Figures 4.2.5 (a-g) show the Energy Landscape maps of the curriculum stages 1 through 7.

Next we explore a curriculum based on the number of nodes n in the grammar. We synthesize a 5-node dependency grammar and its simplifications to $n = 4, 3, 2, 1$ nodes with discretization factor $r = 10$. We sample $m = 200$ sentences $D_j = \{x_i\}$ from each grammar $\theta_j, j = 1, \ldots, 5$. The first stage of the curriculum is trained on the set D_1, and the k-th stage is trained on D_k. Figures 4.2.5 (a-d) show the Energy Landscape maps of the curriculum stages 2 through 5. The ELM for Stage 1 is omitted because it is the same as the ELM in Figure 4.2.5 (a) due to the convexity of the landscape.

For both the curriculum based on the sentence length and the curriculum based on the number of nodes in the grammar, we observe that the Energy Landscape maps becomes more complex in the later stages of the curriculum; the landscapes in the later stages are flatter and have more local minima. In each figure, the closest local minimum to the global minimum of the previous curriculum stage is highlighted in blue. It is evident that for stages 3-7 of the curriculum based on sentence lengths (Figures 4.2.5 b-f) and stages 3-5 of the curriculum based on the number of nodes (Figures and 4.2.5 b-d), the global minimum from curriculum stage n is close to the global minimum of stage $n + 1$. This provides an explanation for the performance benefit of curriculum learning: early stages (which can be learned more easily) provide a good starting guess for later stages, which allows later stages to converge to a better local minimum, which also results in less computation time overall.

We tested curriculum learning with the second curriculum ($n = 1, \ldots, 5$ dependency grammar nodes) and a constrained total running time. Each successive stage of the curriculum is initialized with the output from the previous stage. We allot $s = 18,000$ seconds total running time, and assign each successive stage twice as much time as the previous stage, yielding the distribution: Stage 1 - 581s, Stage 2 - 1161s, Stage 3 - 2323s, Stage 4 - 4645s,



(a) sentence length = 1

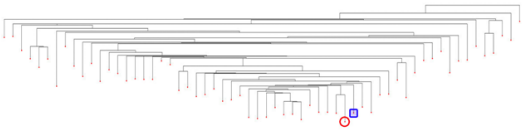
(b) sentence length ≤ 2

(c) sentence length ≤ 3

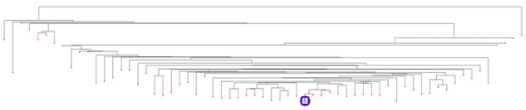
(d) sentence length ≤ 4

Figure 4.4: Curriculum based on training sample sentence length.

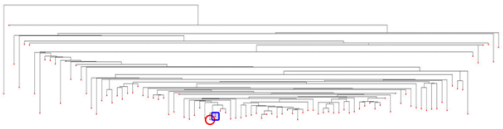
Stage 5 - 9290s. The exponentially increasing time in the curriculum design is chosen because the complexity of the later stages requires more time to converge. We run the Wang Landau algorithm $n = 1,000$ times with this curriculum and find the energy basins of the ELM of the complete 5-node dependency grammar that the learned models belong to. Hence we obtain a histogram of the learned models on the leaf nodes of the ELM as shown in Figure 4.9 (b). For comparison, Figure 4.9 (a) shows the histogram of the models learned without using the curriculum. The utilization of the curriculum results in more frequent convergence to the global minimum as well as energy basins near the global minimum.



