

## Master Thesis

# Probabilistic Modeling of City-scale 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



# Contents

<b>Acknowledgments</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>Symbols</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>3</b>
<b>3 Background</b>	<b>5</b>
3.1 Submodular Set Functions . . . . .	5
3.2 Probabilistic Log-sub/supermodular Models . . . . .	5
3.3 Facility Location Diversity Model (FLID) . . . . .	6
3.3.1 Example: Two Landmarks . . . . .	7
3.4 Learning from Data . . . . .	7
3.4.1 Learning Log-modular Models . . . . .	7
3.4.2 NCE Learning . . . . .	8
3.4.3 Learning FLID via NCE and SGD . . . . .	9
3.4.4 AdaGrad . . . . .	10
<b>4 Extending FLID</b>	<b>11</b>
4.1 Beyond Diversity: FLDC . . . . .	11
4.1.1 Learning FLDC Models . . . . .	12
4.1.2 Example: Two Non-overlapping Clusters . . . . .	12
4.2 Generalizing through Features: FFLDC . . . . .	12
4.2.1 Learning FFLDC Models . . . . .	14
4.2.2 Example: Rated Locations . . . . .	14
<b>5 Synthetic Experiments</b>	<b>17</b>
5.1 Learning Setup . . . . .	17
5.1.1 Noise Distribution . . . . .	17
5.1.2 Initialization . . . . .	17
5.1.3 Learning Rate . . . . .	18
5.1.4 Projection in SGD . . . . .	18
5.1.5 Column Restarts . . . . .	18
5.1.6 Termination Criteria . . . . .	18
5.2 Example Datasets . . . . .	18
5.2.1 FLID: Two Landmarks . . . . .	19
5.2.2 FLDC: Two Non-overlapping Clusters . . . . .	19
5.2.3 FFLDC: Rated Locations . . . . .	20
5.3 Learning Rate Sensitivity . . . . .	21
5.4 Noise-to-data Ratio . . . . .	22

<b>6 Modeling an Image Collection - Setup</b>	<b>25</b>
6.1 Flickr Dataset . . . . .	25
6.2 Path Finding . . . . .	28
6.2.1 Mean-shift Clustering . . . . .	28
6.3 Models . . . . .	33
6.4 Baselines . . . . .	33
6.4.1 Markov Model . . . . .	33
6.4.2 Heuristic Markov Model . . . . .	33
6.4.3 Proximity . . . . .	34
6.4.4 Log-modular . . . . .	34
6.5 Evaluation . . . . .	34
<b>7 Modeling an Image Collection - Results</b>	<b>37</b>
7.1 The Problem with Singletons . . . . .	37
7.2 Small Dataset . . . . .	37
7.2.1 Diversity & Coherence . . . . .	38
7.3 Large Dataset . . . . .	40
<b>8 Conclusion</b>	<b>45</b>
<b>Bibliography</b>	<b>49</b>

# Acknowledgments

Firstly, I would like to sincerely express my gratitude to Prof. Andreas Krause who has been an outstanding advisor not only during the course of this thesis but all of my master studies. His insightful comments and suggestions were always opportune and of great help to move forward.

To my supervisors, Dr. Sebastian Tschiatschek and Alkis Gotovos, I am immensely indebted for their dedication and constant support throughout the last six months. I am thankful for the valuable discussions that we had.

I would also like to thank my colleagues at the ETH for their encouragement and motivation, it has been a great opportunity to share this experience with them.

Lastly but not least, to my loved ones, whether they are close or far away, I am deeply thankful for their unwavering support.



# Abstract

The increasing popularity of photo sharing sites such as Flickr and Instagram has created a unique opportunity for researchers to harness large collections of images and extract useful knowledge from them. One interesting application is the use of geotagged photos to analyze the behavior of tourists in a city, e.g. to identify commonly visited locations. In this work, we use probabilistic submodular models (PSMs), a class of distributions over sets, to model sets of visited locations. In particular, we start with the facility location diversity model, a PSM proposed by Tschiatschek et al. (2016) for modeling diversity, and extend it to also model attractive behavior. Furthermore, we propose the integration of features to enable the model to generalize to previously unseen locations. In experiments, we demonstrate that these models can represent complex distributions over sets of locations. Moreover, using geotagged photos taken between 2005 and 2015 in Zurich, we show how these models can be applied to the task of recommending touristic attractions to visit.



# Symbols

## Symbols

$\eta$	Learning rate
$\gamma$	Model parameter
$\kappa$	Largest subset size
$\nu$	Noise-to-data ratio
$\sigma$	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
$S, T$	Set
$S_t$	Sequence
$V$	Ground set
$Z$	Partition function

## 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

$\theta$	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
TVD	Total Variation Distance

# Chapter 1

## Introduction

The amount of information available on the internet is staggering, even conducting a simple search like *St Peter's Basilica* can turn up millions of results. Fortunately, search engines ensure the first results are the most relevant and often times all we need to look at. However, this may not be true for all types of searches.

Imagine a person planning a trip to Rome for two days. Where to start? Suppose they start searching for attractions to visit or things to do. A popular site for this kind of search is *tripadvisor* which shows over 2000 attractions and activities along with user reviews and recommendations, clearly it is impossible for someone to go through all these results so picking the top results and filtering for some type of activities is a good strategy. Afterwards, they must find the location of each place and organize them in a schedule and map considering travel times, visit times and locations. This certainly sounds like a complicated endeavor.

An alternative would be to find a curated itinerary that contains all the places to visit and in which order. There are many for Rome since it is such a touristic city, but what about smaller cities? Could it be possible to produce such itineraries in an automatic fashion, such that it can be scaled to multiple cities?

This is a complex problem with various questions to answer. How to obtain data about a city that is available regardless of size or popularity? How to extract meaningful information about locations and activities from such data? How to select a subset of those locations and activities that are of interest for a particular user and present it in an organized manner?

We focus on a simplified instance of this problem. Consider a user that is planning a day trip and knows some attractions to visit, however they would like to see more than that. Therefore, a method is needed that can expand the existing set of attractions to be visited in a city. This should be generated automatically from data so that it can be applied to different cities. This addresses all the questions presented before, although it excludes the problems of scheduling based on travel and visit times, which could be done as a post-processing step given a set of locations to visit.

Photos of a place are usually an important factor when selecting destinations, and it can influence the perception that a visitor has of a place. When a tourist takes a photo of some location and uploads it to the internet, they are giving it an implicit value (Donaire et al., 2014). Several authors have used internet photo collections for the task of tourist route planning and recommendation. This is why we chose photos as the starting point for identifying popular attractions, in particular geotagged photos are of interest because they can be directly associated with a place.

One source for these photos is Flickr, a popular site where users can upload and manage their photos for free. Having over 92 million users among 63 countries makes Flickr an excellent source of photos for many locations all over the world, moreover around 1 mil-

lion photos are shared every day<sup>1</sup>. Many of the photos in Flickr are public and available through an API which allows researchers to collect and analyze this data.

In order to model this data, we focus on probabilistic submodular models which have been used for summarization tasks and exhibit a natural capacity of modeling the concept of diversity. This work studies a submodular model known as FLID proposed by Tschiatschek et al. (2016) which was used for summarization of a large image collection and product recommendation from Amazon data, we extend it and show how this can be also applied for the problem of recommending routes for tourists. One of the extensions introduces the concept of attractiveness to the model, which aims to capture situations when it is desirable to visit multiple similar locations, for example visiting two churches on the same day. This attractiveness is achieved by incorporating supermodularity to the model. Additionally, features were introduced to show how the model can generalize to unseen elements.

The contributions of this work are:

- A supermodular extension to the FLID model to integrate the concept of attractiveness.
- An extension to the FLID model able to generalize to unseen elements using feature representations.
- Exploring the behavior of noise contrastive estimation for the FLID model and its extensions, as well as the effect of different parameters in the learning results.
- Presenting an application of the proposed models to solve a recommendation task for touristic locations in a city.

The remainder of this document is organized as follows, Chapter 2 summarizes the existing literature and methods for the problems of summarization of image collections and tourist route recommendation based on geotagged photos, as well as literature on submodular models. Chapter 3 explains in detail the existing models and techniques used throughout this work. Chapter 4 introduces the proposed models and their characteristics, while chapter 5 shows how they can be learned from data and the performance of the learning method. Chapter 6 describes the methods used for the task of recommending tourist locations and how the proposed models were applied to real data, the results of the experiments are then presented and discussed in Chapter 7. Finally, Chapter 8 summarizes this work and presents some conclusions and future research directions.

---

<sup>1</sup>As reported in 2014 <http://techcrunch.com/2014/02/10/flickr-at-10-1m-photos-shared-per-day-170-increase-since-making-1tb-free/>

# Chapter 2

## Related Work

Using online image collections to describe and present visual summaries of places or cities has become recently a topic of interest. Kennedy et al. (2007) and Kennedy and Naaman (2008) use photos from Flickr to produce visual summaries of locations and events, their analysis uses geographical labels, visual features and user-provided textual tags to identify representative photos of a place. Another approach was presented by Jaffe et al. (2006), they apply clustering to the spatial features of photos and then evaluate these clusters based on number of photographers and textual tags in order to identify the most representative ones.

However, the scale of these works was limited to only single cities. Kleinberg et al. (2009) present a clustering approach for identifying landmarks and points of interest based on the number of geotagged photographs and photographers, they show that it can be applied to specific cities or whole countries. Their work also shows how visual features and textual tags can be used to infer the location of a photo when geographic labels are not available, this is done to solve the problem that most of the photos uploaded to Flickr are not geotagged. Alternatively Serdyukov et al. (2009) present a language-based model for predicting the most probable location in which a photo was taken based on the textual annotations provided by users. A different large-scale approach to generating visual summaries of a place is presented by Li et al. (2009) where supervised learning, namely SVMs, is used instead of clustering, they also incorporate temporal information in addition to the visual and textual features and show how it can improve the summaries. Temporal information is an important feature when dealing with the problem of itineraries or touristic routes. Popescu et al. (2009) show how information such as travel and visit times between locations can be extracted from Flickr photos which then could be used to propose personalized trips with detailed schedules.

The methods presented by Kurashima et al. (2010) are more closely related to our current approach, they propose the use of clustering techniques for landmark detection and a Markov chain for modeling photographers' behavior which can then be used to generate travel routes based on previous history of visits. Our contribution uses the same clustering techniques but introduces a novel and richer model for the photographers' behavior.

This novel model is based on current research on submodular probabilistic models and summarization. Submodularity is a property of set functions that has been widely studied, see Krause and Golovin (2014) for a survey. Much of the research is focused on inference (Djolonga and Krause, 2014, 2015) and sampling (Gotovos et al., 2015). Additionally, applications in document summarization have used submodularity to model aspects such as diversity and coverage. Shahaf et al. (2012, 2013) present a method for generating timelines representing structured summaries from large collections of documents, it shows how submodularity can be used to encourage diversity between documents. More specific to the task of visual summarization is the work presented by Tschiatschek et al. (2014). They propose mixtures of submodular functions to generate summaries of large image

collections and compare the performance of these automatically generated summaries to those created by humans. A novel probabilistic submodular model was proposed by Tschiatschek et al. (2016) for the task of summarization of image collections, the authors show how this class of distributions can be used to model diversity in sets for a wide range of applications. Moreover, they prove that FLID can be learned efficiently even with thousand of items while achieving state of the art performance, improving on the scalability of Determinantal Point Processes (DPPs), a prominent submodular probabilistic model.

# Chapter 3

## Background

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

### 3.1 Submodular Set Functions

Functions over sets are of great interest and can be applied to diverse settings. In particular, submodular set functions have been utilized in multiple domains, such as document summarization and information gathering (Krause and Golovin, 2014).

A set function is defined on the powerset  $2^V$  of a ground set  $V$ , i.e.  $F : 2^V \rightarrow \mathbb{R}$  and it is submodular if it exhibits a "diminishing returns" property , namely if it satisfies:

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

Intuitively this indicates that adding an element to a smaller set yields a larger gain, 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 is less effective than adding it to a smaller one. This property also makes them good candidates for modeling the concept of diversity (Tschitschek et al., 2016).

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.

### 3.2 Probabilistic Log-sub/supermodular Models

Set functions may also express probability distributions, in which case they are known as Probability Set Functions. Formally, a set function  $F$  is a probability set function if it satisfies that  $\forall S \subseteq V, P : 2^V \rightarrow \mathbb{R} \mid 0 \leq P(S) \leq 1$  and  $\sum_{S \subseteq V} P(S) = 1$  (Bruno et al., 1999). In this work we are 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):

$$P(S) = \frac{1}{Z} \exp(F(S)), \quad (3.2)$$

where  $F(S)$  is a submodular or supermodular function, respectively. Djolonga and Krause (2014, 2015) indicate that these models encompass many practical probabilistic models such as repulsive Ising models, Determinantal Point Processes (DPPs) and binary pairwise Markov Random Fields (MRFs).

The normalization constant  $Z$  is also known as the *partition function* and it 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 in general (Jerrum and Sinclair, 1990). This makes it impossible in practice to perform such computations except for special cases of the model.

### 3.3 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  $F(\emptyset) = 0$ , i.e.

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

where  $u(i)$  is some function  $u : V \rightarrow \mathbb{R}$ . 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:

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

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) proposed an additional term that models set diversity, its objective is to identify redundant elements and assign lower probabilities to sets that contain them.

This diversity promoting term is based on representing each element  $i$  with an  $L$ -dimensional vector  $\mathbf{w}^b \in \mathbb{R}_{\geq 0}^L$ , where each dimension  $d$ ,  $1 \leq d \leq L$  captures a concept related to set diversity and  $w_{i,d}^b$  quantifies the relevance of each element  $i$  for each concept  $d$ . Hereby, define  $\mathbf{W}^b \in \mathbb{R}_{\geq 0}^{|V| \times L}$  as the matrix where the  $i$ -th row is the representation  $\mathbf{w}_i^b$  of element  $i$ .

