UNSUPERVISED LEARNING OF FINITE MIXTURE MODELS WITH DETERMINISTIC ANNEALING FOR LARGE-SCALE DATA ANALYSIS

JONG YOUL CHOI

INDIANA UNIVERSITY
BLOOMINGTON

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the School of Informatics and Computing
Indiana University
January 2012

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Geoffrey Fox, Ph.D.

Paul Purdom, Ph.D.

David Wild, Ph.D.

Sun Kim, Ph.D.

January 12, 2012

WEBSITE:

http://www.cs.indiana.edu/~jychoi/

E-MAIL:

jychoi@cs.indiana.edu

DEDICATION

This thesis is dedicated to my lovely wife, Hyejong Jang, who always endures everything with me and gives endless love and encouragement

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Geoffrey Fox, for his support and guidance of my research and his endless patience. Without his help, it would be impossible to write this thesis. I am extremely fortunate to have him as my advisor.

I also would like to thank my research committee: Professor Paul Purdom, Professor Sun Kim, and Professor David Wild, for valuable discussions and insightful comments.

I am very grateful to have been a Ph.D. student in the School of Informatics and Computing where I could enjoy intellectual conversations and interactions with bright many professors and researchers.

My sincere thanks also goes to Professor Judy Qiu for her help and guidance on projects I worked with and new challenges that inspired me. I am indebted to my previous advisor, Dr. Markus Jakobsson, who gave me a great opportunity to start my Ph.D. study and provided countless help and advice.

I would like to thank my friend, Youngsang Shin, who didn't hesitate to give invaluable advice and immense help on my researches and life in Bloomington. I also thank to my fellow, Seung-Hee Bae, a fantastic collaborator and a lab mate to discuss many things on research. I wish to express my thanks to my lab fellows Thilina Gunarathne, Yang Ruan, Saliya Ekanayake, Hui Li, Tak-Lon Wu, Yuduo Zhou, and Bingjing Zhang for their collaboration and comments for my research.

I would like to give my special thank to my wife, Hyejong Jang, for everything during my study. My sincere thanks goes to my parents, who gave everything and support me always. Last but not the least, I would like to extend my thanks to my parents-in-law for generous support and encouragement.

ABSTRACT

The finite mixture model forms one of the most fundamental foundations in the fields of statistical pattern recognition, data mining, and machine learning to access the essential structure of observed random sample data. It aims at building a probabilistic generative model by which one can virtually reproduce the observed sample data from a mixture of a finite number of probabilistic distributions called *latent* components or sources. The finite mixture model provides a flexible and convenient way to explain a wide variety of random phenomena of observed sample data in a generative process of mixing finite random sources.

One of the main challenges in the finite mixture model is to search an optimal model parameter set from a large parameter space. The standard method used to fit a finite mixture model is the Expectation-Maximization (EM) algorithm. However, the EM algorithm for finite mixture model has a serious drawback; it can find only local optimum solutions and thus the quality of answer can be heavily affected by initial conditions and vary. Another important problem is the overfitting problem showing poor predicting performance on unseen data.

We have observed that a global optimization heuristic, known as Deterministic Annealing (DA), can outperform the traditional EM algorithm for parameter fitting in certain types of mixture models and provide an overfitting avoidance property. The DA algorithm, developed by K. Rose and G. Fox, has been proven its success in avoiding the local optimum problem and widely used in solving many data mining algorithms. Although

many researches have been performed on both theoretic perspectives and clustering applications, the use of the DA, however, has not been widely reported in many real data mining applications, despite of its superior quality and additional functions, such as learning hierarchical structures of data and overfitting avoidance. This is the main motivation in this work: applying the DA algorithm to finite mixture models and developing new algorithms and functions.

More specifically, in this thesis, we focus two well-known data mining algorithms which are based on the finite mixture model: i) Generative Topographic Mapping (GTM) for dimension reduction and data visualization, and ii) Probabilistic Latent Semantic Analysis (PLSA) for text mining. Those two algorithms have been widely used in the fields of data visualization and text mining, but still suffer from the local optimum problem due to the use of the EM algorithm in their original developments. We extend those EM-based algorithms by using the DA algorithm to improve their qualities in parameter estimation and overfitting avoidance.

LIST OF ACRONYMS

DA	Deterministic Annealing
DA-GTM	Generative Topographic Mapping with Deterministic Annealing
DA-PLSA	Probabilistic Latent Semantic Analysis with Deterministic Annealing
EM	Expectation-Maximization
EM-GTM	Generative Topographic Mapping with Expectation-Maximization
EM-PLSA	Probabilistic Latent Semantic Analysis with Expectation-Maximization
FMM-1	Finite Mixture Model Type-1
FMM-2	Finite Mixture Model Type-2
GTM	Generative Topographic Mapping
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
MLE	Maximum Likelihood Estimation
PLSA	Probabilistic Latent Semantic Analysis
SVD	Singular Value Decomposition

CONTENTS

1 INTRODUCTION

Finite mixture modeling forms one of the most fundamental foundations in the fields of statistical pattern recognition, data mining, and machine learning to access the essential structures of observed random sample data.

It aims at building a probabilistic model in which a random sample is drawn from a mixture of a finite number of probabilistic distributions called *components* or *sources* [24, 36]. The idea is that those components (or sources) can be used to abstract or summarize the random sample data. However, in general, since no direct information of components is available initially from a given random sample data, we call such components in a mixture model as *hidden* or *latent* components and the main task in building a finite mixture model is to uncover such hidden (or latent) components.

Since seeking hidden components or sources to fit observed sample data the best, a finite mixture model is also called as a *latent class model* or a *topic model* (especially in text mining area). The process of learning can fall under the category of *unsupervised learning* in that finding latent components is solely based on observed sample data with no use of any external information. Also, due to its random sample generating capability, a finite mixture model is known as a *generative* model which can randomly generate observable data.

The finite mixture model provides a flexible and convenient way to explain a wide variety of random phenomena of observed sample data as a generative process of mixing finite user-defined random sources [13, 24].

Due to its usefulness to provide a flexible and powerful tool of modeling complex observed data, the finite mixture model has been continued to receive increasing attention over years, from both a practical and a theoretical point of view [13, 24] and applied in broad range of areas involving statistical modeling, such as clustering [18, 24], text mining [5, 15], image processing [35], speech recognition [27], to name a few.

One of the main challenges in the finite mixture model is to search an optimal model parameter set to fit observed sample data, called *mixture model fitting*, from a large parameter space. In general, the mixture model fitting is known as a NP-hard problem [1]. The standard method used to fit a finite mixture model is the Expectation-Maximization (EM) algorithm [12, 13, 24]. However, the EM algorithm for finite mixture model has one serious drawback; it can find only local optimum solutions, not global solutions, and thus the quality of the answer can be largely affected by initial conditions. We have observed that a novel global optimization heuristic, called Deterministic Annealing (DA), can outperform the traditional EM algorithm for searching optimal parameters in certain types of mixture model fitting problems.

The DA algorithm, pioneered by K. Rose and G. Fox [28–31], has been proven its success in avoiding local optimum problem and widely used in solving many data mining algorithms [17, 23, 32, 37]. Although many researches have been performed on both theoretic perspectives [28, 37] and clustering applications [17, 23, 32, 38], the use of the DA, however, has not been widely reported in many real data mining applications, despite of its superior quality and overfitting avoidance with a systematic approach. This is the main topic we study in this work: applying the DA algorithm to solve the finite mixture model problem and developing new algorithms.

More specifically, in this thesis, we focus two well-known data mining algorithms which are based on the finite mixture model: i) Generative Topographic Mapping (GTM) for

dimension reduction and data visualization, and ii) Probabilistic Latent Semantic Analysis (PLSA) for text mining. Those two algorithms have been widely used in the fields of data visualization and text mining, but still suffer from the local optimum problem due to the use of the EM algorithm in their original developments. Although a DA-like approach has been discussed in [15], the proposed solution is different from the point of view of the traditional DA algorithm proposed by K. Rose and G. Fox [28–31]. We extend those EM-based algorithms by using the DA algorithm to improve their qualities in parameter estimation and overfitting avoidance.

Overfitting is often referred in a supervised learning setting to describe a problem that a model looses its generality and thus shows large performance differences between a training set and a validation set. In this thesis, we use overfitting in unsupervised learning, where we do not have a managed testing set, in order to refer a model with poor predictive performance for unseen data.

Our contributions in this thesis are summarized as follows:

i) Propose a generalized approach to solve the finite mixture model problem by using a novel optimization algorithm, called DA, to guard against the local optimum problem and help to achieve global optimum solutions.

ii) Develop a DA-based algorithm for GTM, named DA-GTM.

iii) Present the first and second order differential equations of the new objective function of DA-GTM for completing algorithm in deciding starting parameters.

iv) Propose a new fast and stable convergence scheme for DA-GTM.

v) Develop a DA-based algorithm for PLSA, named DA-PLSA.

vi) Provide the first and second order differential equations of the new objective function of DA-PLSA to determine an initial condition.

vii) Present experimental results of our DA-GTM and DA-PLSA, compared with the traditional EM-based algorithms.

1.1 THESIS ORGANIZATION

The rest of this thesis is orgranized as follows:

- In Chapter 2, we give a broad overview of the finite mixture model and its EM algorithm as the standard model fitting and parameter estimation method. Especially we define two finite mixture models we focus in this thesis. We also review the DA algorithm for the optimal finite mixture model fitting.
- In Chapter 3, we present a DA algorithm for GTM, named DA-GTM, and demonstrate the performance results compared with the original GTM which uses an EM method.
- In Chapter 4, we demonstrate how the PLSA problem can be solved by taking a DA approach and present a new algorithm, named DA-PLSA, which is stemmed from the original PLSA which utilizes an EM optimization.
- Lastly, we discuss the contributions of this thesis and future work.

1.2 BIBLIOGRAPHIC NOTES

The work presented in this thesis is solely the outcome of my own research and includes none of any work in collaboration. Most of the GTM related work in this thesis have been presented and published as conference papers [8–10] and a journal paper [7].

1.3 NOTATION AND CONVENTIONS

In this thesis, we use a normal typeface to indicate scalar values, e.g., σ and β, while using bold typeface for vectors and matrices. To distinguish vectors and matrices, we use a lower case symbol for vectors, e.g., $\mathbf{x}$, $\mathbf{y}$, and an upper case symbol for matrices, e.g., $\mathbf{X}$, $\mathbf{Y}$. We also use an upper case letter for constants without a bold typeface, e.g., N, D. However, exceptions to this convention do appear.

We organize data by using vectors and matrices. We let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ denote an observed or random sample data of size N, where $\mathbf{x}_i$ is a D-dimensional random row vector ($1 \leqslant i \leqslant N$). We organize N-turple sample data into a $N \times D$ matrix denoted by $\mathbf{X} = (\mathbf{x}_1^{Tr}, \ldots, \mathbf{x}_N^{Tr})^{Tr}$ such that row i of $\mathbf{X}$ contains i-th sample data $\mathbf{x}_n$ where Tr represents a transpose. To access each element in a matrix, we use subscripts such that x_{ij} is an (i, j) element of $\mathbf{X}$. Similarly, we let $\mathbf{y}_1, \ldots, \mathbf{y}_K$ denote component data or latent data of size K, where $\mathbf{y}_k$ is a D-dimensional vector ($1 \leqslant k \leqslant K$). We also organize such K-tuple data set into $K \times D$ matrix denoted by $\mathbf{Y} = (\mathbf{y}_1^{Tr}, \ldots, \mathbf{y}_K^{Tr})^{Tr}$ so that k-th latent vector $\mathbf{y}_k$ equals row k. Also, y_{kj} denotes an (k, j) element of $\mathbf{Y}$.

We use vectors for an array of scalar values. For example, we let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ denote mixing proportions or weights where each scalar quantity $\pi_k (1 \leqslant k \leqslant K)$ holds $0 \leqslant \pi_k \leqslant 1$ and in total $\sum_{k=1}^{K} \pi_k = 1$.

We let $|\cdot|$ and $\|\cdot\|$ denote L_1-norm and L_2-norm, respectively, to represent size of vectors or distances between two vectors. For example,

$$|\mathbf{x}| = \sum_{i=1}^{D} |x_i| \tag{1.1}$$

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2} \tag{1.2}$$

2 FINITE MIXTURE MODELS AND DETERMINISTIC ANNEALING

In this chapter, we introduce finite mixture models and the Deterministic Annealing (DA) algorithm.

2.1 FINITE MIXTURE MODELS

In the finite mixture model, we model the probability distribution of observed sample data as a mixture distribution of finite number of components in a way in which each sample is independently drawn from a mixture distribution of *latent* or *hidden* components of size K with mixing weights. In machine learning, this modeling process falls under the category of unsupervised learning as we aim at finding a model and its parameters solely from the given sample data without using any external information. In general, we can assume any form of distributions as a latent component but in practice we use one of well-defined conventional continuous or discrete distributions, such as Gaussian, Poisson, multinomial, and so on.

Formally, in the finite mixture model, we model the probability distribution of the i-th (multivariate) sample data $\mathbf{x}_i$ as a mixture distribution of K components and de-

fine the probability of $\mathbf{x}_i$ by a conditional probability with a mixing weight vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ and component-specific parameters $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K\}$ as follows:

$$P(\mathbf{x}_i \mid \boldsymbol{\Omega}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k P(\mathbf{x}_i \mid \boldsymbol{\omega}_k) \tag{2.1}$$

where the parameter set $\boldsymbol{\Omega}$ represents a general component-specific parameter set; it can be parameters for latent cluster centers or distribution parameters for components, and $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}$ denotes mixing weights constrained by $\sum_k \pi_k = 1$ for all k and each element is bounded by $0 \leqslant \pi_k \leqslant 1$.

In general, the mixing weights $\pi_1, \ldots, \pi_K$ are system-wide parameters in that all sample data will share the same mixing weights. This model, often called as a mixture of unigrams [5], is the traditional finite model widely used in the most algorithms, including density estimation and clustering, where K components are closely related to the centers of clusters [18]. This is also the model used in GTM [4, 10].

Relaxing the condition constrained on the mixing weights, we can further extend the previous model to build a more flexible model, in a way in which each sample has its own mixing weights rather than system-wide shared weights used in the previous definition. This relaxed version of the finite mixture model can be defined by

$$P(\mathbf{x}_i \mid \boldsymbol{\Omega}, \boldsymbol{\Psi}) = \sum_{k=1}^{K} \psi_{ik} P(\mathbf{x}_i \mid \boldsymbol{\omega}_k) \tag{2.2}$$

where $\boldsymbol{\Omega}$ represents a general parameter set as defined above and a new mixing weight $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N\}$ represents a set of N weight vectors of size K, so that each weight vector $\boldsymbol{\psi}_i$ represents K mixing weights $\boldsymbol{\psi}_i = (\psi_{i1}, \ldots, \psi_{iK})$ corresponding to the i-th sample data $\mathbf{x}_i$ and is constrained by $\sum_{k=1}^{K} \psi_{ik} = 1$ and $0 \leqslant \psi_{ik} \leqslant 1$. This is the model we will use for PLSA [15, 16]. More details will be discussed in Chapter 4.

In this thesis, we focus those two mixture models defined in Eq. (2.1) and Eq. (2.2). Hereafter we call those two finite mixture models as Finite Mixture Model Type-1 (FMM-1) and Finite Mixture Model Type-2 (FMM-2) respectively.

2.1.1 Expectation Maximization Algorithm

In analyzing random sample data with finite mixture models, we seek a set of mixture model parameters so that a model can optimally fit a given sample data set. In statistics, this process is called *model fitting* or *parameter estimation*. One of the most well known estimators to measure the goodness of fitting or assess the quality of parameters is called a Maximum Likelihood Estimation (MLE). In the MLE framework, we search a set of parameters which maximizes likelihood, or equivalently log of likelihood known as *log-likelihood*, of a given sample data set.

By using MLE in the finite mixture models defined above, our goal can be stated as follows; Given that the sample data is independent and identically distributed (i.i.d), the likelihood of the sample data is

$$P(\mathbf{X} \mid \boldsymbol{\Omega}) = \prod_{i=1}^{N} P(\boldsymbol{x}_i \mid \boldsymbol{\Omega}) \tag{2.3}$$

and we seek parameters which maximize the following log-likelihood $\mathcal{L}$, defined by,

$$\begin{aligned} \mathcal{L} &= \log P(\mathbf{X} \mid \boldsymbol{\Omega}) && (2.4) \\ &= \sum_{i=1}^{N} \log P(\boldsymbol{x}_i \mid \boldsymbol{\Omega}) && (2.5) \end{aligned}$$

However, finding optimal parameters, i.e., *model fitting*, by using MLE in finite mixture models is in general intractable except the most trivial cases. Traditionally in finite mixture models, an iterative optimization method, called EM algorithm developed by Dempster et al. [12], has been widely used for model fitting.

The EM algorithm searches for a solution by iteratively refining local solutions by taking two consecutive steps per iteration: Expectation step (E-step) and Maximization step (M-step). The high-level description of the EM steps for the finite mixture models can be summarized as follows.

- E-step : we evaluate an expectation denoted by r_{ki} which is the conditional association probability of the k-th component related to the i-th sample data, defined by

$$r_{ki} = P(k \mid i) \tag{2.6}$$
$$= \frac{P(\mathbf{x}_i \mid \boldsymbol{\omega}_k)}{\sum_{k'} P(\mathbf{x}_i \mid \boldsymbol{\omega}_{k'})} \tag{2.7}$$

  where

$$\sum_{k=1}^{K} r_{ki} = 1 \tag{2.8}$$

  The value r_{ki} is called in many different ways; responsibility, membership probability, and association probability. Basically, it represents how likely sample $\mathbf{x}_i$ can be generated by component $\boldsymbol{\omega}_k$.

