TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Nachrichtentechnik

Applications of Information Theory and Factor Graphs for Machine Learning

Rana Ali Amjad

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. Klaus Diepold
Prüfer der Dissertation:	1. Prof. Dr. sc. techn. Gerhard Kramer 2. Prof. Haim Permuter, Ph.D.

Die Dissertation wurde am 27.03.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 17.06.2019 angenommen.

Acknowledgements

I would like to start by thanking Professor Gerhard Kramer for providing me with the excellent opportunity to pursue my doctoral degree at the Institute for Communications Engineering (LNT). He granted me the freedom to explore different research topics to find my own path. This allowed me to learn so much, not only in terms of how to conduct research but also in terms of many other professional traits. I have also learned from him how to approach a research problem more systematically while keeping both theoretical and practical aspects in mind.

Next, I want to thank my key collaborator, Dr. Bernhard Geiger, with whom I worked on a major portion of this thesis. Conducting research in close collaboration with him was not only very productive but also an absolute joy. I also thoroughly enjoyed my two visits to Graz to work with him. I would like to thank Dr. Georg Böcherer, with whom I started my research endeavours in information theory and reliable communication. Besides, working with some of my students was also a fruitful and fun venture, especially with Clemens Blöchl and Rayyan Khan.

The time spent at LNT was made much more joyful due to the great colleagues that I had. I always enjoyed the fun and insightful conversations that I had with many of them at lunch or during coffee breaks. My time at LNT has helped me find some great friends including Lars, Patrick and Andrei. I also want to thank my friends outside LNT who have been like a family to me in Munich. They always supported me when I needed to be cheered up and made my time in Munich so memorable. I also enjoyed my numerous trips throughout Europe with them that always helped me recharge and come back with new inspiration for research. I also want to thank the non-scientific staff at LNT for supporting me with all sorts of administrative tasks.

The support and love of my family is one of the key drivers that helped me in pursuing this path comfortably. My father's sagacious advice has always helped me in figuring out the way ahead. My mother's support and love has always helped me to get motivated again whenever I was feeling down. My brothers and their families have always made my short holidays in United Kingdom an absolute pleasure.

Finally, and most importantly, I would like to thank Allah. Without Allah's guidance I would never be able to achieve this goal.

München, March 2019 Rana Ali Amjad

Contents

Zusammenfassung

Diese Arbeit untersucht drei verschiedene Anwendungen von Informationstheorie und graphischen Modellen im Bereich des maschinelles Lernen. Im ersten Teil entwickeln wir allgemeine informationstheoretisch Kostenfunktion für Markov Aggregation. Dann zeigen wir, wie man diese Kostenfunktion für Co-Clustering anwenden kann. Wir schalgen auch eine heuristische Optimierung mit niedriger Komplexität vor. Die Stärken und die Schwächen dieser Kostenfunktion werden in unterschiedlichen Szenarien und für unterschiedliche Parameterwerte auf synthetischen und realen Datensätzen illustriert.

Im zweiten Teil benutzen wir Factor Graphen und den Max-Product Algorithmus als mathematische Werkzeuge, um einen neuen Clustering-Algorithmus, Extended Affinity Propagation zu entwickeln. Extended Affinity Propagation ist sowohl in der Lage, globale Struckturen zu entdecken, als auch wertvolle lokale Information über individuelle Cluster herauszufinden. Wir beleuchten diese Charakteristika von Extended Affinity Propagation sowohl für synthetische als auch für reale Datensätze.

Der Schwerpunkt des dritten Teils ist eine genaue Untersuchung eines neuen Ansatzes: Trainieren der tiefer neuronaler Netzen mit dem Information Bottleneck Prinzip. Wir zeigen zwei wichtige Probleme auf, die entstehen, wenn man das Information Bottleneck Functional für diesen Zweck verwendet: Berechenbarkeit/Optimierbarkeit und das Versagen die gewünschte Eigenschaften anzutrainieren. Wir schlagen verschiedene Methoden vor, um diese Probleme zu verhindern. Wir diskutieren kurz, welche Auswirkungen unsere Resultate auf die Analyse von Feed-Forward tiefen neuronalen Netzen mit dem Information Bottleneck Prinzip haben.

Abstract

This thesis explores three different applications of tools from information theory and graphical models to problems in machine learning. In the first part we develop a generalized information-theoretic cost function for Markov aggregation. We then show how to transform this parameterized cost function for application to co-clustering. We also propose a low complexity optimization heuristic for the cost function. The strengths and weaknesses of the generalized cost function in different scenarios and for different parameter values are illustrated using synthetic datasets and real world datasets.

In the second part we employ the mathematical framework of factor graphs and Max-Product algorithm to propose a new clustering algorithm, Extended Affinity Propagation. Extended Affinity Propagation is not only effective in discovering the global structure in data but also provides useful local information about individual clusters. We highlight these features of Extended Affinity Propagation for both synthetic datasets and real world datasets.

The third part of this thesis focuses on a deeper investigation of a recently proposed idea to train deep neural networks using the Information Bottleneck principle. We highlight two major issues when using the Information Bottleneck functional for training feed-forward deep neural networks: computability/optimizability and failure to instill desired properties. We propose various ways to repair the Information Bottleneck functional to remedy these issues. We also briefly highlight the impact of our investigation on the analysis of feed-forward deep neural networks using the Information Bottleneck principle.

1

Introduction

Machine learning techniques have enjoyed significant attention in recent years. They are being used to replace classical signal processing methods in various supervised learning scenarios, such as classification and regression tasks. This is especially true for the problems where the input signal comes from a natural source, e.g., natural images or text, in contrast to the problems where the input signal may have artificially imposed structure, e.g., channel codes. Besides, unsupervised learning techniques are now being widely used for exploratory data analysis. Exploratory data analysis can, for example, be used to analyze large amounts of customer data to assist in making business decisions, or it can serve as a preprocessing step before applying supervised learning techniques.

Information theory and graphical models have played a key role in the development of telecommunication technologies. They have provided the right tools to suitably formulate and analyze problems to gain deeper understanding. They have also given guidelines for how to develop practical algorithms and systems, satisfying different requirements such as performance guarantees and complexity constraints. Inspired by this success for communication problems, researchers are investigating whether these tools can be used to gain deeper understanding and insights into machine learning problems and algorithms. This thesis contributes to this direction. We consider three different problems.

The first task is concerned with information-theoretic Markov aggregation and coclustering. Markov aggregation is the process of approximating a Markov chain over a large state space with a Markov chain over a much smaller state space. Markov aggregation finds its applications in, e.g., computation chemistry [1], natural language processing, and the simulation and control of large systems. Information-theoretic cost functions motivated by different properties, e.g., predictability and lumpability, were introduced for deterministic Markov aggregation in [2–5]. In Chap. 2, we start by defining the two operational goals that we want to achieve for Markov aggregation: Markovity and preserving temporal dependence. We then devise a parameterized information-theoretic cost function to achieve these operational goals. We discuss how this new cost function, for different val-

ues of a parameter β, subsumes the cost functions for Markov aggregation proposed in [2] and [4] as well as the Information Bottleneck problem, hence giving rise to a generalized information-theoretic Markov aggregation framework. We also propose an optimization heuristic and verify our insights into the cost function by applying it to synthetic Markov chains and to a natural language processing example. Parts of the work in Chap. 2 are published in [6].

In Chap. 3 we transform the Markov aggregation framework from Chap. 2 and apply it to the co-clustering problem. The parameterized cost function obtained this way includes various key information-theoretic cost functions for co-clustering as special cases, in particular the cost functions in [7–11]. This allows us to study these cost functions jointly using a common framework. We discuss insights gained via this joint framework by applying the new cost function to both synthetic and real world datasets. Parts of the work in Chap. 3 have been published in [12, 13].

The second part of this thesis proposes a new pairwise similarities based clustering algorithm. Cluster analysis has widespread applications in industry as a tool for exploratory data analysis. Due to the wide range of applications, many famous clustering algorithms have been proposed over the past decades [14–16]. These techniques are, however, successful either in recognizing the global structure of data, i.e., meaningful clusters, or in providing a useful local view, e.g., local representatives, but not both. In Chap. 4 we use Affinity Propagation [16] as our starting point. We develop a new algorithm, Extended Affinity Propagation, by exploiting the underlying framework of factor graphs and message passing algorithms. We discuss how this algorithm is not only effective in recognizing global structure but also provides useful local information about individual clusters. We illustrate these features on synthetic and real world datasets. Parts of the work in Chap. 4 are published in [17].

In the last part of this thesis, we investigate the use of the information bottleneck principle to train feed-forward deep neural networks for classification tasks. Deep neural networks attained excellent performance for various complex classification tasks such as image segmentation and text classification [18]. The authors of [19] recently proposed using the Information Bottleneck functional as a cost function for training, where the information-theoretic compression term should enforce regularization. In Chap. 5 we point out two problems with Information Bottleneck functional as a cost function for training deep neural networks. The first issue relates to the computability and optimizability of the cost function whereas the second issue relates to a failure to instill desirable properties in a deep neural network based classifier. We discuss various remedies that, at least partly, alleviate these issues. We relate our findings to representation learning and regularization of latent representations, a domain of future research that we believe holds great promise. In [20], the authors discuss how the Information Bottleneck functional can be used for experimental analysis of deep neural networks. We briefly review how the discussion in this chapter sheds new light in this respect. Most of the work in Chap. 5 has been published in [21].

Part I.

Markov Aggregation and Co-Clustering

2

Information-Theoretic Cost Functions for Markov Aggregation

In this chapter we develop an information-theoretic framework for Markov aggregation.

Markov aggregation is the task of representing a Markov chain with a large alphabet by a Markov chain with a smaller alphabet, thus reducing model complexity while at the same time retaining the computationally and analytically desirable Markov property. Such a model reduction is necessary if the original Markov chain is defined on an alphabet that is too large to admit simulation, estimating model parameters from data, or control (in the case of Markov decision processes). These situations occur often in computational chemistry (where aggregation is called coarse-graining, e.g., [1]), natural language processing, and the simulation and control of large systems (giving rise to the notion of bisimulation, e.g., [22]). Information-theoretic cost functions were proposed for Markov aggregation in [2–5]. Specifically, the authors of [2] proposed a cost function linked to the *predictability* of the aggregated Markov chain. Such an approach is justified if the original model is *nearly completely decomposable*, i.e., if there is a partition of the alphabet such that transitions within each element of the partition occur quickly and randomly, while transitions between elements of the partition occur only rarely. Building on this work, the authors of [4] proposed a cost function linked to *lumpability*, i.e., to the phenomenon where a function of a Markov chain is Markov. Such an approach is justified whenever there are groups of states with similar probabilistic properties (in a well-defined sense). Both [2] and [4] focus on deterministic aggregations, i.e., every state of the original alphabet is mapped to exactly one state of the reduced alphabet. Moreover, the authors of both references arrive at their cost functions by lifting the aggregated Markov chain to the original alphabet. The authors of [3] present an information-theoretic cost function for stochastic aggregations, but they do not justify their choice by an operational characterization (such as predictability or lumpability). Instead, they arrive at their cost function via the composite of the original and the aggregated Markov chain.

In this chapter, we extend the works [2–5] as follows: We follow a two-step approach to Markov aggregation (Sec. 2.3): Observing the original Markov chain through a (stochastic or deterministic) channel, and then approximating this (typically non-Markov) projected process via a Markov chain. This approach has already been taken by [4], albeit only for deterministic aggregations. Using this two-step approach, we propose a parameterized, information-theoretic cost function for Markov aggregation (Sec. 2.4). We arrive at this cost function neither via lifting nor via the composite model, but via requiring specific operational qualities of the projected process: It should be close to a Markov chain and it should retain the temporal dependence structure of the original Markov chain. We show that our cost function contains the cost functions of [2–5] as special cases (Sec. 2.4.4). This allows to jointly investigate these previously proposed cost functions for better understanding. We then propose a simple, low-complexity heuristic to minimize our generalized cost function for deterministic aggregations along with an effective initialization procedure based on graduated optimization (Sec. 2.5). The initialization procedure enables us to avoid bad local optima for a range of parameter values. We illustrate the proposed cost function and heuristic for various examples (Sec. 2.6). Specifically, we investigate the aggregation of synthetic quasi-lumpable and nearly completely decomposable Markov chains, and we look at a toy example from natural language processing. In this work we do not deal with the important issue of choosing/determining the appropriate number of aggregated states. We assume that we know the desired number of aggregated states as this allows us to focus purely on the influence of the cost function independent of the heuristics used to determine the number of aggregated states.

The research presented in this chapter was conducted in close collaboration with Dr. Bernhard Geiger and Clemens Blöchl.

2.1. Preliminaries

In this chapter we deal exclusively with first-order stationary, irreducible and aperiodic Markov chains defined over finite alphabets unless otherwise stated, which we will refer to simply as Markov chains. Lemma. 2.1 relates Markovity of a stochastic process to it's entropy rate, where entropy rate of a process is denoted by $\bar{H}(\cdot)$.

Lemma 2.1 ([23, Prop. 3])**.** Let $\mathbf{Z}$ be a stationary stochastic process. Then $\mathbf{Z}$ is a Markov iff $\bar{H}(\mathbf{Z}) = H(Z_2|Z_1)$.

$\bar{H}(\cdot)$ denotes the entropy rate of a process. If $\mathbf{Z}$ is a stationary process on $\mathcal{Z}$ (not necessarily Markov), then one can approximate this process by a Markov chain $\tilde{\mathbf{Z}} \sim \text{Mar}(\mathcal{Z}, \mathbb{P})$:

Lemma 2.2 ([24, Cor. 10.4])**.** Let $\mathbf{Z}$ be a stationary process on $\mathcal{Z}$ and let $\mathbb{P}$ specify the transition probability matrix from Z_{n-1} to Z_n. Define

$$\mathbb{P}^* \triangleq \arg\min_{\mathbb{P}'} \bar{D}(\mathbf{Z}||\mathbf{Z}') \tag{2.1a}$$

where the optimization is over Markov chains $\mathbf{Z}' \sim \text{Mar}(\mathcal{Z}, \mathbb{P}')$. It turns out that

$$\mathbb{P}^* = \mathbb{P} \tag{2.1b}$$

Furthermore, for $\tilde{\mathbf{Z}} \sim \text{Mar}(\mathcal{Z}, \mathbb{P}^*)$, we have

$$\bar{D}(\mathbf{Z}||\tilde{\mathbf{Z}}) = H(Z_2|Z_1) - \bar{H}(\mathbf{Z}) \tag{2.1c}$$

where $\bar{D}(\cdot||\cdot)$ denotes the Kullback-Leibler Divergence Rate (KLDR) between two processes.

By Lemma 2.1 we know that right-hand side of (2.1c) is 0 iff $\mathbf{Z}$ is Markov. Hence, one can view $\bar{D}(\mathbf{Z}||\tilde{\mathbf{Z}})$ as a measure of how close a process $\mathbf{Z}$ is to a Markov chain.

2.2. Related Work

Information-theoretic cost functions for Markov aggregation had been proposed in, e.g., [2–4]. More generally, aggregations of dynamical systems that are not necessarily Markov were discussed in [25]. In contrast to [2–4], the cost functions proposed by [25] are task-specific in the sense that they aim to predict an observation based on Z_t from the aggregated process.

The topic of *lumpability* is closely related to Markov aggregation, i.e., the question whether a non-injective function of a Markov chain is Markov. Initial research in this area was performed by Kemeny and Snell (strong and weak lumpability, [26, Sec. 6.3-6.4]), Rosenblatt (lumpability of continuous-valued Markov processes [27]), and Buchholz (exact lumpability [28]). Gurvits and Ledoux discovered linear-algebraic conditions on

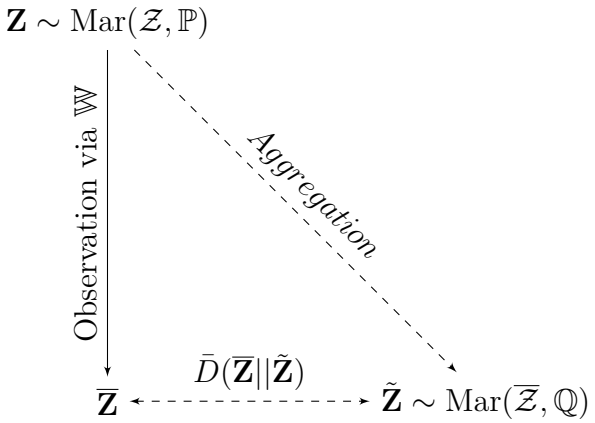


Figure 2.1.: Illustration of the aggregation problem: A stationary first-order Markov chain $\mathbf{Z}$ is given. We are interested in finding a conditional distribution $P_{\overline{Z}|Z}$ and an aggregation of $\mathbf{Z}$, i.e., a Markov chain $\tilde{\mathbf{Z}}$ on $\overline{\mathcal{Z}}$. The conditional distribution $P_{\overline{Z}|Z}$ defines a stationary process $\overline{\mathbf{Z}}$, a noisy observation of $\mathbf{Z}$. $\overline{\mathbf{Z}}$ might not be Markov of any order, but can be approximated by a Markov chain $\tilde{\mathbf{Z}}$.

the transition probability matrix of $\{Z_t\colon t = 1, 2, \dots\}$ and the aggregation function for weak and strong lumpability [29]. An equivalent characterization of strong lumpability in information-theoretic terms has been presented by Geiger and Temmel and Pfante et al. in [23] and [30], respectively. This information-theoretic characterization was used in a cost function for Markov aggregation in [4].