(a) sentence length ≤ 5

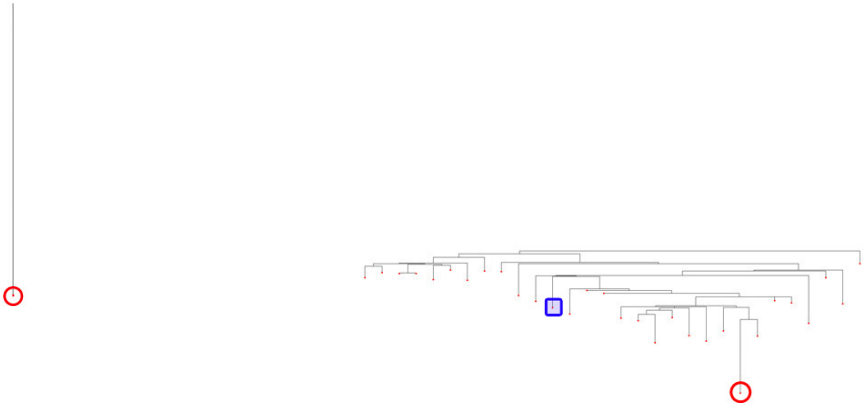


(b) sentence length ≤ 6



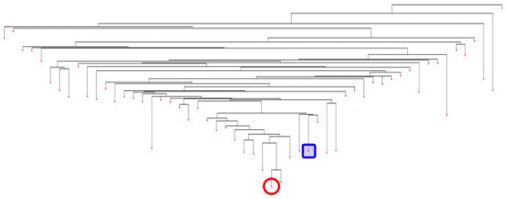
(c) sentence length ≤ 7

Figure 4.5: (cont.) Curriculum based on training sample sentence length.



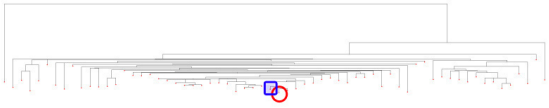
(a) 1 node

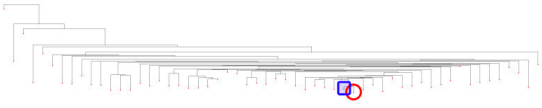
(b) 2 nodes



(c) 3 nodes

Figure 4.6: Curriculum based on number of nodes





(a) 4 nodes

(b) 5 nodes

Figure 4.7: (cont.) Curriculum based on number of nodes

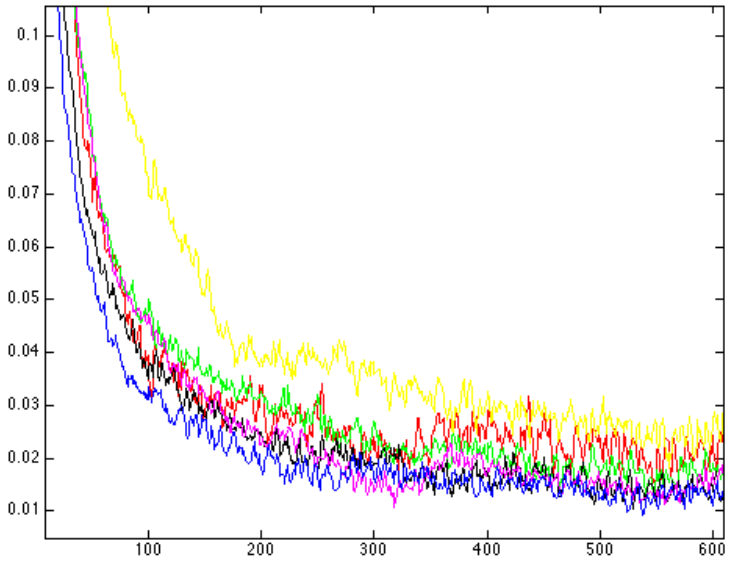
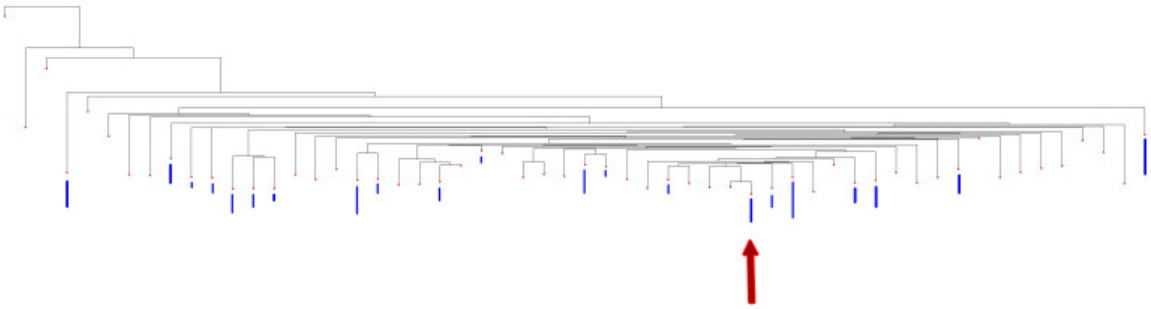
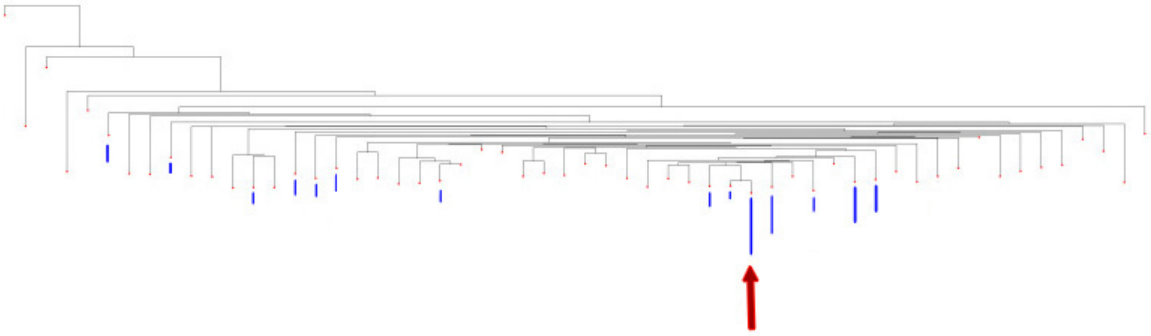