- M-step : using the values computed in E-step, we find parameters which will locally maximize the log-likelihood $\mathcal{L}^*$, defined by,

$$\mathcal{L}^* = \underset{\mathbf{\Omega}}{\operatorname{argmax}}\, \mathcal{L} \tag{2.9}$$

$$= \underset{\mathbf{\Omega}}{\operatorname{argmax}} \sum_{i=1}^{N} \log P(\mathbf{x}_i \,|\, \mathbf{\Omega}) \tag{2.10}$$

  To determine such parameters, we use the first derivative test; i.e., we compute the first-order derivatives of $\mathcal{L}$ with respect to each parameter and test if they become zero, such that $\partial\mathcal{L}/\partial\boldsymbol{\omega}_k = 0$. This requires exact knowledge of probability distributions of components. Thus, details of M-step may vary from different models. The M-step of our focus algorithms, GTM and PLSA, will be discussed in Section 3.1 and Section 4.1 respectively.

Although the EM algorithm has been widely used in many optimization problems including the finite mixture models we are discussing, it has been shown a severe limitation, known as the local optima problem [37], in which the EM method can be easily trapped in local optima, failing to find a global optimum, and so the outputs are very sensitive to initial conditions. The problem can be worse if we need accurate solutions for, such as, density estimation or visualization in scientific data analysis. This may also cause poor quality of query results in text mining.

To overcome such problem occurred in the finite mixture model problems with EM, including our main focus algorithms GTM [3, 4] and PLSA [15, 16], we apply a novel optimization method, called DA [28], to avoid local optimum and seek robust solutions against poor initial conditions. We will discuss more details of DA in the next.

2.2 DETERMINISTIC ANNEALING

The DA algorithm [28–31] has been successfully applied to solve many optimization problems in various machine learning algorithms and applied in many problems, such as clustering [17, 28, 38], visualization [23], protein alignment [6], and so on. The core capability of the DA algorithm is to avoid local optimum and pursue a global optimum solution in a deterministic way [28], which contrasts to stochastic methods used in the simulated annealing [22], by controlling the level of randomness or smoothness. The DA algorithm, adapted from a physical process known as annealing, finds a solution in a way in which an optimal solution is gradually revealed as lowering a numeric *temperature* which controls randomness or smoothness.

At each level of temperature, the DA algorithm chooses an optimal state by following the principle of maximum entropy [19–21], developed by E. T. Jaynes, a rational approach to choose the most unbiased and non-committal answer for a given condition. In short, the principle of maximum entropy, which is a heuristic approach to be used to choose an answer with constrained information, states that if we choose an answer with the largest entropy when other information is unknown, we will have the most unbiased and non-committal answer.

To find a solution with the largest entropy, the DA algorithm introduces a new objective function $\mathcal{F}$, called *free energy*, an analogy to the Helmholtz free energy in statistical physics, defined by

$$\mathcal{F} \quad = \quad \langle \mathcal{D} \rangle - T \, \mathcal{S} \tag{2.11}$$

where $\langle \mathcal{D} \rangle$ represents an expected cost, T is a Lagrange multiplier, also known as a *numeric temperature*, and $\mathcal{S}$ is an entropy.

It is known that minimization of the free energy $\mathcal{F}$ is achieved when the association probabilities defined in Eq. (2.6) forms a Gibbs distribution, such as

$$P(k|i) = \frac{\exp(-d(i,k)/T)}{Z_i} \tag{2.12}$$

where $d(k,i)$ represents an association cost between ω_k and x_i, also called distortion, and Z_i is a normalization function, also known as partition function in statistical physics. With Eq. (2.12), we can restate the free energy defined by Eq. (2.11) as

$$\mathcal{F} = -T\sum_{i=1}^{N}\log Z_i \tag{2.13}$$

where the partition function Z_i is defined by

$$Z_i = \sum_{k=1}^{K}\exp\left(\frac{-d(i,k)}{T}\right) \tag{2.14}$$

In the DA algorithm, we choose at each level of temperatures an answer which minimizes the free energy [28]. A standard method of parameter estimation, also known as model fitting, in finite mixture models is the EM algorithm, which suffers from the local optimum problem characterized by high-variance answers with different random starts. To overcome such problem, we use the DA algorithm which is robust against the random initialization problem and shows a proven ability to avoid local optimum for finding global optimum solutions.

With the DA algorithm, the traditional objective function based on MLE in the finite mixture models will be replaced to use the following new objective function based on the free energy estimation:

$$\mathcal{F}^* = \underset{\Omega}{\operatorname{argmin}}\, \mathcal{F} \tag{2.15}$$

In this thesis, we focus on developing new DA objective functions for GTM and PLSA based on the finite mixture models, FMM-1 and FMM-2, defined as Eq. (2.1) and Eq. (2.2) respectively. By using Eq. (2.13), we propose a general free energy function as follows:

$$\mathcal{F}_{\mathrm{FMM}} = -T \sum_{i=1}^{N} \log \sum_{k=1}^{K} \left\{ c(i,k) P(\mathbf{x}_i \mid \boldsymbol{\omega}_k) \right\}^{1/T} \tag{2.16}$$

where $c(i,k)$ represents a weight coefficient related with a conditional probability of data $\mathbf{x}_i$ given component $\boldsymbol{\omega}_k$, $P(\mathbf{x}_i \mid \boldsymbol{\omega}_k)$.

Details will be discussed in Section 3.2 and Section 4.2 respectively.

2.2.1 Phase Transition

As one of the characteristic behaviors of the DA algorithm, the free energy estimation undergoes an irregular sequence of rapid changes of state, called *phase transitions*, when we are lowering the numeric temperature [28–30]. As a result, at some ranges of temperatures we cannot obtain all distinctive solutions but, instead, we only obtain a limited number of effective solutions [28, 39]. For an example, in the DA clustering algorithm proposed by K. Rose and G. Fox [28, 30], we can see only one effective cluster at a high temperature and observe unique clusters gradually pop out subsequently as the temperature is getting lowered.

Thus, in DA, solutions will be revealed by degrees as the annealing process proceeds, starting with a high temperature and ending in a low temperature. In other words, as we do annealing (i.e., lowering temperatures) during the DA process, we will observe a series of specific temperatures, called *critical temperatures*, at which the problem space radically changes and solutions burst out in a manner in which a tree grows.

A question is how we can find or predict such phase transitions. In DA, we can describe phase transitions as a moment of losing stability of the objective function, the free energy $\mathcal{F}$, and turning to be unstable. Mathematically, that moment corresponds to the point in which the Hessian matrix, the second-order partial derivatives, of the object function loses its positive definiteness.

In our finite mixture framework, the Hessian matrix, the second-order partial derivatives of the free energy $\mathcal{F}$ with respect to component variables $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K$, can be defined as a block matrix:

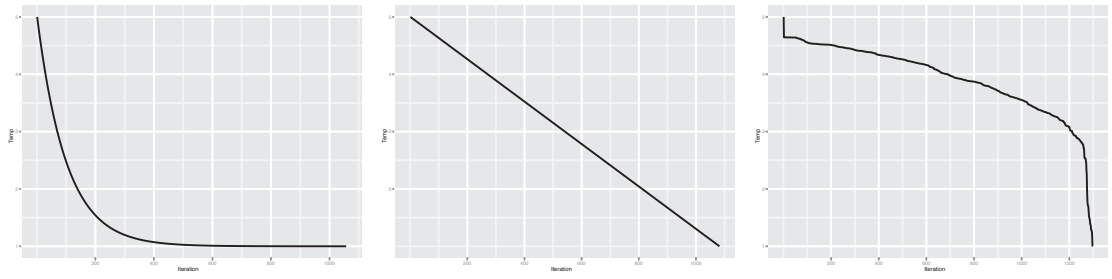
$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \cdots & \mathbf{H}_{1K} \\ \vdots & & \vdots \\ \mathbf{H}_{K1} & \cdots & \mathbf{H}_{KK} \end{bmatrix}, \tag{2.17}$$

where an element $\mathbf{H}_{kk'}$ is

$$\mathbf{H}_{kk'} = \frac{\partial^2 \mathcal{F}}{\partial \boldsymbol{\omega}_k^{\mathsf{Tr}} \partial \boldsymbol{\omega}_{k'}} \tag{2.18}$$

for $1 \leqslant k, k' \leqslant K$.

At a critical temperature (a moment of a phase transition), the Hessian $\mathbf{H}$ will be unstable and lose its positive definiteness. This temperature is called a critical temperature. Thus, we can define critical temperatures as the point to make the determinant of Hessian matrix $\mathbf{H}$ be zero ($\det(\mathbf{H}) = 0$).



(a) exponential **(b)** linear **(c)** adaptive

Figure 1: Various cooling schedule schemes. While exponential (a) and linear (b) is fixed and pre-defined, our new cooling scheme (c) is adaptive that the next temperature is determined in the on-line manner.

We will cover details of derivation and usages for GTM and PLSA in Section 3.3 and Section 4.3 respectively.

2.2.2 Adaptive cooling schedule

The DA algorithm has been applied in many areas and proved its success in avoiding local optimum and pursuing global solutions. However, to the best of our knowledge, no literature has been found about the cooling schedule of the DA algorithm. Commonly used cooling schedule is exponential (Figure 1a), such as $T = \alpha T$, or linear (Figure 1b), such as $T = T - \delta$ for invariant coefficients α and δ. Those scheduling schemes are fixed in that cooling temperatures are pre-defined and the coefficient α or δ will not be changed during the annealing process, regardless of the complexity of a given problem.

However, as we discussed previously, the DA algorithm undergoes the phase transitions in which the problem space (or free energy) can change dramatically. To avoid such drastic changes and make the transitions smoother, one may try to use very small α or δ coefficient. However, the procedure can go too long to be used in practice.

To overcome such problem, we propose an adaptive cooling schedule in which cooling temperatures are determined dynamically during the annealing process. More specifically, at every iteration in the DA algorithm, we predict the next phase transition temperature and move to that temperature as quickly as possible. Figure 1c shows an example of an adaptive cooling schedule, compared with fixed cooling schedules, Figure 1a and Figure 1b.

Another advantage we can expect in using our adaptive cooling schedule scheme is that users have no need to set anymore coefficients regarding cooling schedule. The adaptive cooling process will automatically suggest next temperatures, based on a given problem complexity. The adaptive cooling schedule for GTM will be discussed in Section 3.3.

2.2.3 Overfitting Avoidance

In statistical machine learning, overfitting (or overtraining) is a phenomena that a trained model works too well on the training example but shows poor predictive performance on unseen data [2, 25]. Especially in a supervised learning setting, an overfitting problem can be observed when training errors are getting smaller while validation errors are increasing. Such overfitting problem, in general, contributes to the poor predictive power of a trained model and causes a serious issue in many statistical machine learning, data mining, and information retrieval where predictive power for unseen data is a valuable property. A few general solutions suggested in the areas are regularization, cross-validation, early stopping, and so on.

In this thesis, we use DA in the finite mixture model framework to guard against the overfitting problem in an unsupervised learning setting where we do not have a managed testing set. We refer overfitting to a model with poor generalization quality.

Besides avoiding local optima problem discussed earlier, DA can also provide a capability for overfitting avoidance finding a generalized solution. In fact, DA natively supports generalized solutions. Note that DA starts to find a model in a smoothed probability surface in a high numeric temperature and gradually tracking an optimum solution in a bumpy and complex probability surface on lowering the numeric temperature [26]. In other words, DA has naturally an ability to control smoothness of a solution. We can exploit this feature of DA to obtain a less- or non-overfitted model.

Overfitting avoidance of DA has been investigated in various places [16, 33, 34]. In this thesis, we take a similar approach to exploit DA's overfitting avoidance for improving predicting power for PLSA, a text mining algorithm. Details will be discussed in Section 4.2.

3 GENERATIVE TOPOGRAPHIC MAPPING WITH DETERMINISTIC ANNEALING

In this chapter, we introduce the Generative Topographic Mapping (GTM) algorithm and describe how the GTM problem can be solved in the finite mixture model framework. Especially, we show the GTM algorithm is based on the Finite Mixture Model Type-1 (FMM-1), defined in Eq. (2.1). Then, we propose a new DA algorithm for GTM, named Generative Topographic Mapping with Deterministic Annealing (DA-GTM). In the next, we start with brief overviews of the GTM problem and discuss how we use the DA algorithm for parameter estimation in GTM and predicting phase transition in DA-GTM, followed by experimental results.

3.1 GENERATIVE TOPOGRAPHIC MAPPING

The GTM [3, 4] algorithm is a visualization algorithm designed to find a non-linear manifold embedding in a low dimension space (say, L-dimension) for a given high dimensional data set (say D-dimensional) by using K latent components. More specifically, the GTM algorithm seeks K latent variables, denoted by $z_1, \ldots, z_K$, in L-dimension space, also called *latent space*, such that $z_k \in \mathbb{R}^L (k = 1, ..., K)$, which can optimally represent the

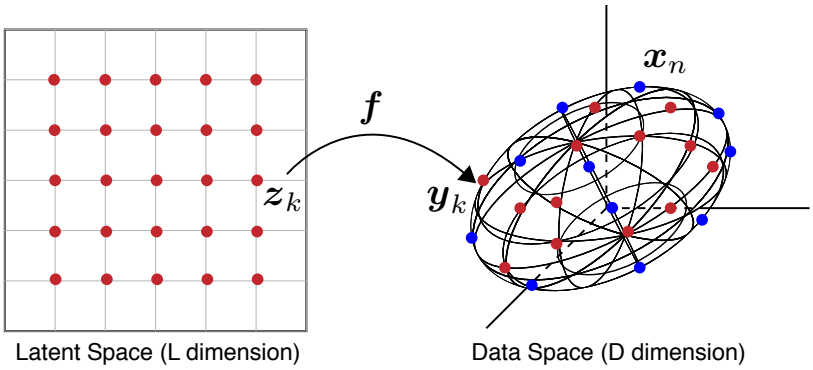


Figure 2: Non-linear embedding by GTM

given N data points, denoted by $x_1, \ldots, x_N$, in the D-dimension space, also called *data space*, which usually $L \ll D$ (See Figure 2).

For this end, the GTM algorithm finds a low dimensional embedding by using the following two steps: First, mapping the K latent variables, $z_1, \ldots, z_K$, in the latent space to the data space with respect to a non-linear mapping $f : \mathbb{R}^L \mapsto \mathbb{R}^D$. Let us denote the mapped points in the data space as $y_1, \ldots, y_K$. Secondly, fitting the mapped points $y_1, \ldots, y_K$, considered as K components, to the N sample data points $x_1, \ldots, x_N$ by using FMM-1 defined in Eq. (2.1).

Note that the GTM algorithm uses explicitly the Gaussian probability as a component distribution, specifically, defined by a Gaussian centered on y_k with covariance Σ_k. Without losing generality, we assume the Gaussian as an isotropic Gaussian with scalar variance σ^2, such that the conditional probability density $P(x_i \mid y_k, \sigma^2)$ is defined by the following Gaussian distribution $\mathcal{N}$:

$$P(x_i \mid y_k, \sigma^2) = \mathcal{N}(x_i \mid y_k, \sigma^2) \tag{3.1}$$

$$= \left(\frac{1}{2\pi\sigma^2}\right)^{D/2} \exp\left\{-\frac{1}{2\sigma^2} \| x_i - y_k \|^2\right\} \tag{3.2}$$

In summary, the GTM algorithm is the FMM-1 in which a sample data is modeled by

$$P(\mathbf{x}_i \,|\, Y, \sigma^2) \;=\; \sum_{k=1}^{K} \frac{1}{K}\, \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2). \tag{3.3}$$

In the GTM algorithm, we uses an uniform mixing weight, such that $\pi_k = 1/K$ for all k $(1 \leqslant k \leqslant K)$, as the Gaussian can control its variance σ^2 for varying mixing weights. Also, the component variables $\mathbf{y}_1, \ldots, \mathbf{y}_K$, serving as centers of Gaussian or means, are mapped by a non-linear function from L-dimension to D-dimension. The choice of the non-linear mapping $f : \mathbb{R}^L \mapsto \mathbb{R}^D$ can be made from any parametric, non-linear model. In the original GTM algorithm [3, 4], a generalized linear regression model has been used, in which the map is a linear combination of a set of fixed M basis functions, such that,

$$\mathbf{y}_k = \boldsymbol{\phi}_k^{Tr} \boldsymbol{W}, \tag{3.4}$$

where a column vector $\boldsymbol{\phi}_k = (\phi_{k1}, ..., \phi_{kM})$ is a mapping of z_k by the M basis function $\phi_m : \mathbb{R}^L \mapsto \mathbb{R}$ for $m = 1, ..., M$, such that $\phi_{km} = \phi_m(z_k)$ and $\boldsymbol{W}$ is a $M \times D$ matrix containing weight parameters. With a matrix notation, we can simplify the above equation by

$$\boldsymbol{Y} = \boldsymbol{\Phi} \boldsymbol{W} \tag{3.5}$$

where $\boldsymbol{\Phi}$ is $K \times M$ matrix of which row k represents $\boldsymbol{\phi}_k = (\phi_{k1}, ..., \phi_{kM})^{Tr}$.