2.3. Problem Statement

Definition 2.1 (Markov Aggregation Problem)**.** Consider a Markov process $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$, a set $\overline{\mathcal{Z}}$, and a cost function $\bar{\mathrm{C}}(\cdot, \cdot)$. The Markov aggregation problem concerns finding a minimizer of

$$\min_{\tilde{\mathbf{Z}}} \ \bar{\mathrm{C}}(\mathbf{Z}, \tilde{\mathbf{Z}}) \tag{2.2}$$

where the optimization is over Markov chains $\tilde{\mathbf{Z}}$ on $\overline{\mathcal{Z}}$.

We address the Markov aggregation problem using the following two-step approach depicted in Fig 2.1.

($S1$) The first step is to use a (possibly stochastic) mapping from $\mathcal{Z}$ to $\overline{\mathcal{Z}}$. Applying this mapping to $\mathbf{Z}$ on a symbol by symbol basis leads to a stationary process $\overline{\mathbf{Z}}$ (i.e., $\overline{Z}_i = P_{\overline{Z}|Z}(Z_i)$) which may not be Markov (in fact, $\overline{\mathbf{Z}}$ is a hidden Markov process). We call $\overline{\mathbf{Z}}$ the projected process.

($S2$) In the second step we look for the optimal approximation $\tilde{\mathbf{Z}}$ of $\overline{\mathbf{Z}}$ in the sense of Lemma 2.2. $\tilde{\mathbf{Z}}$ is the final aggregated Markov chain.

This two-step approach is a popular method of Markov aggregation and has been employed in various works including [2, 4, 5]. In these references, the mapping in the first step was restricted to be deterministic whereas in this work we allow it to be stochastic. In other words, while these references looked for a partition of $\mathcal{Z}$ induced by a function $g\colon \mathcal{Z} \to \overline{\mathcal{Z}}$, in this work we permit stochastic mappings induced by a conditional distribution $P_{\overline{Z}|Z}$. We represent $P_{\overline{Z}|Z}$ as a row stochastic matrix $\mathbb{W}$, where $W(z, \overline{z}) = P_{\overline{Z}|Z}(\overline{z}|z)$.

With this notation, the following corollary to Lemma 2.2 solves the second of the two steps in our approach, i.e., it characterizes the optimal approximation $\tilde{\mathbf{Z}}$ of the projected process $\overline{\mathbf{Z}}$ for a specific $\mathbb{W}$:

Corollary 2.3. Let $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$ and let $\mathbb{W}$ denote a conditional distribution from $\mathcal{Z}$ to $\overline{\mathcal{Z}}$. Let $\overline{\mathbf{Z}}$ be the hidden Markov process obtained by observing $\mathbf{Z}$ through $\mathbb{W}$, and let $\tilde{\mathbf{Z}} \sim \mathrm{Mar}(\overline{\mathcal{Z}}, \mathbb{Q})$ be its best Markov approximation in the sense of minimizing $\bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}})$ (cf. Lemma 2.2). Then,

$$\mathbb{Q} = \mathbb{U}\mathbb{P}\mathbb{W} \tag{2.3}$$

where $\mathbb{U} := \mathrm{diag}(\boldsymbol{\nu})^{-1}\mathbb{W}^T \mathrm{diag}(\boldsymbol{\mu})$ with $\boldsymbol{\nu}^T := \boldsymbol{\mu}^T\mathbb{W}$ being the marginal distribution of $\overline{Z}_k$.

Note that this corollary extends [4, Lem. 3] from deterministic to stochastic mappings. With the second step solved, the two-step approach to the optimization problem stated in Definition 2.1 boils down to optimizing over the mapping $\mathbb{W}$. We can thus restate the Markov aggregation problem as follows:

Definition 2.2 (Markov Aggregation Problem Restated)**.** Consider a Markov process $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$, a set $\overline{\mathcal{Z}}$, and a cost function $\bar{\mathrm{C}}(\cdot, \cdot)$. Let

$$\mathrm{C}(\mathbf{Z}, \mathbb{W}) = \bar{\mathrm{C}}(\mathbf{Z}, \tilde{\mathbf{Z}}) \tag{2.4}$$

where $\tilde{\mathbf{Z}}$ is the best Markov approximation of $\overline{\mathbf{Z}}$ in the sense of Lemma 2.2. The Markov aggregation problem using the two-step approach concerns finding a minimizer of

$$\min_{\mathbb{W}} \ \mathrm{C}(\mathbf{Z}, \mathbb{W}) \tag{2.5}$$

where the optimization is over stochastic mappings from $\mathcal{Z}$ to $\overline{\mathcal{Z}}$. If the optimization is restricted over deterministic mappings g, we abuse notation and write $\mathrm{C}(\mathbf{Z}, g)$ for the cost.

Note that Definition 2.1 and Definition 2.2 are not equivalent in general, i.e., the optimal aggregated chain $\tilde{\mathbf{Z}}$ obtained by solving (2.2) is not the same as the optimal aggregated chain $\tilde{\mathbf{Z}}$ obtained by solving (2.5). The two formulations only become equivalent when we restrict the optimization in (2.2) to aggregated Markov chains which can be obtained as a result of the aforementioned two-step approach.

2.4. Cost Function

The next step in defining the problem is to specify the cost function $\mathrm{C}(\mathbf{Z}, \mathbb{W})$. Our design of the cost function is guided by the following two operational goals:

($O1$) We want $\mathbb{W}$ to make the projected process $\overline{\mathbf{Z}}$ as Markov as possible. The closer $\overline{\mathbf{Z}}$ is to a Markov process, the smaller the approximation loss.

($O2$) The temporal dependence structure of a Markov process is one of the most important characteristic of a Markov process. As a second objective, we want to preserve the temporal dependence structure of the original Markov process $\mathbf{Z}$ after aggregation, i.e., $\tilde{\overline{\mathbf{Z}}}$ should exhibit similar coarse-grained temporal structure as observed in $\mathbf{Z}$.

In the following, we will first explore suitable cost functions for the two goals separately. Subsequently we will combine them to obtain our final cost function, from which we can trade-off between the two goals as we desire.

2.4.1. Markov Aggregation via Lumpability

To achieve the first goal ($O1$), as a consequence of Lemma 2.2, we can use the following cost function

$$\bar{\mathrm{C}}(\mathbf{Z}, \tilde{\mathbf{Z}}) = \bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}}). \tag{2.6}$$

From Lemma 2.2 we know that $\min_{\mathbb{W}} \bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}}) = 0$ if the projected process $\overline{\mathbf{Z}}$ is already a Markov process. $\bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}})$ can be thought of as an information-theoretic metric that specifies how close $\overline{\mathbf{Z}}$ is to being a Markov chain. Closeness between $\overline{\mathbf{Z}}$ and its approximation $\tilde{\mathbf{Z}}$ implies that data obtained by simulating the aggregated model $\tilde{\mathbf{Z}}$ differs not too much from data obtained by simulating the original model in conjunction with the stochastic mapping, i.e., data obtained from $\overline{\mathbf{Z}}$.

The cost function (2.6) suffers from two shortcomings. First, since (2.6) focuses only on satisfying $O1$ (i.e., the loss approximation loss incurred in the second step of our approach), it lead to solutions that are trivial in the sense of $O2$: If $\mathbb{W}$ makes the conditional distribution independent of the conditioning event, i.e., $P_{\overline{Z}|Z} = P_{\overline{Z}}$, or $\mathbb{W} = \mathbf{1}\boldsymbol{\alpha}^T$ for some probability vector $\boldsymbol{\alpha}$, then $\overline{\mathbf{Z}}$ is independent and identically distributed (i.i.d.) and hence Markov. Indeed, in this case $\bar{H}(\overline{\mathbf{Z}}) = H(\overline{Z}_2|\overline{Z}_1) = H(\overline{Z})$, and $\bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}}) = 0$. Hence the cost function needs to be appropriately regularized to use it for Markov aggregation. The second shortcoming is that $\bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}})$ requires, by (2.1c), the computation of the entropy rate $\bar{H}(\overline{\mathbf{Z}})$ of a hidden Markov process. This problem is inherently difficult [31], and analytic expressions do not exist even for simple cases (cf. [32]). In the following, we discuss two relaxations of the Markov aggregation problem for $\bar{\mathrm{C}}(\mathbf{Z}, \tilde{\mathbf{Z}}) = \bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}})$.

The authors of [4] addressed the second shortcoming by relaxing the cost via

$$\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) := H(\overline{Z}_2|\overline{Z}_1) - H(\overline{Z}_2|Z_1) \geq \bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}}). \tag{2.7}$$

This cost does not require computing $\bar{H}(\overline{\mathbf{Z}})$ and is linked to the phenomenon of *lumpability*, namely that a function of a Markov chain has the Markov property [23, Thm. 9]: If $\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) = 0$, then $\overline{\mathbf{Z}}$ is a Markov chain.

We now show that, at least for deterministic mappings, this cost function also has a justification in *approximate probabilistic bisimulations*, or ε-bisimulations. More specifi-

cally, the authors of [33] discussed bisimilarity of Markov processes and showed that two Markov chains are bisimilar if one can be described as a function of the other (see discussion after [33, Def. 5.2]). In other words, if $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$ is a Markov chain, $g\colon \mathcal{Z} \to \overline{\mathcal{Z}}$ a surjective function, and $\overline{\mathbf{Z}} \sim \mathrm{Mar}(\overline{\mathcal{Z}}, \mathbb{Q})$ satisfies $\overline{Z}_k = g(Z_k)$, then $\mathbf{Z}$ and $\overline{\mathbf{Z}}$ are bisimilar. Since this is equivalent to lumpability, bisimilarity is implied by $\mathrm{C}_L(\mathbf{Z}, g) = 0$.

Extending this line of reasoning, we justify the cost function $\mathrm{C}_L(\mathbf{Z}, g)$ in terms of ε-bisimulation of pairs of Markov chains, even if $\mathbf{Z}$ is not lumpable w.r.t. g. To this end, we adapt [34, Def. 4 & 5] for our purposes.

Definition 2.3 (ε-Bisimulation)**.** Consider two finite Markov chains $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$ and $\tilde{\mathbf{Z}} \sim \mathrm{Mar}(\overline{\mathcal{Z}}, \mathbb{Q})$ and assume w.l.o.g. that $\mathcal{Z}$ and $\overline{\mathcal{Z}}$ are disjoint. We say that $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ are *ε-bisimilar* if there exists a relation $\mathcal{R}_\varepsilon \subseteq (\mathcal{Z} \cup \overline{\mathcal{Z}}) \times (\mathcal{Z} \cup \overline{\mathcal{Z}})$ such that for all $x \in \mathcal{Z}$ and $y \in \overline{\mathcal{Z}}$ for which $(\overline{z}, z) \in \mathcal{R}_\varepsilon$, and all $T \subseteq \mathcal{Z} \cup \overline{\mathcal{Z}}$ we have

$$\sum_{x' \in \mathcal{R}_\varepsilon(T) \cap \mathcal{Z}} \mathbb{P}(z, z') \geq \sum_{y' \in T \cap \overline{\mathcal{Z}}} Q_{\overline{z} \to \overline{z}'} - \varepsilon \tag{2.8}$$

where $\mathcal{R}_\varepsilon(T) := \{s_2 \in \mathcal{Z} \cup \overline{\mathcal{Z}} \colon s_1 \in T, (s_1, s_2) \in \mathcal{R}_\varepsilon\}$.

The definitions of ε-bisimulations are typically given for labeled [34, Def. 4 & 5] or controlled [22, Def. 4.4] Markov processes with general alphabets and thus contain more restrictive conditions than our Definition 2.3. Our definition is equivalent if the alphabets are finite and if the set of labels is empty.

Proposition 2.4. Let $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$ and the surjective function $g\colon \mathcal{Z} \to \overline{\mathcal{Z}}$ be given. Let $\mathbb{Q}$ be as in Corollary 2.3, where $\mathbb{W}(z, \overline{z})$ iff $\overline{z} = g(z)$. Let $\tilde{\mathbf{Z}} \sim \mathrm{Mar}(\overline{\mathcal{Z}}, \mathbb{Q})$. Then, $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ are ε-bisimilar with

$$\varepsilon = \sqrt{\frac{\ln(2)\mathrm{C}_L(\mathbf{Z}, g)}{2 \min_{z \in \mathcal{Z}} \mu_z}}. \tag{2.9}$$

Proof: See Appendix A.1. ∎

Despite this justification, the cost function $\mathrm{C}_L(\mathbf{Z}, \mathbb{W})$ is mainly of theoretical interest. The reason is that $\mathrm{C}_L(\mathbf{Z}, \mathbb{W})$ inherits the trivial solutions of $\bar{D}(\overline{\mathbf{Z}} || \tilde{\mathbf{Z}})$ in terms of $O2$, for example $\mathbb{W} = \mathbf{1}\boldsymbol{\alpha}^T$ leads to $\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) = 0$, regardless of $\boldsymbol{\alpha}$ and $\mathbb{P}$. Even restricting $\mathbb{W}$ to be a deterministic partition, as considered in [4], does not solve this problem: the combinatorial search over all partitions may have its global optimum at a partition that makes $\overline{\mathbf{Z}}$ close to an i.i.d. process. Indeed, if the cardinality of $\overline{\mathcal{Z}}$ is not constrained (or if g is not required to be surjective), then the constant function g yields $\mathrm{C}_L(\mathbf{Z}, g) = 0$.

2.4.2. Markov Aggregation via Predictability

Temporal dependence in a process $\mathbf{Z}$ can be captured by redundancy rate $\bar{R}(\mathbf{Z})$, which for a Markov chain $\mathbf{Z}$ equals $\bar{R}(\mathbf{Z}) = I(Z_1; Z_2)$. Since $\overline{Z}_n$ is a (stochastic) mapping of Z_n, any two samples $\overline{Z}_\ell$ and $\overline{Z}_n$, $\ell < n$ are conditionally independent given the sequence Z_ℓ^n. Therefore,

the temporal dependence in $\overline{\mathbf{Z}}$ can only originate from the temporal dependence in $\mathbf{Z}$. Preserving the temporal dependence structure of $\mathbf{Z}$ is thus captured well by maximizing the redundancy rate of $\overline{\mathbf{Z}}$, i.e., by choosing a minimizer of

$$\min_{\mathbb{W}} \ \bar{R}(\mathbf{Z}) - \bar{R}(\overline{\mathbf{Z}}). \tag{2.10}$$

However, the cost function suffers from two shortcomings. First, we need to compute $\bar{H}(\overline{\mathbf{Z}})$ to evaluate the cost function. Secondly, while (2.10) is a suitable loss for the first step, it ignores the temporal information loss incurred by the second step in our method. Hence for the overall goal of Markov aggregation it needs to be adapted. We know that both $\tilde{\mathbf{Z}}$ and $\overline{\mathbf{Z}}$ share the conditional distribution $\mathbb{Q}$ that defines the dependence of the current symbol and the previous symbol for both the projected process and the aggregated process. For $\overline{\mathbf{Z}}$, the current symbol may further depend on the past beyond the last symbol, but $\tilde{\mathbf{Z}}$ ignores the rest of the temporal information contained in $\overline{\mathbf{Z}}$. Hence the following cost function captures the temporal dependence in $\overline{\mathbf{Z}}$, the aggregated process.

$$\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) := I(Z_1; Z_2) - I(\overline{Z}_1; \overline{Z}_2). \tag{2.11}$$

We know by the data processing inequality:

$$\bar{R}(\mathbf{Z}) - \bar{R}(\overline{\mathbf{Z}}) \leq \mathrm{C}_P(\mathbf{Z}, \mathbb{W}). \tag{2.12}$$

Definition (2.11) doesn't require computation of $\bar{H}(\overline{\mathbf{Z}})$ and hence it resolves both shortcomings of (2.10). This cost function has been proposed earlier in [2] for deterministic partitions. Since $\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$ aims to preserve temporal dependence it does not suffer from the same trivial solutions as $\mathrm{C}_L(\mathbf{Z}, \mathbb{W})$ and $\bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}})$: A constant function g or a soft partition $\mathbb{W} = \mathbf{1}\boldsymbol{\alpha}^T$ render $\overline{Z}_1$ and $\overline{Z}_2$ independent, hence the cost is maximized at $\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) = I(Z_1; Z_2)$. Unfortunately, as was shown in [4, Thm. 1], we have $\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) \geq \mathrm{C}_L(\mathbf{Z}, \mathbb{W})$, i.e., (2.11) does not capture the Markovity of $\overline{\mathbf{Z}}$ as well as the relaxation proposed by [4].