Figure 4.8: Curriculum convergence. Yellow - no curriculum, red - exponentially decreasing, green - linearly decreasing, pink - average, black - exponentially increasing, blue - exponentially increasing



(a) no curriculum



(b) exponentially decreasing curriculum

Figure 4.9: Distribution of learned grammars (a) without a learning curriculum (b) with the exponentially decreasing curriculum. The blue bars histogram the number of learned grammars belonging to each energy basin, the red arrow indicates the energy basin of the ground truth solution.

CHAPTER 5

PAC learning in the AOG space

In this chapter, we provide theoretical justifications for using the AOG model (rather than flat models) for object, scene and even representation.

While compositional models are widely used in Machine Learning and there is empirical evidence that compositional models need fewer training samples, there is no theoretical proof in the literature as to why and in what cases they are better than flat representations. To that end, we focus on the AOG graph (a type of compositional model) and provide PAC learning bounds for the number of training samples necessary to guarantee that the model is learned with high probability to a given degree of accuracy.

5.0.1 PAC Learning

Probably Approximately Correct (PAC) [Val84] learning concerns the ability of a probabilistic algorithm to learn a solution from a known target set such that with high confidence the error is small. To fully understand the concept for PAC learning, we must first introduce some terminology.

The instance space X is the set of all objects in the learner's world. In computer vision applications, it can for example be the set of all arrays of sketches associated with x,y coordinates in an image. A concept c over the instance space is a subset of the instance space $c \subseteq X$. For instance, it can be the set of all sketches on an image that form the shape of a cat face. Thus a concept can be thought of as a boolean function $c : X \to \{0, 1\}$ which identifies all instances that adhere to a given rule. That is, in the previous example, $c(x) = 1$ if x is a cat face and $c(x) = 0$ otherwise. A concept class C is a set of concepts over X.

In the PAC learning model, a learner is a function which takes some number of samples drawn from an unknown target concept c and outputs a hypothesis h in the hypothesis space H which is meant to approximate c. The hypothesis class is the space of concepts that the learner may output and is typically a equal to or a subset of C. The performance of the learner is judged based on how 'close' the hypothesis h is to the target concept. More formally, if D is a distribution over the instance space X, then the error of the learned concept h is defined to be

$$error(h) = \Pr_{x \sim D}[c(x) \neq h(x)].$$

We say that the concept class C is PAC learnable if the exists an algorithm L such that for every concept $c \in C$ and every distribution D on the instance space X, then for every ϵ and δ there exists an integer $N(\epsilon, \delta)$ such that if the earner L is given $N(\epsilon, \delta)$ independent samples from c then it will output a hypothesis h such that with probability greater than $1 - \delta$, it will satisfy $error(h) \leq \epsilon$. In other words, Probably ($p > 1 - \delta$), the algorithm will learn a hypothesis that is Approximately Correct ($error(h) \leq \epsilon$). If a hypothesis class is PAC learnable, then finding theoretical bounds on $N(\epsilon, \delta)$ gives us the maximum number of samples necessary to guarantee arbitrarily good performance for the learning algorithm.