For each dimension  $d$ , the diversity of a set  $S$  is quantified by  $\max_{i \in S} w_{i,d}^b - \sum_{i \in S} w_{i,d}^b$ . This assigns a negative value for sets that contain more than one element with nonzero weight  $w_{i,d}^b$  in that dimension (Tschiatschek et al., 2016). Equation (3.5) shows the complete diversity term, this sums over all  $L$  dimensions to account the diversity in each concept.

$$\text{div}(S) = \sum_{d=1}^L \left( \max_{i \in S} w_{i,d}^b - \sum_{i \in S} w_{i,d}^b \right) \quad (3.5)$$

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

$$P(S) = \frac{1}{Z} \exp\left(\sum_{i \in S} u_i + \sum_{d=1}^L \left( \max_{i \in S} w_{i,d}^b - \sum_{i \in S} w_{i,d}^b \right)\right) \quad (\text{FLID})$$

Table 3.1: FLID probability distribution for the scenario in Example 3.3.1.

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

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 in the size of the ground set and significantly better than the cost of enumerating the powerset of  $V$ , i.e.  $O(2^{|V|})$  (Tschitschek et al., 2016). Nevertheless, it is worth noting that this complexity is exponential in  $L$  which means that the partition function computation is only practical for a limited range of FLID models, namely those with  $L \ll |V|$ .

### 3.3.1 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  $\mathbf{u} = (2, 2, 2)^\top$ , indicating that all locations are equally popular. A suitable diversity weight matrix would then be  $\mathbf{W}^b = (0, 20, 20)^\top$ . Table 3.1 shows the resulting probabilities of the subsets, accurately representing the example scenario.

## 3.4 Learning from Data

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

### 3.4.1 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:

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

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

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

where  $N(i \in S)$  is the number of sets containing element  $i$ . Therefore, the utilities can be estimated as:

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

This model also allows computing the partition function in closed form, i.e.:

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

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. 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.

### 3.4.2 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  $v$  be the noise-to-data ratio, i.e.  $v = |\mathcal{N}|/|\mathcal{D}|$ .

The goal is to estimate the posterior probabilities  $P(Y_s = 1 | S; \theta)$  and  $P(Y_s = 0 | S; \theta)$ , in order to discriminate noise from data samples. These probabilities are given by equations (3.10) and (3.11).

$$P(Y_s = 1 | S; \theta) = \frac{\hat{P}_d(S; \theta)}{\hat{P}_d(S; \theta) + vP_n(S)} \quad (3.10)$$

$$P(Y_s = 0 | S; \theta) = \frac{vP_n(S)}{\hat{P}_d(S; \theta) + vP_n(S)} \quad (3.11)$$

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  $\theta$  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):

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

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

### Theoretical Considerations

A couple of theoretical conditions are necessary for obtaining asymptotic convergence 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.  $\exists \theta^* | \hat{P}_d(S; \theta^*) = P_d$ .

2. The noise distribution  $P_n$  is nonzero whenever  $P_d$  is nonzero.

### Practical Considerations

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.

#### 3.4.3 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 been 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  $\nabla \log P(Y_s = y | S; \theta)$  must be computed. For FLID, the parameter vector  $\theta$  is composed by  $\mathbf{u}, \mathbf{W}^b$  and  $\hat{Z}$ , and the corresponding gradients are given by,

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

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

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

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

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

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 gradient with respect to  $\mathbf{W}^b$  (Tschiatschek et al., 2016).

After the gradient is computed, the parameters are updated according to the following equation,

$$\theta^{t+1} = \theta^t + \eta(t) \nabla g(\theta), \quad (3.18)$$

where  $\eta(t)$  is the learning rate.

### Computational Complexity

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

### 3.4.4 AdaGrad

A commonly used learning rate function for SGD is  $\eta(t) = \eta_0 t^{-p}$ , which offers the best theoretical convergence speed for convex functions under certain conditions but often also 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  $\eta_0$  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 presented in later chapters, 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  $\theta$  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  $\gamma$  in  $\theta$  is given by,

$$\theta_\gamma^{\tau+1} \leftarrow \theta_\gamma^\tau - \frac{\eta}{\sqrt{G_\gamma}} \nabla g(\theta)_\gamma^\tau, \quad (3.19)$$

where  $\mathbf{G}^\tau$  is the vector of accumulated gradients at time  $\tau$  and each component  $G_\gamma^\tau$  is given by,

$$G_\gamma^\tau = \sum_{t=1}^{\tau} (\nabla g(\theta)_\gamma^t)^2, \quad (3.20)$$

where  $\nabla g(\theta)_\gamma^t$  denotes the gradient of the  $\gamma$  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 that should be considered. Coherence is another desired property of summaries, for example in structured summaries (Yan et al., 2011). Balancing coherence and diversity is considered a challenge and maximizing 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. We 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  $\mathbf{w}^K \in \mathbb{R}_{\geq 0}^K$ , 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  $\sum_{i \in S} w_{i,c}^e - \max_{i \in S} w_{i,c}^e$  which is a supermodular term mirroring the submodular term in FLID. Summing over all the concepts results in the proposed coherence term. Hereby, define  $\mathbf{W}^e \in \mathbb{R}_{\geq 0}^{|V| \times K}$  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 we refer to as the Facility Location Diversity and Coherence (FLDC) model. Its probability distribution is defined as,

$$P(S) = \frac{1}{Z} \exp \left( \sum_{i \in S} u_i + \text{div}(S) + \sum_{c=1}^K \left( \sum_{i \in S} w_{i,c}^e - \max_{i \in S} w_{i,c}^e \right) \right), \quad (\text{FLDC})$$

where  $\text{div}(S)$  is the diversity term in Equation (3.5).

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 eligible elements for a set given the maximum weights for each dimension. This can be easily extended to include the ordering for the coherence weights as well. This requires evaluating  $K$  more dimensions which

Table 4.1: Probability distribution for Example 4.1.2.

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

results in the added term to the exponent of the complexity expression, which makes the computation of the partition function more expensive than in the FLID case.

### 4.1.1 Learning FLDC Models

As with FLID, the parameters for an FLDC model can be estimated using NCE. For FLDC the vector  $\theta$  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,

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

where  $(\nabla_{\mathbf{W}^e})_{i,c}$  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 because the complexity is still linear in the data and noise.

### 4.1.2 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  $\mathbf{u} = \vec{0}$ , 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 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.

Considering these issues, we propose a further extension to the model. First, 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 feature indicates whether it is indoors or outdoors. The representations of all elements are represented in matrix  $\mathbf{X} \in \mathbb{R}^{|V| \times M}$  where each row  $i$  corresponds to the representation  $\mathbf{x}_i$  of element  $i$ .

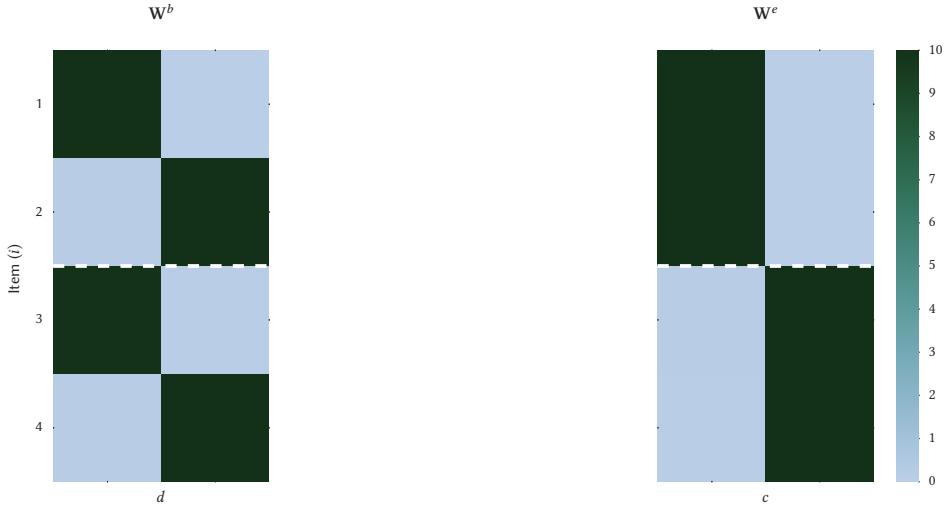


Figure 4.1: Diversity and coherence weights for FLDC model in Example 4.1.2. The dotted line divides the clusters.

Then, the utility vector  $\mathbf{u}$  and the weight matrices  $\mathbf{W}^b, \mathbf{W}^e$  are factorized,

$$\mathbf{u} = \mathbf{X}\mathbf{a}, \quad (4.2)$$

$$\mathbf{W}^b = \mathbf{X}\mathbf{B}, \text{ and} \quad (4.3)$$

$$\mathbf{W}^e = \mathbf{X}\mathbf{E}, \quad (4.4)$$

where  $\mathbf{a} \in \mathbb{R}^M$  represents the contribution of each feature to the total utility of an item, and  $\mathbf{B} \in \mathbb{R}^{M \times L}$  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. For example if  $M < |V|$  and the features are sufficient to represent the items, i.e. each item can be uniquely identified from its features, it is expected that learning requires less computation because of the reduced number of parameters in contrast to learning an equivalent FLDC model.

We refer to the extended model as the Featurized Facility Location Diversity and Coherence (FFLDC) model and its probability distribution is defined as:

$$P(S) = \frac{1}{Z} \exp \left( \sum_{i \in S} \sum_{k=1}^M x_{i,k} a_k + fdiv(S) + fcoh(S) \right), \quad (\text{FFLDC})$$

where (4.5)

$$fdiv(S) = \sum_{d=1}^L \left( \max_{i \in S} \sum_{k=1}^M x_{i,k} b_{k,d} - \sum_{i \in S} \sum_{k=1}^M x_{i,k} b_{k,d} \right) \text{ and} \quad (4.6)$$

$$fcoh(S) = \sum_{c=1}^K \left( \sum_{i \in S} \sum_{k=1}^M x_{i,k} e_{k,c} - \max_{i \in S} \sum_{k=1}^M x_{i,k} e_{k,c} \right) \quad (4.7)$$

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

*Remark.* If  $\mathbf{X} = \mathcal{I}$ , then FFLDC is equivalent to FLDC, with  $\mathbf{u} = \mathbf{a}$ ,  $\mathbf{W}^b = \mathbf{B}$  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 directly on items, to evaluate the probability of a set containing a previously unseen element  $j$  only its representation  $\mathbf{x}_j$  is required.

The computation of the partition function for this model can be performed 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  $O(|V|M(L + K))$  time for the matrix multiplications, therefore the overall complexity of calculating the partition function for an FFLDC model is  $O(|V|M(L + K) + |V|^{L+K+1})$ . This expression can be simplified to  $O(|V|^{L+K+1})$  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.

#### 4.2.1 Learning FFLDC Models

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

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

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

$$\left( \nabla_{\mathbf{E}} \log \hat{P}_d(S; \hat{\mathbf{Z}}, \mathbf{a}, \mathbf{B}, \mathbf{E}) \right)_{m,c} = x_{i^c, m} - \sum_{i \in S} x_{i,m}, \quad (4.10)$$

where  $i^d = \text{argmax}_{i \in S} \sum_{k=1}^M x_{i,k} b_{k,d}$  and  $i^c = \text{argmax}_{i \in S} \sum_{k=1}^M x_{i,k} e_{k,c}$ .  $(\nabla_{\mathbf{a}} \cdot)_m$  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 gradients in (4.8)-(4.10) are equivalent to the FLDC sub-gradients. Because  $x_{i,m} = 1$  only when  $i = m$ .

Regarding 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  $O(|S|M(L+K))$  which makes the running time of a single pass over data and noise samples  $O(|\mathcal{D} \cup \mathcal{N}|M(L+K)\kappa)$ . Even though this is larger than the complexity of FLID or FLDC, it is still linear in the size of the data and noise samples.

#### 4.2.2 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 (4.11). The first column corresponds to the aforementioned rating, the second and third are binary features indicating whether the location is an outdoor one and whether it serves food, respectively.

Table 4.2: Synthetic data for Example 4.2.2.

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

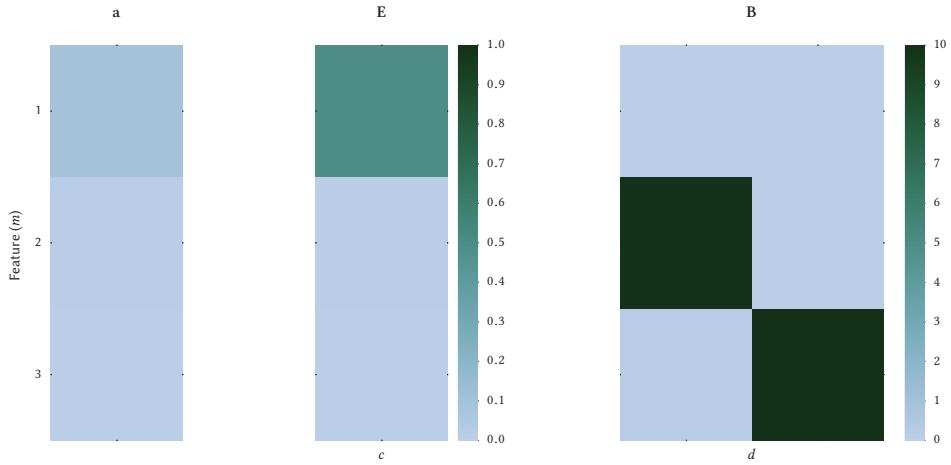


Figure 4.2: FFLDC sample model for Example 4.2.2.

$$\mathbf{X} = \begin{pmatrix} 4 & 1 & 0 \\ 4 & 1 & 1 \\ 3 & 0 & 1 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix} \quad (4.11)$$

A possibility is an FFLDC model that encourages diversity on the second and 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 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

Table 4.3: Synthetic data for Example 4.2.2 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

showing the model's capacity to generalize.