The goal of preserving temporal dependence information is justified in scenarios in which $\mathbf{Z}$ is quasi-static, i.e., runs on different time scales: The process $\mathbf{Z}$ moves quickly and randomly within a group of states, but moves only slowly from one group of states to another. Since all other information contained in $\mathbf{Z}$ is not necessarily preserved by minimizing $\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$, this cost function can also lead to undesired solutions. For example, if $\mathbf{Z}$ is i.i.d. and does not contain any temporal dependence structure, then $I(Z_1; Z_2) = 0$ and $\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) = 0$ for every mapping $\mathbb{W}$.

A third objective for optimization may be worth mentioning. The information contained in $\mathbf{Z}$ splits into a part describing its temporal dependence structure (measured by its redundancy rate $I(Z_1; Z_2)$) and a part describing the new information generated in each time step (measured by its entropy rate $\bar{H}(\mathbf{Z})$). Indeed, we have

$$H(X) = \bar{H}(\mathbf{Z}) + I(Z_1; Z_2). \tag{2.13}$$

While in this work we focus on preserving Markovity via $\mathrm{C}_L(\mathbf{Z}, \mathbb{W})$ and the temporal dependence structure via $\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$, the authors of [23] investigated conditions such that the newly generated information (measured by $\bar{H}(\mathbf{Z})$) is preserved. Developing a Markov aggregation framework that balances three different goals – Markovity, temporal dependence, generated information – is the object of future work.

2.4.3. Regularized Markov Aggregation Cost Function

In order to achieve both goals mentioned earlier we can combine the cost functions above to get the following regularized Markov aggregation cost function:

$$\min_{\mathbb{W}} \underbrace{(1-\beta)\bar{D}(\overline{\mathbf{Z}}||\tilde{\mathbf{Z}}) + \beta(\bar{R}(\mathbf{Z}) - \bar{R}(\overline{\mathbf{Z}}))}_{=:\delta_\beta(\mathbf{Z},\mathbb{W})} \tag{2.14}$$

where $0 \leq \beta \leq 1$. We know that KLDR is a non-negative quantity and the data processing inequality ensures that $\bar{R}(\mathbf{Z}) \geq \bar{R}(\overline{\mathbf{Z}})$, hence the cost (2.14) is non-negative. We moreover have the following property

Lemma 2.5. $\delta_\beta(\mathbf{Z}, \mathbb{W})$ is non-decreasing in β.

Proof: See Appendix A.2. ∎

Due to the aforementioned issues we can replace the two components of the cost function with their upper bounds to get the following cost function

$$\delta_\beta(\mathbf{Z}, \mathbb{W}) \leq (1-\beta)\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) + \beta\mathrm{C}_P(\mathbf{Z}, \mathbb{W}). \tag{2.15}$$

We then propose a final change to the cost function as follows

$$\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) := (1-2\beta)\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) + \beta\mathrm{C}_P(\mathbf{Z}, \mathbb{W}). \tag{2.16}$$

where again $\beta \in [0, 1]$. One can justify going from (2.15) to (2.16) by noticing that for every $0 \leq \beta \leq 1$ for (2.15), one can find a $0 \leq \beta \leq 0.5$ for (2.16) such that the two optimization problems are equivalent, i.e., they have the same optimizer $\mathbb{W}$. Furthermore, for $\beta = 1$, the cost function in (2.16) corresponds to information bottleneck problem, a case that is not covered by (2.15). Hence, not only is C_β a strict generalization of δ_β but also has the information bottleneck problem as an interesting extreme case. C_β can be expressed purely in terms of mutual information quantities as follows:

$$\mathcal{L}_\beta(\zeta) := \beta I(Z_1; Z_2) + (1-2\beta)I(Z_1; \overline{Z}_2) - (1-\beta)I(\overline{Z}_1; \overline{Z}_2). \tag{2.17}$$

In the following we summarize some of the properties of C_β.

Lemma 2.6. For C_β and $0 \leq \beta \leq 1$ we have:

1. $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) \geq 0$

2. $\delta_{0.5}(\mathbf{Z}, \mathbb{W}) = \mathrm{C}_{0.5}(\mathbf{Z}, \mathbb{W}) = \frac{1}{2}\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$
3. $\mathrm{C}_1(\mathbf{Z}, \mathbb{W}) = \mathrm{C}_{\mathrm{IB}}(\mathbf{Z}, \mathbb{W}) := I(Z_1; Z_2|\overline{Z}_2)$
4. For $\beta \leq \frac{1}{2}$, $\beta\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) \leq \delta_\beta(\mathbf{Z}, \mathbb{W}) \leq \mathrm{C}_\beta(\mathbf{Z}, \mathbb{W})$
5. For $\beta \geq \frac{1}{2}$, $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) \leq \delta_\beta(\mathbf{Z}, \mathbb{W}) \leq \beta\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$
6. If $\mathbf{Z}$ is reversible, then $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W})$ is non-decreasing in β

Proof: See Appendix A.3. ■

2.4.4. Special Cases of $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W})$

We now show that specific settings of β lead to cost functions that have been proposed previously in the literature. We list these approaches together with the algorithms that were proposed to solve the respective Markov aggregation problem.

▷ For $\beta = \frac{1}{2}$, optimizing (2.16) is equivalent to optimizing $\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$. The authors of [2] proposed this cost function for deterministic aggregations, i.e., they proposed optimizing $\mathrm{C}_P(\mathbf{Z}, g)$. In [13] we show that this restriction to deterministic aggregations is suboptimal: For example consider the following reversible, three-state Markov chain for which the optimal aggregation to $|\overline{\mathcal{Z}}| = 2$ states is stochastic.

Example 2.1. Let $N = 3$ and

$$\mathbb{P} = \begin{bmatrix} 0.1 & 0.1 & 0.175 \\ 0.1 & 0.15 & 0.075 \\ 0.175 & 0.075 & 0.05 \end{bmatrix}. \tag{2.18}$$

We wish to cluster Z_1 and Z_2 pairwise to $M = 2$ clusters, hence we are looking for a $\mathbb{W} \in \mathcal{M}_{3\times 2}$ that maximizes the mutual information between $\overline{Z}_1$ and $\overline{Z}_2$. We parametrize $\mathbb{W}$ as

$$\mathbb{W} = \begin{bmatrix} p & 1-p \\ q & 1-q \\ r & 1-r \end{bmatrix} \tag{2.19}$$

and, in simulations, sweep all three parameters p, q, and r between 0 and 1 in steps of 0.025. The parameters maximizing the mutual information are $p = 1$, $q = 0.65$, and $r = 0$, giving a mutual information of 0.0316. In comparison, the mutual information obtained by the three (nontrivial) deterministic pairwise clusterings evaluate to 0.0281 for $p = q = 1 - r = 1$, 0.0288 for $p = 1 - q = r = 0$, and 0.0222 for $p = 1 - q = 1 - r = 1$.

In [13] we also discuss scenarios where we can show or have experimental evidence that deterministic aggregations are optimal. For the bi-partition problem, i.e., for

$|\overline{\mathcal{Z}}| = 2$, the authors of [2] propose a relaxation to a spectral, i.e., eigenvector-based optimization problem, the solution of which has a computational complexity of $\mathcal{O}(|\mathcal{Z}|^3)$. In general, this relaxation leads to a further loss of optimality, even among the search over all deterministic bi-partitions. For a general $\overline{\mathcal{Z}}$, [2] suggests to solve the problem by repeated bi-partitioning, i.e., splitting sets of states until the desired cardinality is achieved.

▷ For $\beta = 1$, the problem becomes equivalent to maximizing $I(Z_1; \overline{Z}_2)$. This is exactly the information bottleneck problem [35] for a Lagrangian parameter $\gamma \to \infty$:

$$I(Z_2; \overline{Z}_2) - \gamma I(Z_1; \overline{Z}_2). \tag{2.20}$$

Algorithmic approaches to solving this optimization problem are introduced in [36]. Note that in this case the optimal aggregation will be deterministic, a result that was shown in [13, Th. 1] and [37, Lem. 1].

▷ For $\beta = 0$, the authors of [4] relaxed their cost function $\mathrm{C}_0(\mathbf{Z}, g) = \mathrm{C}_L(\mathbf{Z}, g)$ as

$$\begin{aligned} \mathrm{C}_0(\mathbf{Z}, g) &= H(\overline{Z}_2|\overline{Z}_1) - H(\overline{Z}_2|Z_1) = I(\overline{Z}_2; Z_1|\overline{Z}_1) \\ &\leq I(Z_2; Z_1|\overline{Z}_1) = I(Z_2; Z_1) - I(Z_2; \overline{Z}_1) \end{aligned} \tag{2.21}$$

and proposed using the agglomerative information bottleneck method [38] with the roles of Z_1 and Z_2 in (2.20) exchanged to solve this relaxed optimization problem. The method has a computational complexity of $\mathcal{O}(|\mathcal{Z}|^4)$ [36, Sec. 3.4]. While the mapping minimizing $\mathrm{C}_L(\mathbf{Z}, \mathbb{W})$ may be stochastic, the mapping minimizing (2.21) is deterministic; hence, with this relaxation in mind, the restriction to deterministic aggregations made in [4] comes without an additional loss of optimality compared to what is lost in the relaxation.

▷ The authors of [3] proposed minimizing

$$I(Z_1; Z_2) - I(Z_2; \overline{Z}_1) - \gamma H(\overline{Z}_2|Z_1). \tag{2.22}$$

They suggested using a deterministic annealing approach, reducing γ successively until $\gamma = 0$. In the limiting case, the cost function then coincides with (2.21) and the optimal aggregation is again deterministic. Note that, for reversible Markov chains, we have $I(Z_1; \overline{Z}_2) = I(Z_2; \overline{Z}_1)$, hence both (2.21) and (2.22) (for $\gamma = 0$) are equivalent to $\mathrm{C}_1(\mathbf{Z}, \mathbb{W})$. Analyzing [3, Sec. III.B] shows that in each annealing step the quantity

$$D\left(P_{Z_2|Z_1=z} || P_{Z_2|\overline{Z}_1=\overline{z}}\right) := \sum_{z' \in \mathcal{Z}} P_{Z_2|Z_1}(z'|z) \log \frac{P_{Z_2|Z_1}(z'|z)}{P_{Z_2|\overline{Z}_1}(z'|\overline{z})} \tag{2.23}$$

has to be computed for every x and y. Hence, the computational complexity of this approach is $\mathcal{O}(\overline{\mathcal{Z}} \cdot |\mathcal{Z}|^2)$ in each annealing step.

2.5. Optimization Heuristic

In this section we focus on deterministic aggregations. We can thus replace $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W})$ by $\mathrm{C}_\beta(\mathbf{Z}, g)$ for some g: $\mathcal{Z} \to \overline{\mathcal{Z}}$. In general, the computational complexity of finding the deterministic aggregation with minimum $\mathrm{C}_\beta(\mathbf{Z}, g)$ is exponential in $|\mathcal{Z}|$. We now propose a low complexity iterative method to optimize (2.16) over deterministic aggregations for any given β. The method consists of a sequential optimization algorithm (Algorithm 2.1) and an annealing procedure for β (Algorithm 2.2) that prevents getting stuck in local optima. Our algorithm has a computational complexity of $\mathcal{O}(|\overline{\mathcal{Z}}| \cdot |\mathcal{Z}|^2)$ per iteration. Note, however, that the restriction to deterministic aggregation functions comes, at least for some values of β, with a loss of optimality, as also discussed in Sec. 2.4.4 and [13].

2.5.1. Sequential Algorithm (sGITMA)

We briefly illustrate an iteration of Algorithm 2.1: Suppose $z \in \mathcal{Z}$ is mapped to the aggregate state $\overline{z} \in \overline{\mathcal{Z}}$, i.e., $g(z) = \overline{z}$. We remove z from $\overline{z}$. We then assign z to every aggregate state $\overline{z}'$, $\overline{z}' \in \overline{\mathcal{Z}}$, while keeping the rest of the mapping g the same, and evaluate the cost function. Finally, we assign z to the aggregate state that minimized the cost function (breaking ties, if necessary). This procedure is repeated for every $z \in \mathcal{Z}$.

Algorithm 2.1 Sequential Generalized Information-Theoretic Markov Aggregation.

function g = sGITMA($\mathbb{P}$, β, $|\overline{\mathcal{Z}}|$, $\#\mathrm{iter}_{\max}$, optional: initial aggregation function g_{init})
  if g_{init} is empty **then**
    $g \leftarrow$ Random Aggregation Function
  else
    $g \leftarrow g_{\mathrm{init}}$
  end if
  $\#\mathrm{iter} \leftarrow 0$
  while $\#\mathrm{iter} < \#\mathrm{iter}_{\max}$ **do**
    for all elements $z \in \mathcal{Z}$ **do** ▷ *Optimizing g*
      for all aggregate states $\overline{z} \in \overline{\mathcal{Z}}$ **do**
        $g_{\overline{z}}(z') = \begin{cases} g(z') & z' \neq z \\ \overline{z} & z' = z \end{cases}$ ▷ *Assign z to aggregate state $\overline{z}$*
        $C_{g_{\overline{z}}} = \mathrm{C}_\beta(\mathbf{Z}, g_{\overline{z}})$
      end for
      $g = \arg\min_{g_{\overline{z}}} C_{g_{\overline{z}}}$
    end for
    $\#\mathrm{iter} \leftarrow \#\mathrm{iter} + 1$
  end while
end function

One can verify that the cost function is reduced in each step of Algorithm 2.1, as a state is assigned to a different aggregate state only if the cost function is reduced. Hence, the algorithm modifies g in each iteration to reduce the cost until it either reaches the maximum number of iterations or until the cost converges.

Note that the algorithm is random in the sense that it is started with a random aggregation function g. Depending on the specific application, though, a tailored initialization procedure may improve performance.

Finally, it is worth mentioning that for $\beta = 1$ our Algorithm 2.1 is equivalent to the sequential information bottleneck algorithm proposed in [36, Sec. 3.4].

2.5.2. Annealing Procedure for β

Although Algorithm 2.1 is guaranteed to converge (with proper tiebreaking), convergence to a global optimum is not ensured. The algorithm may get stuck in poor local minima. This happens often for small values of β, as our experiments in Section 2.6.2 show. The reason is that, for small β, $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W})$ has many poor local minima and, randomly initialized, the algorithm is more likely to get stuck in one of them. In contrast, our results suggest that for larger values of β the cost function has only few poor local minima, and the algorithm converges to a good local or a global minimum for a significant portion of random initializations.

A solution for small β would thus be to choose an initialization that is close to a "good" local optimum. A simple idea is thus to re-use the function g obtained for a large value of β as initial aggregation for smaller values of β. We thus propose the following annealing algorithm: We initialize $\beta = 1$ to obtain g. Then, in each iteration of the annealing procedure, β is reduced and the aggregation function is updated, starting from the result of the previous iteration. The procedure stops when β reaches the desired value, β_{target}. The β-annealing algorithm is sketched as Algorithm 2.2. As is clear from the description, the β-annealing algorithm closely follows graduated optimization [39]. The results for synthetic datasets with and without β-annealing are discussed in Section 2.6.2, which show that without restarts one keeps getting stuck in bad local optima for small β, while with β-annealing one can avoid them. Our intuition that β-annealing provides good initialization for the subsequent lower values of β is also empirically confirmed, as for our experiments in Sec. 2.6, we need fewer iterations on average to converge to a local minimum when we initialize based of β-annealing as compared to when we initialize randomly. Furthermore, in our experiments we have observed that with β-annealing included, our heuristic achieves good results for random initializations for $\beta = 1$, hence tailoring initialization procedures is not necessary, at least for the scenarios we considered. Many of the additional steps in choosing the right initialization, or in the optimization heuristics proposed in the literature, effectively change the optimization problem and hence obfuscate the effect of the cost function itself.

Note that the β-annealing algorithm admits producing results for a series of values of β at once: Keeping all intermediate aggregation functions, one obtains aggregations for all values of β in the set $\{1, 1-\Delta, 1-2\Delta, \ldots, 1-\Delta\lceil\frac{1-\beta_{\text{target}}}{\Delta}\rceil, \beta_{\text{target}}\}$. The aggregations

Algorithm 2.2 β-Annealing Information-Theoretic Markov Aggregation

function $g = \text{AnnITMA}(\mathbb{P}, \beta_{\text{target}}, |\overline{\mathcal{Z}}|, \#\text{iter}_{\max}, \Delta)$
  $\beta \leftarrow 1$
  $g = \text{sGITMA}(\mathbb{P}, \beta, |\overline{\mathcal{Z}}|, \#\text{iter}_{\max})$
  while $\beta > \beta_{\text{target}}$ **do**
    $\beta \leftarrow \max\{\beta - \Delta, \beta_{\text{target}}\}$
    $g = \text{sGITMA}(\mathbb{P}, \beta, |\overline{\mathcal{Z}}|, \#\text{iter}_{\max}, g)$
  end while
end function

one obtains are exactly those one would obtain from restarting AnnITMA for each value in this set, each time with the same random initial partition. We used this fact in our experiments: If we were interested in results for β_{target} ranging between 0 and 1 in steps of 0.05, rather than restarting AnnITMA for each value in this set, we started AnnITMA for $\beta_{\text{target}} = 0$ and $\Delta = 0.05$ once, keeping all intermediate results.