In [BEH87], Blumer et al. prove that every finite hypothesis space is PAC learnable, and furthermore that the number of samples $N(\epsilon, \delta)$ necessary to learn a hypothesis within ϵ, δ accuracy is bounded by

$$N(\epsilon, \delta) \geq \frac{1}{\epsilon}(\ln |H| + \ln \frac{1}{\delta}).$$

We define the capacity $C(H)$ of a finite hypothesis space H to be $\ln(|H|)$. Therefore the number of samples is directly proportional to the capacity of the hypothesis space and is equal to

$$N(\epsilon, \delta) \geq \frac{1}{\epsilon}(C(H) + \ln \frac{1}{\delta}). \tag{5.1}$$

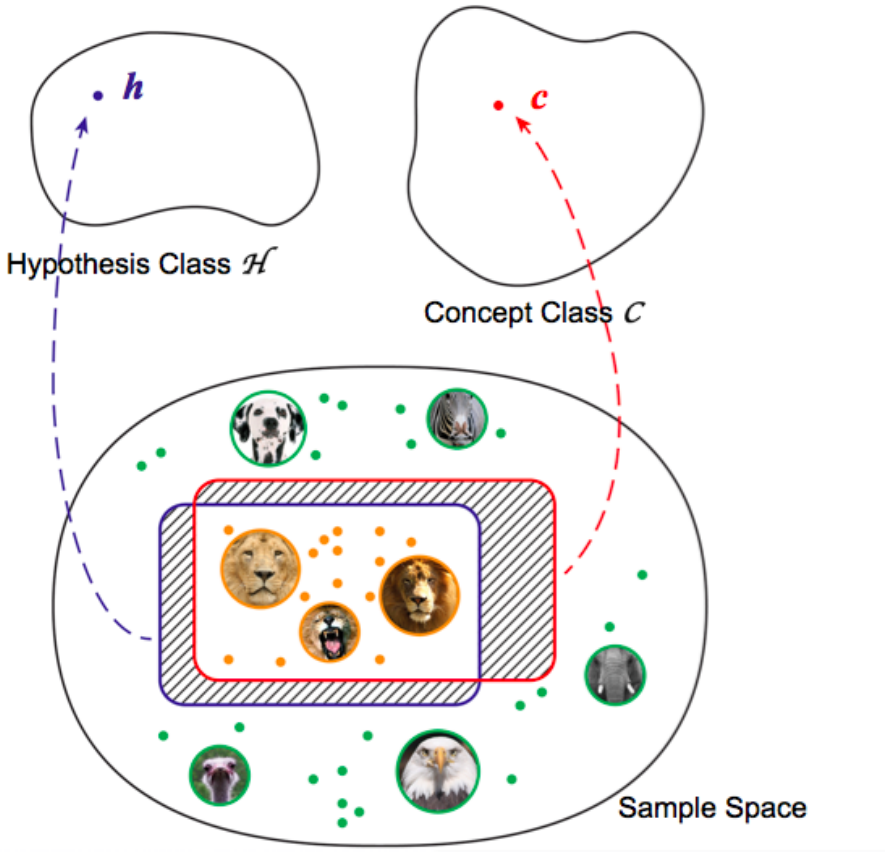


Figure 5.1: Visual representation of PAC learning concept definitions. Here the Sample space is the space of all images. The concept class C is the set of all animals images and the hypothesis class is the set of all sets of images. The target concept $c \in C$ is the function $c(x) = 1$ if x is an image of a lion and $c(x) = 0$ if x is not an image of a lion. The hypothesis h is a function learned by some algorithm L from an input set of images $\{x_i\}$ sampled iid from a distribution D over the sample space, and their labels $c(x_i)$. The error $error(h)$ is the integral over the distribution D of the shaded area in the image, that is $\int_{D(x)} c(x) \neq h(x)$.

5.0.2 AOG hypothesis space

We restrict our study to the case of the boolean AND-OR graph. The boolean AOG is defined on the instance space $X = \{0,1\}^n$ which represents binary reposes of imagine filters (such as Gabor filters) corresponding to the n terminal nodes in the AOG. More formally, the terminal nodes are boolean functions $t_i : I \to \{0,1\}, i = 1, \ldots, n$ evaluated on the image I. The AOG function $g : X \to \{0,1\}$ is evaluated recursively on the nodes of the graph; an AND node evaluates to 1 if and only if all of its children evaluate to 1, while an OR node evaluates to 1 if at least one of its children evaluates to 1.