With this model setting, the GTM algorithm corresponds to a Gaussian mixture model problem of FMM-1 and seeks an optimal set of parameters, $\mathbf{y}_1, \ldots, \mathbf{y}_K$ and σ^2, which maximizes the following log-likelihood of GTM, $\mathcal{L}_{GTM}$, defined by

$$\mathcal{L}_{GTM}(\mathbf{Y}, \sigma^2) = \sum_{i=1}^{N} \log \left\{ \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{x}_i \mid \mathbf{y}_k, \sigma^2) \right\} \tag{3.6}$$

3.2 DETERMINISTIC ANNEALING FOR GENERATIVE TOPOGRAPHIC MAPPING

The GTM algorithm uses an EM method which starts with a random initial matrix $\mathbf{W}$ and iteratively refines a solution to maximize Eq. (3.6), which can be easily trapped in local optima. Thus, an output (which is a mapping) produced by the original GTM algorithm can vary depending on initial parameters, which is known as the random initial value problem. Instead of using the EM, we have applied a DA approach to find a global optimum solution. With the use of the DA algorithm, we can have more robust GTM maps without suffering the random initial value problem.

To apply the DA algorithm, as discussed in Section 2.2, we need a new free energy function for GTM, named $\mathcal{F}_{GTM}$, which we will minimize through iterations. By using the definitions for free energy, used in the paper by K. Rose [28], we can drive a free energy

function for the GTM algorithm as follows; First, we let define the association cost $d(n, k)$ using a Gaussian distribution by

$$\begin{aligned} d(i,k) &= -\log\ P(\mathbf{x}_i, \mathbf{y}_k) && (3.7)\\ &= -\log\ \{P(\mathbf{y}_k)P(\mathbf{x}_i \,|\, \mathbf{y}_k)\} && (3.8)\\ &= -\log\ \left\{\frac{1}{K}\,\mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)\right\} && (3.9)\end{aligned}$$

By using Eq. (2.12), then, we can define Z_i by

$$Z_i = \sum_{k=1}^{K} \left(\frac{1}{K}\right)^{\beta} \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta} \tag{3.10}$$

Here, for brevity, we use an *inverse numeric temperature* denoted by β, such that $\beta = 1/T$.

Finally, by using Eq. (2.13), we can have the free energy function for GTM, $\mathcal{F}_{\text{GTM}}$, as follows:

$$\begin{aligned} \mathcal{F}_{\text{GTM}}(\mathbf{Y}, \sigma^2, \beta) &= -\frac{1}{\beta}\sum_{i=1}^{N} \log Z_i && (3.11)\\ &= -\frac{1}{\beta}\sum_{i=1}^{N} \log\left\{\left(\frac{1}{K}\right)^{\beta} \sum_{k=1}^{K} \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}\right\} && (3.12)\end{aligned}$$

which is the objective function for the DA-GTM algorithm to minimize as changing temperature from high (equivalent β near zero) and to low (equivalently $\beta = 1.0$).

Notice that the free energy function of DA-GTM, $\mathcal{F}_{\text{GTM}}$ (3.12), and the MLE (3.6) of GTM differ only the use of the inverse temperature variable β and the sign. Especially, at $\beta = 1.0$, we have

$$\mathcal{L}_{\text{GTM}}(\mathbf{Y}, \sigma^2) = -\mathcal{F}_{\text{GTM}}(\mathbf{Y}, \sigma^2, \beta) \tag{3.13}$$

and thus we can conclude that the original GTM algorithm's target function $\mathcal{L}_{GTM}$ is just a special case of $\mathcal{F}_{GTM}$.

To minimize (3.12), we need to find parameters to make the following two partial derivatives be zero (Detailed derivations can be found in Appendix A):

$$\frac{\partial \mathcal{F}_{GTM}}{\partial \mathbf{y}_k} = \frac{1}{\sigma^2} \sum_{i=1}^{N} \rho_{ki} (\mathbf{x}_i - \mathbf{y}_k) \tag{3.14}$$

$$\frac{\partial \mathcal{F}_{GTM}}{\partial \sigma^2} = -\sigma^4 \sum_{i=1}^{N} \sum_{k=1}^{K} \rho_{ki} \left(\frac{D\sigma^2}{2} - \frac{1}{2} \|\mathbf{x}_i - \mathbf{y}_k\|^2 \right) \tag{3.15}$$

where ρ_{ki} is a property, known as *responsibility*, such that,

$$\rho_{ki} = \frac{P(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^\beta}{\sum_{k'=1}^{K} P(\mathbf{x}_i \,|\, \mathbf{y}_{k'}, \sigma^2)^\beta} \tag{3.16}$$

By using the same matrix notations used in the GTM paper [3, 4], the DA-GTM algorithm can be written as a process to seek an optimal weight $\boldsymbol{W}$ and variance σ^2 at each temperature T.

$$\boldsymbol{W} = (\boldsymbol{\Phi}^{Tr} \mathbf{G} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{Tr} \mathbf{R} \mathbf{X} \tag{3.17}$$

$$\frac{1}{\sigma^2} = \frac{1}{ND} \sum_{i=1}^{N} \sum_{k=1}^{K} \rho_{ki} \|\mathbf{x}_i - \mathbf{y}_k\|^2 \tag{3.18}$$

where $\mathbf{X}$ is a $N \times D$ data matrix, $\boldsymbol{\Phi}$ is a $K \times M$ basis matrix, $\mathbf{G'}$ is a $K \times K$ diagonal matrix with elements $\gamma_k = \sum_n^N (\rho_{ki})^{\frac{1}{T}}$.

3.3 PHASE TRANSITIONS

As we discussed in Section 2.2, DA algorithms undergoes phase transitions as lowering the temperatures. At some temperature, we can not obtain all distinct solutions but, instead, we can only obtain a number of effective solutions. All solutions will gradually pop out while the annealing process proceeds as with lowering the temperature.

In the DA-GTM algorithm, we can observe the same behavior. As an example, at a very high temperature, the DA-GTM algorithm gives only one effective latent point that all $\mathbf{y}_k$'s are *collapsed* to, corresponding to the center of data, denoted by $\bar{\mathbf{x}}$, such that $\bar{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{x}_i / N$. At a certain temperature as we lowering temperature gradually, components, $\mathbf{y}_1, \ldots, \mathbf{y}_K$, which were *settled* (or stable) in their positions, start to *explode* (or move). We call this temperature as the first critical temperature, denoted by $T_c^{(1)}$ or, equivalently, $\beta_c^{(1)} = 1/T_c^{(1)}$, where the superscript indicates a sequence. As we further lowering the temperature, we can observe a series of subsequent phase transitions and thus multiple critical temperatures, such as $T_c^{(2)}, T_c^{(3)}, \ldots, T_c^{(K)}$. Especially obtaining the first phase transition $T_c^{(1)}$ is an important task since we should start our annealing process with an initial temperature higher than $T_c^{(1)}$.

In the DA algorithm, we define a phase transition as a moment of losing stability of the DA's objective function, the free energy $\mathcal{F}$, and turning to be unstable. Mathematically, that moment corresponds to the status in which the Hessian of the object function loses its positive definiteness.

For our DA-GTM algorithm, we can write the following Hessian matrix as a block matrix:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \cdots & \mathbf{H}_{1K} \\ \vdots & & \vdots \\ \mathbf{H}_{K1} & \cdots & \mathbf{H}_{KK} \end{bmatrix}, \tag{3.19}$$

where a sub matrix H_{ij} is a second derivative of the free energy $\mathcal{F}_{GTM}$ as shown in Eq. (2.11). More specifically, $\mathbf{H}_{ij}$ can be written as follows:

$$\begin{aligned} \mathbf{H}_{kk} &= \frac{\partial^2 \mathcal{F}_{GTM}}{\partial \mathbf{y}_k^{Tr} \partial \mathbf{y}_k} && (3.20) \\ &= -\frac{1}{\sigma^4 T} \sum_{i=1}^{N} \left\{ \rho_{ki}(1-\rho_{ki})(\mathbf{x}_i - \mathbf{y}_k)^{Tr}(\mathbf{x}_i - \mathbf{y}_k) - T\sigma^2 \rho_{ki} \mathbf{I}_D \right\}, \text{ or} && (3.21) \\ \mathbf{H}_{kk'} &= \frac{\partial^2 \mathcal{F}_{GTM}}{\partial \mathbf{y}_k^{Tr} \partial \mathbf{y}_{k'}} && (3.22) \\ &= \frac{1}{\sigma^4 T} \sum_{i=1}^{N} \left\{ -\rho_{ki}\rho_{k'n}(\mathbf{x}_i - \mathbf{y}_k)^{Tr}(\mathbf{x}_i - \mathbf{y}_{k'}) \right\} (k \neq k'), && (3.23) \end{aligned}$$

where $k, k' = 1, \ldots, K$, and $\mathbf{I}_D$ is an identity matrix of size D. Note that H_{kk} and $H_{kk'}$ are $D \times D$ matrices and thus, $\mathbf{H} \in \mathbb{R}^{KD \times KD}$.

As discussed in Section 2.2.1, we can compute the critical points which satisfy $\det(\mathbf{H}) = 0$. However, the size of the hessian matrix $\mathbf{H}$ can be too big to compute in practice. Instead, we compute critical points by dividing the problem into smaller pieces. The sketch of the algorithm is as follows:

1. For each component $\mathbf{y}_k$, let assume $\mathbf{y}_k$ is split into two sub-components, say $\mathbf{y}_k$ and $\mathbf{y}_{k'}$ (See Figure **??**). Note we dropped superscripts for brevity.

2. Compute a local Hessian for $\mathbf{y}_k$ (let denote $\mathbf{H}_k$) defined by

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{kk} & \mathbf{H}_{kk'} \\ \mathbf{H}_{kk'} & \mathbf{H}_{kk} \end{bmatrix} \tag{3.24}$$

   where $\mathbf{H}_{kk}$ and $\mathbf{H}_{kk'}$ are defined by Eq. (3.20) and Eq. (3.22) respectively but we let $\rho_{ki} = \rho_{ki}/2$ as we divide responsibilities too. Then, find a candidate of next critical temperature $T_{c,k}$ by letting $\det(\mathbf{H}_k) = 0$.

3. Choose the most largest yet lower than the current T among $\{T_{c,k}\}$ for all $k = 1, \ldots, K$.

To compute $\det(\mathbf{H}_k) = 0$, let define the following:

$$\mathbf{U}_{\mathbf{x}|\mathbf{y}_k} = \sum_{i=1}^{N} \rho_{ki}(\mathbf{x}_i - \mathbf{y}_k)^{\mathsf{Tr}}(\mathbf{x}_i - \mathbf{y}_k) \tag{3.25}$$

$$\mathbf{V}_{\mathbf{x}|\mathbf{y}_k} = \sum_{i=1}^{N} (\rho_{ki})^2(\mathbf{x}_i - \mathbf{y}_k)^{\mathsf{Tr}}(\mathbf{x}_i - \mathbf{y}_k) \tag{3.26}$$

Then, we can rewrite Eq. (3.20) and Eq. (3.22) by

$$\mathbf{H}_{kk} = -\frac{1}{T\sigma^4}\left(2\mathbf{U}_{\mathbf{x}|\mathbf{y}_k} - \mathbf{V}_{\mathbf{x}|\mathbf{y}_k} - 2T\sigma^2\gamma_k\mathbf{I}_D\right) \tag{3.27}$$

$$\mathbf{H}_{kk'} = -\frac{1}{T\sigma^4}\left(-\mathbf{V}_{\mathbf{x}|\mathbf{y}_k}\right) \tag{3.28}$$

We can also rewrite Eq. (4.19) by

$$\mathbf{H}_k = -\frac{1}{T\sigma^2}\left(\begin{bmatrix} 2\mathbf{U}_{\mathbf{x}|\mathbf{y}_k} - \mathbf{V}_{\mathbf{x}|\mathbf{y}_k} & -\mathbf{V}_{\mathbf{x}|\mathbf{y}_k} \\ -\mathbf{V}_{\mathbf{x}|\mathbf{y}_k} & 2\mathbf{U}_{\mathbf{x}|\mathbf{y}_k} - \mathbf{V}_{\mathbf{x}|\mathbf{y}_k} \end{bmatrix} - 2T\gamma_k\sigma^2\mathbf{I}_{2D}\right) \tag{3.29}$$

Thus, by letting $\det(\mathbf{H}) = 0$, we get the following eigen equation:

$$\operatorname{eig}\left(\begin{bmatrix} 2\mathbf{U}_{\mathbf{x}|\mathbf{y}_k} - \mathbf{V}_{\mathbf{x}|\mathbf{y}_k} & -\mathbf{V}_{\mathbf{x}|\mathbf{y}_k} \\ -\mathbf{V}_{\mathbf{x}|\mathbf{y}_k} & 2\mathbf{U}_{\mathbf{x}|\mathbf{y}_k} - \mathbf{V}_{\mathbf{x}|\mathbf{y}_k} \end{bmatrix}\right) = 2T\gamma_k\sigma^2 \tag{3.30}$$

where $\operatorname{eig}(\mathbf{A})$ denotes eigenvalues of $\mathbf{A}$.

We can further simplify the above equation by using the Kronecker product $\otimes$:

$$\operatorname{eig}\left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \otimes \mathbf{U}_{\mathbf{x}|\mathbf{y}_k} - \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \mathbf{V}_{\mathbf{x}|\mathbf{y}_k}\right) = 2T\gamma_k\sigma^2 \tag{3.31}$$

Finally, $T_{c,k}$ can be computed by

$$T_{c,k} = \frac{1}{2\gamma_k\sigma^2}\lambda_{max,k} \tag{3.32}$$

where $\lambda_{max,k}$ is the largest, but lower than a current temperature T, eigenvalue of the lefthand side of Eq. (3.31).

The first critical temperature $T_c^{(1)}$ is a special case of Eq. (3.32). With the Hessian matrix defined above, we can compute the first phase transition point occurred at $T_c^{(1)}$. Assuming that the system has not yet undergone the first phase transition and the current temperature is high enough, then we will have all $\mathbf{y}_k$'s overlapped in the center of the data point, denoted by $\mathbf{y}_0 = \bar{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{x}_i/N$, and equal responsibilities, such as $\rho_{ki} = \rho_{k'n} = 1/2$ for all k and n.

Then, the second derivatives can be rewritten by

$$\mathbf{H}_{kk} = -\frac{N}{4T\sigma^4}\left(\mathbf{S}_{\mathbf{x}|\mathbf{y}_0} - 2T\sigma^2\mathbf{I}_D\right) \tag{3.33}$$

$$\mathbf{H}_{kk'} = -\frac{N}{4T\sigma^4}\left(-\mathbf{S}_{\mathbf{x}|\mathbf{y}_0}\right) \tag{3.34}$$

Algorithm 1 GTM with Deterministic Annealing

DA-GTM

1: Set $T > T_c$ by using Eq. (3.37)
2: Choose randomly M basis function $\phi_m (m = 1, ..., M)$
3: Compute $\mathbf{\Phi}$ whose element $\phi_{km} = \phi_m(z_k)$
4: Initialize randomly $\boldsymbol{W}$
5: Compute σ^2 by Eq. (3.18)
6: **while** $T \geqslant 1$ **do**
7: Update $\boldsymbol{W}$ by Eq. (3.17)
8: Update σ^2 by Eq. (3.18)
9: $T \leftarrow$ NEXTCRITICALTEMP
10: **end while**
11: **return** $\mathbf{\Phi}, \boldsymbol{W}, \sigma^2$

Algorithm 2 Find the next critical temperature

NEXTCRITICALTEMP

1: **for** $k = 1$ to K **do**
2: $\Lambda_k \leftarrow \{\emptyset\}$
3: **for** each $\lambda \in \text{eig}\left(\mathbf{U}_{\boldsymbol{x}|\boldsymbol{y}_k} - \mathbf{V}_{\boldsymbol{x}|\boldsymbol{y}_k}\right)$ **do**
4: **if** $\lambda < T\gamma_k\sigma^2$ **then**
5: $\Lambda_k \leftarrow \Lambda_k \cup \lambda$
6: **end if**
7: **end for**
8: $\lambda_{\max,k} \leftarrow \max(\Lambda_k)$
9: $T_{c,k} \leftarrow \lambda_{\max,k}/\gamma_k\sigma^2$
10: **end for**
11: **return** $T_c \leftarrow \max(\{T_{c,k}\})$

where $\mathbf{S}_{\boldsymbol{x}|\boldsymbol{y}_0}$ represents a covariance matrix of centered data set such that,

$$\mathbf{S}_{\boldsymbol{x}|\boldsymbol{y}_0} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{y}_0)^{Tr}(\boldsymbol{x}_i - \boldsymbol{y}_0) \tag{3.35}$$

and the Hessian matrix also can be rewritten by

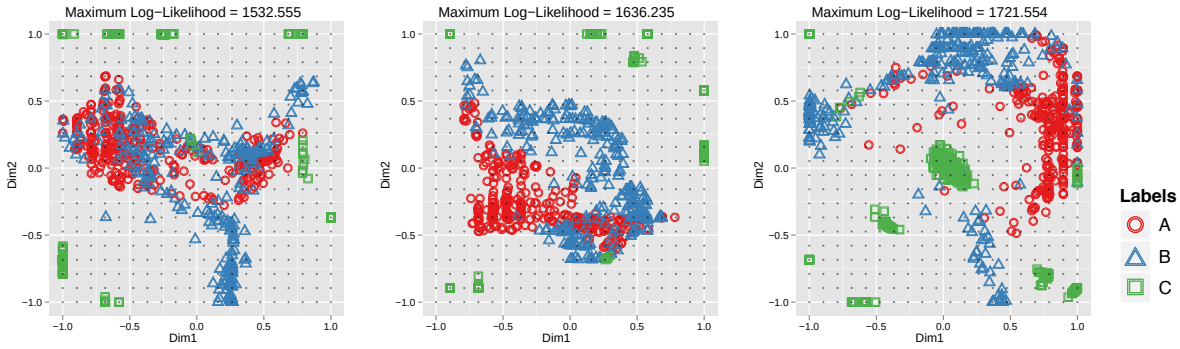
$$\text{eig}\left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes N\mathbf{S}_{\boldsymbol{x}|\boldsymbol{y}_0}\right) = 2T\gamma_k\sigma^2 \tag{3.36}$$

Thus, the first critical temperature is

$$T_c = \frac{1}{\sigma^2}\lambda_{max} \tag{3.37}$$

where λ_{max} is the largest eigenvalue of $\mathbf{S}_{\boldsymbol{x}|\boldsymbol{y}_0}$.