# Chapter 5

## Synthetic Experiments

This chapter describes experiments performed on synthetic data to explore the learning performance of the methods described in Chapter 3.

### 5.1 Learning Setup

This section describes implementation details of NCE and SGD. These are valid for both the synthetic experiments in this chapter as well as those performed on real data, cf. Chapter 7.

#### 5.1.1 Noise Distribution

As mentioned in Section 3.4.2, there are several practical considerations when choosing the noise distribution. 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. Hence, why the log-modular distribution is a natural choice for the noise distribution for training the FLID, FLDC and FFLDC models via NCE.

#### 5.1.2 Initialization

The log-modular model also provides a sensible initialization for the parameter 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),

$$\mathbf{a} = \mathbf{X}^{-1}\mathbf{u} \quad (5.1)$$

Unfortunately, this system is in general under/over-determined, so an approximate solution must be used. The chosen method for finding this solution is ridge regression which minimizes the squared error  $\sum_{i \in S} (u_i - (\mathbf{X}\mathbf{a})_i)^2$  with a regularization term on the  $l_2$  norm of  $\mathbf{a}$ .

The weight matrices,  $\mathbf{W}^e, \mathbf{B}, \dots$ , are initialized with random values drawn from  $\mathcal{U} \sim [0, 0.001]$ , which represents a uniform distribution between  $[0, 0.001]$ . This 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  $\gamma$ , namely  $G_\gamma^0 = 0.01$ .

### 5.1.3 Learning Rate

When not using AdaGrad, the learning rate used in the experiments follows the power law mentioned in Section 3.4.4, i.e.

$$\eta(t) = \frac{\eta_0}{(t+1)^p}, \quad (5.2)$$

where  $\eta_0$  and  $p$  are hyperparameters and  $t$  is the iteration number, with  $t \in \{0, \dots\}$ .  $\eta_0$  is also referred to as the initial learning rate.

### 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  $\theta_h$  be an element of these matrices, e.g.  $\theta_h = w_{i,d}^e$ , then the projection is defined as,

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

where  $\theta'_h$  is the unprojected parameter after the gradient update.

### 5.1.5 Column Restarts

One common problem observed during the experiments is that for the trained models two or more columns in the weight matrices are very similar and therefore, redundant. This can happen for some initialization settings, e.g. when  $i^d$  and  $i^c$  in Equations (4.9) and (4.10) are the same for different values of  $d$  and  $c$  and increase at the same rate in each iteration, hence leading to a local maximum.

In order to address this problem, a heuristic procedure was implemented that resets one of the columns when it is *too* similar to another one. This similarity is measured as:

$$\text{sim}_{\mathbf{W}}(a, b) = \hat{\mathbf{W}}_{*,a}^\top \cdot \hat{\mathbf{W}}_{*,b} \quad (5.4)$$

Where  $a$  and  $b$  are column indexes for a weight matrix  $\mathbf{W}$ , e.g.  $\mathbf{W}^b$ , and  $\hat{\mathbf{W}}_{*,a}$  and  $\hat{\mathbf{W}}_{*,b}$  are the normalized  $a$ -th and  $b$ -th columns of  $\mathbf{W}$ , respectively.

The column pair with maximum similarity is identified every  $N_r$  iterations and if the similarity is larger than a threshold  $\delta$  then the first column is added to the second one, and the first one is restarted to random values as done in the initialization step. The expectation is that the reset column will learn different information with the new initialization values. Usually  $N_r = 500$  and  $\delta = 0.9$  was used.

### 5.1.6 Termination Criteria

The termination criteria for learning was the number of epochs, i.e. the number of 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.  $\nu = 2$ .

Table 5.1: Learned distribution for Example 4.2.2

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

- 100 epochs.
- $\eta_0$  is 0.005 and  $p = 0.1$  for the learning rate, without AdaGrad.

In order to compare the models, the Total Variation Distance (TVD) between the target distribution  $P_d$  and the learned distribution  $\hat{P}_d$  was used, intuitively this measures the quality of the approximation of the learned model to the real distribution. The TVD is defined as (Levin et al., 2008, chap. 4):

$$\|P_d - \hat{P}_d\|_{TV} = \frac{1}{2} \sum_{S \subseteq V} |P_d(S) - \hat{P}_d(S)| \quad (5.5)$$

### 5.2.1 FLID: Two Landmarks

Recall the following characteristics about the dataset in Example 3.3.1:

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

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

$$\begin{aligned} \mathbf{u} &= (5.42, 2.82, 2.79)^\top \\ \mathbf{W}^b &= (0.02, 6.10, 6.10)^\top \end{aligned}$$

This is a slightly different model from the one suggested in Section 3.3.1, however the normalized probability distribution for this model presented in Table 5.1 shows that it closely approximates the distribution from Table 3.1.

### 5.2.2 FLDC: Two Non-overlapping Clusters

Recall the following characteristics about the dataset in Example 4.1.2:

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

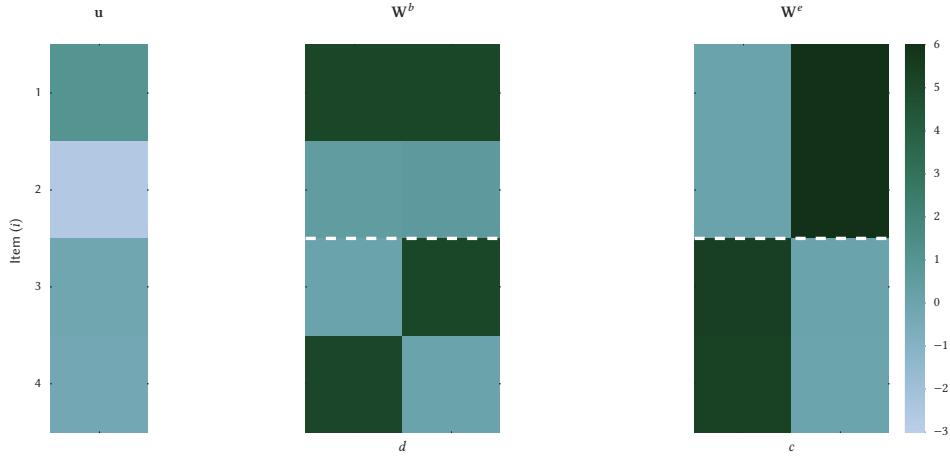


Figure 5.1: Learned model for Example 4.1.2. The white line divides the clusters described in the example.

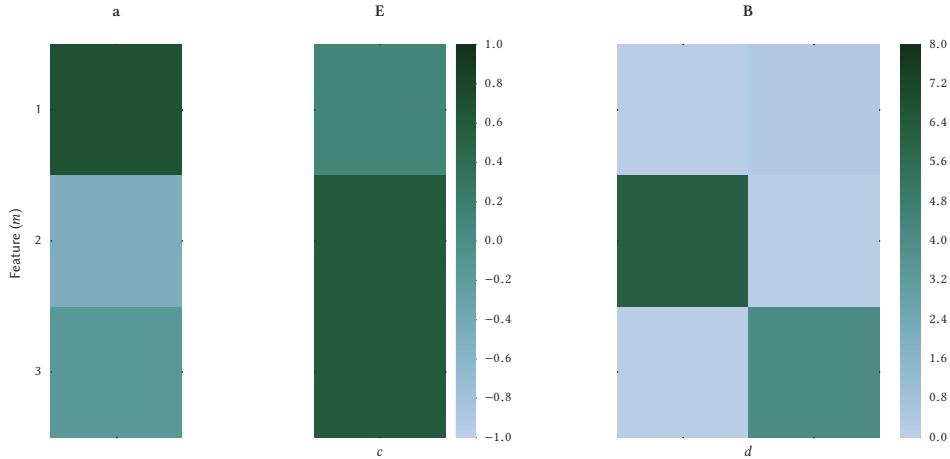


Figure 5.2: Learned model for Example 4.2.2

Figure 5.1 shows the resulting utility vector  $\mathbf{u}$  and weight matrices  $\mathbf{W}^b, \mathbf{W}^e$  after learning the FLDC model. 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 4.1.2 it also approximately realizes the desired distribution with a total variation distance equal to 0.04 which represents a very small error in approximating the distribution from Table 4.1.

### 5.2.3 FFLDC: Rated Locations

Recall the following characteristics about the dataset in Example 4.2.2:

- 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. The total variation distance for the model in Figure 5.2 from the distribution in Table 4.2 is 0.08, indicating a very small error

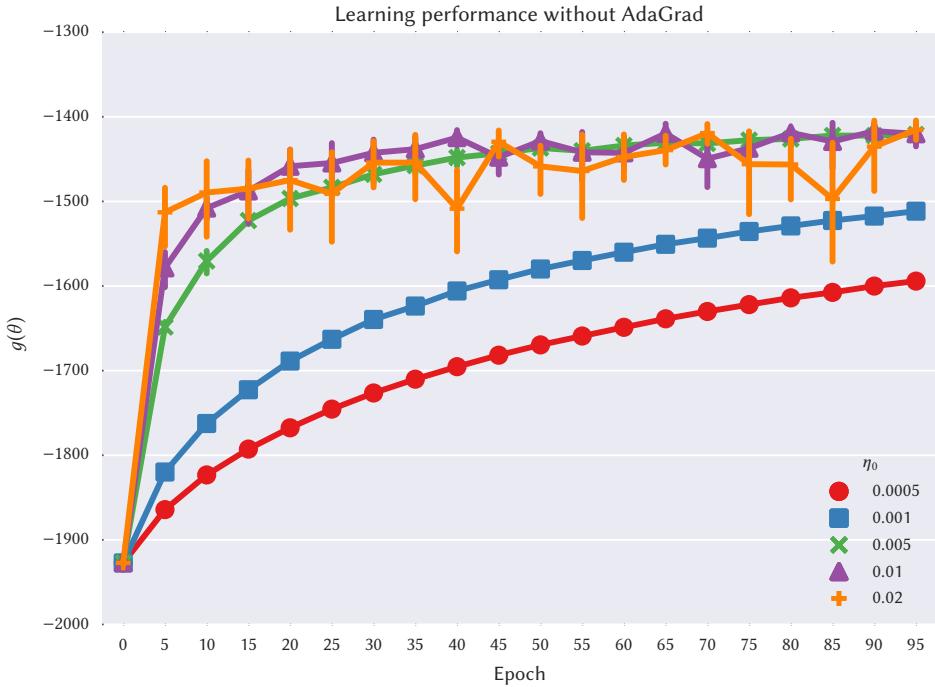


Figure 5.3: Objective function for various values of  $\eta_0$  without using AdaGrad.

in the approximation. 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 effect of the learning rate on the learning performance for the synthetic dataset from Example 4.2.2.

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:

- $\eta_0$  is increased between 0.0005 and 0.02
- The learning was performed 8 times with different random initialization values to obtain statistics on the results.
- For these experiments the reset heuristic for columns was disabled to prevent sudden changes in the objective function due to the resets, hence requiring executing some of the experiments multiple times to avoid local maxima. The local maxima was easily identified by examining the final  $g(\theta)$  values which would be significantly lower for the models that had repeated columns.

Figure 5.3 shows  $g(\theta)$  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  $\eta_0$  the learning can be significantly slow and that large values of  $\eta_0$  may lead to oscillations, as discussed in section 3.4.4.

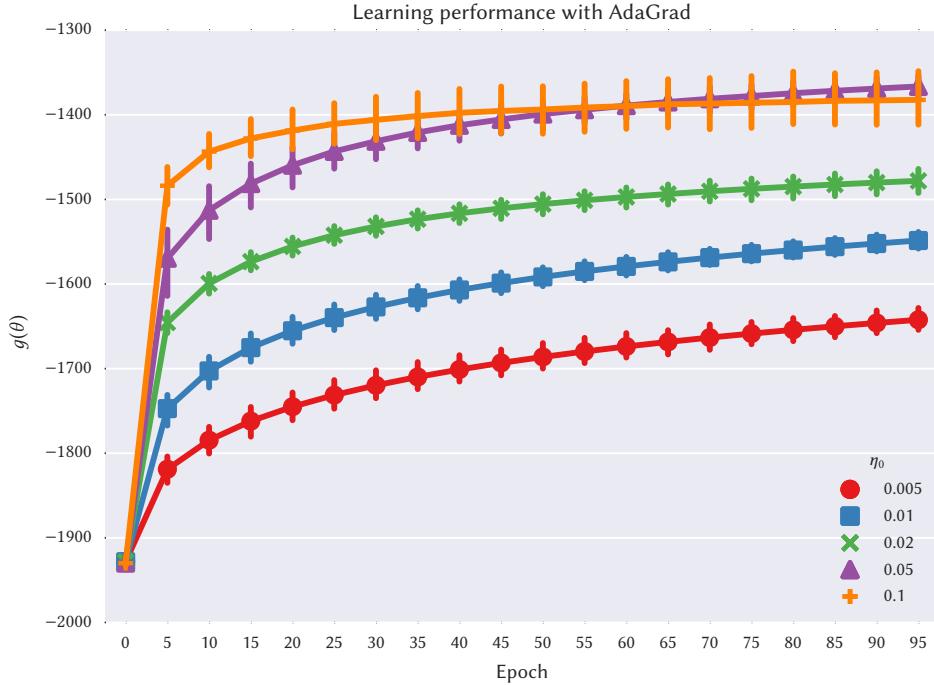


Figure 5.4: Objective function during learning for various values of  $\eta_0$  with AdaGrad.

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 across epochs. 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 may not be always the case for real data.

By contrast, Figure 5.4 shows the objective function for  $\eta_0$  between 0.005 and 0.1 after incorporating AdaGrad to the implementation. These results show that training with AdaGrad is stable for a larger range of  $\eta_0$  without sacrificing convergence speed.

