Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Learning & Adaptive Systems

Master Thesis

Spatial Summarization of Image Collections

Spring Term 2016

This dissertation is submitted for the degree of *Master of Science ETH in Computer Science*

Supervised by:
Prof. Dr. Andreas Krause
Dr. Sebastian Tschiatschek
Alkis Gotovos, M.Sc.

Author:
Diego Alfonso Ballesteros Villamizar

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

List of Figures

List of Tables

Preface

Acknowledgments and such …

Abstract

The Abstract …

Symbols

Symbols

η	Learning rate
γ	Model parameter
κ	Largest subset size
ν	Noise-to-data ratio
σ	Gaussian distribution's standard deviation
$\mathcal{D}$	Set of data samples
K	Latent coherence dimensions
L	Latent diversity dimensions
M	Number of features
$\mathcal{N}$	Set of noise samples
$\mathbb{R}$	Real numbers
S, T	Subset
V	Ground set
Z	Normalization constant

Matrices

$\mathbf{B}$	Feature diversity weights
$\mathbf{E}$	Feature coherence weights
$\mathcal{I}$	Identity
$\mathbf{W}^b$	Coherence weights
$\mathbf{W}^e$	Diversity weights
$\mathbf{X}$	Features

Vectors

$\boldsymbol{\theta}$	Model parameters
$\mathbf{a}$	Utility weights for features
$\mathbf{u}$	Utility weights
$\mathbf{G}$	Accumulated gradient

Indices

i, j	Element
k	Feature
c	Coherence dimension
d	Diversity dimension

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
FLID	Facility Location Diversity
FLDC	Facility Location Diversity and Coherence
FFLDC	Featurized Facility Location Diversity and Coherence
MLE	Maximum Likelihood Estimation
NCE	Noise Contrastive Estimation
LAS	Learning & Adaptive Systems
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

Motivation (tourist routes and summarization), and quick idea of the methodology.

1.1 Contributions

Summarization of contributions.

Chapter 2

Related Work

2.1 Probabilistic Sub/super-modular models

Talk about the importance of these models and the recent interest in them, e.g. DPPs, Sampling.

2.2 Image collection summarization

Generally about the problem and the different approaches.

2.3 Tourist routes recommendation

The initial papers I found on this, basically Markov chains and clustering.

Chapter 3

Background

This chapter presents the supporting concepts and methods that will be used throughout this work.

3.1 Submodular Probability Set Functions

As described before, probability models over sets are of great interest and can be applied to diverse settings. In particular, submodular probability set functions have been utilized in multiple domains, such as document summarization and information gathering (Krause and Golovin, 2014).
Probability set functions are a class of distributions P over subsets S of a ground set V, i.e. functions that satisfy $\forall S \subseteq V, P : 2^V \to \mathbb{R} \mid 0 \leq P(S) \leq 1$ and $\sum_{S \subseteq V} P(S) = 1$. These distributions can also be understood as probabilities over binary sequences $[a_1, \ldots, a_n]$ of fixed size $n = |V|$ (Bruno et al., 1999). Hereby, w.l.o.g., let $V = \{1, \ldots, n\}$.
A set function $F : 2^V \to \mathbb{R}$ is submodular if it exhibits a "diminishing returns" property (Krause and Golovin, 2014), namely if it satisfies:

$$\forall S, T \subseteq V : S \subseteq T, i \notin T \mid F(S \cup i) - F(S) \geq F(T \cup i) - F(T) \tag{3.1}$$

Intuitively this indicates that adding an element to a smaller set yields a larger effect, i.e. in terms of the set function F value, than adding it to a larger one. This is a natural property in the context of summarization where adding more information to a large summary can be less effective than adding it to a smaller one. This property also makes them good candidates for modeling the concept of diversity (Tschiatschek et al., 2016).

3.2 Supermodularity and Modularity

Analogous to the use of submodular functions to model diversity, supermodular functions can be used to model complementarity or coherence between elements.
A set function is supermodular if it satisfies the condition in (3.1) with the inequality sign reversed, i.e. a set function $F(S)$ is supermodular iff $-F(S)$ is submodular. Supermodularity is used extensively in economics to model complementarity between strategies in games (Amir, 2005).
Finally, if a function F is both submodular and supermodular, i.e. it satisfies condition (3.1) with equality, then it is said to be a modular function. These are the simplest examples of submodular or supermodular functions and are akin to linear functions but in a discrete domain (Krause and Golovin, 2014).

3.3 Probabilistic Log-sub/supermodular Models

This work is interested in a particular class of probability set functions, namely probabilistic log-submodular and log-supermodular models, which are probabilities $P(S)$ of the form (Djolonga and Krause, 2014):

$$P(S) \propto \exp(F(S)) \tag{3.2}$$

Where $F(S)$ is a submodular or supermodular function, respectively. These models encompass many practical probabilistic models such as repulsive Ising models, Deterministic Point Processes (DPPs) and binary pairwise Markov Random Fields (MRFs) (Djolonga and Krause, 2014, 2015).
This model class is of interest because the contributions of this work focus on extensions to a novel probabilistic log-submodular model proposed by Tschiatschek et al. (2016) which will be described in the next section.

3.3.1 Partition Function

In log-submodular models the normalization constant Z is known as the *partition function* (Djolonga and Krause, 2014). Z is necessary to fully determine the normalized model and compute quantities such as marginal probabilities, however its exact computation is known to be #P-complete (Jerrum and Sinclair, 1990). This makes it impossible in practice to perform such computations except for special cases of the model.

3.4 Facility Location Diversity Model (FLID)

Every modular function can be written as a sum of individual weights for each element $i \in V$ assuming a normalization such that $F(\emptyset) = 0$ (Krause and Golovin, 2014), i.e.

$$F(S) = \sum_{i \in S} u(i) \tag{3.3}$$

Where $u(i)$ is some function $u : V \to \mathbb{R}$. For convenience, denote $\mathbf{u}$ as the vector of modular weights where $u_i = u(i)$. Therefore any log-modular function can be written as:

$$P(S) \propto \exp\left(\sum_{i \in S} u_i\right) = \prod_{i \in S} \exp(u_i) \tag{3.4}$$

This is the simplest log-submodular probability function and is the modular part for the FLID model proposed by Tschiatschek et al. (2016). In this model, $\mathbf{u}$ can be thought of as modeling the relevance or utility of each element, for example in the context of spatial summarization this utility could be proportional to the popularity of a location or to how many times it has been photographed.
However, these utilities alone can not capture interactions between the elements in a set. To address this, Tschiatschek et al. (2016) propose an additional term that models set diversity, its objective is to characterize redundant elements and assign lower probabilities to sets that contain them.
This term is based on representing each element i with an L-dimensional vector $\mathbf{w}^b \in \mathbb{R}^L_{\geq 0}$, where each dimension $1 \leq d \leq L$ captures a concept related to set diversity and $w^b_{i,d}$ quantifies the relevance of each element i for each concept d (Tschiatschek et al., 2016). Hereby, define $\mathbf{W}^b \in \mathbb{R}^{|V| \times L}_{\geq 0}$ as the matrix where the i-th row is the representation $\mathbf{w}^b$ of element i.
For each dimension d, the diversity of a set S is quantified by $\max_{i \in S} w^b_{i,d} - \sum_{i \in S} w^b_{i,d}$. This assigns a negative value for sets that contain more than one element with nonzero weight

Table 3.1: FLID probability distribution for the scenario in Example 3.4.2.