With Eq. (3.32) and Eq. (3.37), we can process DA-GTM with an adaptive cooling scheme discussed in Section 2.2.2. The overall pseudo code is shown in Algorithm 1 and Algorithm 2.



(a) EM-GTM **(b)** DA-GTM with exp. scheme **(c)** DA-GTM with adaptive scheme

Figure 3: Comparison of (a) EM-GTM, (b) DA-GTM with exponential, and (c) DA-GTM with adaptive cooling scheme for the oil-flow data which has 3-phase clusters (A=Homogeneous, B=Annular, and C=Stratified configuration). Plots are drawn by a median result among 10 random-initialized executions for each scheme. As a result, DA-GTM with adaptive cooling scheme (c) has produced the largest maximum log-likelihood and thus the plot shows better separation of the clusters, while EM-GTM (a) has output the smallest maximum log-likelihood and the result shows many overlaps.

3.4 EXPERIMENTS

To compare the performances of our DA-GTM algorithm with the original EM-based GTM (hereafter EM-GTM for short), we have performed a set of experiments by using two datasets: i) the oil flow data used in the original GTM papers [3, 4], obtained from the GTM website[1], which has 1,000 points having 12 dimensions for 3-phase clusters and ii) a chemical compound data set obtained from PubChem database[2], which is a NIH-funded repository for over 60 million chemical compounds and provides various chemical information including structural fingerprints and biological activities, for the purpose of chemical information mining and exploration. In this paper we have randomly selected a subset of 1,000 elements having 166 dimensions.

1 GTM homepage, http://www.ncrg.aston.ac.uk/GTM/
2 PubChem project, http://pubchem.ncbi.nlm.nih.gov/

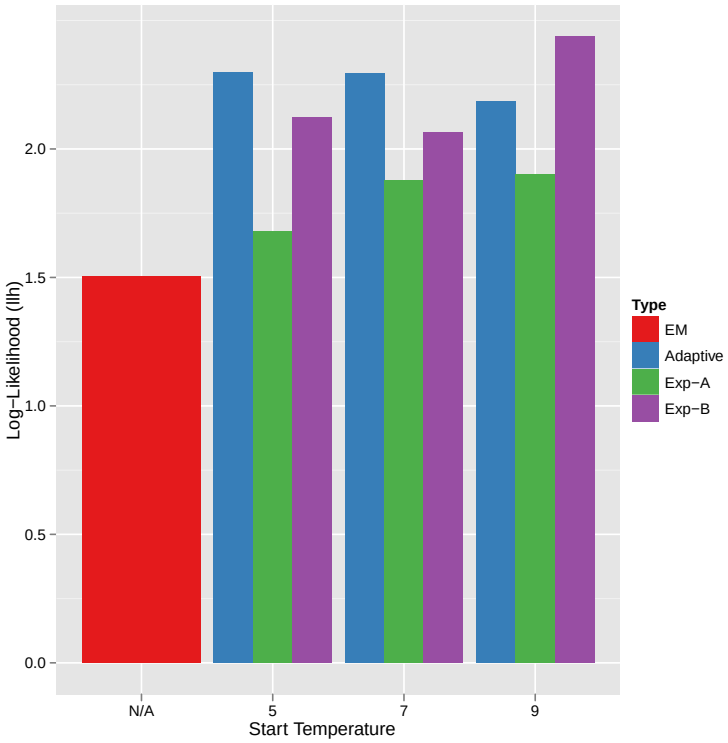
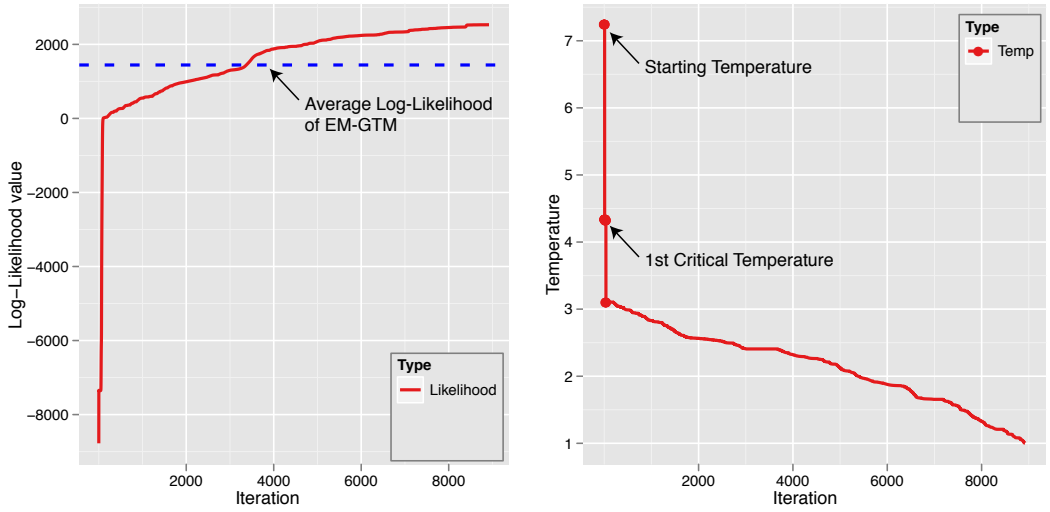


Figure 4: Comparison of EM-GTM with DA-GTM in various settings. Average of 10 random initialized runs are measured for EM-GTM, DA-GTM with 3 cooling schemes (adaptive, exponential with $\alpha = 0.95$ (Exp-A) and $\alpha = 0.99$ (Exp-B).

In Figure 3, we have compared for the oil-flow data maximum log-likelihood produced by EM-GTM, DA-GTM with exponential cooling scheme, and DA-GTM with adaptive cooling scheme and present corresponding GTM plots as outputs, known as *posterior-mean projection plot* [3, 4], in the latent space. For each algorithm, we have executed 10 runs with different random setups, chosen a median result, and drawn a GTM plot. As a result, DA-GTM with adaptive cooling scheme (Figure 3c) has produced the largest maximum log-likelihood (best performance), while EM-GTM (Figure 3a) produced the smallest maximum log-likelihood (worst performance). Also, as seen in the figures, a plot with larger maximum log-likelihood shows better separation of the clusters.



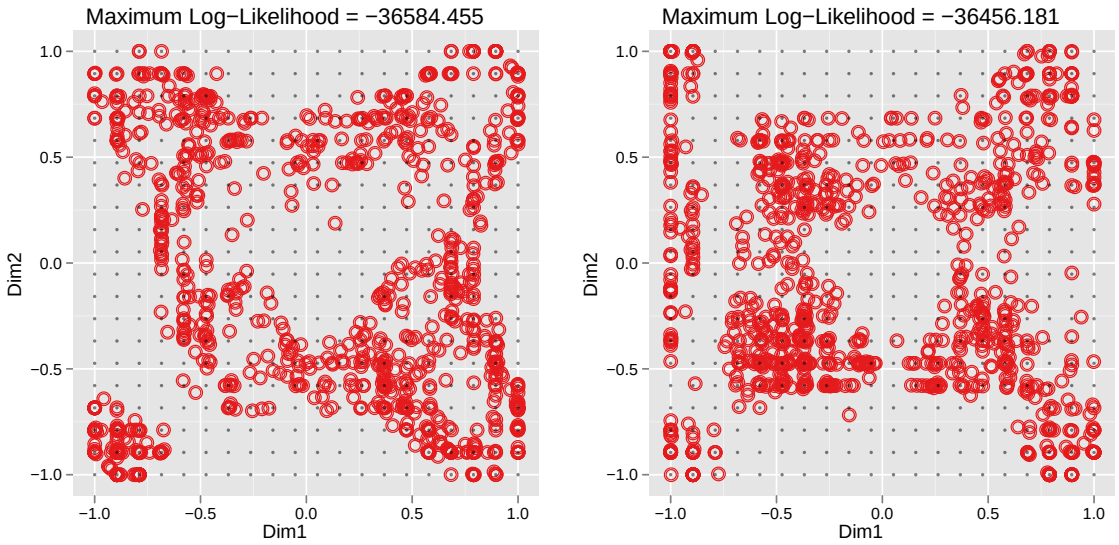
(a) Progress of log-likelihood **(b)** Adaptive changes in cooling schedule

Figure 5: A progress of DA-GTM with adaptive cooling schedule. This example show how DA-GTM with adaptive cooling schedule progresses through iterations

In the next, we have compared the performance of EM-GTM and DA-GTM with 3 different cooling schemes: i) Adaptive schedule, which we have prosed in this thesis, ii) Exponential schedule with a cooling coefficients $\alpha = 0.95$ (denoted Exp-A hereafter), and iii) Exponential schedule with a cooling coefficients $\alpha = 0.99$ (denoted Exp-B hereafter). For each DA-GTM setting, we have also applied 3 different starting temperature 5, 6, and 7, which are all larger values than the 1^{st} critical temperature which is about 4.64 computed by Eq. (3.37). Figure 4 shows the summary of our experiment results in which numbers are estimated by the average of 10 executions with different random initialization.

As a result shown in Figure 4, the DA-GTM outperforms EM-GTM in all cases. Also, our proposed adaptive cooling scheme mostly outperforms other static cooling schemes.

Figure 5 shows an example of execution of DA-GTM algorithm with adaptive cooling schedule.



(a) EM-GTM **(b)** DA-GTM with exp. scheme

Figure 6: Comparison of (a) EM-GTM and (b) DA-GTM with exponential scheme for 1,000 element PubChem dataset having 166 dimensions. Plots are drawn as a median result of 10 randomly initialized executions of EM-GTM and DA-GTM. The average maximum log-likelihood from DA-GTM algorithm $(-36,608)$ is larger than one from EM-GTM $(-36,666)$.

We have also compared EM-GTM and DA-GTM with exponential cooling scheme for the 1,000 element PubChem dataset which has 166 dimensions. As shown in Figure 6, DA-GTM's output is better than EM-GTM's since DA-GTM's average maximum log-likelihood $(-36,608)$ is larger than EM-GTM's $(-36,666)$.

3.5 CONCLUSIONS AND FUTURE WORK

In this chapter, we have showed how the GTM problem can be solved in a finite mixture framework and presented a new method to achieve better optimization. In the line of this approach, We have developed DA-GTM to solve the original GTM problem based on

the EM method. Our new algorithm uses as an optimization method the DA algorithm which is more resilient against the local optima problem and less sensitive to poor initial conditions, from which the original EM-based GTM was suffered. In addition, we have also developed a new cooling scheme, called adaptive cooling schedule. In contrast to the conventional cooling schemes, such as linear or exponential cooling schemes, which are all pre-defined and fixed, our new adaptive cooling scheme can adjust the granularity of cooling speed in an on-line manner. In our experiments, we showed new DA-GTM algorithm and adaptive cooling scheme can outperform the conventional GTM algorithm.

4 PROBABILISTIC LATENT SEMANTIC ANALYSIS WITH DETERMINISTIC ANNEALING

In this chapter, we introduce the Probabilistic Latent Semantic Analysis (PLSA) algorithm and discuss how the PLSA problem can be solved with the DA algorithm in the finite mixture framework. Especially, we show the model used in PLSA is based on the Finite Mixture Model Type-2 (FMM-2) we defined in Eq. (2.2). In the next, we start with brief overviews of the PLSA problem and discuss how we apply the DA algorithm for parameter estimation and learning in the PLSA algorithm and present a new algorithm, named Probabilistic Latent Semantic Analysis with Deterministic Annealing (DA-PLSA), followed by experimental results.

4.1 PROBABILISTIC LATENT SEMANTIC ANALYSIS

The PLSA algorithm [15, 16], also known as a *aspect* or *topic* model, is an algorithm for modeling binary and count data, aiming at recovering or deducting a generative process from which one can obtain essential probabilistic structures or latent aspects of data. Results from PLSA can be used for summarization, clustering, and classification. Among many application areas reported in the literature, such as document indexing [5, 15],

video processing [35], and speech recognition [27], PLSA shows prominent capability in analyzing and retrieval of text document, which is our focus in this thesis.

PLSA was stemmed from an algorithm called Latent Semantic Analysis (LSA) [11, 14], which is based on a linear algebra method, called Singular Value Decomposition (SVD), to extract the most dominant features of sample data by using a linear combination and L_2-norm approximation, adding a statical model which allows us to have principled approaches and statistical foundations. Especially, in the field of text document analysis and linguistics, PLSA has been used for building probabilistic models of text and languages.

The probabilistic model used in PLSA is called a latent variable model in which we assume data (or documents) is generated from a set of latent components (or *topics*). This model exactly corresponds to one of the finite mixture models we discuss in Section 2.1 (especially FMM-2. We will discuss shortly). In other words, in the context of the finite mixture model for text analysis, PLSA seeks a finite number of topics which can represent optimally the documents given in a corpus.

During the optimization process finding a model and model parameters, PLSA uses an EM method. However, this fitting process suffers two severe problems: one is the local optimum problem in which one would observe large variations of solutions, depending on random starting settings; the other is the overfitting problem in that EM chooses a model too close to the given data so that a solution loses its generality. Model generality is an important feature in information retrieval as it is directly related to the predictive power of a model for unseen data.

Overfitting is often referred in a supervised learning setting to describe a problem that a model looses its generality and thus shows large performance differences between a training set and a validation set. We use overfitting in an unsupervised learning setting,

where we do not have a managed testing set, in order to refer a model with poor predictive performance for unseen data.

This is our motivation to solve the PLSA problem with DA. Using DA in the place of EM in PLSA allows us i) to avoid local optimum problem and help to find a global solution with small variations and ii) to prevent overfitting and give more smoothed answers.

Latent Dirichlet Allocation (LDA), one of the most popular text mining algorithms, has been proposed by D. Blei at. al [5] to overcome overfitting problem in PLSA. LDA is an extension of PLSA and pursues a smoothed (generalized) model by using Dirichlet prior, called Dirichlet prior smoothing. Our DA based solution shares the same objective with LDA; seeking a smoothed model. However, we achieve smoothing effects in a DA framework in which no prior knowledge is required. We show a simple performance comparison in Section 4.4

Before discussing details of the DA algorithm for PLSA, we briefly review the PLSA algorithm. In PLSA, we denote a collection of N text documents, called a corpus of size N, as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_i$ $(1 \leqslant i \leqslant N)$ represents a document vector. In this corpus, we have a vocabulary set containing total D unique words (or terms) denoted by $\{w_1, \ldots, w_D\}$ and thus each document $\mathbf{x}_i$ is a D-dimensional vector where its j-th element represents the number of occurrences (or *frequency*) of word w_j. One may summarize the corpus $\mathbf{X}$ in a rectangular $N \times D$ matrix, called co-occurence (or document-term) matrix $\mathbf{X} = [x_{ij}]_{ij}$ for $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant D$, in a way in which an element x_{ij} denotes the frequency of word w_j occurred in a document $\mathbf{x}_i$.

Then, we can define a topic as a generative function that will create a document (i.e, a list of words and word frequencies) by following a multinomial distribution over words.

More specifically, if a document is generated from a certain topic, say k-th topic, its conditional probability can be written by

$$P(\mathbf{x}_i \mid \zeta_k = 1) = \text{Multi}(\mathbf{x}_i \mid \theta_k) \tag{4.1}$$

where ζ_k is called a latent class, a binary random variable indicating association with the k-th latent class, and $\text{Multi}(\mathbf{x}_i \mid \theta_k)$ represents a multinomial probability of $\mathbf{x}_i$ over word probability $\theta_k = (\theta_{k1}, \ldots, \theta_{kD})$ where θ_{kj} represents a word probability $P(w_j \mid \zeta_k = 1)$, defined by

$$\text{Multi}(\mathbf{x}_i \mid \theta_k) = \frac{\Gamma(|\mathbf{x}_i| + 1)}{\prod_{j=1}^{D} \Gamma(x_{ij} + 1)} \prod_{j=1}^{D} (\theta_{kj})^{x_{ij}} \tag{4.2}$$

where $\Gamma(\cdot)$ represents a gamma function, an extension of the factorial function such that $\Gamma(n) = (n-1)!$.

Assuming we have total K topics in a given corpus, the marginal document probability can be defined as a mixture of topics written by

$$P(\mathbf{x}_i \mid \boldsymbol{\Theta}, \boldsymbol{\Psi}) = \sum_{k=1}^{K} \psi_{ik} \text{Multi}(\mathbf{x}_i \mid \theta_k) \tag{4.3}$$

where a word probability set is denoted by $\boldsymbol{\Theta} = \{\theta_1, \ldots, \theta_K\}$ and a mixture weight set is presented by $\boldsymbol{\Psi} = [\psi_{ik}]_{ik}$ for each mixture weight ψ_{ik} with the constraint $0 \leqslant \psi_{ik} \leqslant 1$ and $\sum_k \psi_{ik} = 1$.

Please note this is Type 2 of our finite mixture model, FMM-2, defined in Eq. (2.2), where the component specific parameters $\boldsymbol{\Omega} = \{\omega_1, \ldots, \omega_K\}$ correspond to the word probability $\boldsymbol{\Theta} = \{\theta_1, \ldots, \theta_K\}$. Also, a mixture weight ψ_{ik} is a document level parameter, rather than a corpus level, in that each document will have different mixture weights over

the finite number of topics. This is the key difference that distinguished from the other mixture model in which weights are uniform throughout random samples (e.g., GTM).