Finally, Figure 5.5 shows a direct comparison between training with and without AdaGrad using the best values of  $\eta_0$  for each configuration, i.e.  $\eta_0 = 0.1$  with AdaGrad and  $\eta_0 = 0.005$  without AdaGrad. In this figure both algorithms converge to some value, however it also shows that training with AdaGrad converges significantly faster while maintaining a more stable mean. These combined results indicate that using AdaGrad is the best strategy for learning the FFLDC model for this particular example.

## 5.4 Noise-to-data Ratio

The noise-to-data ratio  $v$  is another hyperparameter for NCE learning, the recommendation is to have as many noise samples as computationally possible (Gutmann and Hyvärinen, 2012). In order to better understand the effect of  $v$ , an experiment was performed where different models for the dataset from Example 4.2.2 are learned with different values of  $v$ . The setting is the same as for the learning rate experiments, but with  $\eta_0 = 0.05$  and always using AdaGrad. The noise parameter  $v$  was varied between 0.5 and 10.

The results for this experiment are shown in Figure 5.6. It shows that increasing the number of noise samples reduces the TVD of the resulting model as expected. However, it only decreases significantly for small values of  $v$ , this may be because the number of epochs is not sufficient for larger  $v$ . This last assumption was tested by running another

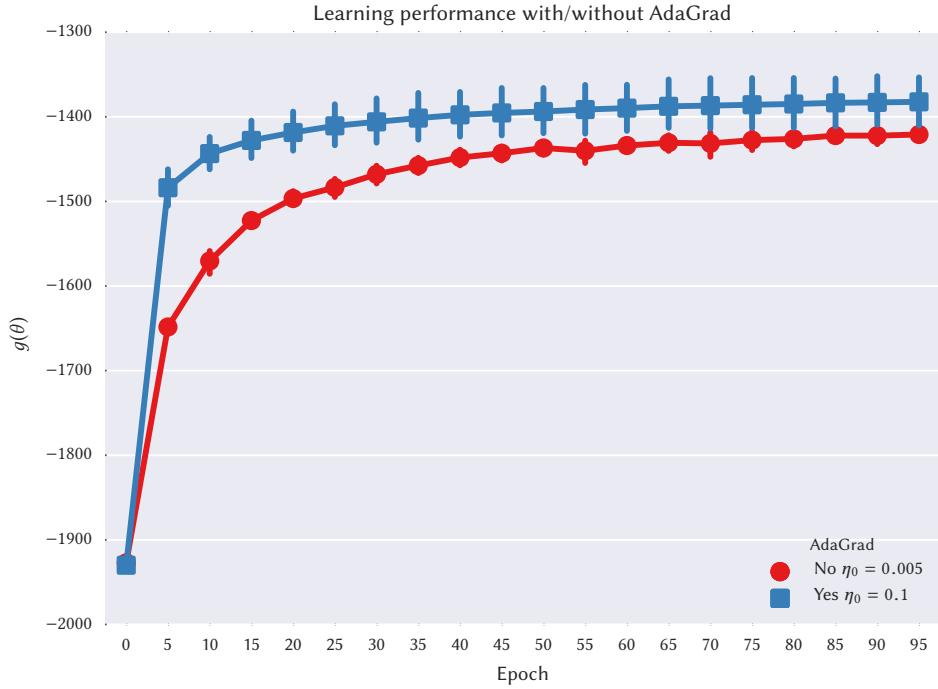


Figure 5.5: Comparison of learning performance with and without AdaGrad.

experiment with fixed  $v = 10$  but increasing the number of epochs from 100 to 1500, the results are presented in Figure 5.7.

Figure 5.7 shows that increasing the number of epochs can significantly improve the quality of the learned model. The decreasing distance indicates that it is possible for the NCE learning procedure to converge to the true distribution given sufficiently many noise samples and training epochs.

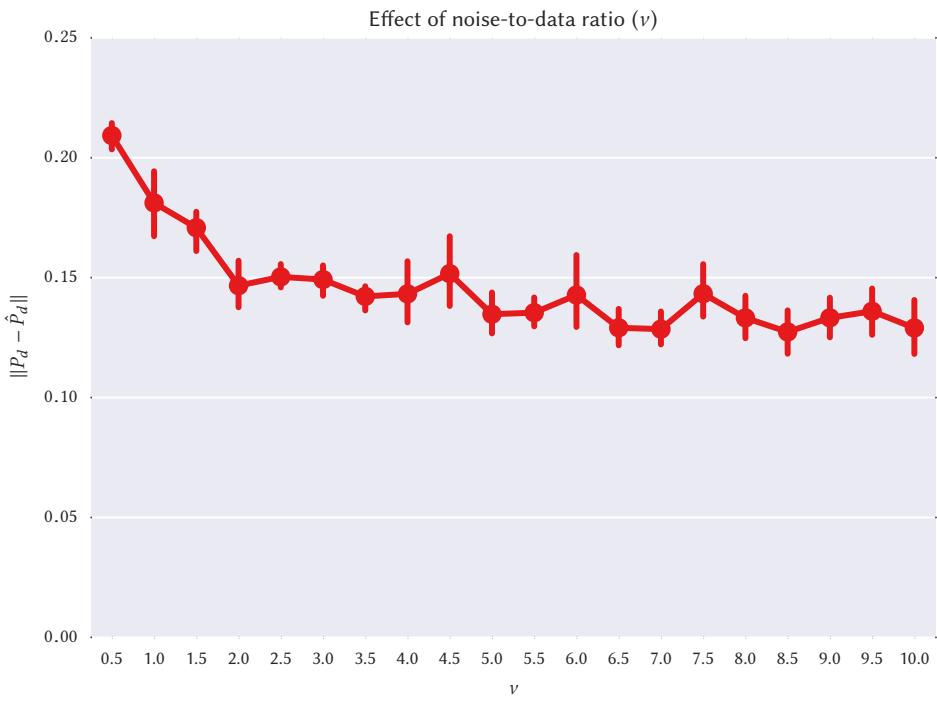
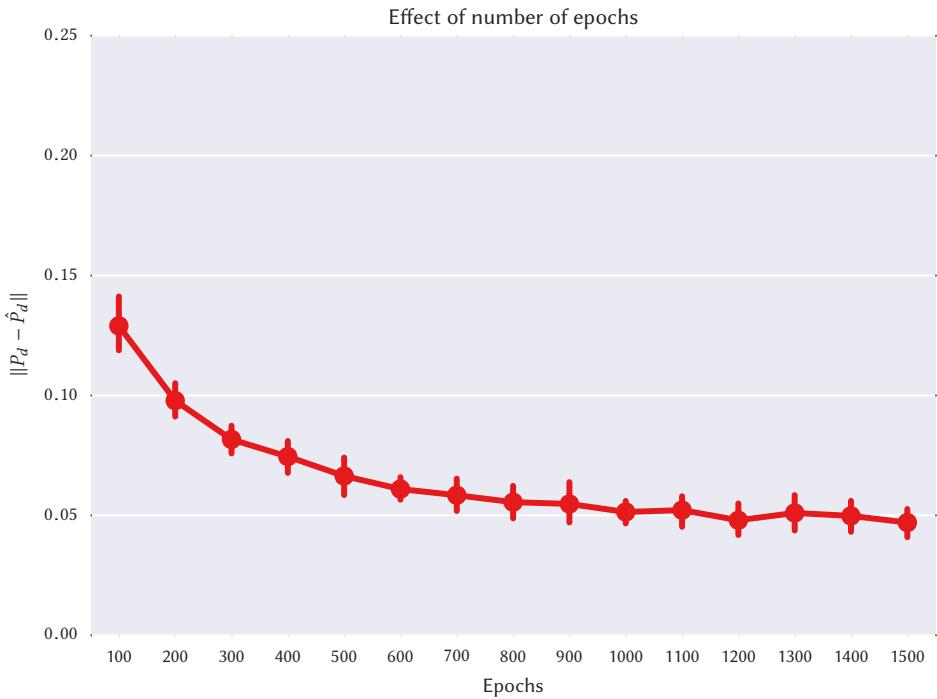
Figure 5.6: Total variation distance for different values of  $\nu$ .

Figure 5.7: Total variation distance for different number of epochs.

# Chapter 6

## Modeling an Image Collection - Setup

This chapter describes an application of the models presented in Chapter 4 to a real-world dataset. As mentioned in the introduction, finding and recommending paths for tourists is an interesting research topic. These paths can be understood as spatial summaries of a city, representing the most important locations to visit in a city.

These tourist paths can be discovered from a large collection of geotagged images by using the accompanying timestamps as explained in Section 6.2. Even though these paths are naturally ordered, they can also be considered as unordered sets of locations that represent the behavior of different visitors taking photographs in the city, which can be modeled using the FLID, FLDC and FFLDC models.

### 6.1 Flickr Dataset

For the experiments on real-world data, public geotagged photos from Flickr were used. There are close to two million geo-tagged photos available on Flickr<sup>1</sup>, this offers a unique and rich dataset for the task of spatial summarization. As discussed before, this work focuses on the city level. Therefore, the experiments were performed on a subset of the Flickr data, namely on public geotagged photos taken within Zürich, Switzerland.

The Flickr API<sup>2</sup> provides multiple functions to explore and use the data. In particular the search endpoint allows developers to retrieve photo records according to a specific criteria. Each photo record contains the following information:

**Owner** The ID of the user who uploaded and owns the photo.

**Title** A title given by the user.

**Description** A description given by the user.

**Visibility** Indicates if a photo is public, private or semi-private, i.e. shared only with friends.

**Timestamps** When the photo was taken, uploaded and last modified.

**Geo** Indicates if the photo has geographical information, and if yes it contains the geo-tag's precision, latitude and longitude.

**License** Copyright license.

---

<sup>1</sup><https://www.flickr.com/map> reports 1,833,307 geotagged photos as of April 2016.

<sup>2</sup>See documentation at <https://www.flickr.com/services/api/>



Figure 6.1: Bounding box used for filtering photos taken in the Zürich area.

**Tags** Text tags set by users.

**Machine tags** Automatic text tags set by Flickr.

**URL** The URL to download the photo.

These properties can be used to filter the search results. For collecting the desired data for Zürich the following criteria was used:

- The photo was taken between 2005 and 2015.
- The photo is marked as public and has no copyright restrictions.
- The photo is geotagged and has street level accuracy, i.e. latitude and longitude coordinates from GPS. The location of the photo must be within the area shown in Figure 6.1.

The API sets some limitations on the search function, namely it does not allow more than 3,600 requests per hour and it restricts the search results to the first 4,000 records. This means that it was not possible to issue a single search request with the outlined criteria, instead the retrieval was done over the course of several hours though single month queries. This ensured that each query had less than 4,000 results, hence retrieving all the available records. This search yielded 168,608 photos taken by 6,159 users.

Figure 6.2 shows the distribution of photos taken in 2015. The majority of them is concentrated at the city center but also around locations of interest outside the center like Uetliberg, Oerlikon and the Zoo. This indicates that the dataset contains information regarding various places to visit in Zürich. Figure 6.3 shows zoomed-in areas around the city showing that the data is mostly concentrated in touristic places such as the historic center, where Fraumünster, Grossmünster and the city hall are.

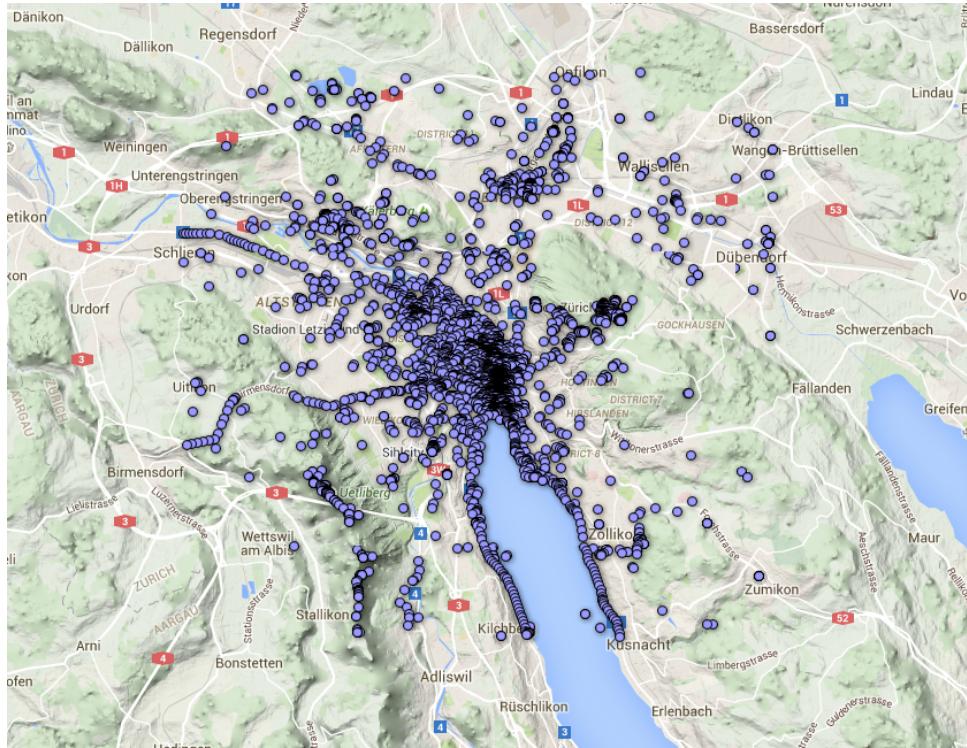
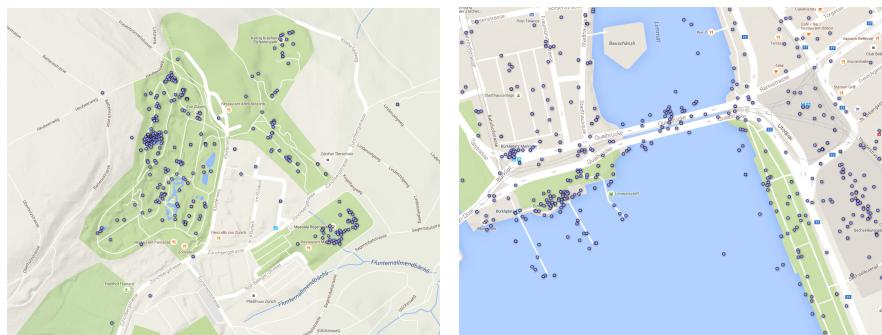
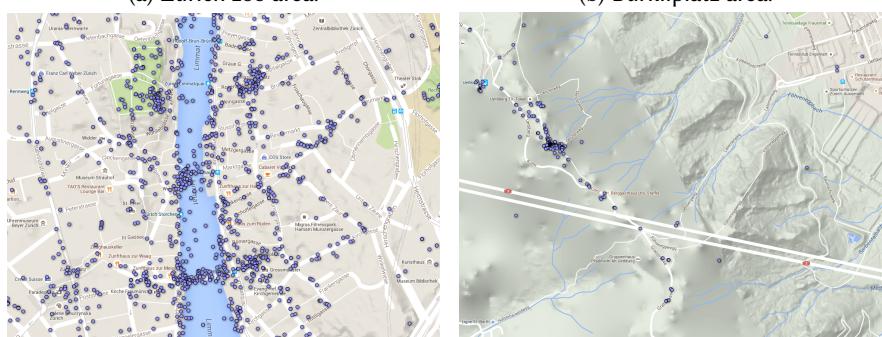


Figure 6.2: Feature map with the photos from the Zürich dataset taken in 2015.