Every AOG g defines an equivalence class in the AOG space because every permutation of child nodes for a given AND/OR node results in an equivalent AOG. Additionally, every AND/OR could be split into two nodes; however, we avoid this scenario by allowing AND nodes to have only OR node children and for OR nodes to only have AND/Terminal node children.

We characterize AOG spaces by the following parameters (as illustrated in Figure 5.0.2):

- AOG depth d: the number of AND/OR node layer pairs
- AND node branching number b_a: the maximum number of children of each AND node
- OR node branching number b_o: the maximum number of children of each OR node
- Terminal node number k: size of the set of all terminal nodes.

The size of the AOG space is determined by these parameters, so the capacitance of an AOG with depth d, branching numbers b_a, b_o and n terminal nodes is $C(d, b_a, b_o, k) = \ln |H(d, b_a, b_o, k)|$.

Theorem 1. *The capacity of the AOG space is bounded by* $C(d, b_a, b_o, n) \leq (b_a b_o)^d \ln(n)$.

Proof. (by induction on the depth d) For the $d = 1$ case, the root node is an OR node with at most b_o children. Each of the children are AND nodes with at most b_a children, which are

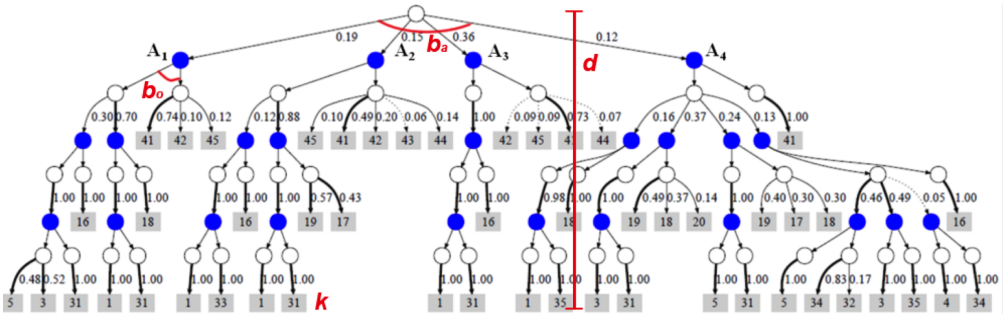


Figure 5.2: Four parameters (d, b_a, b_o, n) characterizing AOG spaces. Depth d: the number of AND/OR node layer pairs, branching number b_a: the maximum number of children of each AND node, branching number b_o: the maximum number of children of each OR node, terminal node number k: size of the set of all terminal nodes.

all terminal nodes. The terminal nodes have at most n variants, so there are at most $n^{b_a b_o}$ variants of the $d = 1$ AOG. Thus $|H(1, b_a, b_o, n)| \leq n^{b_a b_o}$ and therefore

$$C(1, b_a, b_o, n) = \ln |H(1, b_a, b_o, n)| \leq \ln(n^{b_a b_o}) = (b_a b_o)^1 \ln n.$$

Now suppose that $|H(d, b_a, b_o, n)| \leq \left(n^{b_a b_o}\right)^d$ for all $d \leq k$. Then for the $d = k + 1$ case, the top level OR node has at most b_o children and each child is an AND node with at most b_a children. The bottom level children can have at most $|H(k, b_a, b_o, n)| \leq n^{b_a b_o^{(k)}}$ variants, hence

$$|H(k+1, b_a, b_o, n)| \leq n^{(b_a b_o^{(k)})(b_a b_o)} = n^{b_a b_o^{(k+1)}}$$

and therefore

$$C(k+1, b_a, b_o, n) \leq b_a b_o^{k+1} \ln n.$$

It follows by induction on d that $C(d, b_a, b_o, n) \leq (b_a b_o)^d \ln(n)$ for all $d \geq 1$. □

The capacity bound can be further tightened by assuming part locality in the AOG model. In real-world applications, the children of a node represent the parts which make up the parent object and these parts are located in close spatial proximity of one another. Therefore, we can restrict the AOG space by allowing only terminal nodes within a given distance of one another to be children of the same node.