Then, the probability of the full set of documents in the corpus can be written by

$$P(\mathbf{X} \mid \boldsymbol{\Theta}, \boldsymbol{\Psi}) = \prod_{i=1}^{N} \sum_{k=1}^{K} \psi_{ik} \text{Multi}(\mathbf{x}_i \mid \theta_k) \tag{4.4}$$

and the log-likelihood can be defined by

$$\mathcal{L}_{\text{PLSA}}(\mathbf{X}, \boldsymbol{\Theta}, \boldsymbol{\Psi}) = \sum_{i=1}^{N} \log \left\{ \sum_{k=1}^{K} \psi_{ik} \text{Multi}(\mathbf{x}_i \mid \theta_k) \right\} \tag{4.5}$$

Eq. (4.5) is the objective function in the original PLSA algorithm to maximize by using EM. In the following we will discuss how the DA algorithm can be used for the PLSA problem.

4.2 DETERMINISTIC ANNEALING FOR PROBABILISTIC LATENT SEMANTIC ANALYSIS

To maximize the log-likelihood function shown in Eq. (4.5), T. Hofmann has proposed an EM algorithm for model fitting in PLSA [15, 16]. However, as we discussed in Section 2.2, EM finds only local solutions. In addition, the EM algorithm does not provide a systematic way to avoid the overfitting problem, which is an important issue in text mining and retreival. Simply, in the EM algorithm, one may try to stop optimization earlier before reaching convergence (known as early stopping) but it is not enough for better performance.

To overcome such problem, we propose a new DA algorithm for PLSA, named Probabilistic Latent Semantic Analysis with Deterministic Annealing (DA-PLSA). In fact, in the paper on PLSA [16], T. Hofmann has also proposed a DA-like algorithm, called Tempered EM. However, the Tempered EM is different from the traditional DA algorithm in that the cooling schedule is reversed and is only applied to solve overfitting problem (we will discuss soon). Our proposed DA-PLSA algorithm is more closed to the DA approach presented by K. Rose and G. Fox [28, 29].

To optimize the PLSA model fitting with DA, we define a new objective function for PLSA, *free energy* $\mathcal{F}_{\text{PLSA}}$, given by

$$\mathcal{F}_{\text{PLSA}} = -\frac{1}{\beta}\sum_{i=1}^{N}\log\sum_{k=1}^{K}\{\psi_{ki}\text{Multi}\,(\mathbf{x}_i \mid \theta_k)\}^{\beta} \tag{4.6}$$

where β represents *inverse computational temperature*, defined by $\beta = 1/T$. Unlike the Tempered EM where temperatures are changed from a low temperature (1.0) to a high temperature, we will gradually lower a temperature from high to low (equivalently, β will be changed from near zero to 1). At each temperature, we have the following internal EM steps to minimize $\mathcal{F}_{\text{PLSA}}$.

- E-step : we evaluate the responsibility ρ_{ki}, defined by

$$\rho_{ki} = \frac{P(\mathbf{x}_i \mid \theta_k, \psi_k)^{\beta}}{\sum_{k'=1}^{K} P(\mathbf{x}_i \mid \theta_{k'}, \psi_{k'})^{\beta}} \tag{4.7}$$

$$= \frac{\{\psi_{ki}\text{Multi}\,(\mathbf{x}_i \mid \theta_k)\}^{\beta}}{\sum_{k'=1}^{K}\{\psi_{k'i}\text{Multi}\,(\mathbf{x}_i \mid \theta_{k'})\}^{\beta}} \tag{4.8}$$

- M-step : we maximize $\mathcal{F}_{PLSA}$ by computing the following parameters:

$$\theta_k = \frac{\sum_{n=1}^{N} \rho_{ki} x_i}{\left| \sum_{n=1}^{N} \rho_{ki} x_i \right|} \tag{4.9}$$

$$\psi_{ik} = \frac{\rho_{ki}}{\sum_{k=1}^{K} \rho_{ki}} \tag{4.10}$$

  Those parameters are chosen as to make the first derivate of $\mathcal{F}_{PLSA}$, $\partial\mathcal{F}_{PLSA}/\partial\theta_k$ and $\partial\mathcal{F}_{PLSA}/\partial\psi_{ik}$, be zero (Details of derivation is shown in Appendix B).

4.2.1 Parameter Estimation for Prediction

As we mentioned previously, predicting power for unseen data is highly valuable, especially, in the text mining area and many researches have been performed to increase predicting power by avoiding the overfitting problem and seeking a general solution. The original PLSA algorithm with EM suffers from the overfitting problem because the EM model fitting finds parameters too specific to a given dataset, so that a result model may lose its generality for unseen data. This overfitting problem can be prevented by using the DA algorithm.

The DA algorithm is an algorithm which can control *smoothness* or *generality* at each level of temperature while doing annealing. In fact, DA finds a solution in a way it gradually refines a solution iteratively, starting from a state of large entropy, which is called a smoothed solution, and ending at a state achieving an optimal solution specific to a problem. In short, DA refines a solution annealed from a high temperature to a low temperature such as $T = 1.0$. Our intuition in using DA to solve the overfitting problem of PLSA is that we find a smoothed model starting from high temperature but before

reaching at $T = 1.0$, expecting that the model has more predictive power for unseen data which is not included in the sample data used for modeling.

To measure the quality of predictive power of unseen data, we utilize the technique used in Tempered EM, known as V-fold cross validation. In V-fold cross-validation, we randomly partition the original data into V smaller subsets. Then, we use only $(V-1)$ subsets, called the *training set*, for training and learning models and the remaining one subset, called the *testing set* or *validation set*, for only measuring a predictive power (or generality) of the trained model. In a series of PLSA related papers [5, 15, 16], to access the quality of the predictive power of a trained PLSA model, it is proposed to measure a quantity, called *perplexity*, of a testing set as unseen data by using the parameters leaned from the training set. Perplexity is a log-averaged inverse probability [5], defined by,

$$\textit{Perplexity} = \exp\left(\frac{-\sum_{i=1}^{N}\log P(\mathbf{x}_i)}{\sum_{i=1}^{N}|\mathbf{x}_i|}\right) \tag{4.11}$$

where $\mathbf{x}_i$ is a document and $|\mathbf{x}_i|$ represents the total sum of word frequencies of document $\mathbf{x}_i$. In short, a lower perplexity score indicates better generalization performance.

Note, perplexity is closely related to the log-likelihood of a corpus, such that

$$\textit{Perplexity} = \exp\left(\frac{-\mathcal{L}_{\text{PLSA}}}{\sum_{i=1}^{N}|\mathbf{x}_i|}\right) \tag{4.12}$$

and thus,

$$\textit{Perplexity} \propto -\mathcal{L}_{\text{PLSA}} \tag{4.13}$$

In this thesis, however, we propose using the total sum of log-likelihood of both training set and testing set, named *total perplexity*, defined by,

$$\textit{Total Perplexity} = a \cdot \mathcal{L}_{\text{PLSA}}(\mathbf{X}_{\text{training}}, \boldsymbol{\Theta}, \boldsymbol{\Psi}) + b \cdot \mathcal{L}_{\text{PLSA}}(\mathbf{X}_{\text{testing}}, \boldsymbol{\Theta}, \boldsymbol{\Psi}) \tag{4.14}$$

for mixing weight coefficients a, b.

In our proposed DA-PLSA, at each iteration we measure the total perplexity, defined in Eq. (4.14), by using both training (if $a > 0$) and testing set and stop annealing at the temperature in which the total perplexity is maximized. For an example, Figure 7 shows the changes of log-likelihood of both a training set and a testing set by using the Associated Press (AP) dataset (Details are summarized in Table 1). In Figure 7, we can observe that the log-likelihood (LLH) of the training set is steadily improved, as we proceed annealing from a high temperature to a low temperature. However, the LLH of the testing set is decreasing because the model fitting is losing its generality. Our proposed solution using total perplexity is to stop annealing when the total perplexity is maximized. In this example shown in Figure 7, we use $a = b = 0.5$ for measuring the total perplexity (green dotted line). This scheme suggest that we stop annealing at about $T = 49.98$, so that the sum of perplexity is maximized.

In Figure 7, we can also observe the overfitting problem; a steep drop of log-likelihood of the testing set. This is mainly because, during the model fitting process in DA-PLSA, the smoothed word probabilities at high temperatures become rigid so that word probabilities tend to be extreme, either close to one or zero, at lower temperature. This harms the log-likelihood of the testing set. To show the changes of word probabilities at different temperatures, in Figure 8 we have compared (a) log probabilities of words at the optimal temperature found by the total perplexity ($T = 49.98$) and (b) ones at $T = 1.0$. As shown in this example, DA-PLSA finds a general solution at higher temperature larger than 1.0.

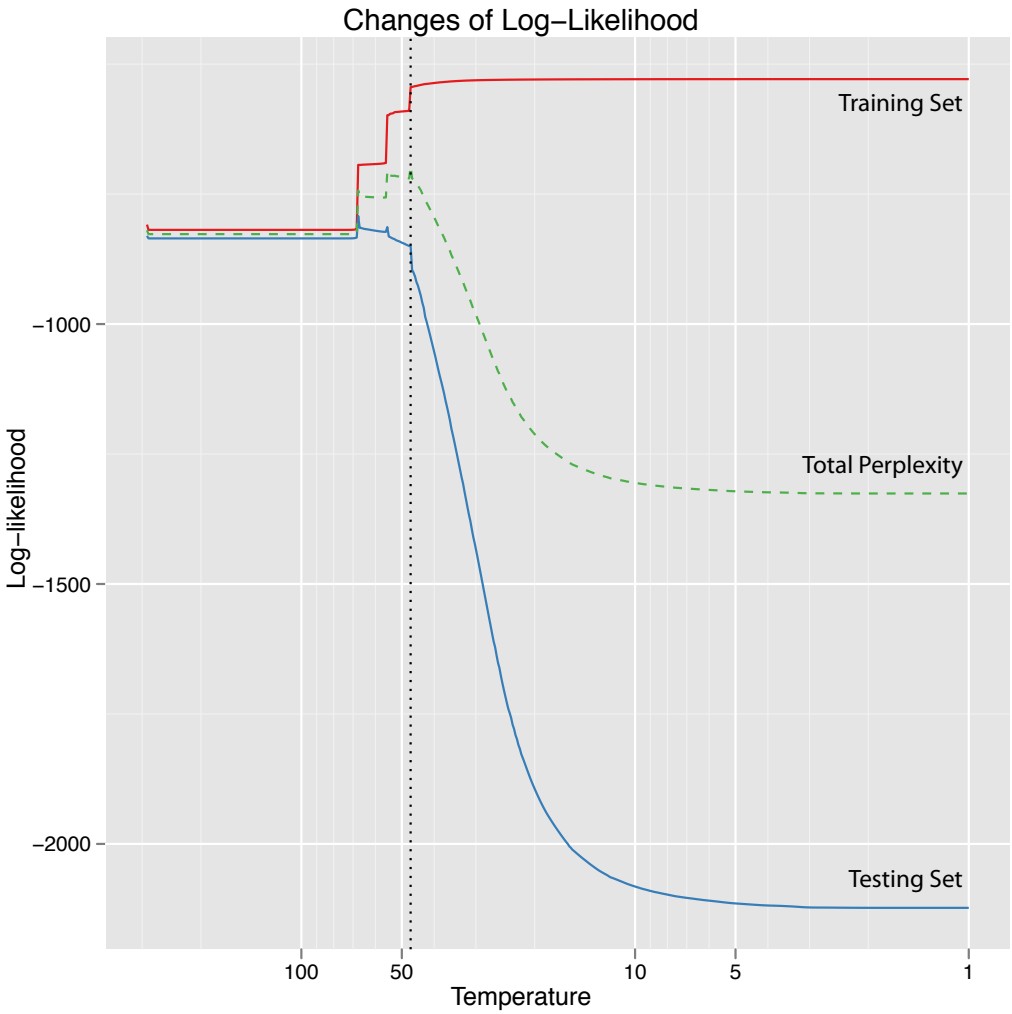
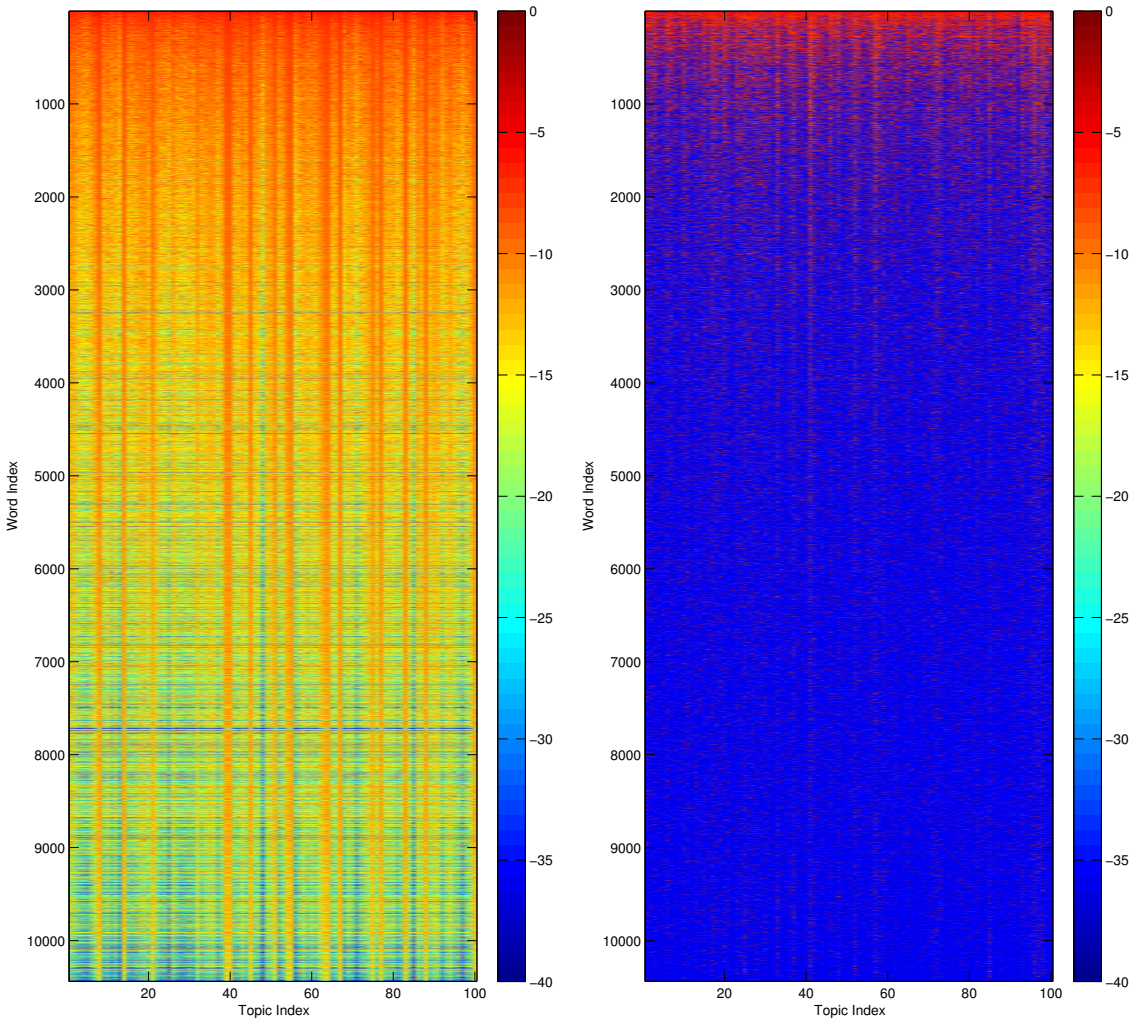


Figure 7: Changes of log-likelihoods of of the training set and the testing set in DA-PLSA for the Associated Press dataset with K=100. The sum (dotted line) represents total perplexity. Temperatures are log-scale as we used an exponential cooling scheme.



(a) $T = 49.98$ **(b)** $T = 1.00$

Figure 8: A Heat map of word probability of the AP data set at different temperatures: (a) one is measured at the optimal temperature in which total perplexity is maximized ($T = 49.98$) and (b) the other is taken at $T = 1.0$. Probabilities are log scaled.

4.3 PHASE TRANSITIONS

As we discussed in Section 2.2, our DA-PLSA algorithm will undergo phase transitions in which the free energy $\mathcal{F}_{\text{PLSA}}$ changes drastically when the temperature is lowering. Also, as we discussed previously in DA-GTM, we can define such phase transitions of DA-PLSA as a moment of losing stability of the objective function, the free energy $\mathcal{F}_{\text{PLSA}}$, and turning to be unstable. I.e., phase transitions can occur at the moment in which the Hessian of DA-PLSA loses its positive definiteness. The Hessian matrix of DA-PLSA can be also defined in the same form shown in Eq. (3.23) with the following sub matrix definitions:

$$\mathbf{H}_{kk} = \frac{\partial}{\partial \boldsymbol{\theta}_k^{\text{Tr}}}\left(\frac{\partial \mathcal{F}_{\text{PLSA}}}{\partial \boldsymbol{\theta}_k}\right) \tag{4.15}$$

$$= -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\rho_{ki}-\rho_{ki}^2\right)\beta^2\left(\mathbf{x}_i\boldsymbol{\theta}_k^{-1}\right)^{\text{Tr}}\left(\mathbf{x}_i\boldsymbol{\theta}_k^{-1}\right)-\rho_{ki}\beta\text{diag}\left(\mathbf{x}_i\boldsymbol{\theta}_k^{-2}\right)\right\} \tag{4.16}$$

$$\mathbf{H}_{kk'} = \frac{\partial}{\partial \boldsymbol{\theta}_{k'}^{\text{Tr}}}\left(\frac{\partial \mathcal{F}_{\text{PLSA}}}{\partial \boldsymbol{\theta}_k}\right) \tag{4.17}$$

$$= -\frac{1}{\beta}\sum_{n=1}^{N}\left\{-\rho_{ki}\rho_{k'i}\beta\left(\mathbf{x}_i\boldsymbol{\theta}_k^{-1}\right)^{\text{Tr}}\left(\mathbf{x}_i\boldsymbol{\theta}_k^{-1}\right)\right\} \tag{4.18}$$

where $\text{diag}(\mathbf{d})$ represents a diagonal matrix whose diagonal element is vector $\mathbf{d}$ and $k, k' = 1, \cdots, K$.