2.5.3. Computational Complexity

Note that the asymptotic computational complexity of Algorithm 2.2 is the same as that of Algorithm 2.1, since the former simply calls the latter $\lceil (1 - \beta_{\text{target}})/\Delta \rceil + 1$ times. We thus only evaluate only the complexity of Algorithm 2.1. From (2.17), we can express $\mathrm{C}_\beta(\mathbf{Z}, g)$ in terms of three mutual information terms:

$$\mathrm{C}_\beta(\mathbf{Z}, g) = \beta I(Z_1; Z_2) + (1 - 2\beta) I(Z_1; g(Z_2)) - (1 - \beta) I(g(Z_1); g(Z_2)). \tag{2.24}$$

The first term $I(Z_1; Z_2)$ is constant regardless of the aggregation, hence the computation of $\mathrm{C}_\beta(\mathbf{Z}, g)$ depends upon the computation of the other two terms.

In each iteration of the main loop, we evaluate $\mathrm{C}_\beta(\mathbf{Z}, g_y)$ in line 12 for each $z \in \mathcal{Z}$ and $\overline{z} \in \overline{\mathcal{Z}}$. Note that $g_{\overline{z}}$ differs from the current g only for one element as defined in line 11. Thus, the joint PMF $P_{Z_1, g_{\overline{z}}(Z_2)}$ differs from $P_{Z_1, g(Z_2)}$ in only two rows and hence can be computed from $P_{Z_1, g(Z_2)}$ in $\mathcal{O}(|\mathcal{Z}|)$ computations. Moreover, $I(Z_1; g_{\overline{z}}(Z_2))$ can be computed from $I(Z_1; g(Z_2))$ in $\mathcal{O}(|\mathcal{Z}|)$ computations, cf. [38, Prop. 1]. This is because we can write [40, eq. (2.28)]

$$\begin{aligned} I(Z_1; g_y(Z_2)) = I(Z_1; g(Z_2)) \\ + \sum_{\substack{x_1 \in \mathcal{X} \\ y_2 \in \{y, g(x)\}}} P_{Z_1, g_y(Z_2)}(x_1, y_2) \log \left(\frac{P_{Z_1, g_y(Z_2)}(x_1, y_2)}{P_{Z_1}(x_1) P_{g_y(Z_2)}(y_2)} \right) \\ - \sum_{\substack{x_1 \in \mathcal{X} \\ y_2 \in \{y, g(x)\}}} P_{Z_1, g(Z_2)}(x_1, y_2) \log \left(\frac{P_{Z_1, g(Z_2)}(x_1, y_2)}{P_{Z_1}(x_1) P_{g(Z_2)}(y_2)} \right). \end{aligned} \tag{2.25}$$

The term $I(g_y(Z_1); g_y(Z_2))$ can be computed from $I(g(Z_1); g(Z_2))$ in $\mathcal{O}(|\overline{\mathcal{Z}}|)$ computations, but requires the updated joint PMF $P_{g_{\overline{z}}(Z_1), g_{\overline{z}}(Z_2)}$. This PMF can be computed from $P_{g(Z_1), g(Z_2)}$ in $\mathcal{O}(|\mathcal{Z}|)$ computations. Line 12 is executed once for each aggregate state in $\overline{\mathcal{Z}}$ and once for each state in $\mathcal{Z}$ in every iteration, so we get that optimizing g has a computational complexity of $\mathcal{O}(|\overline{\mathcal{Z}}| \cdot |\mathcal{Z}|^2)$ per iteration.

2.6. Experiments and Examples

Now we look at examples and experiments to study the behaviour of the proposed cost function $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W})$ and the optimization heuristic.

2.6.1. A Non-Reversible Markov Chain

The last property of Lemma 2.6 cannot be generalized to non-reversible Markov chains. Specifically, as the proof of Lemma 2.6 shows, C_β is non-decreasing in β iff $\mathrm{C}_P \geq 2\mathrm{C}_L$. Since one can find also non-reversible Markov chains for which this holds, reversibility is sufficient but not necessary for C_β to be non-decreasing in β. We next consider a non-reversible Markov chain $\mathbf{Z} \sim \mathrm{Mar}(\{1, 2, 3\}, \mathbb{P})$ with

$$\mathbb{P} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.25 & 0.3 & 0.45 \\ 0.15 & 0.425 & 0.425 \end{bmatrix} \tag{2.26}$$

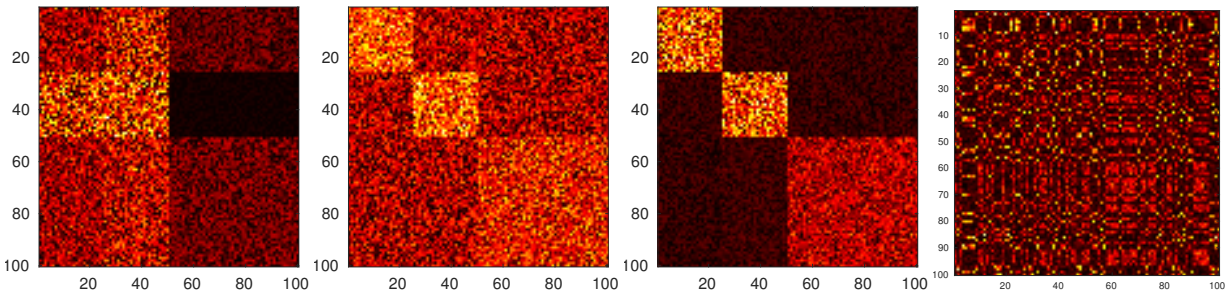
and let g be such that $g(1) = 1$ and $g(2) = g(3) = 2$. Then, $\mathrm{C}_L = 0.0086$ and $\mathrm{C}_P = 0.0135$, i.e., $\mathrm{C}_P < 2\mathrm{C}_L$. In this case, C_β is decreasing with increasing β.

2.6.2. Quasi-Lumpable and Nearly Completely Decomposable Markov Chains

Suppose we have a partition $\{\mathcal{Z}_i\}$, $i = 1, \ldots, M$, of $\mathcal{Z}$ with $|\mathcal{Z}_i| = N_i$. Then for any $\mathbb{A}'$ and $\mathbb{P}'_{ij}$ that are $M \times M$ and $N_i \times N_j$ row stochastic matrices, respectively, define $\mathbb{A} = [a_{ij}] = (1 - \alpha)\mathbb{A}' + \alpha\mathbf{I}$, $\alpha \in [0, 1]$, and let

$$\mathbb{P}' = \begin{bmatrix} a_{11}\mathbb{P}'_{11} & a_{12}\mathbb{P}'_{12} & \cdots & a_{1M}\mathbb{P}'_{1M} \\ a_{21}\mathbb{P}'_{21} & a_{22}\mathbb{P}'_{22} & \cdots & a_{2M}\mathbb{P}'_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1}\mathbb{P}'_{M1} & a_{M2}\mathbb{P}'_{22} & \cdots & a_{MM}\mathbb{P}'_{MM} \end{bmatrix}. \tag{2.27}$$

Suppose further that $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P}')$. If g induces the partition $\{\mathcal{Z}_i\}$, then it can be shown that $\overline{\mathbf{Z}}$ is Markov with transition probability matrix $\mathbb{A}$, i.e., $\tilde{\mathbf{Z}} \equiv \overline{\mathbf{Z}}$, and $\mathrm{C}_0(\mathbf{Z}, \tilde{\mathbf{Z}}) = 0$. The Markov chain $\mathbf{Z}$ is *lumpable* w.r.t. the partition g. The matrix $\mathbb{P}'$ is block stochastic and



(a) $\alpha = \varepsilon = 0$ (b) $\alpha = 0.95$, $\varepsilon = 0.8$ (c) $\alpha = 0.95$, $\varepsilon = 0.4$ (d) $\alpha = 0.95$, $\varepsilon = 0.4$; rows and columns are permuted

Figure 2.2.: (a)-(c): Colorplots of the transition probability matrices $\mathbb{P}$ for different values of α and ε. For large α the block diagonal structure becomes more dominant. (d): A random permutation of the rows and columns of (c) hides the block structure.

the parameter α specifies how dominant the diagonal blocks are. Specifically, if $\alpha = 1$, then $\mathbb{P}'$ is block diagonal and we call $\mathbf{Z}$ *completely decomposable*. Such a Markov chain is not irreducible. We hence look at Markov chains $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$ with

$$\mathbb{P} = (1 - \varepsilon)\mathbb{P}' + \varepsilon\mathbb{E} \tag{2.28}$$

where $\varepsilon \in [0, 1]$ and where $\mathbb{E}$ (which can be interpreted as noise) is row stochastic and irreducible. For small values of ε we call $\mathbf{Z}$ nearly completely decomposable (NCD) if α is close to one, otherwise we call it quasi-lumpable.

The structure of the final Markov process $\mathbf{Z} \sim \mathrm{Mar}(\mathcal{Z}, \mathbb{P})$ to be aggregated can be described as follows: The elements of $\mathbb{A}$ basically define the transition probability among different subgroups, α specifies (approximately) the likelihood of staying within the subgroup, $\mathbb{P}'_{ij}$ defines the transition probabilities from the elements subgroup i to subgroup j and ϵ specifies the strength of noise in comparison to the block structure imposed by $\mathbb{P}'$.

We now perform experiments with these types of Markov chains. We set $M = 3$, $N_1 = N_2 = 25$, and $N_3 = 50$, and chose the parameters from $\alpha \in \{0, 0.5, 0.95\}$ and $\varepsilon \in \{0, 0.4, 0.8\}$. For each pair (α, ε), we generated 250 random matrices $\mathbb{A}'$ and $\mathbb{P}'_{ij}$. A selection of the corresponding matrices $\mathbb{P}$ is shown in Fig. 2.2(a) to Fig. 2.2(c). The darker an element is in the figures, the closer is the matrix entry to 0.

Note that in practice the states of even a completely decomposable Markov chain $\mathbf{Z}$ are rarely ordered such that the transition probability matrix is block diagonal. Rather, the state labeling must be assumed to be random. In this case, $\mathbb{P}$ is obtained by a random permutation of the rows and columns of a block diagonal matrix (see Fig. 2.2(d), which is a permuted version of Fig. 2.2(c)), which prevents the optimal aggregation function being "read off" simply by looking at $\mathbb{P}$. That $\mathbb{P}$ has a block structure in our case provides us with a plausible "ground truth" to compare the outcome of our heuristic to. It is important to

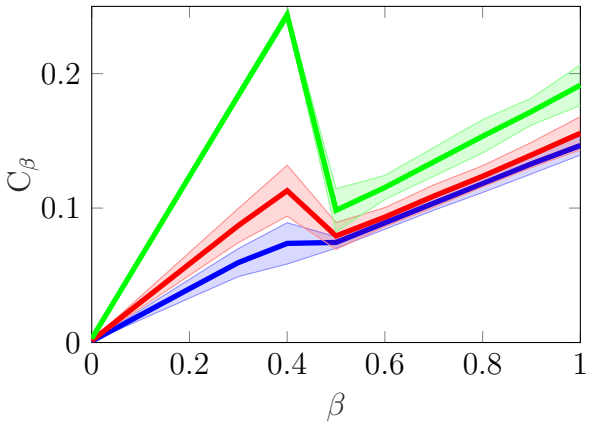
note that the block structure does not affect the performance of our algorithms, since they 1) are unaware of this structure and 2) are initialized randomly. Hence the performance of our algorithm will be independent of whether it is presented with $\mathbb{P}$ depicted in Fig. 2.2(c) or in Fig. 2.2(d).

We applied our aggregation algorithm both with and without the annealing procedure for $\beta \in \{0, 0.1, \ldots, 0.9, 1\}$ and compared the results to the partition $\{\mathcal{Z}_i\}$. We measure the success, i.e., the degree to which the function g obtained from the algorithm agrees with the partition $\{\mathcal{Z}_i\}$, using the adjusted Rand index (ARI). An ARI of one indicates that the two partitions are equivalent. We always assume that the number M of sets in the partition $\{\mathcal{Z}_i\}$ is known.

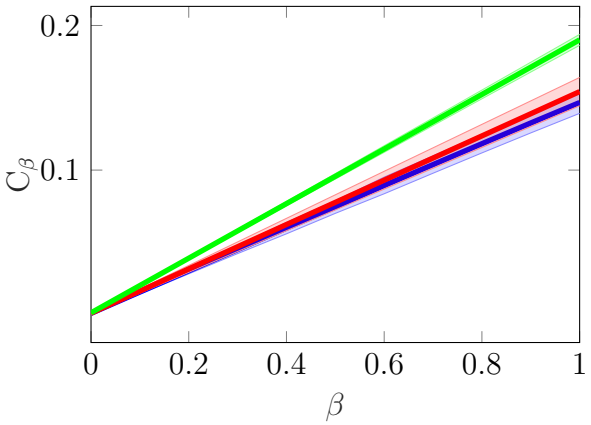
The results are shown in Fig. 2.3. Specifically, Fig. 2.3(b) shows that the cost for the aggregation found by our algorithm with β-annealing decreases monotonically with decreasing β: We obtain a partition for a given value of β. This partition has, assuming $\mathrm{C}_P \geq 2\mathrm{C}_L$, an even lower cost for a smaller value of β. Further optimization for this smaller value of β reduces the cost, leading to the depicted phenomenon. In contrast, the sequential Algorithm 2.1 without the annealing procedure fails for values of β less than 0.5. This is apparent both in the cost in Fig. 2.3(a) (which has a sharp jump around $\beta = 0.5$) and in the ARI in Fig. 2.3(c) (which drops to zero). Apparently, the algorithm gets stuck in a bad local optimum.

Figs. 2.3(e) to 2.3(f) show the ARI of the aggregations obtained by our algorithm with β-annealing. It can be seen that performance improves with increasing α, since the dominant block structure makes discovering the correct partition easier. Moreover, it can be seen that for $\alpha = 0$ the optimum β lies at smaller values, typically smaller than 0.5. The position of this optimum increases with increasing noise: while in the noiseless case the correct partition is typically obtained for β close to zero, in the high noise case of $\varepsilon = 0.8$ we require $\beta \approx 0.4$ to achieve good results. The reason may be that the higher noise leads to more partitions being quasi-lumpable by leading to an i.i.d. $\overline{\mathbf{Z}}$, hence for small values of β one may get drawn into these "false solutions" more easily. In contrast, for NCD Markov chains (i.e., for $\alpha = 0.95$) sometimes noise helps to discover the correct partition. Comparing Figs. 2.3(e) and 2.3(d), we see that a noise of $\varepsilon = 0.4$ allows us to perfectly discover the partition. We believe that a small amount of noise helps in escaping bad local minima.

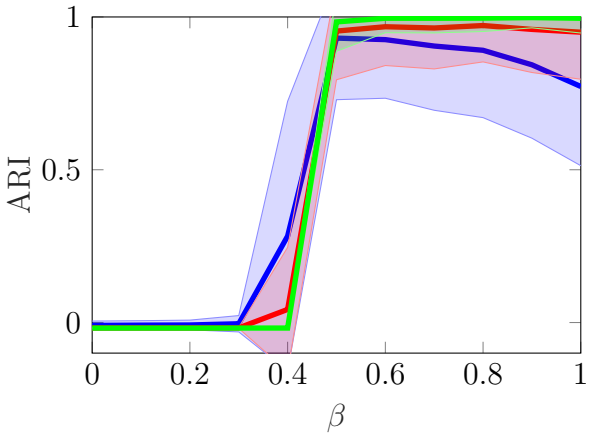
Observe that the β for which the highest ARI is achieved not necessarily falls together with the values 0, 0.5, or 1. This indicates that our generalized aggregation framework has the potential to strictly outperform aggregation cost functions and algorithms that have been previously proposed (cf. Section 2.4.4).