S	$P(S)$
$\{h,s\},\{h,f\}$	≈ 0.41
$\{h\},\{s\},\{f\}$	≈ 0.06
$\{\},\{s,f\},\{h,s,f\}$	≈ 0.00

$w^b_{\cdot,d}$ in that dimension (Tschiatschek et al., 2016). Equation (3.5) shows the complete term, this sums over all L dimensions to account the diversity in each concept.

$$\operatorname{div}(S) = \sum_{d=1}^{L} \left(\max_{i \in S} w^b_{i,d} - \sum_{i \in S} w^b_{i,d} \right) \tag{3.5}$$

Finally, the complete FLID model proposed by Tschiatschek et al. (2016) combines these two terms and is given by:

$$P(S) = \frac{1}{Z} \exp\left(\sum_{i \in S} u_i + \sum_{d=1}^{L} \left(\max_{i \in S} w^b_{i,d} - \sum_{i \in S} w^b_{i,d} \right) \right) \tag{FLID}$$

3.4.1 Partition Function

As mentioned before, the computation of the partition function is generally intractable for log-submodular models. However, for FLID this can be computed in time $O(|V|^{L+1})$ which is polynomial on the size of the ground set and significantly better than the cost of enumerating the powerset of V, i.e. $O(2^{|V|})$ (Tschiatschek et al., 2016). Nevertheless, it is worth noting that this complexity is exponential in L which means that the partition function computation is only pratical for a limited range of FLID models, namely those with $L \ll |V|$.

3.4.2 Example: Two Landmarks

In order to better illustrate the model, consider a town with 3 popular locations: A town hall (h), a statue (s) and a fountain (f). Assume that historic data shows that visitors only take photos at either the town hall and the statue, or at the town hall and the fountain. This can be modeled with FLID by introducing a latent dimension representing some concept that is present in both the statue and the fountain but not in the town hall, e.g. "is not a building".

Concretely, let $V = \{h,s,f\}$ and $\mathbf{u} = (2,2,2)^\intercal$, indicating that all locations are equally popular. A suitable diversity weight matrix would then be $\mathbf{W}^b = (0,20,20)^\intercal$. Table 3.1 shows the resulting probabilities of the subsets, accurately representing the aforementioned problem. Note that the probabilities are not exactly $^{0.5}/_{0.5}$ for the sets of interest but this could be easily fixed by increasing the utilities to ensure that sets of size 2 have a larger unnormalized magnitude compared to individual ones.

3.5 Learning from Data

Much of the interest in log-submodular models is concentrated on performing inference for known functions (Tschiatschek et al., 2016). However, Tschiatschek et al. (2016) explore how to learn such models from data, i.e. estimate the model parameters from observations with an unknown distribution.

3.5.1 Learning Log-modular Models

A special case of log-submodular models where the parameters can be estimated efficiently using Maximum Likelihood Estimation (MLE) is the log-modular model presented in Equation (3.4). For this basic model, it is possible to estimate the utilities $\mathbf{u}$ through the maximum likelihood estimator for the marginals $\hat{P}(i \in S)$. These marginals are given by[1]:

$$P(i \in S) = \frac{1}{1 + \exp(-u_i)} \tag{3.6}$$

And its maximum likelihood estimator for a dataset $\mathcal{D}$ is:

$$\hat{P}(i \in S) = \frac{N(\#i \in S)}{|\mathcal{D}|} \tag{3.7}$$

Therefore, the utilities can be estimated as:

$$u_i = -\log\left(\frac{1}{\hat{P}(i \in S)} - 1\right) \tag{3.8}$$

This model also allows computing the partition function in closed form, this is given by:

$$Z = \prod_{i \in S}(1 + \exp(u_i)) \tag{3.9}$$

Because these quantities can be computed in closed form, it is possible to efficiently estimate the parameters of a log-modular model and sample from it. This is useful for producing noise samples and constructing baselines for the experiments.
Even though MLE is the commonly used method for the task of parameter estimation from data, it only works efficiently for normalized models (Gutmann and Hyvärinen, 2012). This makes it intractable for the general class of log-submodular models, and also a wide range of FLID models because the computation of Z is exponential on L. Gutmann and Hyvärinen (2012) propose an alternative method for parameter estimation in unnormalized models known as Noise Contrastive Estimation (NCE) and this is the method used by Tschiatschek et al. (2016) to learn the FLID model.

3.5.2 NCE Learning

The idea behind NCE is to transform the unsupervised learning task of estimating a probability density from data into a supervised classification task. In order to do this, the observed data $\mathcal{D}$, assumed to be drawn from an unknown distribution P_d, is compared to an artificially generated set of noise samples $\mathcal{N}$ drawn from a known distribution P_n that can be efficiently normalized. The classification task is to maximize the likelihood of correctly discriminating each sample as either observed data or artificial noise.
Formally, denote $\mathcal{A}$ as the complete set of labeled samples, i.e. $\mathcal{A} = \{(S, Y_s) : S \in \mathcal{D} \cup \mathcal{N}\}$ where $Y_s = 1$ for $S \in \mathcal{D}$ and $Y_s = 0$ for $S \in \mathcal{N}$. Additionally, let ν be the noise-to-data ratio, i.e. $\nu = |\mathcal{N}|/|\mathcal{D}|$.
The goal is to estimate the posterior probabilities $P(Y_s = 1 \mid S; \boldsymbol{\theta})$ and $P(Y_s = 0 \mid S; \boldsymbol{\theta})$, in order to discriminate noise from data samples. These probabilities are given by equations (3.10) and (3.11).

$$P(Y_s = 1 \mid S; \boldsymbol{\theta}) = \frac{\hat{P}_d(S; \boldsymbol{\theta})}{\hat{P}_d(S; \boldsymbol{\theta}) + \nu P_n(S)} \tag{3.10}$$

$$P(Y_s = 0 \mid S; \boldsymbol{\theta}) = \frac{\nu P_n(S)}{\hat{P}_d(S; \boldsymbol{\theta}) + \nu P_n(S)} \tag{3.11}$$

[1]Refer to the appendix for the derivation of these equations.

It is worth noting that $\hat{P}_d$ is used instead of P_d, because the real density is not known. As indicated by Gutmann and Hyvärinen (2012), $\hat{P}_d$ can be an unnormalized distribution for NCE where the partition function Z is included in the set of parameters $\boldsymbol{\theta}$ as $\hat{Z}$, hence resulting in an approximately normalized distribution after the optimization.
In order to obtain these posterior probabilities, the following conditional log-likelihood objective is maximized (Gutmann and Hyvärinen, 2012):

$$g(\boldsymbol{\theta}) = \sum_{S \in \mathcal{D}} \log P(Y_s = 1 \mid S; \boldsymbol{\theta}) + \sum_{S \in \mathcal{N}} \log P(Y_s = 0 \mid S; \boldsymbol{\theta}) \tag{3.12}$$

This maximization can be performed using a gradient-based optimization method such as Stochastic Gradient Descent (SGD).