Now we can compute the critical points which satisfy $\det(\mathbf{H}) = 0$. However, the size of the hessian matrix $\mathbf{H}$ can be too big to compute. Instead, we compute critical points by dividing the problem into pieces. The sketch of the algorithm for each iteration (t) is as follows:

1. For each component θ_k, let assume $\theta_k^{(t)}$ is duplicated into two sub components, say $\theta_k^{(t+1)}$ and $\theta_{k'}^{(t+1)}$. For brevity, we will drop superscripts for a iteration index.

2. Compute a local Hessian for θ_k (let denote $\mathbf{H}_k$) defined by

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{kk} & \mathbf{H}_{kk'} \\ \mathbf{H}_{kk'} & \mathbf{H}_{kk} \end{bmatrix} \tag{4.19}$$

   where $\mathbf{H}_{kk}$ and $\mathbf{H}_{kk'}$ are defined by Eq. (3.20) and Eq. (3.22) respectively but we let $\rho_{ki} = \rho_{ki}/2$ as we split responsibilities too.

   Find a candidate of next critical temperature $T_{c,k}$ by letting $\det(\mathbf{H}_k) = 0$.

3. Choose the most largest but lower than the current T among $\{T_{c,k}\}$.

To compute $\det(\mathbf{H}_k) = 0$, let define the following:

$$\mathbf{U}_{\mathbf{X}\,|\,\theta_k} = \sum_{i=1}^{N} \left(\sqrt{\rho_{ki}}\mathbf{x}_i\theta_k^{-1}\right)^{\mathsf{Tr}} \left(\sqrt{\rho_{ki}}\mathbf{x}_i\theta_k^{-1}\right) \tag{4.20}$$

$$\mathbf{V}_{\mathbf{X}\,|\,\theta_k} = \sum_{i=1}^{N} \left(\rho_{ki}\mathbf{x}_i\theta_k^{-1}\right)^{\mathsf{Tr}} \left(\rho_{ki}\mathbf{x}_i\theta_k^{-1}\right) \tag{4.21}$$

$$\mathbf{G}_{\mathbf{X}\,|\,\theta_k} = \operatorname{diag}\left(\sum_{i=1}^{N} \rho_{ki}\mathbf{x}_i\theta_k^{-2}\right) \tag{4.22}$$

Then, we can rewrite Eq. (4.16) and Eq. (4.18) by

$$\mathbf{H}_{kk} = -\frac{\beta}{4}\left(2\mathbf{U}_{\mathbf{X}\,|\,\theta_k} - \mathbf{V}_{\mathbf{X}\,|\,\theta_k} - \frac{2}{\beta}\mathbf{G}_{\mathbf{X}\,|\,\theta_k}\right) \tag{4.23}$$

$$\mathbf{H}_{kk'} = -\frac{\beta}{4}\left(-\mathbf{V}_{\mathbf{X}\,|\,\theta_k}\right) \tag{4.24}$$

And,

$$\mathbf{H}_k = -\frac{\beta}{4}\left(\begin{bmatrix} 2\mathbf{U}_{\mathbf{X}\mid\theta_k} - \mathbf{V}_{\mathbf{X}\mid\theta_k} & -\mathbf{V}_{\mathbf{X}\mid\theta_k} \\ -\mathbf{V}_{\mathbf{X}\mid\theta_k} & 2\mathbf{U}_{\mathbf{X}\mid\theta_k} - \mathbf{V}_{\mathbf{X}\mid\theta_k} \end{bmatrix} - \frac{2}{\beta}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \mathbf{G}_{\mathbf{X}\mid\theta_k}\right) \quad (4.25)$$

$$= -\frac{\beta}{4}\left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \otimes \mathbf{U}_{\mathbf{X}\mid\theta_k} - \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \mathbf{V}_{\mathbf{X}\mid\theta_k} - \frac{2}{\beta}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \mathbf{G}_{\mathbf{X}\mid\theta_k}\right) \quad (4.26)$$

Since we can decompose G such that $\mathbf{G}_{\mathbf{X}\mid\theta_k} = \mathbf{A}^{\mathsf{Tr}}\mathbf{A}$, we can rewrite Eq. (4.26) by

$$\mathbf{H}_k = -\frac{\beta}{4}\mathbf{A}^{\mathsf{Tr}}\left\{\mathbf{\Lambda}_k - \frac{2}{\beta}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \mathbf{I}\right\}\mathbf{A} \quad (4.27)$$

where

$$\mathbf{\Lambda}_k = \left(\mathbf{A}^{\mathsf{Tr}}\right)^{-1}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \mathbf{U}_{\mathbf{X}\mid\theta_k} - \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \mathbf{V}_{\mathbf{X}\mid\theta_k}\right)\mathbf{A}^{-1} \quad (4.28)$$

Then, by letting $\det \mathbf{H} = 0$, we get the following eigen equation:

$$\mathrm{eig}\,(\Lambda_k) = \frac{2}{\beta} \quad (4.29)$$

Thus, a critical temperature $T_{c,k}$ can be computed by

$$T_{c,k} = 1/\beta \quad (4.30)$$

$$= \lambda_{max,k}/2 \quad (4.31)$$

where $\lambda_{max,k}$ represents one of the eigenvalues of $\mathbf{\Lambda}_k$ which is the largest but lower than current temperature T.

Table 1: Summary of the Associated Press dataset for 10-fold cross validation.

AP dataset	Train set	Test set (10%)	Total
Num. of documents	2,022	224	2,246
Num. of words	10,439	10,439	10,439
Sparseness (ratio of zero frequency elements)	0.9871	0.9874	
NIPS dataset	**Train set**	**Test set (10%)**	**Total**
Num. of documents	1,350	150	1,500
Num. of words	12,339	12,339	12,419
Sparseness (ratio of zero frequency elements)	0.9598	0.9590	

Now we can compute the first critical temperature $T_c^{(1)}$. When $K = 1$, we have $\mathbf{U}_{\mathbf{X}|\theta_k} = \mathbf{V}_{\mathbf{X}|\theta_k}$. Then,

$$\begin{aligned} \Lambda_k &= \left(\mathbf{A}^{\mathsf{Tr}}\right)^{-1} \left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \mathbf{U}_{\mathbf{X}|\theta_k} \right) \mathbf{A}^{-1} && (4.32) \\ &= \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \left\{ \left(\mathbf{A}^{\mathsf{Tr}}\right)^{-1} \mathbf{U}_{\mathbf{X}|\theta_k} \mathbf{A}^{-1} \right\} && (4.33) \end{aligned}$$

Thus, the first critical temperature $T_c^{(1)}$ can be computed by

$$T_c^{(1)} = \lambda_{max} \quad (4.34)$$

where λ_{max} is the largest eigenvalue of eig $\left(\left(\mathbf{A}^{\mathsf{Tr}}\right)^{-1} \mathbf{U}_{\mathbf{X}|\theta_k} \mathbf{A}^{-1} \right)$

4.4 EXPERIMENTS

In this section, we show experiment results of DA-PLSA in comparison of the original PLSA algorithm (hereafter EM-PLSA for short). For our experiments, we used known datasets, called Associated Press news dataset (hereafter AP data) and NIPS conference papers dataset (hereafter NIPS data). Details are summarized in Table 1.

4.4.1 Performance of DA

First, we compared the full performance of DA-PLSA (annealing until $T = 1.0$) and traditional EM-PLSA by using the AP data with various latent dimensions (also called the number of hidden components). Figure 9 shows the changes of maximum log-likelihood obtained by DA and EM algorithm of PLSA with different number of latent components. As we can see, DA-PLSA consistently outperforms EM-PLSA by showing larger log-likelihood values.

4.4.2 Avoiding overfitting

Without early stopping, i.e., fitting a model lowering temperatures until $T = 1.0$, as shown in the previous experiment, we may find an optimal PLSA model which generates the maximum log-likelihood for a given sample, i.e., fully optimized, by using DA. However, we may also suffer from an overfitting problem. Especially, in the text mining, a model with predicting power for unseen data is more valuable than a model with over-specific to a given sample. Avoiding such overfitting problem is an important issue. As

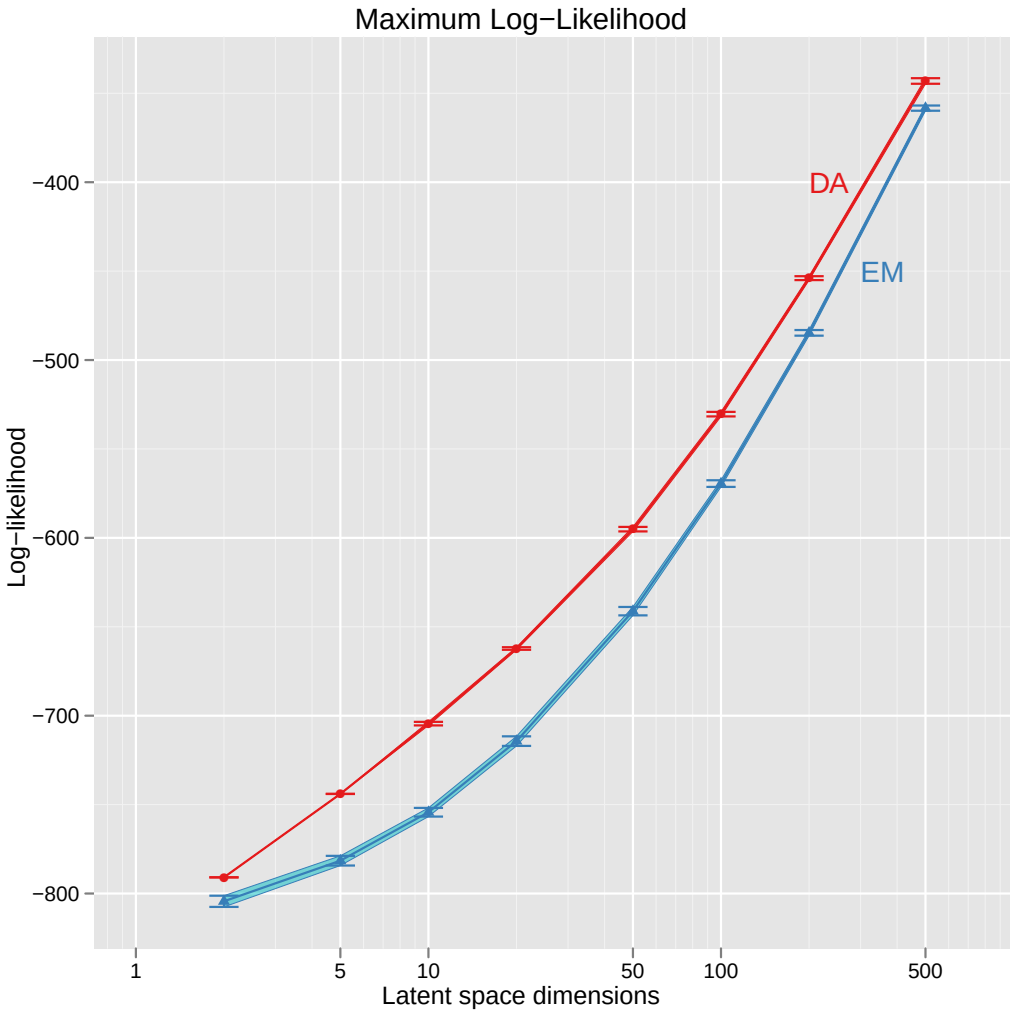


Figure 9: Quality comparison between the EM-PLSA and DA-PLSA by using the maximum log-likelihood values at $T = 1.0$ with respect to various latent dimensions. DA-PLSA outperforms EM-PLSA for all latent dimensions. Numbers are measured by the average of 100 randomly initialized experiments. A bar represents variation.

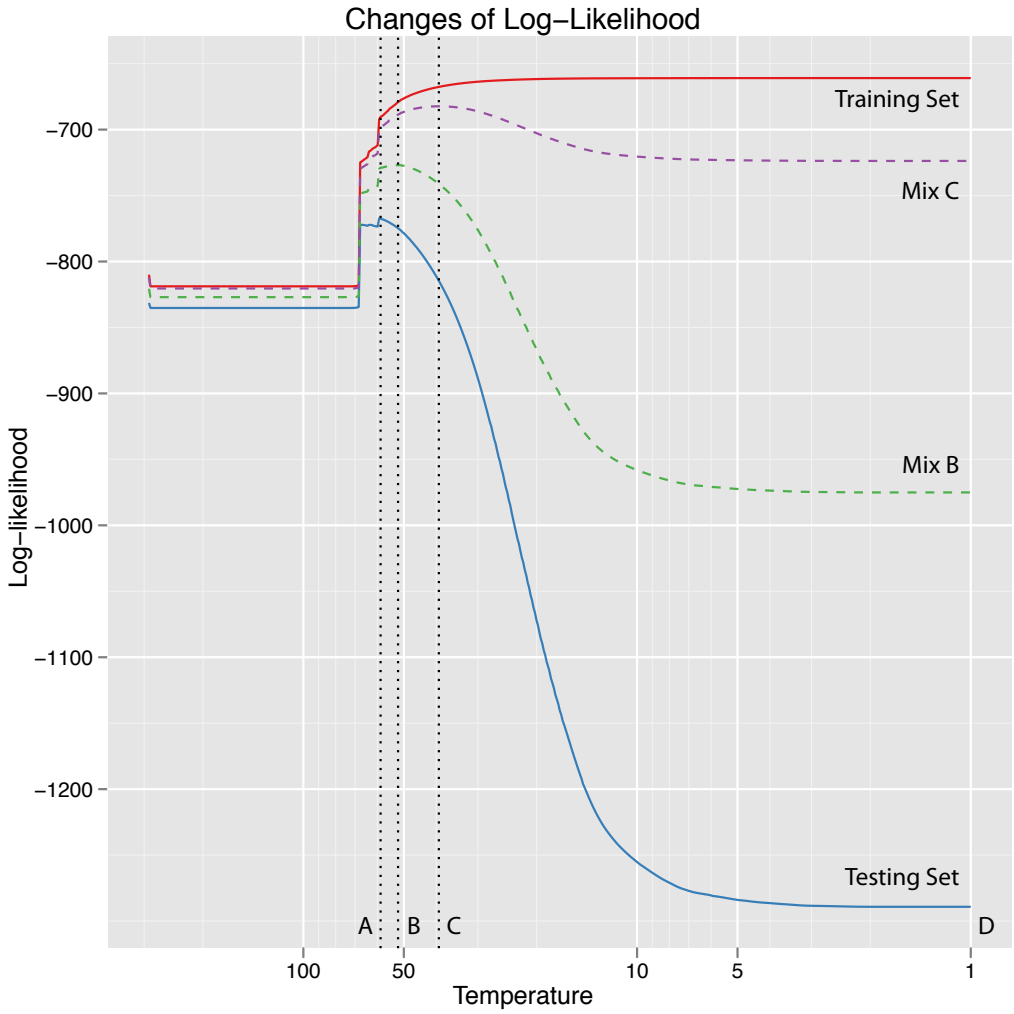
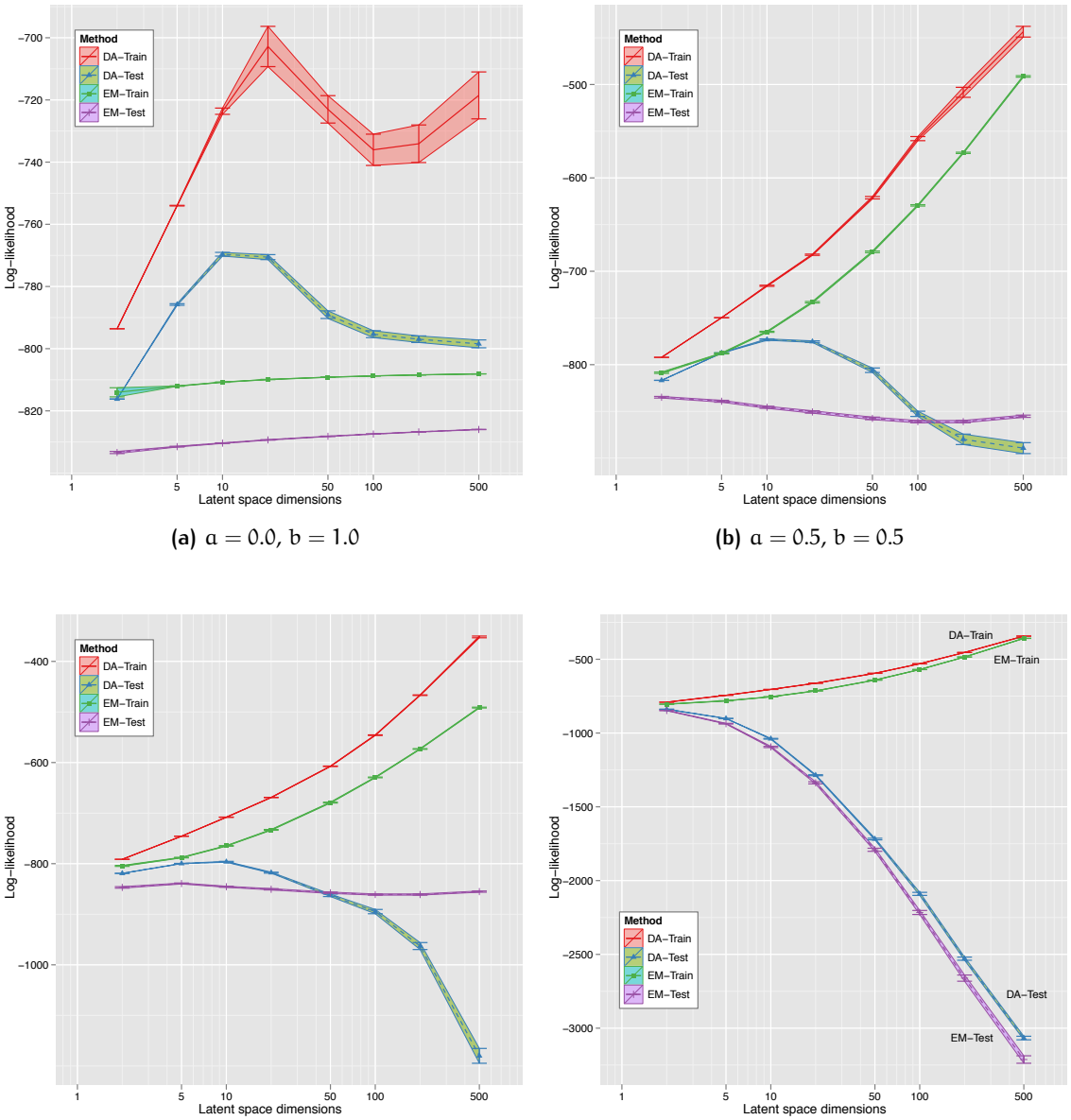


Figure 10: Comparison of DA and EM with the AP dataset. Lines are representing LLH of DA and EM of train set and test set when test set's LLH is maximized.