Theorem 1. *If the terminal nodes of a part can take on at most l values relative to the first terminal node, then we say the AOG has part locality factor l. For an AOG with locality factor l, the capacitance bound is*

$$C(d, b_a, b_o, k, l) \leq b_a^{d-1} b_o^d \ln(k l^{b_o - 1}).$$

Proof. (by induction on the depth d) For the base case where the depth $d = 1$, the first terminal child node of each AND node has k variants, and the remaining $b_a - 1$ child nodes have at most l variants. Since there are b_o AND nodes, the total number of variants is bounded by

$$H(1, b_a, b_o, k, l) \leq \left(k l^{b_a - 1}\right)^{b_o}.$$

Following the same inductive reasoning argument as outlined in the previous theorem, we obtain

$$C(d, b_a, b_o, k, l) = \ln |H(d, b_a, b_o, k, l)| \leq b_a^{d-1} b_o^d \ln(k l^{b_o - 1}).$$

□

We compare the capacity of bounded AOGs to the capacity of equivalent k-DNFs for which learning bounds are found in the literature. A 100-DNF has capacity on the order of $10^4 00$. An equivalent AOG with depth $d = 2$ and branching numbers $b_a = b_o = 5$ has capacity less than 5300. Furthermore, an AOG with part locality and the same parameters can have capacity of less than 2600. Given that the number of training samples necessary to approximately correctly learn the model is proportional to the capacity, it is clear that the AOG model provides a huge computational advantage over the traditional k-DNF model.

5.0.3 Supervised Learning Bounds

AOGs can be learned recursively from the bottom up in the supervised learning case where the graph structure is known and both the terminal nodes and the intermediate nodes have been manually annotated in the training images. Here we provide bounds on the number of training samples needed to PAC learn AND nodes and OR nodes of an AOG to a specific accuracy if all of the child nodes have been annotated in the training set.

Theorem 1. *Suppose the number of samples necessary to PAC learn nodes* $\{o_i\}_{i=1}^{b_a}$ *is* $n_i(\epsilon, \delta)$. *Then a bound on the number of samples necessary to learn their parent AND node is given by*

$$n(\epsilon, \delta) \leq \max_i n_i(\epsilon/b_a, \delta/b_a).$$

Proof. Let μ be a distribution over the instance space X defining a measure over the space. Let the concept C and the function $f : X \to \{0, 1\}$ correspond to the AND node we wish to learn and let the concepts C_i and the functions $f_i : X \to \{0, 1\}$ correspond to its child nodes. Then $f(x) = 1$ iff $f_i(x) = 1$ for all $i = 1, \ldots, b_a$. Because f is an AND node, the concept C is the intersection of all of its child concepts: $C = \cap_i C_i$. The error of the learned AND node is the measure of the symmetrical difference between the sets C and the set $X_f = \{x_i | f(x_i) = 1\}$:

$$\begin{aligned} error_\mu(C, f) &= \mu(X_f \setminus C) + \mu(C \setminus X_f) \\ &\leq \sum \mu(X_{f_i} \setminus C_i) + \sum \mu(C_i \setminus X_{f_i}) \\ &= \sum error_\mu(C_i, f_i). \end{aligned}$$

By assumption, we have a bound on the error of each child node is at most $error_\mu(C, f) \leq \epsilon/b_a$ with probability greater of equal to $1 - \delta/b_a$. Thus with probability greater than $(1 - \delta/b_a)^{b_a}$, the error of the parent node is bounded by

$$error_\mu(C, f) \leq \sum_i error_\mu(C_i, f_i) \leq \sum_i^{b_a} \epsilon/b_a = \epsilon.$$

By the Taylor expansion, $(1 - \delta/b_a)^{b_a} \geq 1 - \delta$, so with probability greater than $1 - \delta$, the error or the learned parent AND node is bounded by ϵ.