Practical Considerations

A couple of considerations are necessary for obtaining good estimates with NCE (Gutmann and Hyvärinen, 2012):

1. The parameterized probability function $\hat{P}_d(S;\theta)$ must be of the same family as the real distribution P_d, i.e. $\exists \boldsymbol{\theta}^* \mid \hat{P}_d(S;\boldsymbol{\theta}^*) = P_d$.
2. The noise distribution P_n is nonzero whenever P_d is nonzero.

Additionally, the following statements must be considered when choosing the noise distribution P_n (Gutmann and Hyvärinen, 2012):

1. A distribution that can be sampled easily.
2. Noise that is as similar as possible to the data, otherwise the classification problem could be too easy.
3. A noise sample as large as possible.

3.5.3 Learning FLID via NCE and SGD

Stochastic Gradient Descent was used by Tschiatschek et al. (2016) to learn the FLID model through NCE. SGD is a gradient-based method that has proven effective in large-scale learning tasks due to its efficiency when the computation time is a limiting factor (Bottou, 2010; Zhang, 2004), hence making it appropriate for the scale of data sourced from the internet, such as public user photos in Flickr.
In each iteration, the gradient $\nabla \log P(Y_s = y \mid S; \boldsymbol{\theta})$ must be computed. For FLID, the parameter vector $\boldsymbol{\theta}$ is composed by $\mathbf{u}, \mathbf{W}^b$ and $\hat{Z}$, and the corresponding gradients are given by:

$$\nabla \log P(Y_s = y \mid S; \boldsymbol{\theta}) = \left(y - \frac{1}{1 + \nu \frac{P_n(S)}{\hat{P}_d(S;\boldsymbol{\theta})}} \right) \nabla \log \hat{P}_d(S; \boldsymbol{\theta}) \tag{3.13}$$

$$\hat{P}_d(S;\boldsymbol{\theta}) = \frac{1}{\hat{Z}} P(S; \mathbf{u}, \mathbf{W}^b) \tag{3.14}$$

$$\left(\nabla_{\mathbf{u}} \log \hat{P}_d(S; \hat{Z}, \mathbf{u}, \mathbf{W}^b)\right)_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

$$\left(\nabla_{\mathbf{W}^b} \log \hat{P}_d(S; \hat{Z}, \mathbf{u}, \mathbf{W}^b)\right)_{i,d} = \begin{cases} -1 & \text{if } i \in S \text{ and } i \neq \operatorname{argmax}_{j \in S} w^b_{j,d} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

$$\nabla_{\hat{Z}} \log \hat{P}_d(S; \hat{Z}, \mathbf{u}, \mathbf{W}^b) = \frac{-1}{\hat{Z}} \tag{3.17}$$

Where $P(S; \mathbf{u}, \mathbf{W}^b)$ is the unnormalized (FLID) equation, $(\nabla_{\mathbf{u}} \cdot)_i$ represents the i-th entry of the gradient with respect to $\mathbf{u}$ and $(\nabla_{\mathbf{W}^b} \cdot)_{i,d}$ represents the (i,d)-th entry of the sub-gradient with respect to $\mathbf{W}^b$ (Tschiatschek et al., 2016).

Computational Complexity

As described by Tschiatschek et al. (2016), the running time requirement for calculating the sub-gradient for FLID is $O(L|S|)$, and for a complete pass over data and noise samples is then $O(|\mathcal{D} \cup \mathcal{N}|\kappa L)$ where $\kappa = \max_{S \in \mathcal{D} \cup \mathcal{N}} |S|$. This shows that learning is efficient as it is only linear on the size of the data and the noise, assuming small values for κ and L.

3.5.4 AdaGrad

A commonly used learning rate function for SGD is $\eta(t) = \eta_0 t^{-p}$, which offers the best yergence speed in theory and also does it often in practice (Bottou, 2012), however it can lead to poor performance due to slow rates of convergence to the solution (Darken and Moody, 1992). By contrast, choosing a larger η_0 may not lead to better results due to the instability in the parameters for small t (Darken and Moody, 1990).
This behavior was observed during the experiments with real data to be presented in later sections, therefore an alternative for the learning rate was sought. In particular, Adaptive Gradient (AdaGrad) was implemented.
AdaGrad was proposed by Duchi et al. (2011), the idea of this method is to adapt the learning rate for each parameter in $\boldsymbol{\theta}$ individually in a way that frequently updated parameters have slower learning rates while infrequent parameters have larger ones. The intuition is that an update to a rare parameter is more informative than one to a parameter that is frequently updated.
Concretely, with AdaGrad the update step for each parameter γ in $\boldsymbol{\theta}$ is given by:

$$\theta_\gamma^{\tau+1} \leftarrow \theta_\gamma^\tau - \frac{\eta}{\sqrt{G_\gamma}} \nabla g(\theta)_\gamma^\tau \tag{3.18}$$

Where $\mathbf{G}^\tau$ is the vector of accumulated gradients at time τ and each component G_γ^τ is given by:

$$G_\gamma^\tau = \sum_{t=1}^{\tau} \left(\nabla g(\theta)_\gamma^t \right)^2 \tag{3.19}$$

Where $\nabla g(\theta)_\gamma^t$ denotes the gradient of the γ parameter at time step t.
It is worth noting that AdaGrad does not incur in an significant increase in running time or memory requirements for the training which makes it inexpensive to include in the implementation of NCE.

Chapter 4

Extending FLID

4.1 Beyond Diversity: FLDC

Diversity is an important property in the context of summarization (Tschiatschek et al., 2016), however it is not the only one. Coherence is also a desired property of summaries, particularly for structured summaries (Yan et al., 2011). Balancing coherence and diversity is considered a challenge and maxiziming diversity alone may lead to disconnected summaries, while focusing on coherence can hinder the breadth of the resulting summaries (Shahaf et al., 2012).

To better understand why coherence is important, consider the construction of a spatial summary for photos taken in a city. A model based on spatial diversity would create sets containing photos taking at distant locations in the city, however another possibility is to have summaries that contain photos that were taken close to some starting point, in this case the desired property is not spatial diversity but rather coherence.

Therefore, a natural extension to the FLID model is to add the capacity of modeling coherence along with diversity. I propose the addition of a log-supermodular term analogous to the log-submodular term (3.5) in order to model the complementarity between elements. This supermodular term introduces a representation of the elements using K-dimensional vectors $\mathbf{w}^K \in \mathbb{R}^K_{\geq 0}$, where each dimension $1 \leq c \leq K$ captures a concept that relates to set coherence and $w_{i,c}$ quantifies the contribution of each element i to each coherence concept c. The coherence of a set S with respect to each concept c is quantified as $\sum_{i\in S} w^e_{i,c} - \max_{i\in S} w^e_{i,c}$ which is a supermodular term mirroring the submodular term in FLID. Summing over all the concepts results in the proposed coherence term. Hereby, define $\mathbf{W}^e \in \mathbb{R}^{|V|\times K}_{\geq 0}$ as the matrix where the i-th row is the representation $\mathbf{w}^e$ of element i.