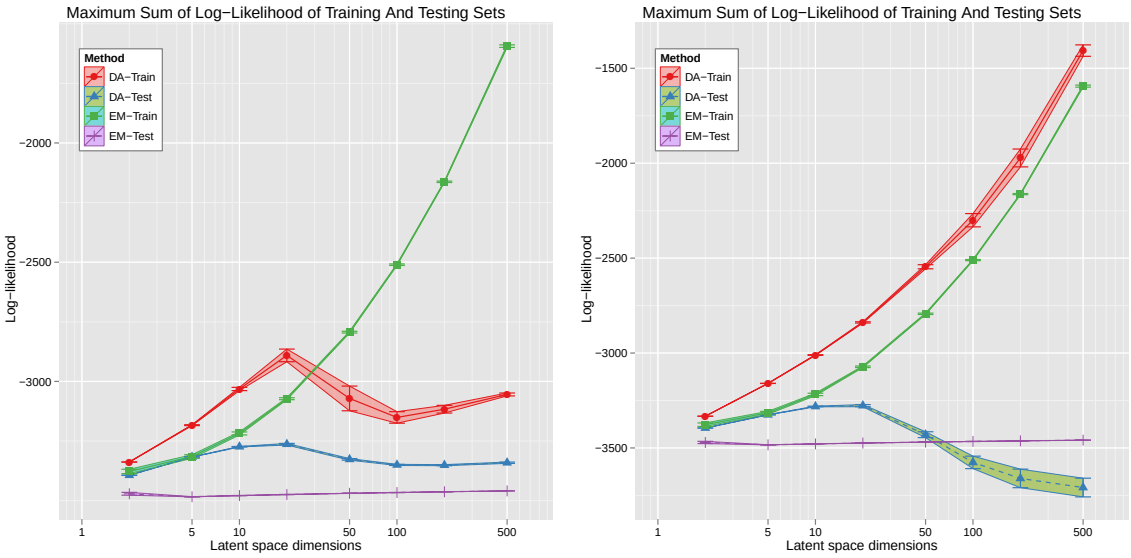
(a) $a = 0.0$, $b = 1.0$

(b) $a = 0.5$, $b = 0.5$

(c) $a = 0.9$, $b = 0.1$

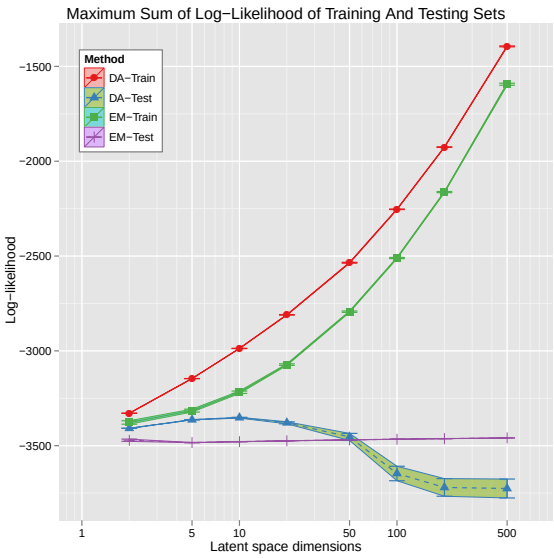
(d) $a = 1.0$, $b = 0.0$

Figure 11: Maximum perplexity for different stop conditions with AP data.



(a) $a = 0.0$, $b = 1.0$

(b) $a = 0.5$, $b = 0.5$



(c) $a = 0.9$, $b = 0.1$

(d) $a = 1.0$, $b = 0.0$

Figure 12: Maximum perplexity for different stop conditions with NIPS data.

discussed, we avoid overfitting in DA-PLSA by stoping annealing process before reaching the terminal temperature $T = 1.0$.

In this experiment, we compare four different stopping conditions with different mixing weight coefficients for measuring total perplexity defined in Eq. (4.14). We consider the following different stopping conditions: A ($a = 0.0, b = 1.0$), B ($a = 0.5, b = 0.5$), C ($a = 0.9, b = 0.1$), and D ($a = 1.0, b = 0.0$) for the AP data with different latent components. For an example, shown in Figure 10 ($K = 20$), we stopped the annealing with different conditions and measured LLH of a training and a testing set. Results are shown in Figure 11. Obviously the case with D (Figure 11(d)) shows the serious overfitting problem. Namely, the log-likelihoods of the training set (both DA and EM) are increasing but LLHs of the testing set are decreasing. However, the case with A (Figure 11(a)) shows a success story. The LLH values of training and testing set are moving in a synchronized way and thus with no serious overfitting problem. We have observed similar trends in NIPS data shown in Figure 12.

4.4.3 Comparison with LDA

We have compared our DA-PLSA performance with LDA [5]. As the definitions of quality in both algorithms are slightly different, it is impossible to compare two algorithms side by side. Instead, we use the program[1] developed by D. Blei, the original author of LDA, and plug the word probabilities generated as an output from our DA-PLSA. A result is shown in Figure 13. DA-PLSA shows a comparable performance result with LDA. This is our initial work in performing quality study compared with LDA. More thorough performance study remains as future work.

1 Available at http://www.cs.princeton.edu/~blei/lda-c/

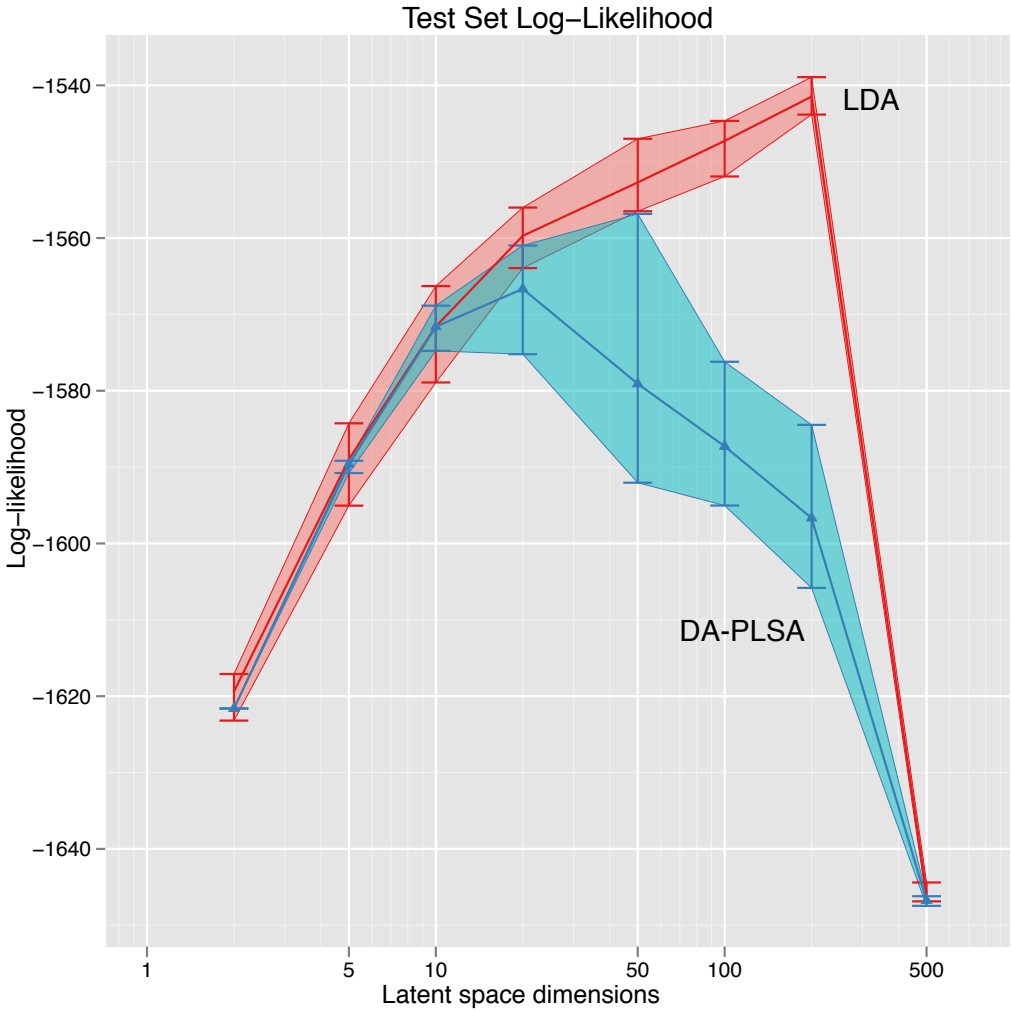


Figure 13: Performance comparison between LDA and DA-PLSA. We plugged the word probabilities generated as an output from our DA-PLSA into the LDA program developed by D. Blei.

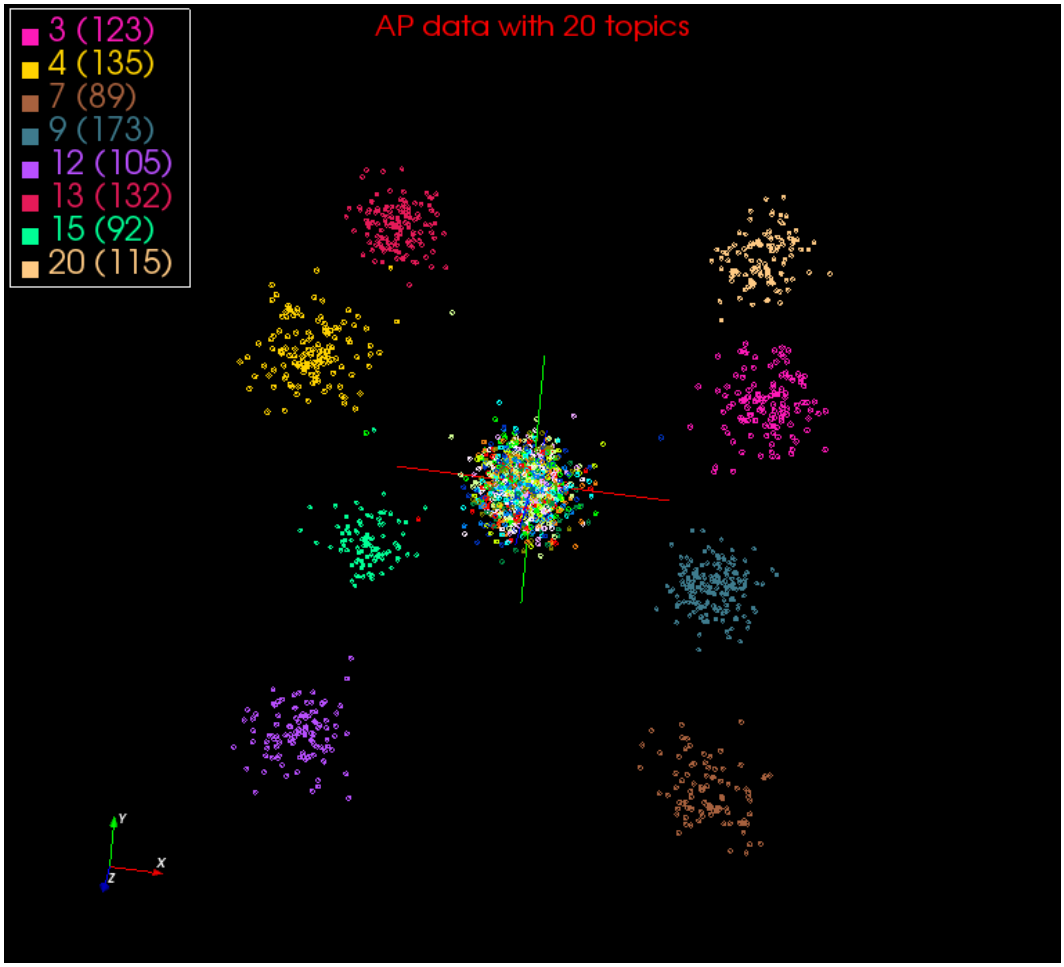
Figure 14: A workflow to visualize large and high-dimensional document corpus in a virtual 3-dimensional space. A high-dimensional document corpus will be processed by DA-PLSA to reduce the original dimension to an arbitrary smaller dimension, then, followed by DA-GTM to compute an optimal embedding in a 3-dimensional space.

4.4.4 Corpus visualization with GTM

Visualization of high-dimensional data in a low-dimension space is the core of exploratory data analysis in which users seek the most meaningful information hidden under the intrinsic complexity of data mostly due to high dimensionality. Especially, we have visualized a large and high-dimensional document corpus in a virtual 3-dimensional space by using a workflow shown in Figure 14. In the workflow, a high-dimensional document corpus will be processed by DA-PLSA to reduce the original dimension to an arbitrary smaller dimension, say K, then, followed by DA-GTM to compute an optimal embedding in a 3-dimensional space.

As an example shown in Figure 15, the AP data is visualized in a 3D space by using a visualization program, called PlotViz3[2], developed by the author. 20 topics ($K = 20$) is chosen for DA-PLSA and the 3-dimensional embedding is computed by DA-GTM. The legend shows well-clustered 8 topics among 20 topics and the top 8 largest probability words in each topic are shown in the table.

2 PlotViz3, a cross-platform tool for visualizing large and high-dimensional data. Available at http://salsahpc.indiana.edu/pviz3/



Topic 3	Topic 4	Topic 7	Topic 9	Topic 12	Topic 13	Topic 15	Topic 20
marriage	mandate	mandate	lately	lately	mandate	mandate	oferrell
kuwaits	kuwaits	resolve	informal	overdue	fcc	commuter	van
algerias	cardboard	fabrics	PSY	ACK	fabrics	kuwaits	fcc
commuter	commuter	kuwaits	referred	fcc	ACK	cardboard	attorneys
exam	fabrics	cardboard	oferrell	oferrell	campbell	fcc	Anticomm
cardboard	minnick	fcc	ACK	corroon	cardboard	turbulence	lately
accuse	glow	commuter	Anitcomm	resolve	solis	fabrics	formation
exceed	theyd	oferrell	clearly	van	sikhs	exam	ACK

Figure 15: The AP data is visualized in a 3D space. 20 topics (K = 20) is chosen for DA-PLSA and the 3-dimensional embedding is computed by DA-GTM. The legend shows well-clustered 8 topics among 20 topics and the top 8 largest probability words in each topic are shown in the table.

4.5 CONCLUSIONS AND FUTURE WORK

In this chapter, we have showed how the PLSA problem can be solved by a mixture model framework and we solve the problem by using a novel algorithm, called DA.

As a result, we have presented a new algorithm, named DA-PLSA, to solve the PLSA problem which originally utilized the EM algorithm for optimal model fitting. Instead of using EM which can cause the local optima problem and the overfitting problem, we applied the DA algorithm to avoid the problems EM suffered. In DA-PLSA, we proposed a new early stopping condition to avoid the overfitting problem and add generality to a model for better predictive power. In addition, we proposed a workflow to visualize large and high-dimensional corpus in a virtual 3-dimensional space, processed by DA-PLSA followed by DA-GTM.

Our experimental results support the new DA-PLSA algorithm outperforms the original EM-PLSA algorithm for getting better optimal solutions and avoiding overfitting problems.

5 SUMMARY AND FUTURE WORK

In this thesis, we have presented finite mixture models to solve two of the well-known data mining algorithms based on a dimension reduction method; i) the GTM algorithm for dimension reduction and visualization and ii) the PLSA algorithm for text mining and retrieval. To solve the challenging model fitting problem arising in the finite mixture models, we propose a new approach using a novel optimization method, called DA. The standard method, using the EM algorithm, notably causes the local optima problem and the overfitting problem. Our newly proposed DA algorithms, named DA-GTM and DA-PLSA, are developed to avoid those problems and give better optimization performance and model fitting for unseen data to increase predicting power which is more important, especially, in the text mining area.