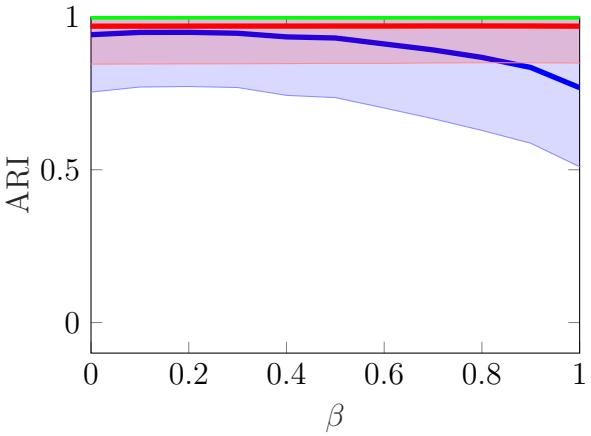
(a) Cost without β-annealing, $\varepsilon = 0.4$



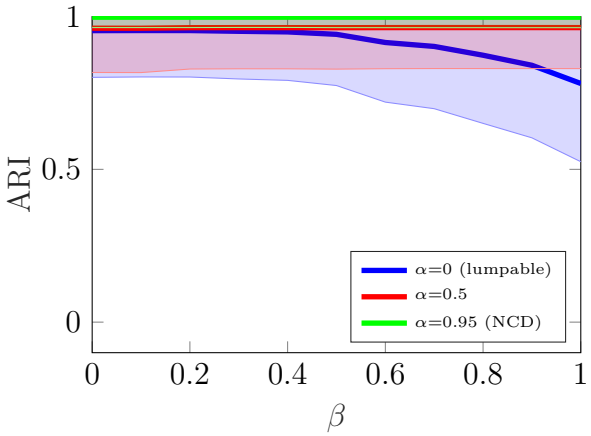
(b) Cost with β-annealing, $\varepsilon = 0.4$



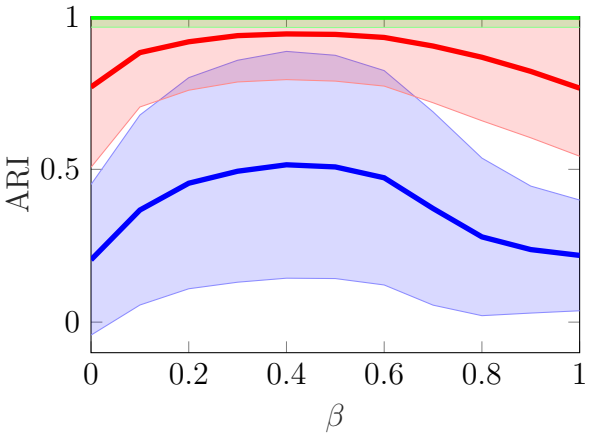
(c) ARI without β-annealing, $\varepsilon = 0.4$



(d) ARI with β-annealing, $\varepsilon = 0.4$



(e) ARI with β-annealing, $\varepsilon = 0$



(f) ARI with β-annealing, $\varepsilon = 0.8$

Figure 2.3.: (a) and (b): Curves showing the cost function C_β with and without β-annealing for $\epsilon = 0.4$. (c) and (d): Curves showing the adjusted Rand index (ARI) with and without β-annealing for $\epsilon = 0.4$. (d)-(f): Curves showing ARI obtained via β-annealing for different settings of ϵ and α. Mean values (solid lines) are shown together with the standard deviation (shaded areas).

2.6.3. A Natural Language Processing Experiment

Consider the letter bi-gram model from [41] which was obtained by analyzing the co-occurrence of letters in F. Scott Fitzgerald's book "The Great Gatsby". The text was modified by removing chapter headings, line breaks, underscores, and by replacing é by e. With the remaining symbols, we obtained a Markov chain with an alphabet size of $N = 76$ (upper and lower case letters, numbers, punctuation, etc.).

We applied Algorithm 2.2 for $|\overline{\mathcal{Z}}| \in \{2, \dots, 7\}$ and $\beta \in \{0, 0.1, \dots, 0.9, 1\}$. To get consistent results, we restarted the algorithm 20 times for $\beta = 1$ and chose the aggregation g that minimized $\mathrm{C}_1(\mathbf{Z}, g)$; we used this aggregation g as an initialization for the β-annealing procedure. Looking at the results for $|\overline{\mathcal{Z}}| = 4$ in Table 2.1, one can observe that the results for $\beta = 0.8$ appear to be most meaningful when compared to other values of β such as $\beta = 1$ (information bottleneck), $\beta = 0.5$ (as proposed in [2]), and $\beta = 0$ (as proposed in [4]). Specifically, for $\beta = 0$ not even the annealing procedure was able to achieve meaningful results. This conclusion is supported by calculating the ARI of these aggregations for a plausible reference aggregation of the alphabet into upper case vowels, upper case consonants, lower case vowels, lower case consonants, numbers, punctuation, and the blank space as shown in the first row of the Table 2.1. The absolute ARI values are not a good performance indicator in this case since we are comparing to a reference partition with seven sets whereas $|\overline{\mathcal{Z}}| = 4$.

In Table 2.2 the same experiment is repeated for $|\overline{\mathcal{Z}}| \in \{2, 7\}$. We again observe that $\beta = 0.8$ leads to the most meaningful results which is also supported by ARI values.

β Value	ARI	Partitions, shown for $\lvert\mathcal{Y}\rvert = 4$
Ref.	–	{␣},{!"$'(),-.:;?[]},{aeiou},{0123456789},{AEIOU},{BCDFGHJKLMNPQRSTVWYZ},{bcdfghjklmnpqrstvwxyz}
$\beta = 1$	0.43	{␣!'),-.0:;?]},{aeioy},{"$(123456789ABCDEFGHIJKLMNOPQRSTUVWY[h},{Zbcdfgjklmnpqrstuvwxz}
$\beta = 0.8$	0.46	{␣!'),-.:;?Z]},{aeiouy},{"$(0123456789ABCDEFGHIJKLMNOPQRSTUVWY[h},{bcdfgjklmnpqrstvwxz}
$\beta = 0.5$	0.35	{␣!3?Z},{'2456789AOUaeiou},{"$(-01BCDEFGHIJKLMNPQRSTVWY[bhjqw},{),.:;]cdfgklmnprstvxyz}
$\beta = 0$	0.12	{␣-2CEFMPSTcfgopst},{"'456789AOUZaeiu},{!$1?BDGHJLNQRVW[bhjklmqrvwz},{(),.03:;IKY]dnxy}

Table 2.1.: Aggregating a letter bi-gram model. The partitions are shown together with the ARI ARI w.r.t. the reference partition (first row) for $|\overline{\mathcal{Z}}| = 4$

β Value	ARI	Partitions, shown for $\lvert\mathcal{Y}\rvert \in \{2, 7\}$
Ref.	–	{␣},{!"$'(),-.:;?[]},{aeiou},{0123456789},{AEIOU},{BCDFGHJKLMNPQRSTVWYZ},{bcdfghjklmnpqrstvwxyz}
$\beta = 1$	0.2	{␣!"'),-.01235689:;?KU]aehioy},{$(47ABCDEFGHIJLMNOPQRSTVWYZ[bcdfgjklmnpqrstuvwxz}
	0.34	{␣!'),-.:;?]},{aeioy},{"$(0123456789ABCDEFGHIJLMNOPQRSTVWY[},{bcfjmpqstw},{dgx},{KUh},{Zklnruvz}
$\beta = 0.8$	0.24	{␣!"'),-.01235689:;?EU]aehiouy},{$(47ABCDFGHIJKLMNOPQRSTVWYZ[bcdfgjklmnpqrstvwxz}
	0.35	{␣!'),-.:;?]},{aeioy},{$"(0123456789ABCDEFGHIJLMNOPQRSTUVWYZ[},{Kh},{bcfjkmpqstw},{dg},{lnruvxz}
$\beta = 0.5$	0.15	{␣!'-12368?EOUZaeiou},{"$(),.04579:;ABCDFGHIJKLMNPQRSTVWY[]bcdfghjklmnpqrstvwxyz}
	0.31	{␣'},{!),.:;?]dy},{aeiou},{"$(-0123589ACEIMOPRSTUWZ},{BDFGHJKLNQVYhj},{7[bcfgkmpqstw},{46lnrvxz}
$\beta = 0$	0.01	{␣!$(-0124578?ABCFHLMNOPRSTUVWaceglnostuwxz},{"'),.369:;DEGIJKQYZ[]bdfhijkmpqrvy}
	0.02	{␣4689ao},{$'AKOiux},{!?HVZhjkmvz},{"(-25CEFLMNRUWY[egnprs},{37BPQbl},{1:;STctw},{),.ODGIJ]dfy}

Table 2.2.: Aggregating a letter bi-gram model. The partitions are shown together with ARI w.r.t. the reference partition (first row) for $|\overline{\mathcal{Z}}| \in \{2, 7\}$

3

Co-Clustering via Information-Theoretic Markov Aggregation

In this chapter we discuss how to transform the information-theoretic framework for Markov aggregation, developed in Chap. 2, to apply it to the co-clustering problem.

Co-clustering is the task of the simultaneous clustering of two sets, typically represented by rows and columns of a data matrix, based on the relationships between the elements of two sets. Aside from being a clustering problem in its own right, co-clustering is also applied for clustering only one dimension of the data matrix. In these scenarios, co-clustering is an implicit method for feature clustering and provides an alternative to feature selection with, purportedly, increased robustness to noisy data [7, 9, 11].

A popular approach to co-clustering employs information-theoretic cost functions and is based on transforming the data matrix into a probabilistic description of the two sets and their relationship. For example, if the entries in the data matrix are all nonnegative, one can normalize the data matrix to obtain a joint probability distribution of two random variables taking values in the two sets. This approach has been taken by, e.g., Slonim et al. [7], Bekkerman et al. [10], El-Yaniv and Souroujon [42], and Dhillon et al. [9]. A different approach to co-clustering is to identify the data matrix with the weight matrix of a bipartite graph and subsequently apply graph partitioning methods to cluster the rows and columns of the data matrix. This approach has been taken by, e.g., Dhillon [43], Labiod and Nadif [44], and Ailem et al. [45]. Other popular approaches are model-based (e.g., latent block models as in [46] and the references therein) or based on nonnegative matrix factorization (e.g., [47, Sec. 4.4]).

Our Markov aggregation framework from Chap. 2 allows us to combine ideas from the graph-based and the information-theoretic approaches. Specifically, we use the data matrix to define a simple random walk on a bipartite graph, i.e., a first-order, stationary Markov

chain. Clustering this bipartite graph (i.e., co-clustering) thus becomes equivalent to clustering the state space of a Markov chain (i.e., Markov aggregation). This, in turn, allows us to transfer the information-theoretic cost function from the latter problem to the former. The resulting cost function, parameterized by a single parameter β, derives its justification from the corresponding Markov aggregation problem. This justification also applies to other information-theoretic cost functions in the literature [7–11], which we obtain as special cases for appropriate choices of β.

In several examples we discuss weaknesses inherent in the cost function for certain values (or value ranges) of β. We adapt the optimization heuristic from Sec. 2.5 for co-clustering and analyze the influence of the choice of β on the co-clustering performance. For the synthetic data sets, we confirm that co-clustering outperforms one-sided clustering if the data matrix is noisy or if there is strong intra-cluster coupling. We then analyze the performance on real-world datasets. For the Newsgroup20 dataset we observed that performance is insensitive to β as long as the number of word clusters is sufficiently large. Performance drops for a few word clusters, a fact for which we provide a theoretical explanation. The parameter β has a somewhat stronger influence on the performance on the MovieLens100k dataset, for which we obtained movie clusters largely consistent with genres. Finally, for the Southern Women Event Participation dataset, our results are remarkably similar to the reference co-clusterings from [48, 49].

Our contribution in the domain of information-theoretic co-clustering can be summarized as follows: We provide a generalized framework for information-theoretic co-clustering by connecting it with Markov aggregation (Sec. 3.4). The cost function, parameterized with a single parameter and connected with the information bottleneck formalism, is justified by well-defined operational goals of the Markov aggregation problem.Our generalized framework contains previously proposed information-theoretic cost functions as special cases (Sec. 3.6). Since the parameter of our cost function has an intuitive meaning, our framework leads to a deeper understanding of the previously proposed approaches. This understanding is further developed by considering the strengths and limitations of information-theoretic cost functions for co-clustering with the help of examples and experiments on synthetic datasets (Section 3.7). We also discuss the influence of the single parameter on the co-clustering results and present general guidelines for setting this parameter depending on the characteristics of the dataset (Sec. 3.8). We then turn to experiments with real-world datasets (Sec. 3.9). Our cost function contains previously proposed cost functions for co-clustering as special cases which allows us to compare them *fairly*, i.e., with the same initialization steps and the same optimization heuristic. For example, the insensitivity to β in our experiments with the Newsgroup20 datasets provides a new perspective on the differences reported in [7, 9–11], suggesting that they are due to differences in optimization heuristics, preprocessing steps, or choice of data subsets rather than the differences in the cost function. We do not deal with the important issue of determining the number of clusters We assume that we know the desired number of clusters as this allows us to focus on the influence of the cost function independent of the heuristics used to determine the number of clusters.

This work was done in collaboration with Dr. Bernhard Geiger and Clemens Blöchl.

3.1. Co-Clustering

Suppose we have two disjoint finite sets $\mathcal{X}$ and $\mathcal{Y}$ and a $|\mathcal{X}| \times |\mathcal{Y}|$ matrix $\mathbb{D}$ containing, e.g., similarities, the number of co-occurrences, or correlations between elements of these two sets. As an example, if $\mathcal{X}$ is a set of documents and $\mathcal{Y}$ a set of words, then the (i, j)-th entry of $\mathbb{D}$ could be the number of times the word j appeared in document i. Co-clustering is concerned with grouping elements of $\mathcal{X}$ and $\mathcal{Y}$ separately into $|\overline{\mathcal{X}}|$ and $|\overline{\mathcal{Y}}|$ clusters (document clusters and word clusters in this example), sacrificing information about the individual data elements to make the group characteristics more prominent and accessible. The original alphabets $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ are not important in clustering problems so we can make the assumption w.l.o.g. that $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ are disjoint. In case of soft clustering these groupings are indicated by the stochastic mappings $P_{\overline{X}|X} = \mathbb{W}_1$ and $P_{\overline{Y}|Y} = \mathbb{W}_2$. Incase of hard clusters we will represent the clustering functions by $\Phi : \mathcal{X} \to \overline{\mathcal{X}}$ and $\Psi : \mathcal{Y} \to \overline{\mathcal{Y}}$.

3.2. Preliminaries

In the spirit of the IB formalism, mutual information can be used to measure relevance. Relevant information loss measures the information about some relevant RV S that is lost by observing a statistically related RV Z via a stochastic mapping $\mathbb{W}$. The quantity was introduced by Plumbley in the context of unsupervised neural networks [50].

Definition 3.1 (Relevant Information Loss)**.** Let S and Z be RVs with finite alphabet, and let $\overline{Z}$ be obtained by applying a stochastic mapping $P_{\overline{Z}|Z} = \mathbb{W}$ to Z, i.e., $\overline{Z} = P_{\overline{Z}|Z}(Z)$. Then, the relevant information loss w.r.t. S that is induced by $\mathbb{W}$ is

$$L_S(Z \to \overline{Z}) := I(S; Z) - I(S; \overline{Z}) = I(S; Z|\overline{Z}) \geq 0. \tag{3.1}$$

With this definition, we can rewrite the cost function for Markov aggregation in terms of relevant information loss:

Lemma 3.1. Let $\mathbf{Z}$ be the Markov process that we want to aggregate and $\overline{\mathcal{Z}}$ be the alphabet of the aggregated process, then (2.16) can be rewritten in terms of relevant information loss as follows:

$$\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) = \beta L_{Z_1}(Z_2 \to \overline{Z}_2) + (1 - \beta) L_{\overline{Z}_2}(Z_1 \to \overline{Z}_1). \tag{3.2}$$

3.3. Related Work

Information-theoretic approaches to co-clustering require a probability distribution over the sets to be clustered. For example, if the data matrix $\mathbb{D}$ is nonnegative, then one can normalize it such that its entries sum to one. One can thus define RVs X and Y over the sets $\mathcal{X}$ and $\mathcal{Y}$ that have a joint distribution $P_{X,Y} \propto \mathbb{D}$.

An early information-theoretic approach to co-clustering was proposed by Slonim and Tishby [7] and is based on the IB method [35]. There, the authors proposed first finding the clustering function Φ maximizing $I(\Phi(X);Y)$, and then, after fixing Φ, finding the clustering function Ψ that maximizes $I(\Phi(X);\Psi(Y))$. Their approach was improved later by El-Yaniv and Souroujon, who suggested iterating this procedure multiple times [42]. Also based on the IB method is the work of Wang et al. [11]. They used a multivariate extension of mutual information to compress "input information" – captured by the mutual information terms $I(X;Y)$, $I(X;\Phi(X))$, and $I(Y;\Psi(Y))$ – while preserving relevant information – captured by the information shared between the clusters, $I(\Phi(X);\Psi(Y))$, and the predictive power of the clusters, $I(\Phi(X);Y)$ and $I(X;\Psi(Y))$.