(a) Zürich zoo area.

(b) Bürkliplatz area.



(c) Historic center area.

(d) Uetliberg area.

Figure 6.3: Zoomed-in maps displaying popular locations in Zürich.

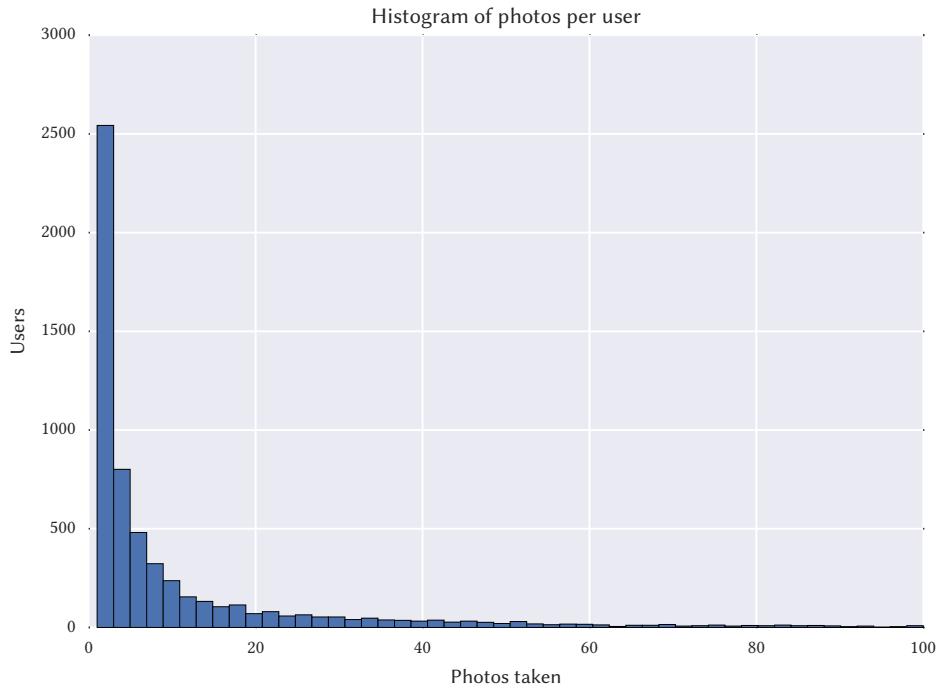


Figure 6.4: Histogram of photos taken per user.

Additional to the spatial coverage of the dataset, one interesting property to explore is how the photos are distributed across users. The histogram of photos taken by each user is shown in Figure 6.4. The x-axis is cut off at 100 photos because the distribution has a heavy tail which is due to professional users like newspapers and event venues that have uploaded hundreds of photos under a single user. These users are not of particular interest for the use case of this work because they do not reflect the interests of a typical visitor. The histogram shows that most of the users uploaded only few pictures which indicates that most of the photos come from very active users that take hundred of photos and this should be taken into account when analyzing the results.

## 6.2 Path Finding

The collected photos were used to identify popular locations that are photographed by many people. These locations define the ground set  $V$  for the considered models. In order to identify the popular locations automatically, the method proposed by Kleinberg et al. (2009) was used.

Kleinberg et al. (2009) define the problem of finding highly-photographed locations as a clustering problem in a two-dimensional feature space, namely the space defined by the spatial coordinates of each photo. The clustering method used for this problem is mean-shift clustering.

### 6.2.1 Mean-shift Clustering

Mean-shift clustering is a technique proposed by Cheng (1995), it is commonly used for image segmentation tasks (Comaniciu and Meer, 2002). However, as proposed by Kleinberg et al. (2009), it can also be used for the task of identifying highly photographed locations in an collection of geotagged photos.

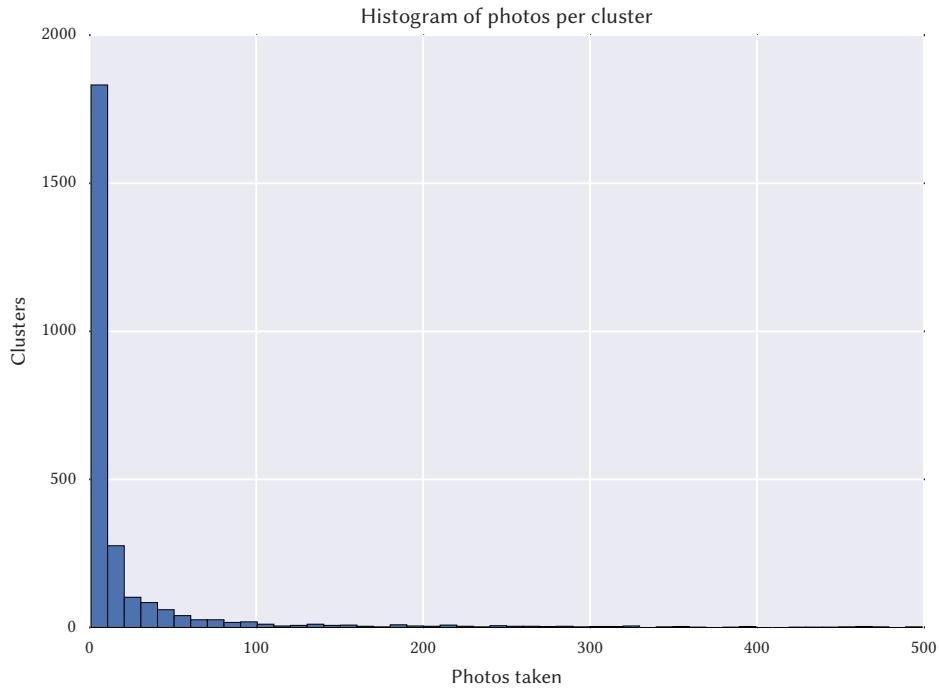


Figure 6.5: Histogram of the number of photos per cluster. The x-axis is cut off at 500 photos because the distribution has a heavy tail.

It is a non-parametric technique for locating the modes of a probability distribution over a feature space, this works by using Kernel Density Estimation (KDE) to estimate the gradient of the underlying distribution from the data and then performing a mean-shift procedure to identify the local maxima or modes, this is also known as hill-climbing. This can be naturally applied to a collection of geotagged photos by considering the feature space of latitude and longitude coordinates.

This approach assumes that there is an underlying distribution where the modes are locations of interest or landmarks where many of the photos were taken. Mean-shift has an advantage over other clustering techniques such as k-means for this problem, because the bandwidth used for the kernel has a physical meaning, namely it defines the radius of the area around each landmark.

Scikit<sup>3</sup> was used for the mean-shift clustering implementation. The clustering was performed on the complete photo dataset with the unnormalized latitude and longitude features using a Gaussian kernel. The kernel's bandwidth was 0.001 degrees which approximately represents a radius of 100m without significant errors for the geographic scale of Zürich.

The clustering procedure yielded 2,686 clusters covering 137,019 photos, the left over photos represent outliers that are not inside any of the resulting kernels. Figure 6.5 displays a histogram of photos per cluster, it shows that over half of the clusters only contain one or two photos. This indicates that many of the clusters are actually not landmarks or highly photographed places, so in order to concentrate the model around the most interesting locations only the top  $N$  clusters sorted by the number of photos were used.

Two datasets were constructed, a small one with the top 10 clusters and a large one with the top 100 clusters. The small dataset was mainly used during the development of the models to test the implementation and try out different settings. On the other hand, the

<sup>3</sup><http://scikit-learn.org/stable/>

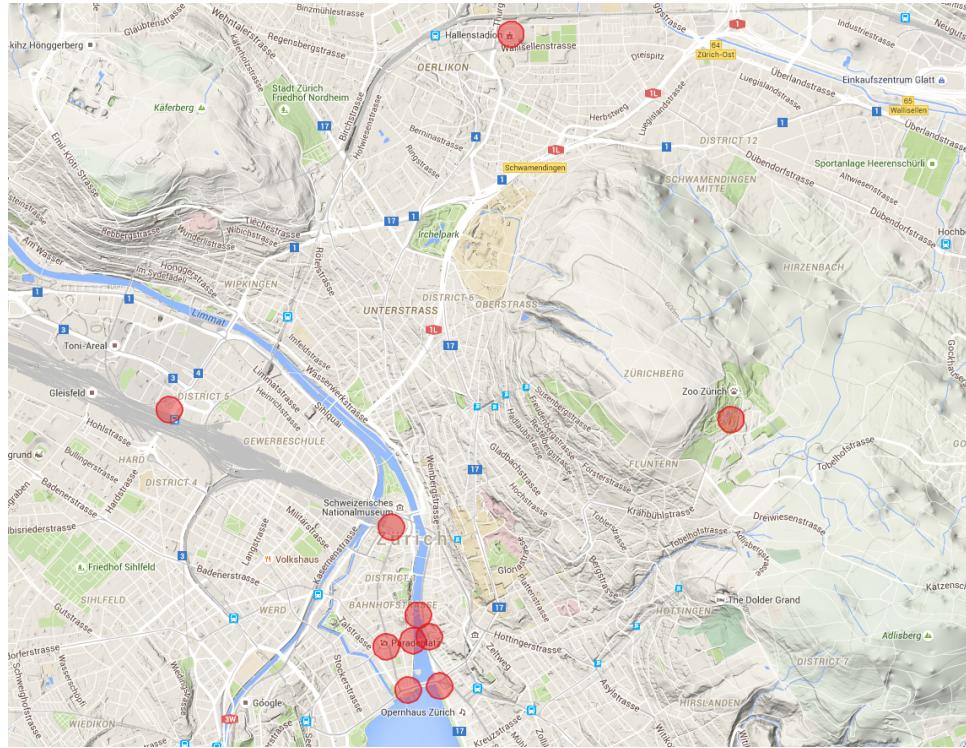


Figure 6.6: Top 10 clusters in Zürich according to the photo count.

large dataset was used for assessing the quality of the final models and comparing them against the baseline models.

Figures 6.6 and 6.7 show the selected 10 and 100 settings clusters, respectively. Additionally, Table 6.1 presents additional information about the top 10 clusters in the small dataset.

To extract paths taken by individual users, the following procedure was used:

1. The photos were grouped by user and day when they were taken.
2. The photos in each user and date group represent a single path, ordered by date and time.
3. The photos were replaced in the paths by the id of the corresponding clusters, including the cluster only once. If the photo did not belong to one of the top clusters, then the photo is discarded.
4. The resulting sequence of clusters represent the paths taken by users through the city.

These sequences were used as the unordered sets for the set models, e.g. FLID, FLDC, and as ordered sequences for the baseline models that incorporate order. Table 6.2 shows the statistics for the final datasets.

Additionally, Figures 6.8 and 6.9 show the histograms of the length of the sets in the small and large datasets, respectively. This shows that most of the sets are either singletons or pairs, which matches the fact that many users just take a few photos as it was shown in Figure 6.4.

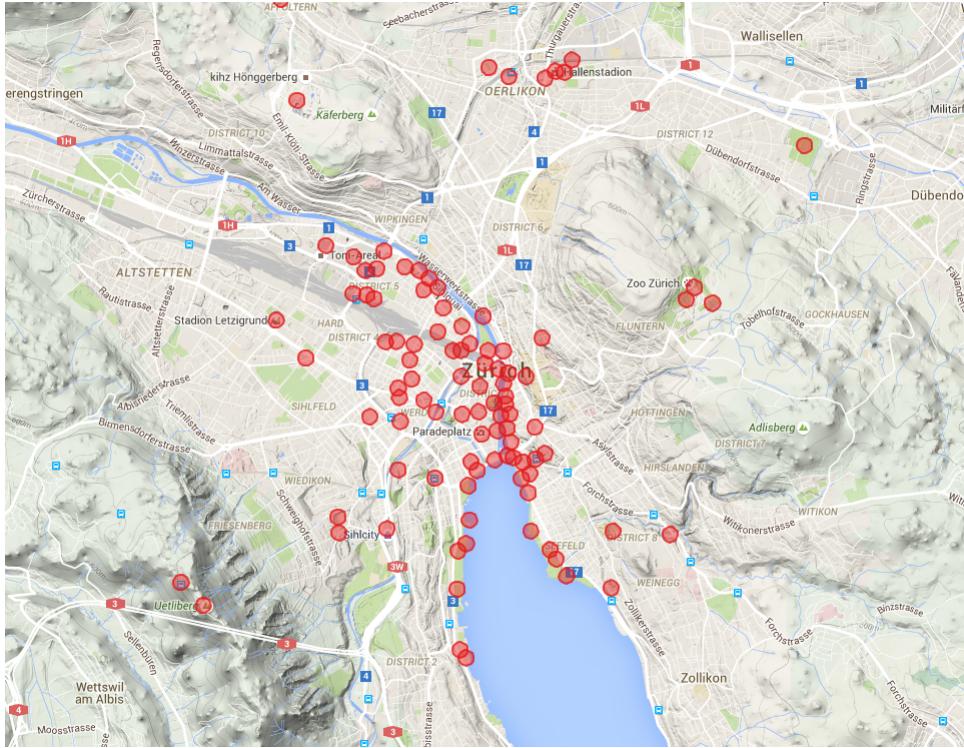


Figure 6.7: Top 100 clusters in Zürich according to the photo count.