More specifically, in Chapter 3, we show how the GTM problem can be solved in a finite mixture framework and present a DA-based algorithm, DA-GTM, to solve the original GTM problem based on the EM method. Our experiments show the new algorithm is more resilient against the local optima problem and thus less sensitive to initial conditions, from which the original EM-based GTM suffered. In addition, we present a new cooling scheme, called adaptive cooling schedule. In contrast to the conventional cooling schemes, such as a linear and a exponential cooling scheme, which are all pre-defined and fixed, our new adaptive cooling scheme can adjust the granularity of cooling speed in an on-line manner.

In Chapter 4, we show how the PLSA problem can be solved by a mixture model framework and present DA-PLSA as a solution to avoid problems caused by the original EM-based PLSA algorithm suffered. Especially, we propose a new early stopping condition to avoid the overfitting problem and add generality to a model for better predictive power as an important feature in information retrieval. Lastly, we propose a workflow to visualize large and high-dimensional corpus in a virtual 3-dimensional space, processed by DA-PLSA followed by DA-GTM. We provide our experimental results to support the new DA-PLSA algorithm outperforms the original EM-PLSA algorithm for getting better optimal solutions and avoiding the overfitting problem.

There remain many new and exciting research directions to explore for future work. First, it would be interesting to further investigate a dynamic method to decide the optimal number of latent components in the finite mixture model. As the DA algorithm gradually grows the number of latent components in a way as a binary tree grows, it may possible to develop a method to decide a stopping condition for growing the tree. Second, by exploiting the tree structure generated by DA, we can develop a method to infer a hierarchical structure of sample data. Lastly, it would be interesting to perform a performance study for data-intensive analysis by implementing in various high-performance platforms and parallel programming models, such as MPI, MapReduce, or partitioned global address space (PGAS).

Appendices

A DERIVATIVES OF THE FREE ENERGY FUNCTION OF DA-GTM

In this appendix, we derive the first and second derivatives of the free energy for GTM, $\mathcal{F}_{\text{GTM}}$ defined in Eq. (3.12), which we re-write by

$$\mathcal{F}_{\text{GTM}}(\mathbf{Y}, \sigma^2, \beta) = -\frac{1}{\beta}\sum_{i=1}^{N}\log\left\{\left(\frac{1}{K}\right)^{\beta}\sum_{k=1}^{K}\mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}\right\} \tag{A.1}$$

where $\mathcal{N}$ represents a Gaussian distribution defined by

$$\mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{D/2}\exp\left\{-\frac{1}{2\sigma^2}\,\|\,\mathbf{x}_i - \mathbf{y}_k\,\|^2\right\} \tag{A.2}$$

We re-write the responsibility, ρ_{ki}, by

$$\rho_{ki} = \frac{P(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}}{\sum_{k'=1}^{K} P(\mathbf{x}_i \,|\, \mathbf{y}_{k'}, \sigma^2)^{\beta}} \tag{A.3}$$

$$= \frac{\mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}}{\sum_{k=1}^{K} \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}} \tag{A.4}$$

A.1 FIRST DERIVATIVES

The first order derivatives of $\mathcal{F}_{\text{GTM}}$ with respect to latent component $\mathbf{y}_k$ can be written by

$$\frac{\partial \mathcal{F}_{\text{GTM}}}{\partial \mathbf{y}_k} = -\frac{1}{\beta} \sum_{n=1}^{N} \frac{\left(\frac{1}{K}\right)^{\beta} \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}}{\sum_{k=1}^{K} \left(\frac{1}{K}\right)^{\beta} \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}} \beta \left(\frac{1}{\sigma^2}\right) (\mathbf{x}_i - \mathbf{y}_k) \tag{A.5}$$

$$= -\frac{1}{\beta} \sum_{n=1}^{N} \rho_{ki} \beta \left(\frac{1}{\sigma^2}\right) (\mathbf{x}_i - \mathbf{y}_k) \tag{A.6}$$

Since the first order derivatives of $\mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}$ with respect to σ is

$$\frac{\partial \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta}}{\partial \sigma} = \mathcal{N}(\mathbf{x}_i \,|\, \mathbf{y}_k, \sigma^2)^{\beta} \left\{ -D\beta \frac{1}{\sigma} + \beta \frac{1}{\sigma^3} \|\mathbf{x}_i - \mathbf{y}_k\|^2 \right\}, \tag{A.7}$$

the first order derivatives of $\mathcal{F}_{\text{GTM}}$ with respect to σ can be written by,

$$\frac{\partial \mathcal{F}_{\text{GTM}}}{\partial \sigma} = -\frac{1}{\beta} \sum_{n=1}^{N} \rho_{ki} \left\{ -D\beta \frac{1}{\sigma} + \beta \frac{1}{\sigma^3} \|\mathbf{x}_i - \mathbf{y}_k\|^2 \right\} \tag{A.8}$$

Also, the first order derivatives of ρ_{ki} with respect to latent component $\mathbf{y}_k$ and $\mathbf{y}_{k'}$ $(k \neq k')$ can be written by, respectively,

$$\frac{\partial \rho_{ki}}{\partial \mathbf{y}_k} = \left(\rho_{ki} - \rho_{ki}^2\right) \beta \frac{1}{\sigma^2} (\mathbf{x}_i - \mathbf{y}_k) \tag{A.9}$$

$$\frac{\partial \rho_{ki}}{\partial \mathbf{y}_{k'}} = -\rho_{ki} \rho_{k'i} \beta \frac{1}{\sigma^2} (\mathbf{x}_i - \mathbf{y}_{k'}) \tag{A.10}$$

A.2 SECOND DERIVATIVES

The second order derivatives of $\mathcal{F}_{\text{GTM}}$ with respect to latent component $\mathbf{y}_k$ can be written by

$$\frac{\partial}{\partial \mathbf{y}_k^{\text{Tr}}}\left(\frac{\partial \mathcal{F}_{\text{GTM}}}{\partial \mathbf{y}_k}\right) = -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\frac{\partial \rho_{ki}}{\partial \mathbf{y}_k^{\text{Tr}}}\right)\beta\frac{1}{\sigma^2}\left(\mathbf{x}_i - \mathbf{y}_{k'}\right) - \rho_{ki}\beta\frac{1}{\sigma^2}\right\} \tag{A.11}$$

$$= -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\rho_{ki} - \rho_{ki}^2\right)\beta^2\frac{1}{\sigma^4}\left(\mathbf{x}_i - \mathbf{y}_k\right)^{\text{Tr}}\left(\mathbf{x}_i - \mathbf{y}_k\right) - \rho_{ki}\beta\frac{1}{\sigma^2}\right\} \tag{A.12}$$

Similarly, the second order derivative with respect to latent component $\mathbf{y}_{k'}$ can be written by

$$\frac{\partial}{\partial \mathbf{y}_{k'}^{\text{Tr}}}\left(\frac{\partial \mathcal{F}_{\text{GTM}}}{\partial \mathbf{y}_k}\right) = -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\frac{\partial \rho_{ki}}{\partial \mathbf{y}_{k'}^{\text{Tr}}}\right)\beta\frac{1}{\sigma^2}\left(\mathbf{x}_i - \mathbf{y}_{k'}\right)\right\} \tag{A.13}$$

$$= -\frac{1}{\beta}\sum_{n=1}^{N}\left\{-\rho_{ki}\rho_{k'i}\beta^2\frac{1}{\sigma^4}\left(\mathbf{x}_i - \mathbf{y}_{k'}\right)^{\text{Tr}}\left(\mathbf{x}_i - \mathbf{y}_k\right)\right\} \tag{A.14}$$

B DERIVATIVES OF THE FREE ENERGY FUNCTION OF DA-PLSA

In this appendix, we derive the first and the second derivatives of the free energy for PLSA, $\mathcal{F}_{\text{PLSA}}$, defined in Eq. (4.6), which we re-write by

$$\mathcal{F}_{\text{PLSA}} = -\frac{1}{\beta}\sum_{i=1}^{N}\log\sum_{k=1}^{K}\psi_{ki}^{\beta}\text{Multi}\left(\mathbf{x}_i \mid \boldsymbol{\theta}_k\right)^{\beta} \tag{B.1}$$

where Multi represents a multinomial distribution of document $\mathbf{x}_i$ over word probabilities $\boldsymbol{\theta}_k$, defined by

$$\text{Multi}(\mathbf{x}_i \mid \boldsymbol{\theta}_k) = \frac{\Gamma\left(|\,\mathbf{x}_i\,|+1\right)}{\prod_{j=1}^{D}\Gamma(x_{ij}+1)}\prod_{j=1}^{D}\left(\theta_{kj}\right)^{x_{ij}} \tag{B.2}$$

where $\Gamma(\cdot)$ is the gamma function, an extension of the factorial function such as $\Gamma(n) = (n-1)!$.

We re-write the responsibility, ρ_{ki}, by

$$\rho_{ki} = \frac{\psi_{ki}^{\beta}\text{Multi}\left(\mathbf{x}_i \mid \boldsymbol{\theta}_k\right)^{\beta}}{\sum_{k'=1}^{K}\psi_{k'i}^{\beta}\text{Multi}\left(\mathbf{x}_i \mid \boldsymbol{\theta}_{k'}\right)^{\beta}} \tag{B.3}$$

B.1 FIRST ORDER DERIVATIVES

The first order derivatives of $\mathcal{F}_{\text{PLSA}}$ with respect to word probability θ_k can be written by

$$\frac{\partial \mathcal{F}_{\text{PLSA}}}{\partial \theta_k} = -\frac{1}{\beta}\sum_{i=1}^{N}\frac{\psi_{ki}^{\beta}\text{Multi}\left(x_i \mid \theta_k\right)^{\beta}}{\sum_{k'=1}^{K}\psi_{k'i}^{\beta}\text{Multi}\left(x_i \mid \theta_{k'}\right)^{\beta}}\left(\beta x_i \theta_k^{-1}\right) \tag{B.4}$$

$$= -\frac{1}{\beta}\sum_{i=1}^{N}\rho_{ki}\beta x_i\theta_k^{-1} \tag{B.5}$$

The first order derivatives of ρ_{ki} with respect to word probability θ_k and $\theta_{k'}$ $(k \neq k')$ can be written by, respectively,

$$\frac{\partial \rho_{ki}}{\partial \theta_k} = \left(\rho_{ki} - \rho_{ki}^2\right)\beta x_i\theta_k^{-1} \tag{B.6}$$

$$\frac{\partial \rho_{ki}}{\partial \theta_{k'}} = -\rho_{ki}\rho_{k'i}\beta x_i\theta_k^{-1} \tag{B.7}$$

B.2 SECOND ORDER DERIVATIVES

The second order derivatives of $\mathcal{F}_{\text{PLSA}}$ with respect to word probability θ_k can be written by

$$\frac{\partial}{\partial \theta_k^{\text{Tr}}}\left(\frac{\partial \mathcal{F}_{\text{PLSA}}}{\partial \theta_k}\right) = -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\frac{\partial \rho_{ki}}{\partial \theta_k^{\text{Tr}}}\right)\beta x_i\theta_k^{-1} + \rho_{ki}\beta\frac{\partial\left(x_i\theta_k^{-1}\right)}{\partial \theta_k^{\text{Tr}}}\right\} \tag{B.8}$$

$$= -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\rho_{ki} - \rho_{ki}^2\right)\beta^2\left(x_i\theta_k^{-1}\right)^{\text{Tr}}\left(x_i\theta_k^{-1}\right) - \rho_{ki}\beta\text{diag}\left(x_i\theta_k^{-2}\right)\right\} \tag{B.9}$$

where $\text{diag}(\mathbf{d})$ represents a diagonal matrix whose diagonal element is vector $\mathbf{d}$.

Similarly, the second order derivative with respect to word probability $\theta_{k'}$ can be written by

$$\frac{\partial}{\partial \theta_{k'}^{Tr}}\left(\frac{\partial \mathcal{F}_{PLSA}}{\partial \theta_k}\right) = -\frac{1}{\beta}\sum_{n=1}^{N}\left\{\left(\frac{\partial \rho_{ki}}{\partial \theta_{k'}^{Tr}}\right)\beta x_i \theta_k^{-1} + \rho_{ki}\beta\frac{\partial\left(x_i\theta_k^{-1}\right)}{\partial \theta_{k'}^{Tr}}\right\} \tag{B.10}$$

$$= -\frac{1}{\beta}\sum_{n=1}^{N}\left\{-\rho_{ki}\rho_{k'i}\beta\left(x_i\theta_k^{-1}\right)^{Tr}\left(x_i\theta_k^{-1}\right)\right\} \tag{B.11}$$

BIBLIOGRAPHY

[1] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75:245–248, 2009. 10.1007/s10994-009-5103-0. (Cited on page 2.)

[2] C. Bishop and S. S. en ligne). *Pattern recognition and machine learning*, volume 4. springer New York, 2006. (Cited on page 17.)

[3] C. Bishop, M. Svensén, and C. Williams. GTM: A principled alternative to the self-organizing map. *Advances in neural information processing systems*, pages 354–360, 1997. (Cited on pages 11, 19, 21, 24, 30, and 31.)

[4] C. Bishop, M. Svensén, and C. Williams. GTM: The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998. (Cited on pages 8, 11, 19, 21, 24, 30, and 31.)

[5] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003. (Cited on pages 2, 8, 35, 37, 42, and 55.)

[6] L. Chen, T. Zhou, and Y. Tang. Protein structure alignment by deterministic annealing. *Bioinformatics*, 21(1):51–62, 2005. (Cited on page 12.)

[7] J. Y. Choi, S.-H. Bae, J. Qiu, B. Chen, and D. Wild. Browsing large scale cheminformatics data with dimension reduction. *Concurrency and Computation: Practice and Experience*, 2011. (Cited on page 5.)

[8] J. Y. Choi, S.-H. Bae, J. Qiu, G. Fox, B. Chen, and D. Wild. Browsing large scale cheminformatics data with dimension reduction. In *Workshop on Emerging Computational Methods for Life Sciences (ECMLS), in conjunction with the 19th ACM International Symposium on High Performance Distributed Computing (HPDC) 2010*, HPDC '10, pages 503–506, Chicago, Illinois, June 2010. ACM. (Cited on page 5.)

[9] J. Y. Choi, S.-H. Bae, X. Qiu, and G. Fox. High performance dimension reduction and visualization for large high-dimensional data analysis. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:331–340, 2010.

[10] J. Y. Choi, J. Qiu, M. Pierce, and G. Fox. Generative Topographic Mapping by Deterministic Annealing. In *Proceedings of the 10th International Conference on Computational Science and Engineering (ICCS 2010)*, 2010. (Cited on pages 5 and 8.)

[11] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990. (Cited on page 36.)

[12] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. (Cited on pages 2 and 10.)

[13] M. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on pattern analysis and machine intelligence*, pages 381–396, 2002. (Cited on pages 1 and 2.)

[14] G. Furnas, S. Deerwester, S. Dumais, T. Landauer, R. Harshman, L. Streeter, and K. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th annual international ACM SIGIR*

conference on Research and development in information retrieval, pages 465–480. ACM, 1988. (Cited on page 36.)

[15] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999. (Cited on pages 2, 3, 8, 11, 35, 39, and 42.)

[16] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, 2001. (Cited on pages 8, 11, 18, 35, 39, 40, and 42.)

[17] T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997. (Cited on pages 2 and 12.)

[18] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, 2000. (Cited on pages 2 and 8.)

[19] E. Jaynes. Information theory and statistical mechanics. II. *Physical review*, 108(2):171–190, 1957. (Cited on page 12.)

[20] E. Jaynes. Information theory and statistical methods I. *Physics Review*, 106(1957):620–630, 1957.

[21] E. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, 1982. (Cited on page 12.)

[22] S. Kirkpatric, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. (Cited on page 12.)

[23] H. Klock and J. Buhmann. Multidimensional scaling by deterministic annealing. *Lecture Notes in Computer Science*, 1223:245–260, 1997. (Cited on pages 2 and 12.)

[24] G. McLachlan and D. Peel. *Finite mixture models*, volume 299. Wiley-Interscience, 2000. (Cited on pages 1 and 2.)

[25] T. Mitchell. Machine learning and data mining. *Communications of the ACM*, 42(11):30–36, 1999. (Cited on page 17.)

[26] K. Nigam. *Using unlabeled data to improve text classification*. PhD thesis, Carnegie Mellon University, 2001. (Cited on page 18.)

[27] D. Reynolds, T. Quatieri, and R. Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000. (Cited on pages 2 and 36.)

[28] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998. (Cited on pages 2, 3, 11, 12, 13, 14, 22, and 40.)

[29] K. Rose, E. Gurewitz, and G. Fox. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 11(9):589–594, 1990. (Cited on page 40.)

[30] K. Rose, E. Gurewitz, and G. Fox. Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948, 1990. (Cited on page 14.)

[31] K. Rose, E. Gurewitz, and G. Fox. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, 1992. (Cited on pages 2, 3, and 12.)

[32] V. Sindhwani, S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of the 23rd international conference on Machine learning*, pages 841–848. ACM New York, NY, USA, 2006. (Cited on page 2.)

[33] D. Smith and J. Eisner. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 787–794. Association for Computational Linguistics, 2006. (Cited on page 18.)

[34] N. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 486. Association for Computational Linguistics, 2004. (Cited on page 18.)

[35] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999. (Cited on pages 2 and 36.)

[36] D. Titterington, A. Smith, U. Makov, et al. *Statistical analysis of finite mixture distributions*, volume 38. Wiley New York, 1985. (Cited on page 1.)

[37] N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998. (Cited on pages 2 and 11.)

[38] X. Yang, Q. Song, and Y. Wu. A robust deterministic annealing algorithm for data clustering. *Data & Knowledge Engineering*, 62(1):84–100, 2007. (Cited on pages 2 and 12.)

[39] A. Yuille and J. Kosowsky. Statistical physics algorithms that converge. *Neural Computation*, 6(3):341–356, 1994. (Cited on page 14.)