In 2003, Dhillon et al. proposed a co-clustering algorithm simultaneously determining clustering functions Φ and Ψ with the goal to maximize $I(\Phi(X);\Psi(Y))$ [9]. They showed that the problem is equivalent to a constrained nonnegative matrix tri-factorization problem [9, Lemma 2.1] with Kullback-Leibler divergence as cost function. (An iterative update rule for the entries of the three matrices is provided in [47, Sec. 4.4].) The work in [9] was generalized into various directions. On the one hand, Bekkerman et al. investigated simultaneous clustering of more than two sets in [10]. Rather than maximizing one of the multivariate extensions of mutual information, the authors suggested maximizing the sum of mutual information terms between pairs of clusters; the pairs of clusters considered in the sum are determined by an undirected graph that must be provided by the user. On the other hand, Banerjee et al. viewed co-clustering as a matrix approximation problem [51], of which the nonnegative matrix tri-factorization problem of [9, Lemma 2.1] is a special case. Their generalized framework admits any Bregman divergence (e.g., Kullback-Leibler divergence or squared Euclidean distance) as cost function and several co-clustering schemes characterized by the type of summary statistic used to approximate the matrix.

Finally, Laclau et al. formulate the co-clustering problem as an optimal transport problem with entropic regularization [52]. Their formulation turns into a probability matrix approximation problem with Kullback-Leibler divergence as cost function, but 1) the order of original and approximate distribution is swapped compared to [9, Lemma 2.1], and 2) the approximate distribution is obtained differently. They proposed solving the co-clustering problem with the Sinkhorn-Knopp algorithm and suggested a heuristic to determine the number of clusters.

3.4. Tranforming C_β for Co-clustering

We now apply the Markov aggregation framework from Chap. 2 to the co-clustering problem. For this purpose we have to first transform the matrix $\mathbb{D}$ to define a Markov process. If $\mathbb{D}$ is nonnegative, we can interpret it as the weight matrix of an undirected, weighted, bipartite graph, cf. [43]. Throughout this work we will assume that $\mathbb{W}$ is such that the bipartite graph is irreducible. On this graph, one can then define a simple random walk,

i.e., a Markov chain $\mathbf{Z}$ with alphabet $\mathcal{X} \cup \mathcal{Y}$ and state transition matrix

$$\mathbb{A} = \mathbb{M}^{-1} \begin{bmatrix} 0 & \mathbb{D} \\ \mathbb{D}^T & 0 \end{bmatrix} \tag{3.3}$$

where $\mathbb{M}$ is a diagonal matrix collecting sums of all connected edge weights of respective nodes. The matrix $\mathbb{M}$ normalizes each row of $\mathbb{A}$ to make it a probability distribution. Since the graph is bipartite and undirected, the Markov chain $\{Z_t\}$ is 2-periodic and reversible. In the case when $\mathbb{D}$ contains negative elements one can do appropriate transformation to obtain a weight matrix to define the weighted bipartite graph in a way that one can hopefully preserve the original structures present in the data.

To apply the Markov aggregation framework to co-clustering, we have to further add the constraint that the aggregation mapping $\mathbb{W}$ maps $\mathcal{X}$ and $\mathcal{Y}$ to disjoint "fan-out" sets, i.e., for some finite and disjoint sets $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ with $\overline{\mathcal{X}} \ll \mathcal{X}$ and $\overline{\mathcal{Y}} \ll \mathcal{Y}$, we have stochastic mappings $\mathbb{W}_1 = P_{\overline{X}|X}$ and $\mathbb{W}_2 = P_{\overline{Y}|Y}$ such that

$$\forall i \in \mathcal{X}\colon \quad \mathbb{W}(i,j) = P_{\overline{Z}|Z}(j|i) = \begin{cases} P_{\overline{X}|X}(j|i), & j \in \overline{\mathcal{X}} \\ 0, j \in \overline{\mathcal{Y}} \end{cases} \tag{3.4}$$

and

$$\forall i \in \mathcal{Y}\colon \quad \mathbb{W}(i,j) = P_{\overline{Z}|Z}(j|i) = \begin{cases} P_{\overline{Y}|Y}(j|i), & j \in \overline{\mathcal{Y}} \\ 0, j \in \overline{\mathcal{X}} & . \end{cases} \tag{3.5}$$

The following proposition then transfers the cost function from C_β to the co-clustering setting:

Proposition 3.2. Consider two disjoint finite sets $\mathcal{X}$ and $\mathcal{Y}$ and a nonnegative $|\mathcal{X}| \times |\mathcal{Y}|$ matrix $\mathbb{D}$ containing similarities between elements of these two sets are given. Define two discrete RVs X and Y over these sets, where the joint distribution $P_{X,Y}$ is obtained by normalizing $\mathbb{D}$. Let $\mathbf{Z}$ be a stationary Markov chain with alphabet $\mathcal{X} \cup \mathcal{Y}$ and state transition matrix $\mathbb{A}$ given in (3.3). Let $\beta \in [0,1]$ and suppose the disjoint sets $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ are given.

For every stochastic mapping $\mathbb{W}$ satisfying the mutual exclusivity constraint ((3.4) and (3.5)), we have

$$\begin{aligned} 2 \cdot \mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) = \beta \left[L_X(Y \to \overline{Y}) + L_Y(X \to \overline{X}) \right] \\ + (1-\beta) \left[L_{\overline{X}}(Y \to \overline{Y}) + L_{\overline{Y}}(X \to \overline{X}) \right] =: \mathcal{L}_\beta(\mathbb{W}_1, \mathbb{W}_2) \end{aligned} \tag{3.6}$$

where $\overline{X} := P_{\overline{X}|X}(X)$ and $\overline{Y} := P_{\overline{Y}|Y}(Y)$.

Proof: Suppose $\boldsymbol{\mu} = [\mu_i]$ is the invariant distribution of $\mathbb{A}$, i.e., $\boldsymbol{\mu}^T = \boldsymbol{\mu}^T \mathbb{A}$. It follows that $\mathrm{diag}(\boldsymbol{\mu}) \propto \mathbb{M}$. The marginal distributions for X and Y are $P_X = \sum_{y \in \mathcal{Y}} P_{X,Y}(\cdot, y) \propto \mathbb{D}\mathbf{1}$

and $P_Y^T = \sum_{x\in\mathcal{X}} P_{X,Y}(x,\cdot) \propto \mathbf{1}^T\mathbb{D}$, respectively. The 2-periodicity of $\{Z_t\}$ thus implies

$$\mu_i = \frac{1}{2}\begin{cases} P_X(i), & i \in \mathcal{X} \\ P_Y(i), & i \in \mathcal{Y}. \end{cases} \tag{3.7}$$

Now assume that the Markov chain $\mathbf{Z}$ is stationary, i.e., the distribution of Z_1 coincides with the invariant distribution $\boldsymbol{\mu}$. Let U be a RV that indicates whether Z_1 was drawn from $\mathcal{X}$ or $\mathcal{Y}$, i.e.,

$$U := \begin{cases} 1, & Z_1 \in \mathcal{X} \\ 0, & Z_1 \in \mathcal{Y}. \end{cases} \tag{3.8}$$

Note that U is a function not only of Z_1 but, by periodicity, of Z_t for every t. The RV U thus connects P_{Z_t} with P_X or P_Y; e.g., if $U = 1$, then $P_{Z_3} = P_X$. It follows from (3.7) that $\Pr(U = 1) = \Pr(U = 0) = \frac{1}{2}$.

Finally, suppose that $\mathbb{W}$ satisfies the mutual exclusivity constraint (3.4) and (3.5); then $U = 1$ if and only if $\overline{Z}_1 \in \overline{\mathcal{X}}$.

We now investigate $I(Z_1'; Z_2')$, where Z_i' is either Z_i or $\overline{Z}_i$. We get

$$\begin{aligned} I(Z_1'; Z_2') &\overset{(a)}{=} I(Z_1', U; Z_2') \\ &\overset{(b)}{=} I(Z_1'; Z_2'|U) + I(U; \tilde{Z}_2) \\ &\overset{(c)}{=} \frac{1}{2} I(Z_1'; Z_2'|U = 1) + \frac{1}{2} I(Z_1'; Z_2'|U = 0) + H(U) \end{aligned} \tag{3.9}$$

where (a) is because U is a function of Z_1 and $\overline{Z}_1$, (b) is the chain rule of mutual information, and (c) follows because U is also a function of Z_2 and $\overline{Z}_2$ and from the definition of conditional mutual information.

Now suppose $Z_1' = \overline{Z}_1$ and $Z_2' = Z_2$. If $U = 1$, then $\overline{Z}_1 \in \overline{\mathcal{X}}$ and $Z_2 \in \mathcal{Y}$, and the joint distribution $P_{\overline{Z}_1, Z_2}$ equals the joint distribution $P_{\overline{X},Y}$. With similar considerations for $U = 0$ we hence get

$$I(\overline{Z}_1; Z_2) = \frac{1}{2} I(\overline{Z}_1; Z_2|U = 1) + \frac{1}{2} I(\overline{Z}_1; Z_2|U = 0) + H(U) \tag{3.10a}$$

$$= \frac{1}{2} I(\overline{X}; Y) + \frac{1}{2} I(X; \overline{Y}) + H(U). \tag{3.10b}$$

Along the same lines we obtain

$$I(Z_1; Z_2) = I(X; Y) + H(U), \tag{3.10c}$$

$$I(\overline{Z}_1; \overline{Z}_2) = I(\overline{X}; \overline{Y}) + H(U), \tag{3.10d}$$

$$I(Z_1; \overline{Z}_2) = \frac{1}{2} I(X; \overline{Y}) + \frac{1}{2} I(\overline{X}; Y) + H(U). \tag{3.10e}$$

Inserting these in the cost function in Lemma 3.1 and applying the definition of relevant

information loss in Definition 3.1 completes the proof. ■

We now present the transformed cost function for information-theoretic co-clustering:

Definition 3.2 (Generalized Information-Theoretic Co-Clustering)**.** The generalized information-theoretic co-clustering problem is

$$\min_{\mathbb{W}_1,\mathbb{W}_2} \mathcal{L}_\beta(\mathbb{W}_1, \mathbb{W}_2) \tag{3.11}$$

where the minimization is over all stochastic mappings $\mathbb{W}_1 = P_{\overline{X}|X}$, $\mathbb{W}_2 = P_{\overline{Y}|Y}$ and where $\mathcal{L}_\beta(\mathbb{W}_1, \mathbb{W}_2)$ is as in the setting of Proposition 3.2.

The cost function $\mathcal{L}_\beta$ admits an intuitive explanation for the effect of the parameter β: In the context of the words/documents co-clustering example above, minimizing $L_X(Y \to \overline{Y})$ means that we are looking for word clusters that tell us much about documents. In contrast, minimizing $L_{\overline{X}}(Y \to \overline{Y})$ means that we are looking for word and document clusters such that the word clusters tell us much about the document clusters. The parameter β thus determines how strongly the two clusterings should be coupled. We show in Sec. 3.7 and Sec. 3.9 that the choice of β can have a large on the clustering performance.

3.5. Adapting the Optimization Heuristic

In this and the next sections we will focus on hard clusters, i.e., deterministic mappings Φ and Ψ. The restriction is motivated by various reasons: First, as we show later, for some of the more widely used values of β, such as 0.5 and 1, one of the minimizers of (3.11) is hard clustering. Hence in these cases restricting to hard clusters does not lead to suboptimality. Secondly, the previous literature for information-theoretic co-clustering mainly focuses on hard clusters. Focusing on hard clusters allows us to compare our results to the ones in literature. Finally, finding a low complexity heuristic to search over all the stochastic mappings which provides "good" results in different settings and for different values of β is a hard problem. Our initial attempts to develop such heuristics were unfruitful and led to worse results even for hard clustering when compared to the heuristic proposed in the following.

In general, finding a minimizer of our cost function (3.11) when limiting ourselves to hard clustering functions Φ and Ψ is a combinatorial problem with exponential computational complexity in $|\mathcal{X}|$ and $|\mathcal{Y}|$. Since our cost function is derived from the Markov aggregation problem, we can adapt the optimization heuristic proposed in Sec. 2.5 to account for the mutual exclusivity constraint to obtain good sub-optimal co-clustering solutions. The pseudocodes for the adapted sequential heuristic, sGITCC, and the annealing heuristic, AnnITCC, are given in Algorithms 3.1 and 3.2, respectively. The annealing procedure helps in the same way as it did for Markov aggregation, assisting in escaping the bad local minima for smaller values of β. It can be shown along the same lines as in Sec. 2.5 that, by storing intermediate results, the computational complexity of computing $\mathcal{L}_\beta(\Phi, \Psi_\ell)$ and

$\mathcal{L}_\beta(\Phi_j, \Psi)$ can be brought down to $\mathcal{O}(|\mathcal{X}|)$ and $\mathcal{O}(|\mathcal{Y}|)$, respectively. Thus, one iteration of Algorithm 3.1 has computational complexity of $\mathcal{O}(|\mathcal{X}| \cdot |\mathcal{Y}| \cdot \max\{|\overline{\mathcal{Y}}|, |\overline{\mathcal{X}}|\})$.

Algorithm 3.1 Sequential Generalized Information-Theoretic Co-Clustering (sGITCC)

function (Φ, Ψ) = sGITCC($P_{X,Y}$, β, $|\overline{\mathcal{X}}|$, $|\overline{\mathcal{Y}}|$, $\#\text{iter}_{\max}$, tol, optional: initial clustering $(\Phi_{\text{init}}, \Psi_{\text{init}})$)
  if $(\Phi_{\text{init}}, \Psi_{\text{init}})$ is empty **then**
    $(\Phi, \Psi) \leftarrow$ Random Clustering
  else
    $(\Phi, \Psi) \leftarrow (\Phi_{\text{init}}, \Psi_{\text{init}})$
  end if
  $\#\text{iter} \leftarrow 0$
  while $\#\text{iter} < \#\text{iter}_{\max} \wedge \delta > \text{tol}$ **do**
    $C_{old} \leftarrow \mathcal{L}_\beta(\Phi, \Psi)$
    for all elements $i \in \mathcal{X}$ **do** ▷ *Optimizing* Φ
      for all clusters $j \in \overline{\mathcal{X}}$ **do**
        $$\Phi_j(x) = \begin{cases} \Phi(x) & \forall x \neq i \\ j & x = i \end{cases}$$
      end for
      $\Phi(i) = \arg\min_j \mathcal{L}_\beta(\Phi_j, \Psi)$
    end for
    for all elements $k \in \mathcal{Y}$ **do** ▷ *Optimizing* Ψ
      for all clusters $\ell \in \overline{\mathcal{Y}}$ **do**
        $$\Psi_\ell(y) = \begin{cases} \Psi(y) & \forall y \neq k \\ \ell & y = k \end{cases}$$
      end for
      $\Psi(k) = \arg\min_\ell \mathcal{L}_\beta(\Phi, \Psi_\ell)$
    end for
    $\delta \leftarrow C_{old} - \mathcal{L}_\beta(\Phi, \Psi)$
    $\#\text{iter} \leftarrow \#\text{iter} + 1$
  end while
end function

The alternative optimization of Ψ and Φ in sGITCC may introduce further suboptimality as compared to sGITMA for Markov aggregation where there is only one function to optimize over. The following example shows how sGITCC can get stuck in a poor local optimum for $\beta = \frac{1}{2}$. The same example is unproblematic for $\beta = 1$. Since one can certainly find heuristics that perform optimally in this example even for $\beta = \frac{1}{2}$, matching the heuristic to the cost function seems to be an important issue. We will see further evidence for the impact of heuristics on performance in our experiments with the Newsgroup20 dataset in Section 3.9.1.

Algorithm 3.2 β-Annealing Information-Theoretic Co-Clustering (AnnITCC)

function $(\Phi, \Psi) = \text{AnnITCC}(P_{X,Y}, \beta, |\overline{\mathcal{X}}|, |\overline{\mathcal{Y}}|, \#\text{iter}_{\max}, \text{tol}, \Delta)$
  $\alpha \leftarrow 1$
  $(\Phi, \Psi) = \text{sGITCC}(P_{XY}, \beta, |\overline{\mathcal{X}}|, |\overline{\mathcal{Y}}|, \#\text{iter}_{\max}, \text{tol})$
  while $\alpha > \beta$ **do**
    $\alpha \leftarrow \max\{\alpha - \Delta, \beta\}$
    $(\Phi, \Psi) = \text{sGITCC}(P_{XY}, \alpha, |\overline{\mathcal{X}}|, |\overline{\mathcal{Y}}|, \#\text{iter}_{\max}, \text{tol}, (\Phi, \Psi))$
  end while
end function

Example 3.1. Consider the following 3×4 matrix describing the joint probability distribution between X and Y:

$$P_{X,Y} = \left[\begin{array}{c|c|cc} 0.25 & 0 & 0 & 0 \\ \hline 0 & 0.25 & 0 & 0 \\ \hline\hline 0 & 0 & 0.25 & 0.25 \end{array}\right]$$

We are interested in two row clusters and two column clusters, i.e., $|\overline{\mathcal{X}}| = |\overline{\mathcal{Y}}| = 2$. Suppose that during some iteration, the clustering functions Φ and Ψ induce the partition indicated by the thin black lines in the matrix $P_{X,Y}$. At this stage, for $\beta = \frac{1}{2}$ the sequential algorithm will terminate since this Φ is the optimal choice for Ψ fixed, and this Ψ is the optimal choice for Φ fixed. In other words, changing either clustering function alone increases the cost $\mathcal{L}_{\frac{1}{2}} = I(X;Y) - I(\overline{X};\overline{Y})$. Nevertheless, it is clear from looking at $P_{X,Y}$ that the cost is minimized ($I(\overline{X};\overline{Y})$ is maximized) for the partition indicated by the thick black lines. The algorithm thus gets stuck for $\beta = \frac{1}{2}$ because the cost function in this case depends only on the clustered variables, and because it updates the clustering functions subsequently rather than jointly. For larger values of β, the coupling between the clustering functions is weaker. In particular, for $\beta = 1$, the clustering functions can be optimized independent of each other, and the algorithm hence terminates at a partition consistent with the vertical thick line, even if it was started at the partition indicated by the thin lines.