Table 6.1: Top 10 clusters for Zürich.

Landmark	Photos taken	Unique users
Hauptbahnhof (Main train station)	4224	932
Grossmünster (Church)	3797	951
Bürkliplatz (Outdoors)	3728	612
Hallenstadion (Event venue)	3575	82
Fraumünster (Church)	3414	889
Bellevueplatz (Outdoors)	2584	437
Rathaus (Town hall)	2569	690
Zoo	2462	151
Paradeplatz (Outdoors)	2372	438
Prime Tower	1930	257

Table 6.2: Path dataset statistics for Zürich.

	Small ( $ V  = 10$ )	Large ( $ V  = 100$ )
Included photos (% of total)	30655 (18%)	91661 (54%)
Paths extracted, i.e. $ \mathcal{D} $	6781	16439
Singleton paths, i.e. $ S  = 1$	5396	12927
Paths with 2 elements, i.e. $ S  = 2$	799	1980

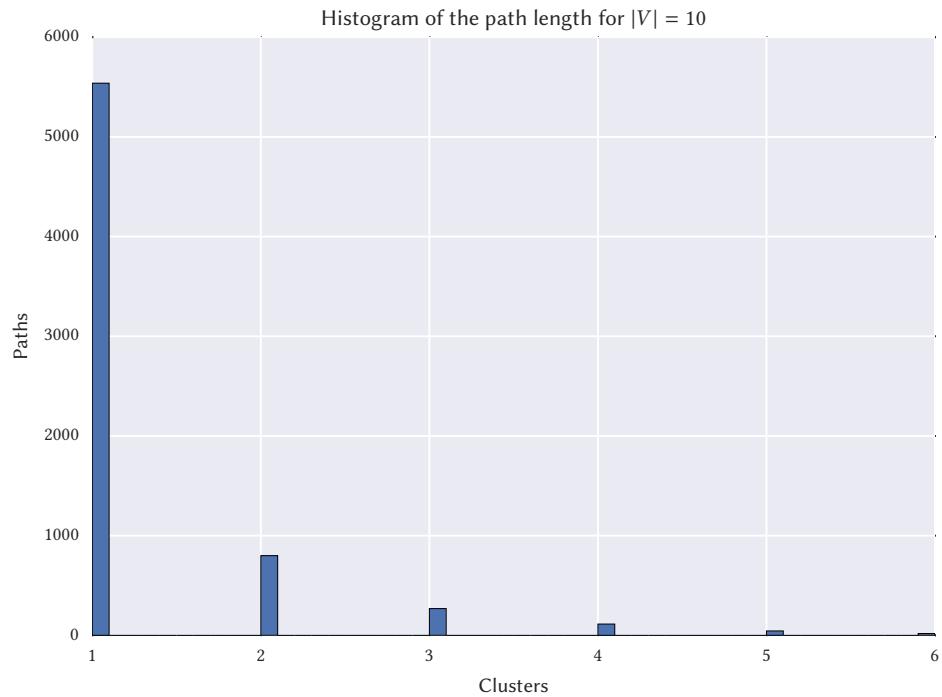


Figure 6.8: Histogram of the lengths of the paths for the small dataset.

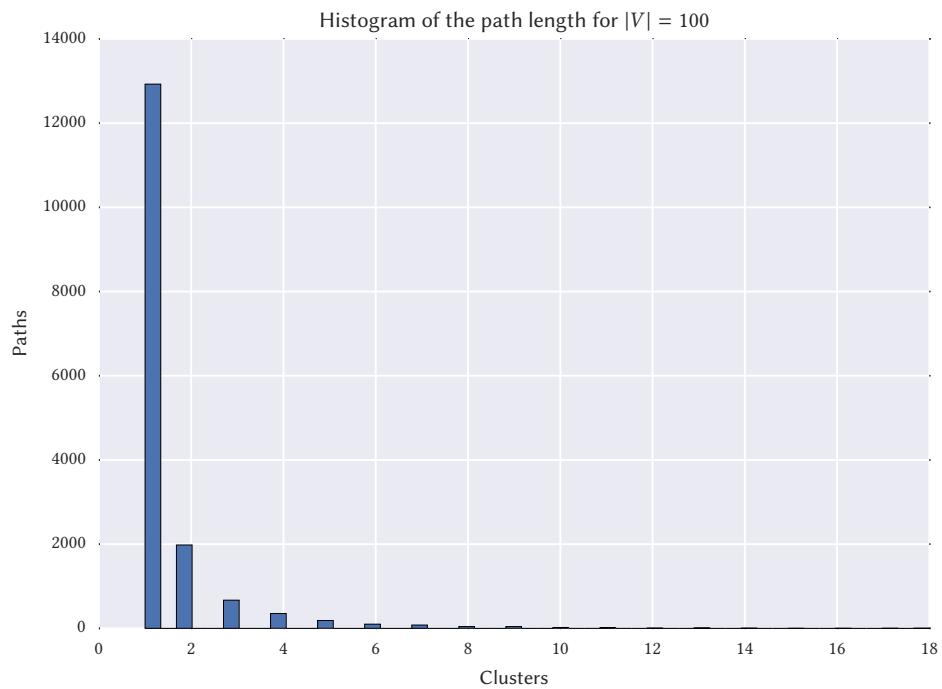


Figure 6.9: Histogram of the lengths of the paths for the large dataset.

## 6.3 Models

For the experiments, the FLID, FLDC and FFLDC models were used. The following implementation details were used in the experiments, unless noted otherwise:

- The data was split into 8 random folds, with 87.5% of the data used for training and 12.5% for testing in each one.
- The noise was generated using the log-modular model as described for the synthetic experiments.
- AdaGrad was used and  $\eta_0$  was 1 and 0.1 for the small and large datasets, respectively.
- The number of epochs was 1000.
- The noise-to-data ratio  $\nu$  was 5.

## 6.4 Baselines

This section describes the baseline models that were used as a comparison for the proposed models.

### 6.4.1 Markov Model

A first-order Markov chain can be used to model the behavior of Flickr users, making the simplifying assumption that each tourist selects the next location to visit based only on the current location. This model depends on the transition probabilities between the different landmarks in the dataset, i.e.  $P(l_{t+1} = i \mid l_t = j)$  where  $l_{t+1}$  represents the next location and  $l_t$  the current one. These probabilities can be estimated by counting all transitions in the dataset, i.e.

$$\hat{P}(l_{t+1} = i \mid l_t = j) = \frac{N(i,j)}{N(*,j)}, \quad (6.1)$$

where  $N(i,j)$  is the number of times that the location  $i$  was photographed immediately after location  $j$ , and  $N(*,j)$  is the number of times that any location was photographed after  $j$ . This model was used successfully by Kurashima et al. (2010) with good results for recommending tourist routes for a city based on Flickr data, therefore it is a natural choice for comparison against the newly proposed models. A full path can be constructed by selecting a starting location and choosing the next one according to the highest transition probability, then repeating the procedure until a desired length is reached. Note that this model can be estimated efficiently from data, concretely it requires  $O(|\mathcal{D}|)$ .

### 6.4.2 Heuristic Markov Model

A strict Markov model can only consider one transition to make decisions, however a natural heuristic that can improve the quality of the model is to consider the complete path history to avoid recommending locations that were already visited. We refer to this baseline as the heuristic Markov model.

The transition probabilities for predicting an element  $l_j$  in the middle of a sequence  $S_t$  are then defined as:

$$P(l_j = i \mid l_1, \dots, l_{j-1}, l_{j+1}, \dots, l_t) = \begin{cases} 0 & \text{if } i \in \{l_1, \dots, l_{j-1}, l_{j+1}, \dots, l_t\} \\ \frac{N(i, l_{j-1})}{N(*, l_{j-1})} & \text{otherwise} \end{cases} \quad (6.2)$$

This heuristic is included to provide a fairer comparison to the models defined over sets, such as FLID, which also consider all the other elements in the set when recommending an element to include.

### 6.4.3 Proximity

Another model is to assume that users photograph the next closest location to their current one. This can be useful to model paths that are concentrated in certain areas, for example in the Zürich dataset many of the paths only visit locations in the center and users seldom take photographs at the Zoo and a landmark in the center on the same day. Formally, this model is characterized by a distance  $d(i,j)$  computed from the coordinates of elements  $i$  and  $j$ . This can be used to predict paths that minimize the distance between elements. For example to predict the an element  $s_j$  in the middle of a sequence  $S_t$ , the element closest to  $s_{j-1}$  and  $s_{j+1}$  is selected, i.e.  $s_j = \operatorname{argmax}_{i \notin S_t} (d(s_{j-1}, i) + d(s_{j+1}, i))$ . This is a natural choice for a baseline model because it provides a very quick model that is not completely random, the distances in this model can be computed in  $O(|V|^2)$  time, which is less than  $|\mathcal{D}|$  for both datasets. To construct a full path, the same procedure as for the Markov chain can be used.

### 6.4.4 Log-modular

As described before, a log-modular model is the simplest log-submodular model therefore it is a natural comparison for the FLID and FLDC models. The log-modular model can also be used with features by solving the linear system in Equation (5.1) to make it comparable to the FFLDC model.

The log-modular model can be estimated from data following the procedure defined in section 3.4.1, which only requires  $O(|\mathcal{D}|)$  time.

This model is significantly different from the two previous baselines because it does not model the order of the elements, simply the membership. However, it is also possible to build a path by first constructing a set of fixed size  $n$  and then ordering the elements in order to minimize the total travel distance. A set of size  $n$  can be constructed by selecting the  $n$  locations with highest utility, i.e.  $\operatorname{argmax}_i u_i$ .

## 6.5 Evaluation

The evaluation of the models aims to assess their quality for the task of recommending additional locations to visit based on an initial set of locations that the user wants to visit. This recommendation can be structured as a path either directly from the output of the model, e.g. for the Markov chain case, or as a post-processing step, i.e. for the set based models. The evaluation procedure focuses on one-step recommendations which can be used both to construct a full path greedily or for an interactive system where the user requests a single location at each step.

The evaluation task is to predict the missing location in a sequence given the other elements, both before and after. This can be constructed by removing one element at a time from the paths in the test dataset. Thus each path of length  $k$  generates  $k$  partial paths for testing with a missing element. For example the path [1, 5, 6], generates 3 partial paths, namely [?, 5, 6], [1, ?, 6] and [1, 5, ?]. Note that only paths of at least size 2 are used in the task.

After each model is trained, the predictions for each path  $S_t = [s_1, \dots, s_{j-1}, s_{j+1}, \dots, s_t]$  where  $s_j$  is missing, are made as follows:

**FLID, FLDC, FFLDC, Log-modular** The element with maximum probability, that is not already in the path is predicted, i.e.  $\operatorname{argmax}_{i \notin S} P(S \cup \{i\})$  where  $S$  is the set of the elements in path  $S_t$ .

**Markov, Heuristic Markov** The element with the highest probability given  $s_{j-1}$  and  $s_{j+1}$  is predicted, i.e.  $\operatorname{argmax}_i P(s_{j+1} | i)P(i | s_{j-1})$ . When using the heuristic the prediction becomes,  $\operatorname{argmax}_{i \notin S_t} P(s_{j+1} | i)P(i | s_{j-1})$

**Proximity** Using the unordered set  $S$  of the elements in  $S_t$ , it predicts the element closest to all elements in the set, i.e.  $\operatorname{argmin}_{i \notin S} \sum_{j \in S} d(i, j)$ .

**Proximity Ordered** Similar to the previous model, but considering the order. The element closest to  $j - 1$  and  $j + 1$  that is not already in the path is predicted, i.e.  $\operatorname{argmin}_{i \notin S_t} d(j - 1, i) + d(i, j + 1)$ .

The metric used for the quality of the predictions is overall accuracy, i.e. the percentage of correct predictions. Formally,

$$Acc_{model} = \frac{1}{|\mathcal{T}|} \sum_{S_t \in \mathcal{T}} (\mathbb{I}_{s_j(i)}) \quad (6.3)$$

Where  $\mathcal{T}$  is the set of all partial paths available for testing and  $\mathbb{I}_{s_j}$  is an indicator function that evaluates to 1 when the predicted element  $i$  is equal to the missing element in the sequence  $s_j$  and 0 otherwise.

Accuracy is the most used metric for recommender systems (Herlocker et al., 2004) and it was the metric used by Kurashima et al. (2010) for evaluating single step recommendations in tourist routes, although they only perform forward evaluation because it is dealing only with sequences as in the Markov case. This accuracy metric is also similar to the leave-one-out procedure used by Tschiatschek et al. (2016) to evaluate the FLID model for the product recommendation task. Therefore, it is expected to provide a good assessment on the quality of the proposed models for this particular task while balancing the fact that some models work on paths and others deal with sets. The accuracy results will be presented as the average of the various data folds with 95% confidence intervals, unless stated otherwise.