□

Theorem 1. *Suppose the number of samples necessary to PAC learn nodes* $\{a_i\}_{i=1}^{b_o}$ *is* $n_i(\epsilon, \delta)$. *Then a bound on the number of samples necessary to learn their parent OR node is given by*

$$n(\epsilon, \delta) \leq \sum_i n_i(\epsilon/b_a, \delta/b_a).$$

Proof. Let μ be a distribution over the instance space X defining a measure over the space. Let the concept C and the function $f : X \to \{0, 1\}$ correspond to the OR node we wish to learn and let the concepts C_i and the functions $f_i : X \to \{0, 1\}$ correspond to its child nodes. Then $f(x) = 1$ iff $f_i(x) = 1$ for all $i = 1, \ldots, b_o$. Because f is an OR node, the concept C is the union of all of its child concepts: $C = \cup_i C_i$. As in the previous theorem, the error of the learned OR node is the measure of the symmetrical difference between the sets C and the set $X_f = \{x_i | f(x_i) = 1\}$. Then by a similar argument, we get that with probability greater than $(1 - \delta/b_a)^{b_a} \geq 1 - \delta$, the target concept is learned with error less than ϵ. □

CHAPTER 6

Conclusion

Energy Landscape Maps are a powerful tool for analyzing high-dimensional non-convex problems. The ELMs allow us to visualize a target energy function by representing its energy basins and the energy barriers between them in a tree-like structure. This visualization reveals the number of local minima, the smoothness of the energy landscape, and the depth of the energy basins; which all together provide insight into the complexity of the learning problem. The ELMs are constructed using an advanced MCMC sampling method that dynamically re-weights the energy function to facilitate efficient traversal of the hypothesis space. The hypothesis space is partitioned into regions based on the energy wells of the energy function. The weights assigned to each region are updated proportionally to how frequently the sampler has visited those regions. The mechanic of selecting samples in less-visited regions with higher probability allows the sampler to frequently transit across energy barriers and efficiently traverse the hypothesis space. From the samples generated from this weighted random walk, we can discover local minima of the energy function, estimate the energy barriers between them, and construct the ELM.

By providing an intuitive visualization of energy functions, ELMs could help researchers gain new insight into the non-convex problems and facilitate the design and analysis of non-convex optimization algorithms. In Chapter 4, We demonstrate this on two classic machine learning problems: clustering with Gaussian mixture models and biclustering. We illustrate how the shape and complexity of ELM is influenced by (a) the input data generated from different ground-truth models, and (b) the hyperparameters of the energy function. We also visualize the behavior of a variety of learning algorithms using ELMs. In Chapter 5, we provide an example of ELMs being used to explain the effectiveness of curriculum learning

of dependency grammars. By plotting the ELM of each curriculum stage, we show that the effectiveness of a curriculum is a result of starting with a highly smoothed energy function and then gradually reducing the smoothness over the curriculum stages, as hypothesized by Bengio et al. [BLC09].

Bibliography

[AL10] Yves F. Atchade and Jun S. Liu. "The Wang-Landau algorithm for Monte Carlo computation in general state spaces." *Statistica Sinica*, **20**:209–233, 2010.

[BEH87] Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. "Occam's razor." *Inf. Process. Lett.*, **24**(6):377–380, April 1987.

[BK97] Oren M. Becker and Martin Karplus. "The topology of multidimensional potential energy surfaces: Theory and application to peptide structure and kinetics." *The Journal of Chemical Physics*, **106**(4):1495–1517, 1997.

[BLC09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning." In *ICML*, p. 6, 2009.

[BM98] C L Blake and C J Merz. "UCI repository of machine learning databases.", 1998. Robustness of maximum boxes.

[BZ05] Adrian Barbu and Song-Chun Zhu. "Generalizing swendsen-wang to sampling arbitrary posterior probabilities." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**:1239–1253, 2005.

[CC00] Yizong Cheng and George M. Church. "Biclustering of Expression Data." In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00)*, pp. 93–103, Menlo Park, CA, August 16–23 2000. AAAI Press.

[CDG04] Hyuk Cho, Inderjit S. Dhillon, Yuqiang Guan, and Suvrit Sra. "Minimum Sum-Squared Residue Co-Clustering of Gene Expression Data." In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*, 2004.

[Cha01] Eugene Charniak. "Immediate-head parsing for language models." In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pp. 124–131. Association for Computational Linguistics, 2001.

[Col99] Michael Collins. *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania, 1999.

[DS00] Sanjoy Dasgupta and Leonard J. Schulman. "A two-round variant of EM for Gaussian mixtures." In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, UAI'00, pp. 152–159, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Elm93] Jeffrey L. Elman. "Learning and development in neural networks: The importance of starting small." *Cognition*, **48**:71–99, 1993.