3.6. Special Cases of $\mathcal{L}_\beta(\Phi, \Psi)$

We next show that our cost function $\mathcal{L}_\beta$ contains, for appropriate settings of the parameter β, previously proposed cost functions for co-clustering as special cases. For example, for $\beta = 1$, we obtain

$$\mathcal{L}_1(\Phi, \Psi) = L_X(Y \to \overline{Y}) + L_Y(X \to \overline{X}). \tag{3.12}$$

This cost function consists of two IB functionals: The first term considers clustering Y with X as the relevant variable, while the second term considers clustering X with Y as the relevant variable. This approach rewards clustering solutions for X and Y that are completely decoupled. In [13], we showed that hard co-clusters minimize $\mathcal{L}_\beta$, hence the

restriction to hard co-clusters in sGITCC is optimal for $\beta = 1$. To minimize $\mathcal{L}_1$, one can use the fixed-point equations derived in [35] or the agglomerative IB method (aIB) that merges clusters until the desired cardinality is reached [38]. Finally, a sequential IB method (sIB) has been proposed that iteratively moves an element from its current cluster to the cluster that minimizes the cost until a local minimum is reached [8].

A simple corollary of the observation that $\mathcal{L}_1$ is minimized by hard co-clusters is obtained by setting $\mathbb{P} = \operatorname{diag} \boldsymbol{\mu}^{(2)}$: If one wants to restrict the alphabet of a RV while preserving most of its information, i.e., if one is looking for a $\mathbb{W} \in \mathcal{M}_{N_2 \times M_2}$ that maximizes $I(X_2; Y_2)$, then our results in [13] shows that hard clustering is optimal.

More interestingly, we can rewrite the cost function that Dhillon et al. proposed in [9] for information-theoretic co-clustering (ITCC) and obtain

$$\mathcal{L}_{\text{ITCC}}(\Phi, \Psi) := I(X;Y) - I(\overline{X};\overline{Y}) = \mathcal{L}_{\frac{1}{2}}(\Phi, \Psi). \tag{3.13}$$

Thus, ITCC is a special case of $\mathcal{L}_\beta$ for $\beta = \frac{1}{2}$. The authors of [9] proposed a sequential algorithm, similar to sIB, alternating between optimizing Φ and Ψ. Furthermore, $\mathcal{L}_{\text{ITCC}}(\Phi, \Psi)$ can be optimized via non-negative matrix tri-factorization [9, Lemma 2.1] and thus yields a generative model as a result. We are not aware if a similar connection to generative models holds for other values of β.

In [10], the cost function $\mathcal{L}_{\frac{1}{2}}$ is generalized to pairwise interactions of multiple variables (the two-dimensional case is equivalent to co-clustering). The authors introduce a multilevel heuristic that schedules the splitting of clusters, merges clusters following the ideas of aIB [7], and optimizes intermediate results sequentially with sIB.

The authors of [7] proposed applying aIB twice to obtain the co-clustering. In the first step, in which the set $\mathcal{X}$ is clustered, they treat Y as the relevant variable; in the second step, in which the set $\mathcal{Y}$ is clustered, they treat the clustered variable $\overline{X}$ as relevant. In essence, the authors of [7] thus minimize the functional

$$\mathcal{L}_{\text{IB-double}}(\Phi, \Psi) = L_Y(X \to \overline{X}) + L_{\overline{X}}(Y \to \overline{Y}) = \mathcal{L}_{\frac{1}{2}}(\Phi, \Psi) \tag{3.14}$$

in a greedy manner: They first optimize over Φ to minimize only the first term and then optimize over Ψ to minimize the second term. Comparing (3.13) and (3.14) reveals that [7] and [9] optimize the same cost function; the different performance results reported in [7] and [9] can only be explained by differences in the optimization heuristic and (possibly) preprocessing steps. We will elaborate on this topic in our experiments with the Newsgroup20 dataset in Section 3.9.1.

For $\beta = \frac{1}{2}$, we also showed in [13] that hard co-clusters are optimal. This also implies that, for $\beta = 1$ and $\beta = 12$, $\mathcal{L}_\beta$ is not a suitable choice for a cost function if one aims to look for soft co-clustering.

Another approach related to IB, called information bottleneck co-clustering (IBCC), was proposed in [11]. The functional being maximized by IBCC is

$$\mathcal{L}_{\text{IBCC}}(\Phi, \Psi) := I(X;\overline{Y}) + I(\overline{X};Y) + I(\overline{X};\overline{Y}) \tag{3.15}$$

$$= 3I(X;Y) - 2\mathcal{L}_{\frac{3}{4}}(\Phi, \Psi). \tag{3.16}$$

Hence, optimizing IBCC is equivalent to optimizing $\mathcal{L}_{\frac{3}{4}}$. The authors of [11] propose two algorithms: One is agglomerative, i.e., a greedy merging algorithm, the other is an iterative update of fixed-point equations in the spirit of [35].

Finally, for $\beta = 0$ we obtain the functional

$$\mathcal{L}_0(\Phi, \Psi) = L_{\overline{X}}(Y \to \overline{Y}) + L_{\overline{Y}}(X \to \overline{X}). \tag{3.17}$$

As previously mentioned, for Markov aggregation and $\beta = 0$ the cost function is linked to the phenomenon of lumpability. In the co-clustering framework, lumpability means that the two clustering solutions are coupled. More precisely, we have $\mathcal{L}_0(\Phi, \Psi) = 0$ if the rows X and columns Y do not share more information with the column clusters $\overline{Y}$ and row clusters $\overline{X}$, respectively, than the row clusters and column clusters share with each other. Unfortunately, we also have $\mathcal{L}_0(\Phi, \Psi) = 0$ if $\overline{X}$ and $\overline{Y}$ are independent, which suggests an inherent drawback of $\mathcal{L}_0$ for co-clustering (despite its justification in Markov aggregation as discussed in Sec. 2.4). This leads to $\mathcal{L}_0$ (and, in general, $\mathcal{L}_\beta$ for small β) having multiple bad local optima in which any heuristic tends to get stuck.

3.7. Strengths and Limitations of Generalized Information-Theoretic Co-Clustering

In this section we use examples and experiments on synthetic datasets to highlight different aspects of using $\mathcal{L}_\beta$ and our proposed optimization heuristic for co-clustering. Specifically, we will point at limitations and strengths of co-clustering in comparison with one-sided clustering ($\beta = 1$), which leads to guiding principles for the choosing β depending on characteristics of the considered dataset.

3.7.1. Examples

In the previous section we mentioned that an inherent shortcoming of $\mathcal{L}_0$ is that it leads to co-clusterings with (near-)independent cluster RVs. In this subsection, we point at further limitations of information-theoretic cost functions for co-clustering. These shortcomings are independent of the employed optimization heuristic, but rather reflect that in some scenarios not even the global optimum of the cost function coincides with the ground truth (or an otherwise desired co-clustering solution). Sometimes this is simply because that the cost function does not fit the underlying model – e.g., if $\mathbb{W}$ is generated according to a Poisson latent block model, then maximizing the likelihood of the co-clustering is equivalent to minimizing $\mathcal{L}_{\frac{1}{2}}$ only if the clusters have all the same cardinality [46, Sec. 2.2]. In contrast, the following two scenarios make no assumptions about an underlying model but illustrate shortcomings inherent to the considered information-theoretic cost functions.

Largely Different $|\overline{\mathcal{X}}|$ and $|\overline{\mathcal{Y}}|$:

An advantage of information-theoretic co-clustering approaches over, e.g., spectral [43, 45] or certain block model-based approaches [46] is that the former admit different cardinalities for the clustered sets $|\overline{\mathcal{X}}|$ and $|\overline{\mathcal{Y}}|$. If, however, these cardinalities differ greatly, then minimizing $\mathcal{L}_\beta$ becomes problematic especially for small values of β. Suppose w.l.o.g. that $|\overline{\mathcal{Y}}| < |\overline{\mathcal{X}}|$. Then, the optimization term $L_{\overline{Y}}(X \to \overline{X})$ is limited by the information contained in $\overline{Y}$ rather than by the information loss induced by clustering X to $\overline{X}$; many functions Φ may bring $L_{\overline{Y}}(X \to \overline{X})$ close to zero simply because $\overline{Y}$ itself already contains little information. Similarly, the term $L_{\overline{X}}(Y \to \overline{Y})$ may be large for many choices of Φ, because, again, the limiting factor is the coarse clustering from Y to $\overline{Y}$. These terms in (3.11) get more important if β is small. In other words, coupled co-clustering fails because the clustered variables contain little information. We illustrate this with a particular example, in which the joint probability distribution between X and Y is

$$P_{X,Y} = \left[\begin{array}{cc|cc} 0.125 & 0 & 0 & 0 \\ \hline 0.125 & 0 & 0 & 0 \\ \hline\hline 0 & 0.125 & 0 & 0 \\ 0 & 0.125 & 0 & 0 \\ \hline\hline 0 & 0 & 0.125 & 0 \\ 0 & 0 & 0.125 & 0 \\ \hline 0 & 0 & 0 & 0.125 \\ \hline 0 & 0 & 0 & 0.125 \end{array}\right].$$

Our aim is to obtain a co-clustering with $|\overline{\mathcal{Y}}| = 2$ and $|\overline{\mathcal{X}}| = 4$. In $P_{X,Y}$, the thick vertical line indicates one possibility for Ψ (a plausible ground truth). The horizontal lines indicate two possible options, Φ_1 (thick lines) and Φ_2 (thin lines) for the row clustering, where Φ_1 corresponds to a plausible ground truth.

For $\beta = 1$, (Φ_1, Ψ) has a lower cost than (Φ_2, Ψ), as desired. Furthermore, one can show that (Φ_1, Ψ) minimizes the cost function; $\mathcal{L}_1$ has its global minimum at the ground truth. For $\beta = \frac{1}{2}$, by evaluating $I(\overline{X}; \overline{Y})$ we see that both (Φ_1, Ψ) and (Φ_2, Ψ) have the same cost. In fact, any row clustering function Φ that shares the cluster boundary with the thick horizontal line in the middle has the same $I(\overline{X}; \overline{Y})$ for the given column clustering function Ψ: In this case, $\overline{X}$ determines $\overline{Y}$, hence we achieve the maximum $I(\overline{X}; \overline{Y}) = H(\overline{Y}) = 1$; the cost function has multiple global minima, only one of which lies at the ground truth. Finally, for $\beta = 0$, (Φ_1, Ψ) has a higher cost than (Φ_2, Ψ). This implies that even if we initialize our algorithm at the ground truth (this could be the case if we do β-annealing) we move away from this clustering solution when we optimize the cost function for smaller values of β.

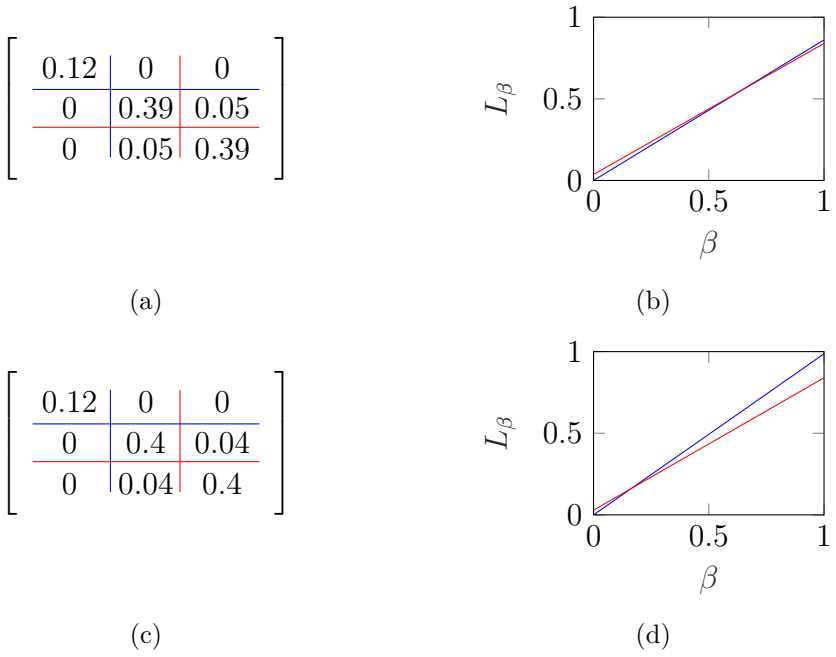


Figure 3.1.: Trading entropy for conditional entropy. (a) and (c) show joint distributions $P_{X,Y}$ together with two possible co-clusterings, while (b) and (d) show the corresponding values of the cost function for different values of β. Blue and red curves in (b) and (d) correspond to co-clusterings indicated by blue and red lines in (a) and (c).

Trading Entropy for Conditional Entropy:

Consider the joint distribution in Fig. 3.1(a) that describes a dataset with a well-separated co-cluster structure for $|\overline{\mathcal{X}}| = |\overline{\mathcal{Y}}| = 2$ (based on zeros and indicated by blue lines, denoted by $(\Phi^\bullet, \Psi^\bullet)$). We evaluate our cost function for different values of β, both for $(\Phi^\bullet, \Psi^\bullet)$ and for an alternative co-clustering indicated by red lines, denoted by (Φ, Ψ). It can be seen in Fig. 3.1(b) that, for $\beta \in [0.65, 1]$, we have $\mathcal{L}_\beta(\Phi^\bullet, \Psi^\bullet) > \mathcal{L}_\beta(\Phi, \Psi)$, i.e., the "incorrect" solution has a lower cost than the ground truth. While in this case, e.g., ITCC [9] would probably terminate with $(\Phi^\bullet, \Psi^\bullet)$, it is easy to construct an example where ITCC fails. Changing our example only slightly leads to generalized information-theoretic co-clustering preferring (Φ, Ψ) over $(\Phi^\bullet, \Psi^\bullet)$ for all β in $[0.15, 1]$ (see Figs. 3.1(c) and 3.1(d)).

These examples show that even for datasets with a well-separated co-cluster structure, for a range of β there can be (local and global) minima having a lower cost $\mathcal{L}_\beta$ than the ground truth. This is because optimizing the cost function for a given value of β boils down to maximizing/minimizing a combination of several mutual information terms. For example, for $\beta = \frac{1}{2}$ we aim to maximize (cf. (3.13))

$$I(\overline{X}; \overline{Y}) = H(\overline{X}) - H(\overline{X}|\overline{Y}). \tag{3.18}$$

This leads to two competing goals: entropy maximization (preferring clusters with roughly equal probabilities) and conditional entropy minimization (e.g., preferring row clusters that determine column clusters, and vice-versa). For the range of β where $\mathcal{L}_\beta(\Phi^\bullet, \Psi^\bullet)$ is not the global minimum, the first goal outweighs the second.

For joint distributions with a well-separated co-cluster structure we have $\mathcal{L}_0(\Phi^\bullet, \Psi^\bullet) = 0$ since $I(X;\overline{Y}) = I(\overline{X};Y) = I(\overline{X};\overline{Y})$. Nevertheless, due to the shortcoming discussed in Section 3.6, this global optimum may not be found because many other co-clusterings lead to $\mathcal{L}_0(\Phi, \Psi) \approx 0$.

3.7.2. Synthetic Datasets

Next, we perform experiments with two different synthetic datasets to explore further the relation between suitable choices of β and the characteristics of the dataset. Since our focus is on providing a better understanding of information-theoretic co-clustering, we assume that the true numbers of clusters and the true clustering functions $\Phi^\bullet$ and $\Psi^\bullet$ are known. As an accuracy measure, we employ the micro-averaged precision.

$$\mathrm{MAP}(\Phi, \Phi^\bullet) := \max_{\pi} \frac{\sum_{j\in\overline{\mathcal{X}}} |\Phi^{-1}(j) \cap \Phi^{\bullet -1}(\pi(j))|}{|\mathcal{X}|} \tag{3.19}$$

where the maximization is over all permutations π of the set $\overline{\mathcal{X}}$. $\mathrm{MAP}(\Psi, \Psi^\bullet)$ is computed along the same lines. Note that $\mathrm{MAP}(\cdot,\cdot)$ requires that the clustering solution found by the algorithm has the same number of clusters as are present in the ground truth. Since we assume the true number of clusters to be known, this is unproblematic. If the number of clusters is unknown, one can resort to more sophisticated measures such as the adjusted Rand index or normalized mutual information. In the present case, all of these measures will lead to similar qualitative results.