Adding the coherence term to FLID results in the extended model, which I refer to as the Facility Location Diversity and Coherence (FLDC) model. Its probability distribution is defined as:

$$P(S) = \frac{1}{Z}\exp\left(\sum_{i\in S} u_i + \operatorname{div}(S) + \sum_{c=1}^{K}\left(\sum_{i\in S} w^e_{i,c} - \max_{i\in S} w^e_{i,c}\right)\right) \tag{FLDC}$$

4.1.1 Partition Function

For the FLDC model, the partition function can be computed in $O(|V|^{L+K+1})$ time. This complexity can be derived using a similar algorithm as the one presented by Tschiatschek et al. (2016) for FLID, the key observation is that the algorithm they present only requires the ordering of weights to determine elegible element for a set given a set of maximum weights per dimension, therefore this can be easily extented to include the the ordering

Table 4.1: Probability distribution for Example 4.1.3.

S	$P(S)$
$\{1,2\}$	0.5
$\{3,4\}$	0.5
Otherwise	0.0

for the coherence weights as well. This requires evaluating K more dimensions which results in the added term to the exponent of the complexity expression, which makes the computation of the partition function more impractical than in the FLID case.

4.1.2 Learning FLDC Models

As with FLID, the parameters for an FLDC model can be estimated using NCE. For FLDC the vector $\boldsymbol{\theta}$ is composed of $[\mathbf{u}, \mathbf{W}^b, \mathbf{W}^e, \hat{Z}]$ and its gradient is the same as in equations (3.13)-(3.17) with the addition of a gradient with respect to the new parameter, $\mathbf{W}^e$, which is given by:

$$\left(\nabla_{\mathbf{W}^e} \log \hat{P}_d(S; \hat{Z}, \mathbf{u}, \mathbf{W}^e, \mathbf{W}^e)\right)_{i,c} = \begin{cases} 1 & \text{if } i \in S \text{ and } i \neq \operatorname{argmax}_{j \in S} w^e_{j,c} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Where $(\nabla_{\mathbf{W}^e} \cdot)_{i,c}$ represents the (i,c)-th entry in the gradient with respect to $\mathbf{W}^e$.
This also affects the overall complexity of the learning algorithm. Computing the gradient takes $O(L+K)|S|$ for FLDC at each step, therefore the running time of a full pass over the data and noise samples takes $O(|\mathcal{D} \cup \mathcal{N}|\kappa(L+K))$. This is not significantly more expensive than learning FLID under the assumption that $K \simeq L$.

4.1.3 Example: Two Non-overlapping Clusters

As an example of the extended model, consider the distribution presented in table 4.1 for $V = \{1,2,3,4\}$, representing a set of two non-overlapping clusters with 2 elements each. Intuitively, this can be modeled by considering that there is diversity between elements in different clusters while inside a cluster there is a coherence component that ties them together.
Concretely, the weight matrices $\mathbf{W}^b, \mathbf{W}^e$ in Figure 4.1 illustrate one possible instance of the model. The corresponding utility vector is $\mathbf{u} = \overrightarrow{0}$, because there is no indication that an individual element is favored over the pairs. This model is easily interpretable and accurately realizes the distribution.

4.2 Generalizing through Features: FFLDC

An important characteristic of the FLID model, and by extension the FLDC one, is that it is agnostic to the type of elements in the ground set, this allows its application to a wide range of problems without prior knowledge. However the downside is that the model has no capability to make use of information about the elements, if available, to improve the modeling of the data. Moreover, if a new element is added to the set there is no way to generalize the existing knowledge about similar elements.
In order to solve these problems, I propose a further extension to the model. Firstly, an element ($i \in V$) is represented as a vector $\mathbf{x}_i \in \mathbb{R}^M$ where each entry is a feature, e.g. for venues one feature could be its aggregated rating while another indicates whether it is

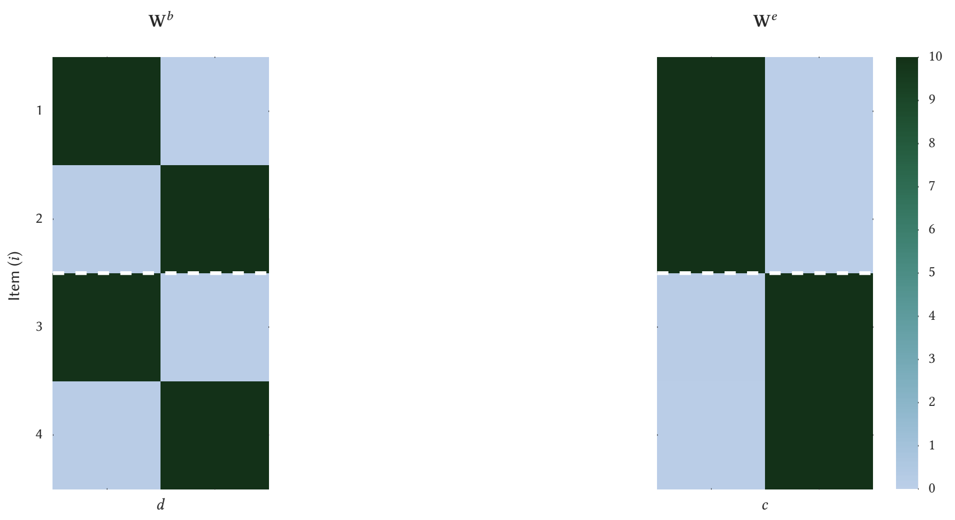

Figure 4.1: Diversity and coherence weights for FLDC model in Example 4.1.3. The dotted line divides the clusters.

indoors or outdoors. The representations of all elements is collected by matrix $\mathbf{X} \in \mathbb{R}^{|V| \times M}$ where each row i corresponds to the repreesntation $\mathbf{x}_i$ of element i.
Then, the utility vector $\mathbf{u}$ and weight matrices $\mathbf{W}^b, \mathbf{W}^e$ are factorized to include this feature matrix $\mathbf{X}$ as follows:

$$\mathbf{u} = \mathbf{Xa} \tag{4.2}$$
$$\mathbf{W}^b = \mathbf{XB} \tag{4.3}$$
$$\mathbf{W}^e = \mathbf{XE} \tag{4.4}$$

Where $\mathbf{a} \in \mathbb{R}^M$ represents the contribution of each feature to the total utility of an item, while $\mathbf{B} \in \mathbb{R}^{M \times L}$ and $\mathbf{E} \in \mathbb{R}^{M \times K}$ encode the contribution of each feature to each latent diversity and coherence dimension, respectively. The intuition behind this factorization is that the information about the items can enhance the latent representations, hence producing a richer model. For example if $M < |V|$ and the features are sufficient to represent the items, it is expected that learning will be easier because of the reduced number of parameters with respect to learning the equivalent FLDC model.
I refer to the extended model as the Featurized Facility Location Diversity and Coherence (FFLDC) model and its probability distribution is defined as:

$$P(S) = \frac{1}{Z} \exp\left(\sum_{i \in S} \sum_{k=1}^{M} x_{i,k} a_k + fdiv(S) + fcoh(S) \right) \tag{FFLDC}$$
$$fdiv(S) = \sum_{d=1}^{L} \left(\max_{i \in S} \sum_{k=1}^{M} x_{i,k} b_{k,d} - \sum_{i \in S} \sum_{k=1}^{M} x_{i,k} b_{k,d} \right) \tag{4.5}$$
$$fcoh(S) = \sum_{c=1}^{K} \left(\sum_{i \in S} \sum_{k=1}^{M} x_{i,k} e_{k,c} - \max_{i \in S} \sum_{k=1}^{M} x_{i,k} e_{k,c} \right) \tag{4.6}$$

Where a_i is the i-th entry of $\mathbf{a}$, $b_{k,d}$ is the (k,d)-th entry of $\mathbf{B}$, $e_{k,d}$ is the (k,e)-th entry of $\mathbf{E}$ and $x_{i,k}$ is the (i,k)-th entry of X.