An alternative metric is the log-likelihood relative improvement (LLRI) proposed by Tschiatschek et al. (2016). This measures the model fit on the test dataset and indicates its relative improvement over the log-likelihood of the log-modular model. Formally,

$$LLRI = 100 \frac{\mathcal{L}_{model} - \mathcal{L}_{log-modular}}{|\mathcal{L}_{log-modular}|} \quad (6.4)$$

Where  $\mathcal{L}_{log-modular}$  is the log-likelihood of the test dataset for the log-modular model and  $\mathcal{L}_{model}$  for the evaluated model, e.g. FLID, FLDC.



# Chapter 7

## Modeling an Image Collection - Results

This chapter presents the results for different experiments performed on the datasets introduced in the previous chapter. Different models are evaluated and their outcome is discussed.

### 7.1 The Problem with Singletons

The histograms in Figures 6.8 and 6.9 show that most of the path datasets are composed on singleton paths representing users that only took photos around a small area on a single day. However, the evaluation procedure described in Section 6.5 only deals with paths of at least 2 items, this represents a problem because the kind of data used for learning is not the same as for the evaluation phase. Moreover, initial experiments showed that the distribution of singleton paths is significantly different from that of paths with 2 or more elements.

In order to illustrate this, a special dataset only containing paths of length 2 or more was constructed. The log-modular model was then trained and evaluated on both versions of the datasets. This was done for both the small ( $|V| = 10$ ) and large ( $|V| = 100$ ) datasets.

The accuracy of the resulting models for the test data is shown in Figure 7.1. It shows that there is a significant improvement in the accuracy when only the distribution of non-singleton paths is considered for the log-modular model. This effect is also expected for the more complex models, e.g. FLID, because their initialization is based on the log-modular model. Note that for the Markov model there is no difference between including singletons or not because transition probabilities can only be learned from paths of 2 or more elements.

Because of these results, the rest of the experiments were performed on the modified datasets without singletons which contain more significant data.

### 7.2 Small Dataset

First, the results from the baseline models are presented in Figure 7.2. These show that the log-modular model outperforms the Markov baselines in this case. This is encouraging as it serves as the base for the more complex models. The results also show that the heuristic Markov model outperforms the strict Markov, this is expected in all cases because the dataset was constructed in such a way that elements are not repeated, hence the proposed heuristic always improves the prediction by removing the already visited elements from

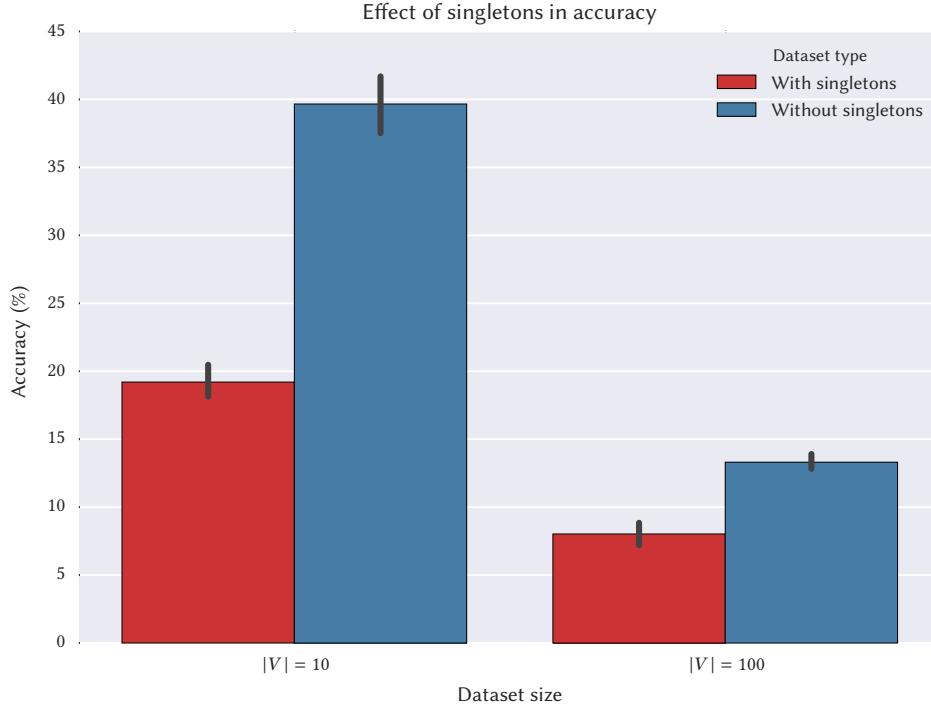


Figure 7.1: Accuracy of log-modular models for different dataset configurations.

the possible choices. The proximity model is the best for the small dataset, and the different between the proximity model with and without ordering show that adding ordering information improves the score.

### 7.2.1 Diversity & Coherence

The first approach to the problem was to use the FLID model, i.e. modeling only diversity. Figure 7.3 shows the model accuracy for different numbers of dimensions. Comparing these values with the log-modular model shows that there is no significant improvement even with a large number of diversity dimensions. Increasing the number of noise samples or epochs did not improve these results either.

It is possible that for this dataset, diversity is not sufficient to capture the underlying distribution. Therefore, the next step was to use the FLDC model. After executing a grid search for  $0 \leq L \leq 10$  and  $0 \leq K \leq 0$  with the FLDC model the accuracy did not improve either.

The fact that the FLID and FLDC models do not result in improved accuracy with respect to the log-modular baseline can be explained by looking at the distribution of the paths in the data. The top 10 paths and their proportions in the dataset are listed in Table 7.1.

The paths in Table 7.1 are all pairs, which are harder to predict because there is only information about one item. Simply ordering by frequency should give a high accuracy, as shown by the log-modular model's accuracy in Figure 7.2.

However, the lack of improvement in accuracy does not mean that the FLID and FLDC models are not learning beyond the log-modular model. This can be seen by comparing the total variation distance between the empiric distribution of the paths, after removing the ordering, and the distributions produced by the learned models. Note that in this case the number of items is small enough that the computation of the partition function is tractable even using the exhaustive method that requires  $O(2^{|V|})$  time. Figure 7.4 shows the total

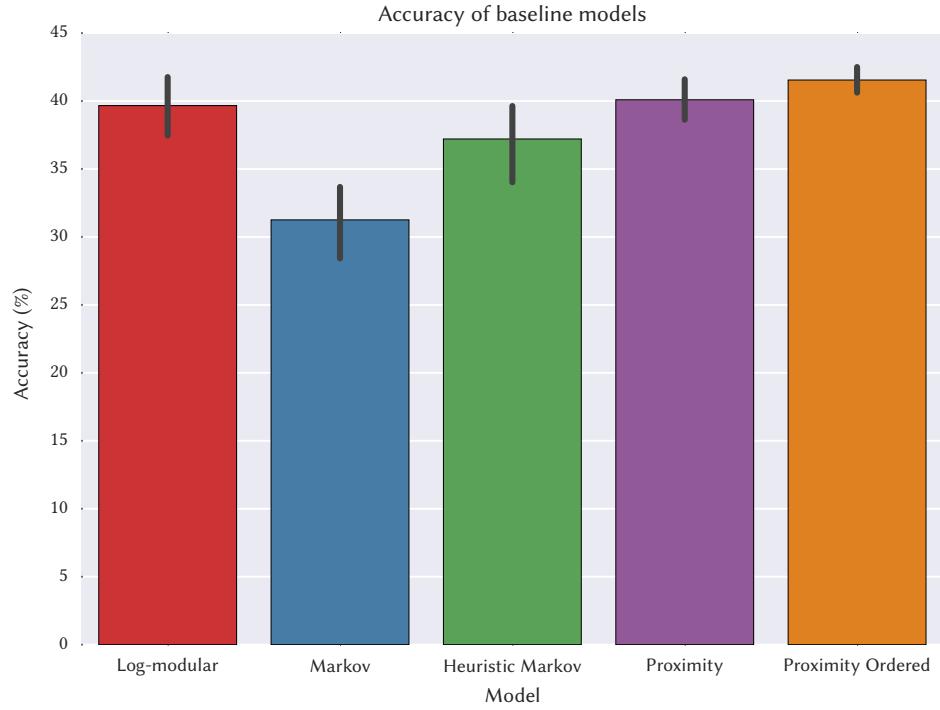


Figure 7.2: Accuracy of baseline models for the small dataset.

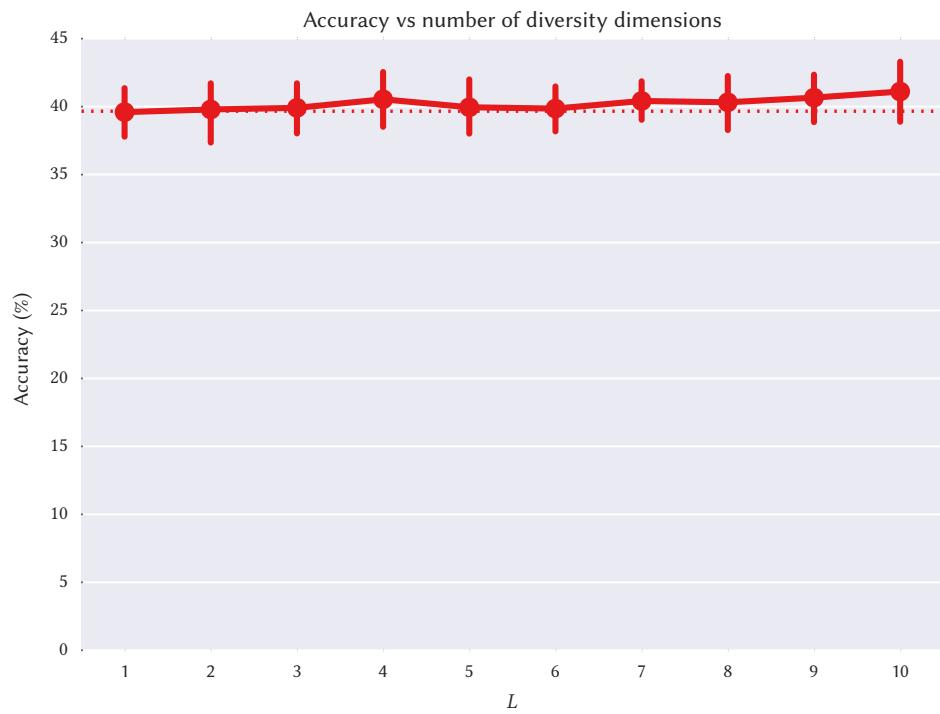


Figure 7.3: Accuracy for various values of  $L$ . The dotted line shows the mean accuracy of the log-modular model.

Table 7.1: Most frequent paths in the small Zürich dataset.

$S$	$N(S)$
[1, 4]	71(5.7%)
[4, 1]	58(4.7%)
[6, 1]	47(3.8%)
[6, 4]	43(3.5%)
[1, 6]	37(3.0%)
[4, 6]	33(2.6%)
[2, 4]	26(2.1%)
[2, 5]	25(2.0%)
[5, 2]	24(1.9%)
[4, 8]	23(1.8%)

variation distance for different FLID and FLDC models. It shows that combining diversity and coherence improves the approximation of the model to the empiric distribution when compared to the TVD for the log-modular model, which was 0.40.

Finally, Figure 7.5 shows the LLRI metric for the FLDC models. This shows more clearly that there is an improvement over the log-modular model, even though it is not reflected on the accuracy metric. The best choice of dimensions is  $L = 5, K = 5$  according to both the LLRI and TVD metrics.

### 7.3 Large Dataset

The baseline models were trained on the large dataset, i.e.  $|V| = 100$ , and the accuracy results are presented in Figure 7.7. In this case, there is a more significant difference between the log-modular and the other baselines. This indicates that simply choosing the most frequent element is not as good of a strategy as for the small dataset, therefore complex models like FLDC and FFLDC should yield better results. Additionally, the significantly higher accuracy of the (Heuristic) Markov and Proximity Ordered models shows how adding information about the ordering, i.e. using paths, can improve the results in comparison to modeling only sets.

Different FLDC models were trained on the large dataset, in order to find the best setting for the dimensions  $L$  and  $K$  a grid search was performed. The average accuracy for each setting is presented in Figure 7.6. This shows that the FLDC model is significantly better for the path completion task than the log-modular baseline, however there is not a visible difference between the various settings for the model. Moreover, these results suggest that models with large number of dimensions have worse accuracy in the test set, this is due to the large number of parameters in the model which would require more data to estimate properly.

Finally, the best FLDC model is compared to the baselines in Figure 7.7. Even though the FLDC model is an improvement over the log-modular, it still does not reach the accuracy of the Markov or Proximity Ordered models. This can be explained by the fact that the first three models in the figure include information about the order of the elements in a set, i.e. they model probabilities over paths, while the others only model membership of the elements in a set. This extra information helps these models perform the prediction task more accurately.