[Gew92] John Geweke. "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments." In *IN BAYESIAN STATISTICS*, pp. 169–193. University Press, 1992.

[GLD00] G. Getz, E. Levine, and E. Domany. "Coupled two-way clustering analysis of gene microarray data." *Proc Natl Acad Sci U S A*, **97**(22):12079–12084, October 2000.

[GR92] Andrew Gelman and Donald B. Rubin. "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**(4):457–472, 1992.

[HJM09] William P. Headden, III, Mark Johnson, and David McClosky. "Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing." In *HLT-NAACL*, pp. 101–109, 2009.

[KM04] Dan Klein and Christopher D. Manning. "Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency." In *Proceedings of ACL*, 2004.

[KMN09] Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2009.

[Lia05a] Faming Liang. "A generalized Wang-Landau algorithm for Monte Carlo computation." *JASA. Journal of the American Statistical Association*, **100**(472):1311–1327, 2005.

[Lia05b] Faming Liang. "A Generalized WangLandau Algorithm for Monte Carlo Computation." *Journal of the American Statistical Association*, **100**(472):1311–1327, 2005.

[LLC07] Faming Liang, Chuanhai Liu, and Raymond J. Carroll. "Stochastic approximation in Monte Carlo computation." *Journal of the American Statistical Association*, **102**:305–320, 2007.

[Mel88] Igor Aleksandrovič Melčuk. *Dependency syntax: theory and practice*. State University of New York Press, 1988.

[MO04] S. C. Madeira and A. L. Oliviera. "Biclustering algorithms for biological data analysis: a survey." *IEEE Trans. on Comp. Biol. and Bioinformatics*, **1**(1):24–45, 2004.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[PJZ11] Mingtao Pei, Yunde Jia, and Song-Chun Zhu. "Parsing video events with goal inference and intent prediction." In *ICCV*, 2011.

[Pot52] Renfrey B. Potts. "Some Generalized Order-Disorder Transformation." In *Transformations, Proceedings of the Cambridge Philosophical Society*, volume 48, pp. 106–109, 1952.

[RP99] D. Rohde and D. Plaut. "Language acquisition in the absence of explicit negative evidence: How important is starting small?" *Cognition*, **72**:67–109, 1999.

[SAJ10] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. "From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing." In *NAACL*, 2010.

[SCR12] Rajhans Samdani, Ming-Wei Chang, and Dan Roth. "Unified expectation maximization." In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 688–698. Association for Computational Linguistics, 2012.

[SW87] R. H. Swendsen and J. S. Wang. "Nonuniversal critical dynamics in Monte Carlo simulations." *Physical Review Letters*, **58**(2):86–88, January 1987.

[TH08] Kewei Tu and Vasant Honavar. "Unsupervised Learning of Probabilistic Context-Free Grammar Using Iterative Biclustering." In *Proceedings of 9th International Colloquium on Grammatical Inference (ICGI 2008), LNCS 5278*, September 2008.

[TH11] Kewei Tu and Vasant Honavar. "On the Utility of Curricula in Unsupervised Learning of Probabilistic Grammars." In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 2011.

[TH12] Kewei Tu and Vasant Honavar. "Unambiguity Regularization for Unsupervised Learning of Probabilistic Grammars." In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL 2012)*, 2012.

[TW87] Martin A Tanner and Wing Hung Wong. "The calculation of posterior distributions by data augmentation." *Journal of the American statistical Association*, **82**(398):528–540, 1987.

[Val84] L. G. Valiant. "A theory of the learnable." In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 436–445, New York, NY, USA, 1984. ACM.

[WMW98] David J. Wales, Mark A. Miller, and Tiffany R. Walsh. "Archetypal energy landscapes." *Nature*, **394**(6695):758–760, August 1998.

[YWW02] Jiong Yang, Wei Wang, Haixun Wang, and Philip S. Yu. "δ-Clusters: Capturing Subspace Correlation in a Large Data Set." In *ICDE*, pp. 517–528. IEEE Computer Society, 2002.

[Zho11] Qing Zhou. "Random Walk over Basins of Attraction to Construct Ising Energy Landscapes." *Phys. Rev. Lett.*, **106**:180602, May 2011.

[ZM06] Song-Chun Zhu and David Mumford. "A stochastic grammar of images." *Found. Trends. Comput. Graph. Vis.*, **2**(4):259–362, 2006.