Remark. If $\mathbf{X} = \mathcal{I}$, then FFLDC is equivalent to FLDC, with $\mathbf{a} = \mathbf{u}$, $\mathbf{W}^b = \mathbf{B}$ and $\mathbf{W}^e = \mathbf{E}$.

The use of features also allows the application of the model to previously unseen elements, hence solving the aforementioned problem of generalization. This is possible because the model is defined on the space of features and not items, and for a previously unseen element j only its representation $\mathbf{x}_j$ is required in order to evaluate the probability of sets containing this element.

4.2.1 Partition Function

The computation of the partition function for this model can be done using the fact that any FFLDC model can be transformed into an equivalent FLDC model through the factorizations in equations (4.2)-(4.4). This transformation requires $\mathcal{O}(|V|M(L+K))$ time for the matrix multiplications, therefore the overall complexity of calculating the partition function for an FFLDC model is $\mathcal{O}(|V|M(L+K)+|V|^{L+K+1})$. This expression can be simplified to $\mathcal{O}(|V|^{L+K+1})$ because the exponential term is significantly larger assuming sensible values for M. Given that the complexity of computing the partition function for FFLDC is equivalent to the FLDC model, it is also intractable in practice.

4.2.2 Learning FFLDC Models

For FFLDC, the parameter vector $\boldsymbol{\theta}$ is different from the previous models, it is composed by $[\mathbf{a},\mathbf{B},\mathbf{C},\hat{Z}]$ and the gradient of the NCE objective $g(\boldsymbol{\theta})$ is given by:

$$\left(\nabla_{\mathbf{a}} \log \hat{P}_d(S;\hat{Z},\mathbf{a},\mathbf{B},\mathbf{E})\right)_m = \sum_{i\in S} x_{i,m} \tag{4.7}$$

$$\left(\nabla_{\mathbf{B}} \log \hat{P}_d(S;\hat{Z},\mathbf{a},\mathbf{B},\mathbf{E})\right)_{m,d} = x_{i^d,m} - \sum_{i\in S} x_{i,m} \tag{4.8}$$

$$\left(\nabla_{\mathbf{E}} \log \hat{P}_d(S;\hat{Z},\mathbf{a},\mathbf{B},\mathbf{E})\right)_{m,c} = x_{i^c,m} - \sum_{i\in S} x_{i,m} \tag{4.9}$$

Where $i^d = \operatorname{argmax}_{i\in S} \sum_{k=1}^{M} x_{i,k} b_{k,d}$ and $i^c = \operatorname{argmax}_{i\in S} \sum_{k=1}^{M} x_{i,k} e_{k,c}$. $(\nabla_{\mathbf{a}}\cdot)_m$ represents the m-th entry of the sub-gradient with respect to a, $(\nabla_{\mathbf{B}}\cdot)_{m,d}$ represents the (m,d)-th entry of the sub-gradient with respect to B and $(\nabla_{\mathbf{E}}\cdot)_{m,c}$ represents the (m,c)-th entry of the sub-gradient with respect to E.

Remark. For $\mathbf{X} = \mathcal{I}$, the sub-gradients in 4.7-4.9 are equivalent to the FLDC sub-gradients. Because $x_{i,m} = 1$ only when $i = m$.

Regarding running time complexity, the FFLDC model requires more operations for the gradient due to the inclusion of features. Concretely, computing the gradient for a set S takes $\mathcal{O}(|S|M(L+K))$ which makes the running time of a single pass over data and noise samples $\mathcal{O}(|\mathcal{D}\cup\mathcal{N}|M(L+K)\kappa)$. Even though this is larger than the complexity of FLID or FLDC, it is still linear on the data.

4.2.3 Example: Rated Locations

A small town has 6 popular locations, each of them has been rated from 0 to 5, where 0 represents a bad review and 5 an excellent review. Collected data shows that a typical visitor is more likely to visit and take photos at places with high ratings, however this is not the only factor driving their behavior. For example, an average visitor does not visit two places that serve food on the same day instead the data shows that visitors usually go to one place that serves food and then to an outdoor one, or they just visit one that has both characteristics. This data is summarized in Table 4.2, the task is to model this behavior using the FFLDC model.
In this example there is knowledge about the items and what features are relevant for the data. These features are summarized in equation (4.10). The first column corresponds to

Table 4.2: Synthetic data for Example 4.2.3.

Locations visited (S)	$P(S)$
$\{1,3\}$	0.30
$\{3,4\}$	0.25
$\{3,6\}$	0.15
$\{2\}$	0.10
$\{1\}$	0.06
$\{3\},\{4\}$	0.04
$\{5\},\{6\}$	0.03
Otherwise	0.00

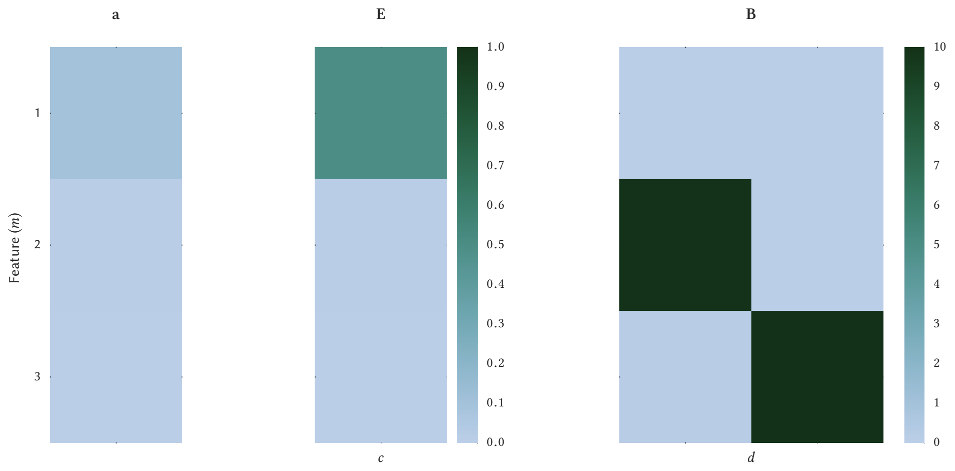

Figure 4.2: FFLDC sample model for Example 4.2.3.

the aforementioned rating, the second and third are binary features indicating whether the location is an outdoor one and whether it serves food, respectively.

$$\mathbf{X} = \begin{pmatrix} 4 & 1 & 0 \\ 4 & 1 & 1 \\ 3 & 0 & 1 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix} \tag{4.10}$$

A possiblity is an FFLDC model that encourages diversity on the second a third feature, i.e. models the idea that visitors do not go to more than one place that serves food or is outdoor, while assigning a positive utility and coherence value to the first feature, i.e. modeling the preference for places with high ratings. One such model is presented in Figure 4.2, this approximates distribution from Table 4.2 and illustrates the type of model that is useful in this example.