Additional experiments with features were performed on both datasets, the objective was to identify features that could be generalized to other cities from the available data. Some

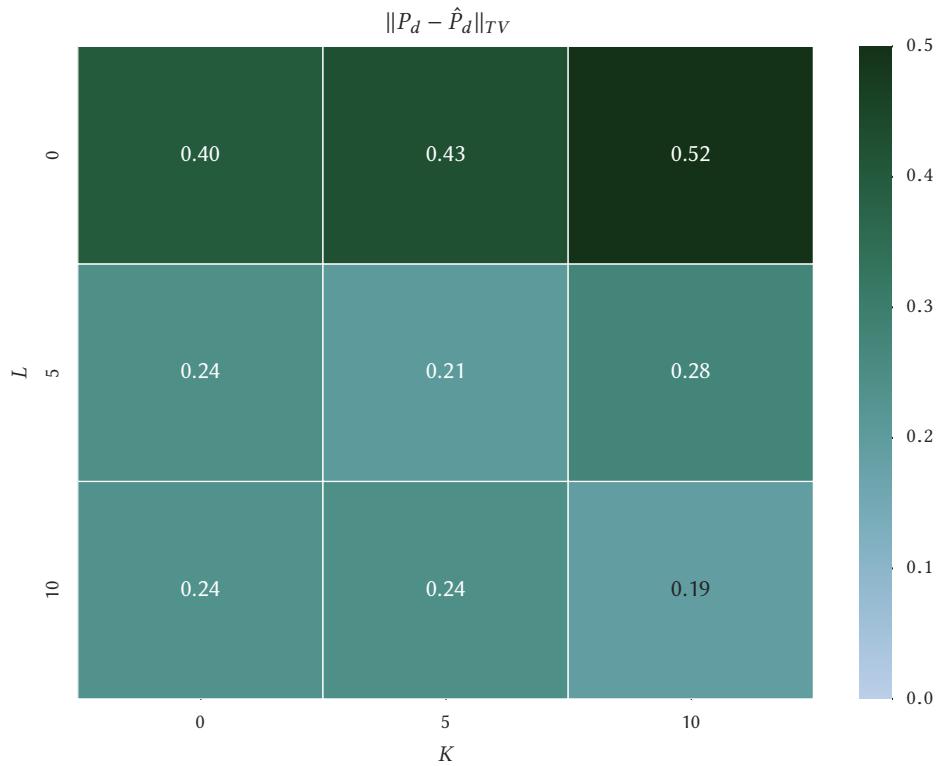


Figure 7.4: Total variation distance for FLDC models on the small dataset.

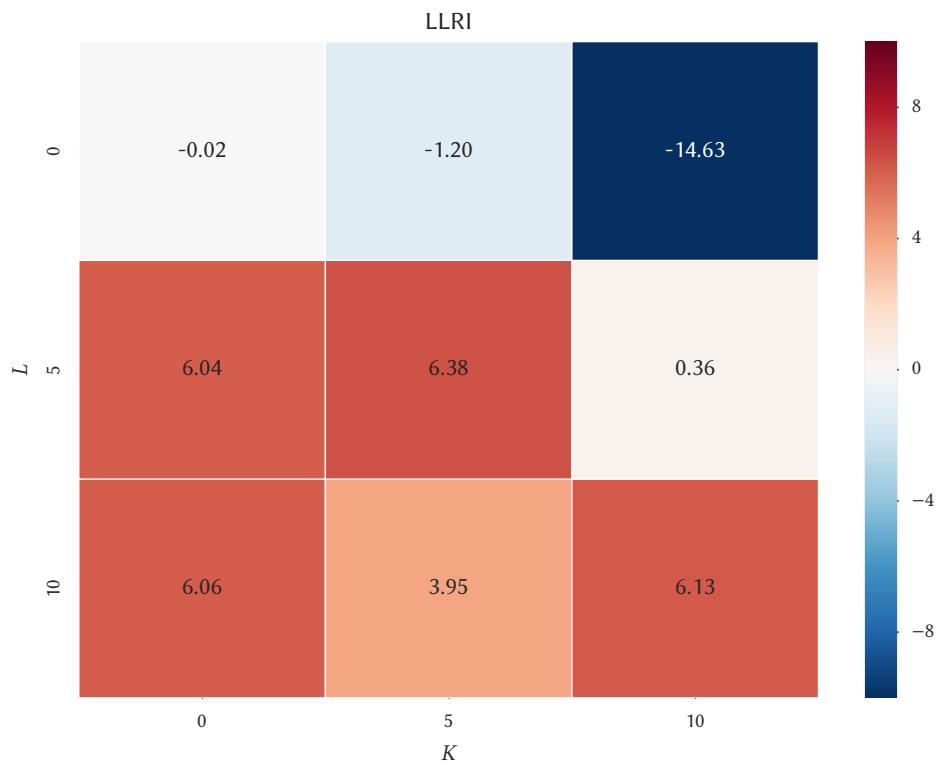


Figure 7.5: Log-likelihood relative improvement (LLRI) for FLDC models on the small dataset.

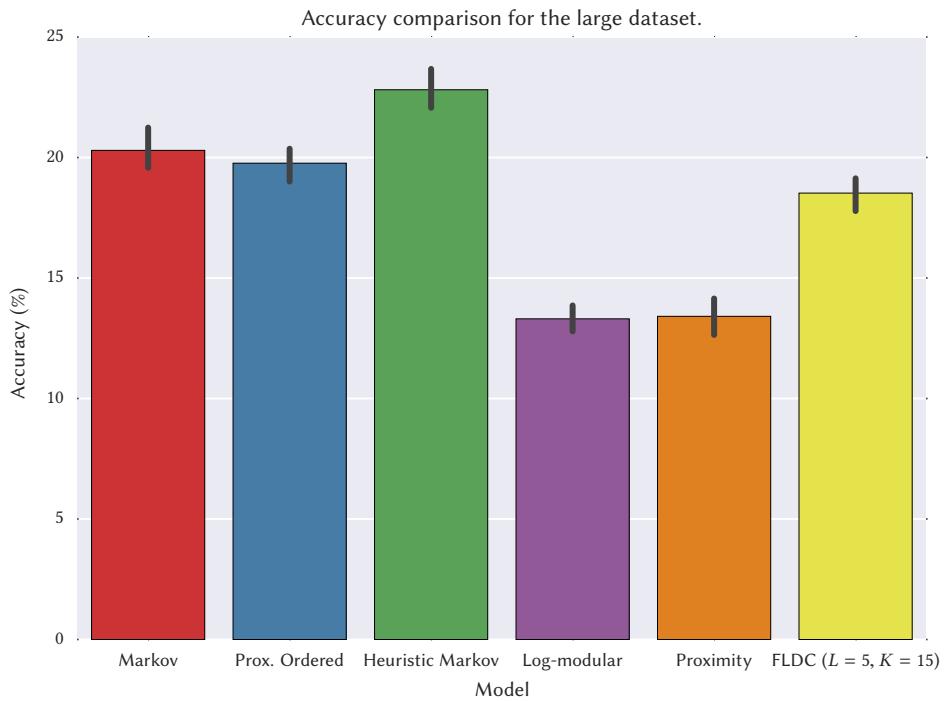
Figure 7.6: Accuracy of FLDC models with varying  $L$  and  $K$  values.

Figure 7.7: Comparison of baseline models to the best FLDC model for the large dataset.

of the features used were:

- Number of photos per cluster
- Number of photographers per cluster
- Characteristics about the venue, e.g. outdoors or indoors.
- Type of venue, e.g. a church or a museum.

The last two sets of features were used only in the small dataset where it was possible to quickly characterize each location. Experiments with these features did not produce successful results in terms of accuracy and the feature matrices needed to be augmented with the identity to achieve similar performance as the FLDC model. However, it is not clear that this is a problem with the FFLDC model itself. We believe that it is an issue of the data and features chosen, we expect that other settings would prove successful and take advantage of the generalization capabilities of the FFLDC model.



# Chapter 8

## Conclusion

We have proposed two new models, FLDC and FFLDC, that extend the capabilities of a novel submodular probabilistic model, namely FLID. We have shown how the FLDC model can capture coherence in addition to diversity, and how FFLDC allows generalization through the use of features. Synthetic experiments show the performance of noise contrastive estimation for learning these models from data and the practical considerations that must be addressed such as how to avoid local maxima and the effect of various hyperparameters.

This type of models has been commonly used for document and image summarization, but we show how it can also be applied for recommending touristic locations using information extracted from an online image collection, namely Flickr. A complex probabilistic model such as FLDC captures higher order information resulting in a better approximation of the underlying data distribution.

Existing techniques for landmark identification allowed us to extract information and useful datasets from geo-tagged photos taken in the city of Zürich. Around 160,000 photos were collected representing perspectives of thousands of users, these were then translated into popular locations and user trips from which tourist behavior and routes were learned. For the large dataset, our best model was able to outperform the baselines that only model membership of elements in a set. However the score was lower than for models that include information about the order of the elements, i.e. those that can model sequences. Devising a model that naturally incorporates information about order of the elements is an interesting direction for future work. However, this could significantly increase the complexity of models which in its current form already make computations intractable due to the combinatorial nature of the set domains.

Spatial features of the photos were mainly used for the identification of landmarks and modeling the user's behavior, however more study is required to identify other features that can be used with the FFLDC model to generalize what can be learned from one city to another. Some experiments were performed in this direction but no conclusive results were obtained.

Despite the limitations, we believe that the proposed models can be applied to a wide range of problems because they can model complex distributions and identify characteristics about the data, such as item similarity.



# 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.
- Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799.
- Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619.
- 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)*.
- Donaire, J. A., Camprubí, R., and Galí, N. (2014). Tourist clusters from Flickr travel photography. *Tourism Management Perspectives*, 11:26–33.
- 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.
- Gotovos, A., Hassani, S. H., and Krause, A. (2015). Sampling from probabilistic submodular models. In *Neural Information Processing Systems (NIPS)*.
- 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.
- Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

- Jaffe, A., Naaman, M., Tassa, T., and Davis, M. (2006). Generating summaries and visualization for large collections of geo-referenced photographs. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval - MIR '06*.
- 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.
- Kennedy, L., Naaman, M., Ahern, S., Nair, R., and Rattenbury, T. (2007). How flickr helps us make sense of the world: context and content in community-contributed media collections. In *Proceedings of the 15th international conference on Multimedia*, pages 631–640.
- Kennedy, L. S. and Naaman, M. (2008). Generating diverse and representative image search results for landmarks. *Proceeding of the 17th international conference on World Wide Web - WWW '08*.
- Kleinberg, J., Huttenlocher, D., Backstrom, L., and Crandall, D. (2009). Mapping the World’s Photos. In *Proceedings of the 18th International Conference on World Wide Web*, pages 761–770.
- Krause, A. and Golovin, D. (2014). Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, chapter 3. Cambridge University Press.
- Kurashima, T., Iwata, T., Irie, G., and Fujimura, K. (2010). Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 579–588.
- Levin, D. A., Peres, Y., and Wilmer, E. L. (2008). *Markov chains and mixing times*. American Mathematical Society.
- Li, Y., Crandall, D. J., and Huttenlocher, D. P. (2009). Landmark classification in large-scale image collections. In *Proceedings of the IEEE International Conference on Computer Vision*, number 12, pages 1957–1964.
- Popescu, A., Grefenstette, G., and Moëllic, P.-A. (2009). Mining tourist information from user-supplied collections. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*.
- Serdyukov, P., Murdock, V., and van Zwol, R. (2009). Placing flickr photos on a map. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 484–491, New York, NY, USA. ACM.
- 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.
- Shahaf, D., Yang, J., Suen, C., Jacobs, J., Wang, H., and Leskovec, J. (2013). Information cartography. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, pages 1097–1105, New York, New York, USA. ACM Press.
- Tschiatschek, S., Bilmes, J., Haochen, W., and Iyer, R. (2014). Learning Mixtures of Submodular Functions for Image Collection Summarization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1413–1421.
- 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.



# List of Figures

4.1	Diversity and coherence weights for FLDC model in Example 4.1.2. The dotted line divides the clusters.	13
4.2	FFLDC sample model for Example 4.2.2.	15
5.1	Learned model for Example 4.1.2. The white line divides the clusters described in the example.	20
5.2	Learned model for Example 4.2.2.	20
5.3	Objective function for various values of $\eta_0$ without using AdaGrad.	21
5.4	Objective function during learning for various values of $\eta_0$ with AdaGrad.	22
5.5	Comparison of learning performance with and without AdaGrad.	23
5.6	Total variation distance for different values of $v$ .	24
5.7	Total variation distance for different number of epochs.	24
6.1	Bounding box used for filtering photos taken in the Zürich area.	26
6.2	Feature map with the photos from the Zürich dataset taken in 2015.	27
6.3	Zoomed-in maps displaying popular locations in Zürich.	27
6.4	Histogram of photos taken per user.	28
6.5	Histogram of the number of photos per cluster. The x-axis is cut off at 500 photos because the distribution has a heavy tail.	29
6.6	Top 10 clusters in Zürich according to the photo count.	30
6.7	Top 100 clusters in Zürich according to the photo count.	31
6.8	Histogram of the lengths of the paths for the small dataset.	32
6.9	Histogram of the lengths of the paths for the large dataset.	32
7.1	Accuracy of log-modular models for different dataset configurations.	38
7.2	Accuracy of baseline models for the small dataset.	39
7.3	Accuracy for various values of $L$ . The dotted line shows the mean accuracy of the log-modular model.	39
7.4	Total variation distance for FLDC models on the small dataset.	41
7.5	Log-likelihood relative improvement (LLRI) for FLDC models on the small dataset.	41
7.6	Accuracy of FLDC models with varying $L$ and $K$ values.	42
7.7	Comparison of baseline models to the best FLDC model for the large dataset.	42

# List of Tables

3.1	FLID probability distribution for the scenario in Example 3.3.1. . . . .	7
4.1	Probability distribution for Example 4.1.2. . . . .	12
4.2	Synthetic data for Example 4.2.2. . . . .	15
4.3	Synthetic data for Example 4.2.2 after adding a new item. . . . .	16
5.1	Learned distribution for Example 4.2.2 . . . . .	19
6.1	Top 10 clusters for Zürich. . . . .	31
6.2	Path dataset statistics for Zürich. . . . .	31
7.1	Most frequent paths in the small Zürich dataset. . . . .	40



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):

Probabilistic Modeling of City-scale Image Collections

**Authored by** (in block letters):

*For papers written by groups the names of all authors are required.*

**Name(s):**

Ballesteros Villamizar

**First name(s):**

Diego Alfonso

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**

Zürich, 19.04.2016

**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.*