If a new location j is considered then it is possible to apply the model and estimate the probability of sets that include it. For example, consider a seventh location with the following features $\mathbf{x}_7 = [5, 1, 0]$, using the same model parameters a new distribution can be computed without changing the model parameters, as it would be the case with FLID or

Table 4.3: Synthetic data for Example 4.2.3 after adding a new item.

Locations visited (S)	$P(S)$
$\{3,7\}$	0.24
$\{1,3\}$	0.21
$\{3,4\}$	0.19
$\{3,6\}$	0.10
$\{7\}$	0.06
$\{1\},\{2\}$	0.04
$\{3\},\{4\}$	0.03
$\{5\},\{6\}$	0.02
Otherwise	0.00

FLDC. The updated distribution is shown in Table 4.3 and it can be considered a sensible distribution given the problem description and the high rating of the new location, hence showing the model's capacity to generalize.

Chapter 5

Synthetic Experiments

This chapter describes experiments done on synthetic data to explore the learning performance of the methods described in the background section.

5.1 Learning Setup

This section describes implementation details of NCE and SGD, these are applicable for both the experiments in this chapter as well as those performed on real data, which will presented in subsequent chapters.

5.1.1 Noise Distribution

As mentioned in section 3.5.2, there are several considerations when choosing the noise distribution. A log-modular distribution is a natural choice for the noise when learning the FLID model (Tschiatschek et al., 2016), and this distribution will also be used for the noise with FLDC and FFLDC models.
The parameters of a log-modular distribution can be estimated efficiently from data through MLE, as given by Equation (3.8). Also the partition function can be computed in closed form, as shown in Equation (3.9). This makes the distribution a good candidate because it can be used efficiently for sampling as well as for computing the normalized probability of sets, i.e. $P_n(S)$ during the learning procedure.

5.1.2 Initialization

The log-modular model also provides a sensible initialization for vector $\mathbf{u}$ in the FLID and FLDC models. For FFLDC, the initialization for $\mathbf{a}$ is done by solving the linear system derived from the factorization in Equation (4.2), this linear system is given by:

$$\mathbf{a} = \mathbf{X}^{-1}\mathbf{u} \tag{5.1}$$

Unfortunately, this system can be under/over-determined in many cases so an approximated solution must be used. The chosen method for finding this solution is Ridge regression which minimizes the squared error $\sum_{i \in S}(u_i - (\mathbf{Xa})_i)^2$ with a regularization term on the l_2 norm of $\mathbf{a}$.
The weight matrices, e.g. $\mathbf{W}^e, \mathbf{B}$, are initialized with random values drawn from a uniform distribution $\mathcal{U} \sim [0, 0.001]$. These is the same initialization setting proposed by Tschiatschek et al. (2016).
In the cases where AdaGrad is used, the accumulated gradient $\mathbf{G}$ is initialized with a constant value for each parameter γ, namely $G^0_\gamma = 0.01$.

5.1.3 Learning rate

When not using AdaGrad, the learning rate used in the experiments follows the power law mentioned in section 3.5.4, i.e.

$$\eta(t) = \frac{\eta_0}{t^p} \tag{5.2}$$

5.1.4 Projection in SGD

The weight matrices, e.g. $\mathbf{W}^e, \mathbf{B}$, are defined only for positive values. Therefore, they must be projected in each gradient step to ensure that the solution stays in the feasible space. This is done following the procedure proposed by Tschiatschek et al. (2016) which produces better results than simply clipping the values to 0. Let θ_h be an element of these matrices, e.g. $\theta_h = w^e_{i,d}$, then the projection is defined as:

$$\theta_h = \begin{cases} \theta'_h & \text{if } \theta'_h \geq 0 \\ \mathcal{U} \sim [0, 0.001] & \text{otherwise} \end{cases} \tag{5.3}$$

Where θ'_h is the unnormalized parameter after the gradient update and $\mathcal{U} \sim [0, 0.001]$ means that the value is drawn from an uniform distribution between $[0, 0.001]$ as in the initialization.

5.1.5 Termination Criteria

The termination criteria for learning was the number of epochs, i.e. full passes over the data and noise samples.

5.2 Example Datasets

This section presents the results obtained after learning the datasets from the examples given for each model. For these experiments, the settings were:

- 1000 samples of data drawn from the corresponding distribution, i.e. $|\mathcal{D}| = 1000$.
- 2000 noise samples, i.e. $\nu = 2$.
- 100 epochs.
- η_0 is 0.005 and $p = 0.1$ for the learning rate, without AdaGrad.

It is worth noting that the presented results for these experiments correspond to the best of multiple runs. It was observed that the learning algorithm does not always arrive to a good solution. This is because $g(\boldsymbol{\theta})$ is not a concave function and some initializations may lead to local maxima.

5.2.1 FLID: Two Landmarks

Recall the following characteristics about the dataset in Example 3.4.2:

- There are 3 elements, h, s, f.
- There are only 2 possible sets, $\{h, s\}$ and $\{h, f\}$. Both equally likely, i.e. $P(S) = 0.5$.
- A model can be constructed with a single diversity dimension, i.e. $L = 1$.

Table 5.1: Learned distribution for Example 4.2.3

S	$P(S)$
$\{h,s\}$	0.48
$\{h,f\}$	0.47
$\{h\}$	0.03
$\{h,s,f\}$	0.02

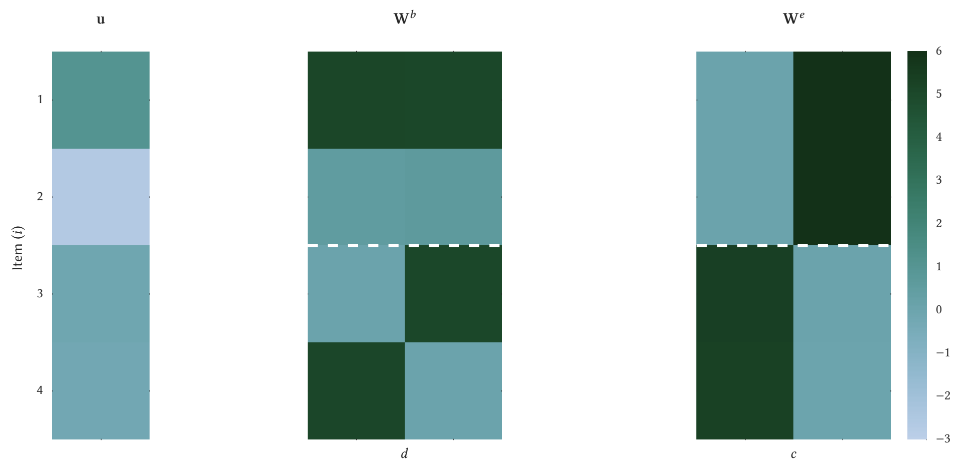

Figure 5.1: Learned model for example 4.1.3. The white line divides the clusters described in the example.

After estimating the parameters from this data, the resulting utility vector and weight matrix are:

$$\mathbf{u} = (5.42, 2.82, 2.79)^{\intercal}$$
$$\mathbf{W}^b = (0.02, 6.10, 6.10)^{\intercal}$$

This is a slightly different model from the one suggested in section 3.4.2, however the normalized probability distribution for this model presented in Table 5.1 shows that it closely approximates the distribution.

5.2.2 FLDC: Two Non-overlapping Clusters

Recall the following characteristics about the dataset in Example 4.1.3:

- There are 4 items, i.e. $V = \{1,2,3,4\}$.
- There are only 2 possible sets, $\{1,2\}$ and $\{3,4\}$. Both equally likely, i.e. $P(S) = 0.5$.
- A model can be constructed with two diversity and two coherence dimensions, i.e. $L = 2, K = 2$.

Figure 5.1 shows the resulting utility vector $\mathbf{u}$ and weight matrices $\mathbf{W}^b, \mathbf{W}^e$ after the learning procedure.
The model shown in the figure displays the desired properties of diversity between the clusters while having coherence between the items in each one. Despite being different from the proposed model in section 4.1.3 it also approximately realizes the desired distribution.

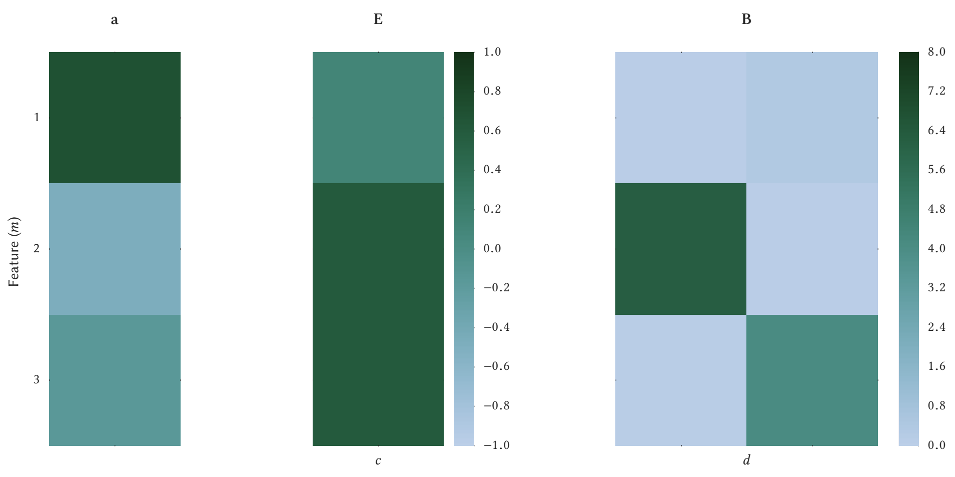

Figure 5.2: Learned model for example 4.2.3

FFLDC: Rated Locations

Recall the following characteristics about the dataset in Example 4.2.3:

- There are 6 items, i.e. $V = \{1, 2, 3, 4, 5, 6\}$.
- The full distribution is related to the features and is given in Table 4.2.
- The proposed model has 2 diversity dimensions and one coherence dimension.

Figure 5.2 shows the resulting model after learning. Unfortunately, this model can be not as easily interpreted as the one proposed in figure 4.2, nevertheless it accurately models the distribution from the example thus showing the learning algorithm's ability to estimate a distribution using features.

5.3 Learning rate sensitivity

This section explores the learning rate's effect on the learning performance for the synthetic dataset from Example 4.2.3.
The first experiment considers the stability of the optimization without AdaGrad as the learning rate increases. The settings are the same as for the previous experiments with the following additions:

- η_0 is increased between 0.0005 and 0.02
- The learning is performed independently 8 times to obtain statistics on the results.

Figure 5.3 shows $g(\boldsymbol{\theta})$ during the optimization, each point corresponds to the mean value calculated after a given number of epochs and the error bars show the corresponding 95% confidence interval for the mean.
The results in figure 5.3 show that for small values of η_0 the learning can be significantly slow and that large values of η_0 may lead to instability, as discussed in section 3.5.4. Concretely, for $\eta_0 \leq 0.01$ the learning is slow but stable, while for $\eta_0 \geq 0.01$ the objective value converges quickly but oscillates significantly. Fortunately, in the particular case of this synthetic dataset there exists a learning rate for which the objective function converges and is stable, i.e. $\eta_0 = 0.005$ however this is not always the case for real data.

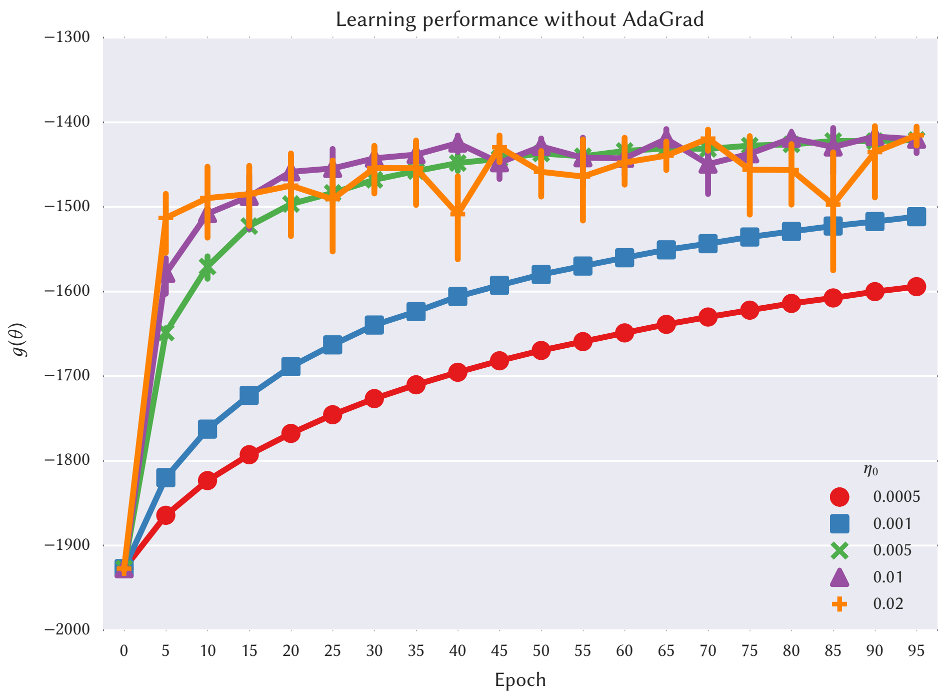

Figure 5.3: Objective function for various values of η_0 without using AdaGrad.

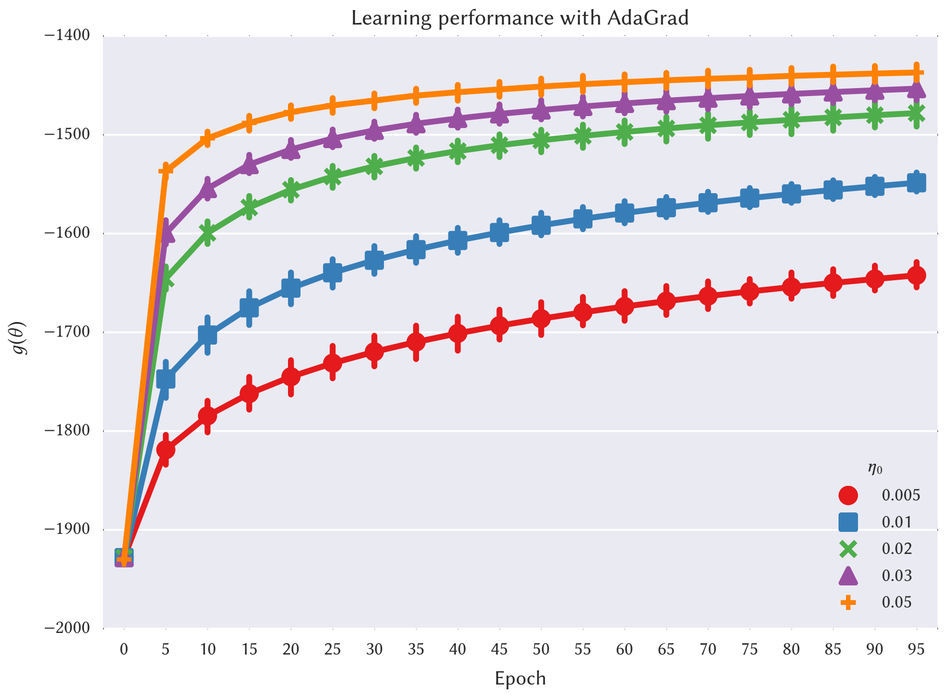

Figure 5.4: Objective function during learning for various values of η_0 with AdaGrad.

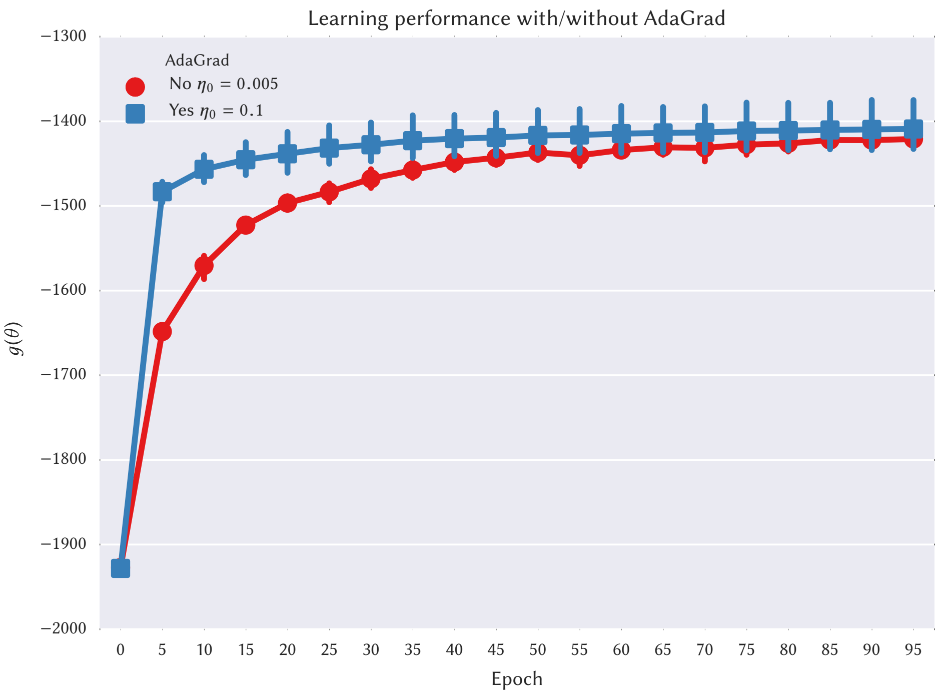

Figure 5.5: Comparison of learning performance with and without AdaGrad.

By contrast, figure 5.4 shows the objective function for η_0 between 0.005 and 0.05 after incorporating AdaGrad to the implementation. These results show that training with AdaGrad is stable for a larger range of eta_0.
Finally, figure 5.5 shows a direct comparison between training with and without AdaGrad using the best values of η_0 for each configuration. In this figure both algorithms converge to an optimal value, however it also shows that training with AdaGrad converges significantly faster while mantaining a stable mean. These combined results indicate that using AdaGrad is the best strategy for learning the FFLDC model.

Chapter 6

Experimental Setup

6.1 Flickr Dataset

Dataset description and how it was collected.

6.2 Path finding

How the actual sets (paths) are built from the data, and the 2 sets of data to be explore 10 items (small) and 100 items (big).

6.3 NCE learning

Details on the implementation of NCE, noise generation.

6.4 Baselines

The baseline models (Markov, distance).

6.5 Evaluation

10 fold evaluation, accuracy and MRR.

Chapter 7

Results

7.1 Small dataset

Results on the small dataset. Difference between FLID, FLDC, FFLDC models. Different feature sets.

7.2 Large dataset

Results on the large dataset. Difference between FLID, FLDC, FFLDC models. Different feature sets.

Chapter 8

Conclusion

Bibliography

Amir, R. (2005). Supermodularity and complementarity in economics: an elementary survey. *Southern Economic Journal*, pages 636–660.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics*, pages 177–186.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer.

Bruno, W. J., Rota, G.-C., and Torney, D. C. (1999). Probability set functions. *Annals of Combinatorics*, 3(1):13–25.

Darken, C. and Moody, J. (1990). Note on learning rate schedules for stochastic optimization. In *Neural Information Processing Systems*, pages 832–838.

Darken, C. and Moody, J. (1992). Towards faster stochastic gradient search. In *Neural Information Processing Systems*, pages 1009–1016.

Djolonga, J. and Krause, A. (2014). From MAP to marginals: Variational inference in bayesian submodular models. In *Neural Information Processing Systems (NIPS)*.

Djolonga, J. and Krause, A. (2015). Scalable variational inference in log-supermodular models. In *International Conference on Machine Learning (ICML)*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Jerrum, M. and Sinclair, A. (1990). Polynomial-time approximation algorithms for the ising model. In *Automata, Languages and Programming*, pages 462–475. Springer Berlin Heidelberg.

Krause, A. and Golovin, D. (2014). Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, chapter 3. Cambridge University Press.

Shahaf, D., Guestrin, C., and Horvitz, E. (2012). Trains of thought: Generating information maps. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 899–908, New York, NY, USA. ACM.

Tschiatschek, S., Djolonga, J., and Krause, A. (2016). Learning probabilistic submodular diversity models via noise contrastive estimation. In *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Yan, R., Wan, X., Otterbacher, J., Kong, L., Li, X., and Zhang, Y. (2011). Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 745–754, New York, NY, USA. ACM.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 116–123, New York, NY, USA. ACM.