Unless noted otherwise, we set $\mathrm{tol} = 0$, $\#\mathrm{iter}_{\max} = 20$, and $\Delta = 0.1$ and ran AnnITCC for values of β between 0 and 1 in steps of 0.1. The simulation code for these and the real-world experiments in Section 3.9 is publicly accessible.[1]

The first experiment looks at the clustering performance in the presence of noise. We generated a joint probability distribution $T_{X,Y}$ with 80 rows and 50 columns, i.e., $|\mathcal{X}| = 80$ and $|\mathcal{Y}| = 50$, and planted co-clusters such that $T_{X,Y}$ is constant within each co-cluster. A colorplot of $T_{X,Y}$ is shown in Fig. 3.2(a). The figure also shows the ground truth $\Phi^\bullet$ ($|\overline{\mathcal{X}}| = 5$) and $\Psi^\bullet$ ($|\overline{\mathcal{Y}}| = 3$). We moreover constructed a random probability distribution N and defined

$$P_{X,Y} = (1-\varepsilon)T_{X,Y} + \varepsilon N \tag{3.20}$$

where $\varepsilon \in \{0, 0.5, 0.7, 0.8\}$. Colorplots of $P_{X,Y}$ are shown in Fig. 3.2(b) and 3.2(c) for $\varepsilon = 0.5$ and $\varepsilon = 0.8$, respectively. We repeated the procedure for 500 different probability matrices N. The MAP values, averaged over these 500 runs, are reported in Fig. 3.2(d) and 3.2(e) (solid lines).

[1] bitbucket.org/bernhard_geiger/coclustering_markovaggregation

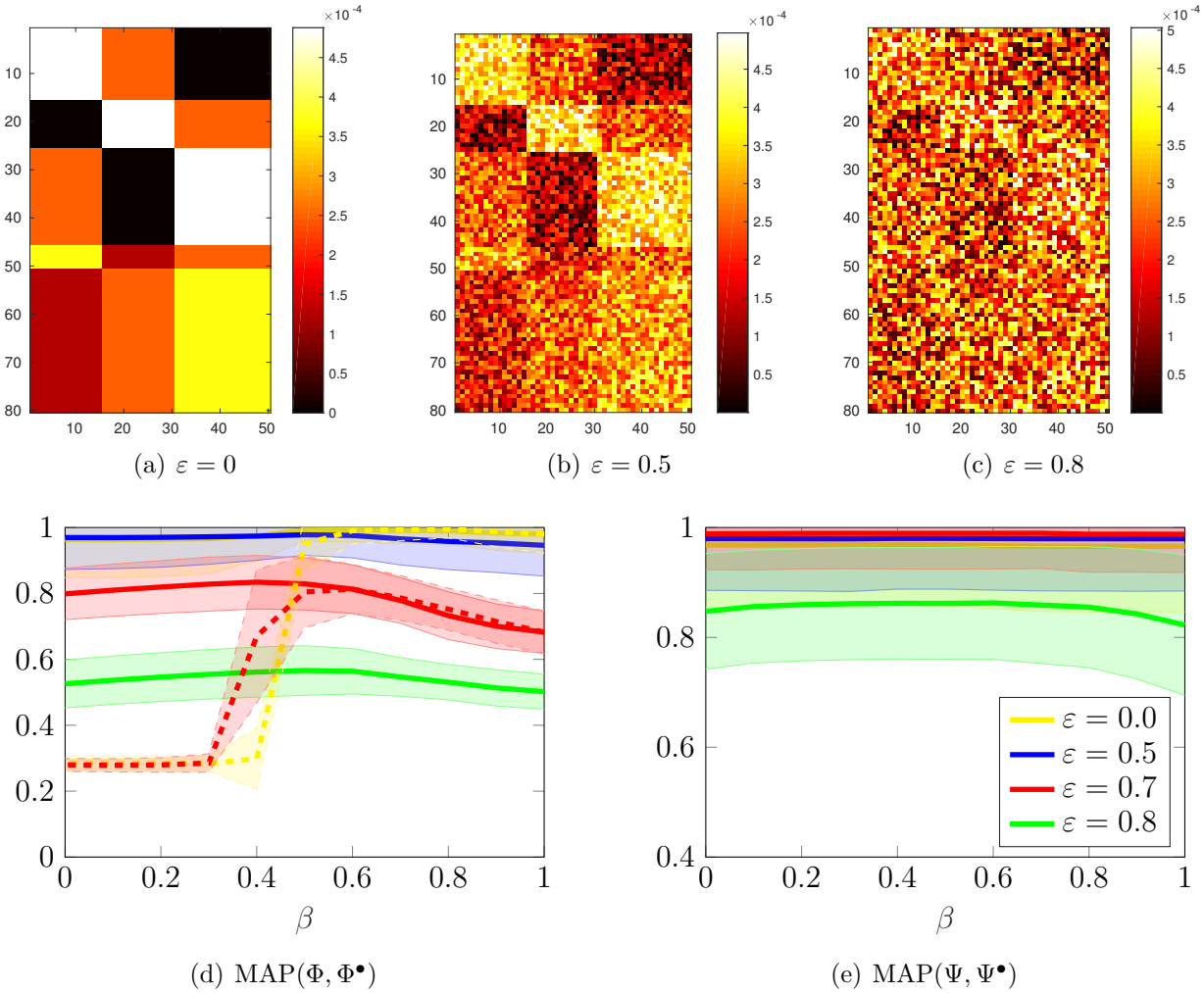


Figure 3.2.: (a)-(c): Colorplots of $P_{X,Y}$ for different noise levels ε. It can be seen that the true cluster structure becomes less obvious with increasing noise levels. (d), (e): Micro-averaged precision curves show the average over 500 random experiments (center line) and the standard deviation (shaded area). Solid curves correspond to AnnITCC, dashed curves to sGITCC. See text for details.

The results show that, even in the noiseless case, the clusters are not always identified correctly. Since we identified the correct clusters in over 90% of the simulation runs, we believe that this effect can be explained by the algorithm getting stuck in a local optimum.

Second, one can observe the expected effect that large noise levels lead to lower MAP values – interestingly, though, co-clustering seems to be robust to noise, as the MAP values in this experiment seem to decrease significantly only for $\varepsilon > 0.5$, i.e., when noise starts to dominate the data matrix. Finally, for large noise levels, it turns out that the intermediate values of β perform better. The performance drop for larger values of β is not due to

the optimization heuristic getting stuck in bad local optima: we found that the cost of the co-clustering solution found by AnnITCC for large β is lower than the cost of the ground truth. Rather, the reason is that for $\beta = 1$ the clustering solutions are uncoupled, i.e., the relevant RV for clustering rows is the noisy column RV. For a certain amount of coupling, i.e., for intermediate values of β, the relevant RV for clustering rows is more strongly related to the column *clusters*, in which noise is reduced due to the averaging effect of clustering. Performance drops again when decreasing β further; the reason is the inherent shortcoming of $\mathcal{L}_0(\Phi, \Psi)$ which is discussed at the end of Section 3.6.

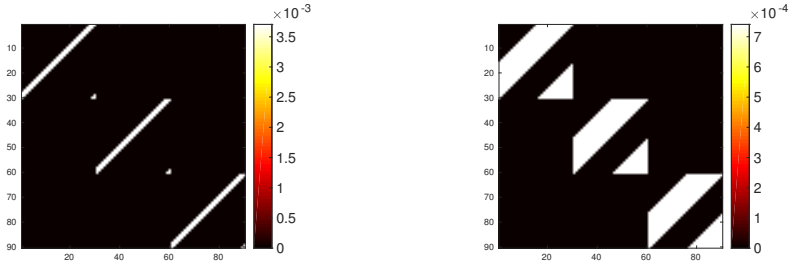
The second experiment investigates the effect of intra-cluster coupling between X and Y. We choose $|\mathcal{X}| = |\mathcal{Y}| = 90$ and $|\overline{\mathcal{X}}| = |\overline{\mathcal{Y}}| = 3$, to avoid the effects discussed in the examples presented earlier in this section and generate a joint probability distribution

$$P_{X,Y} = \begin{bmatrix} \mathbf{C} & 0 & 0 \\ 0 & \mathbf{C} & 0 \\ 0 & 0 & \mathbf{C} \end{bmatrix} \tag{3.21}$$

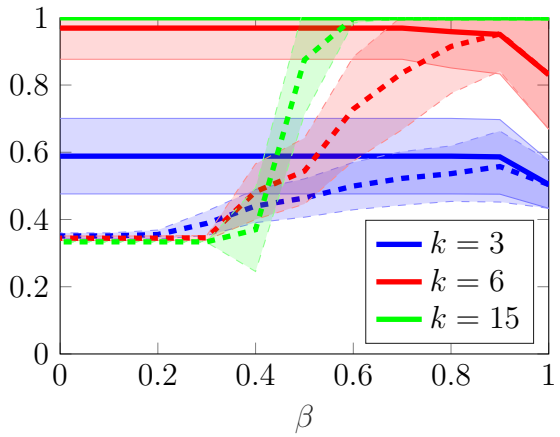
where $\mathbf{C}$ is a 30×30 circulant matrix, the first row of which consists of $30-k$ zeros followed by k entries equal to $\frac{1}{k|\mathcal{X}|}$. Each subsequent row of $\mathbf{C}$ is obtained by a circular shift of the previous row. Fig. 3.3(a) and Fig. 3.3(b) show $P_{X,Y}$ for $k = 3$ and $k = 15$, respectively. The ground truth co-clustering is given by the block structure of $P_{X,Y}$.

As k decreases, the intra-cluster coupling between X and Y increases. To see this note that, for $k = 30$, X does not contain more information about Y than the ground truth cluster $\overline{X}$ does, whereas for $k = 1$, X specifies Y uniquely. Fig. 3.3(c) shows the average MAP values obtained by running AnnITCC 500 times with random initializations. Since the experimental setup is symmetric we show the results only for Φ. First, we observe that with decreasing k the performance deteriorates. This is intuitive considering that with decreasing k the clustering structure becomes less obvious. For $k = 30$, $P_{X,Y}$ is uniform in the the blocks whereas for $k = 1$, the colums of $P_{X,Y}$ can be reordered such that $P_{X,Y}$ is a diagonal matrix with no clear co-clustering structure. Second, $\beta = 1$ does not lead to the best results for increased coupling, despite the fact that the global optimum of $\mathcal{L}_1$ coincides with the ground truth. Apparently, the optimization heuristic tends to terminate in poor local optima more often for $\beta = 1$ than for smaller values of β. This is because for $\beta = 1$ the two clustering solutions are decoupled, i.e., Φ and Ψ are determined independent of each other, while smaller β explicitly assumes coupled clusterings. We thus conclude that smaller values of β detect intra-cluster coupled co-clusters more robustly.

Finally, for both synthetic datasets, the MAP curves are relatively flat in many scenarios. One may think that this is due to AnnITCC getting stuck in a local optimum for a certain β, which it is not able to escape from for the subsequent lower β values. This is not the case: Figs. 3.3(c) and 3.2(d) show that the results obtained by running sGITCC (dotted lines) are almost identical to those obtained from AnnITCC for larger values of β until both of them reach the peak performance. Subsequently, for smaller values of β, the performance of sGITCC dropped significantly due to the reasons outlined at the end of



(a) $k = 3$ (b) $k = 15$



(c) MAP($\Phi, \Phi^\bullet$)

Figure 3.3.: (a)-(c), (f)-(g): Colorplots of $P_{X,Y}$ for different noise levels ε and different parameters k. The true cluster structure becomes less obvious with increasing noise levels. (d), (e), (h): Micro-averaged precision curves show the average over 500 random experiments (center line) and the standard deviation (shaded area). Solid curves correspond to AnnITCC, dashed curves to sGITCC. See text for details.

Section 3.6, justifying using AnnITCC for these values of β.

3.8. Guiding Principles for Choosing β

Although we do not propose a heuristic to find the suitable value (or range) of β for a given dataset, the examples and experiments discussed in Sec. 3.7 provide the following guiding principles:

- ▷ For large differences between target cardinalities $|\overline{\mathcal{X}}|$ and $|\overline{\mathcal{Y}}|$, larger values of β may lead to better results due to the increasingly decoupled nature of the cost function for increasing β.
- ▷ For datasets with highly imbalanced (co-)clusters, smaller values of β are more suitable (but only when one can manage to avoid optimization issues linked to smaller

values of β).

- ▷ In general, co-clustering using $\mathcal{L}_\beta$ and β-annealing seems to be robust to noise. For large noise levels, however, intermediate values of β tend to perform better due to noise averaging.
- ▷ In the presence of intra-cluster coupling, local optima of $\mathcal{L}_\beta$ are more prominent for β close to 1. The correct co-clusterings are found more robustly for intermediate values of β.

3.9. Real World Experiments

3.9.1. Document Classification by Co-Clustering of Words and Documents - Newsgroup20 Data Set

Dataset, Preprocessing, and Simulation Settings

The Newsgroup20 (NG20) dataset[2] consists of approximately 18800 documents containing 50000 different words. In this section, we evaluate co-clustering performance only via document clusters since there is no ground truth for word clusters. Nevertheless, word clustering was claimed to improve the document clustering performance, cf. [7, 9].

We refer to the RV over words as X, the set of words as $\mathcal{X}$, the RV over the documents as Y, and the set of documents as $\mathcal{Y}$. The respective clustered RVs and sets are denoted by an overline. The joint distribution of X and Y is obtained by normalizing the contingency table (counting the number of times a word appears in a document) to a probability distribution. During preprocessing, we removed newsgroup-identifying headers and lowered upper-case letters. We moreover reduced $\mathcal{X}$ to the 2000 words with the highest contribution to $I(X;Y)$, which is consistent with the preprocessing in [7–9]. Finally, we constructed various subsets of the NG20 dataset by randomly selecting 500 documents evenly distributed among the document classes. An overview of the used document classes used for each experiment is given in Table 3.9.1. Note that there are significant differences in the preprocessing steps performed in previous studies. For example, [8] included the newsgroup-identifying header, which may improve clustering performance. We ran AnnITCC with tol $= 10^{-3}$, $\Delta = 0.05$ and $\#\text{iter}_{\max} = 20$. For initialization, we slightly changed line 3 in Algorithm 3.2: Instead of running sGITCC with $\beta = 1$, which is equivalent to the completely decoupled case, we run sIB for both the word and document clusterings separately, where 25 restarts are performed and the best result w.r.t. the cost function is taken. Since there is no ground truth available for the word clusters, we executed AnnITCC for $|\overline{\mathcal{W}}| \in \{2, 4, 8, 16, 32, 64, 128\}$. This is consistent with the simulation settings described in [9], for example.

For a fair comparison of different values of β, we do not apply further heuristics to improve the performance of AnnITCC. In contrast, the authors of [9] initialize their

[2] qwone.com/~jason/20Newsgroups

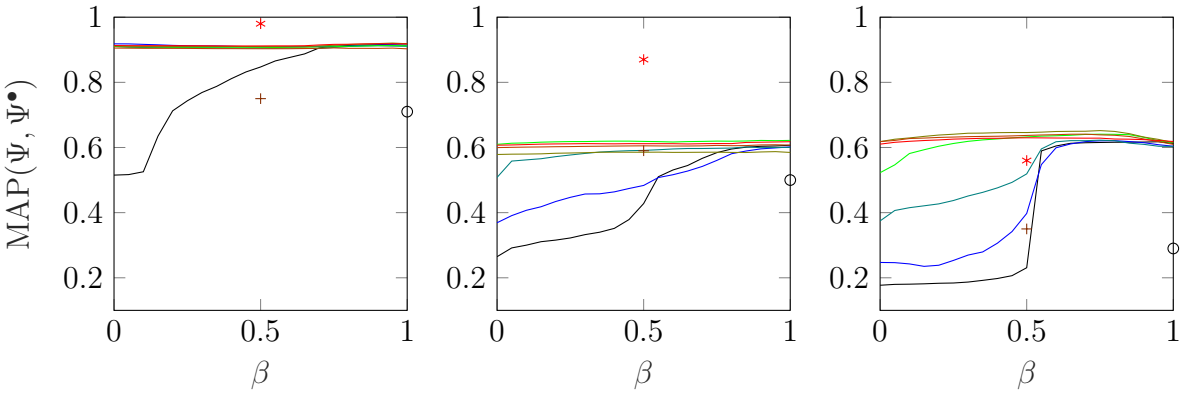
Dataset	Discussion Groups	$\frac{\text{docs}}{\text{class}}$	$\lvert\mathcal{D}\rvert$
Binary	talk.politics.mideast, talk.politics.misc	250	500
Multi5	rec.motorcycles, comp.graphics, sci.space, rec.sport.basketball, talk.politics.mideast	100	500
Multi10	comp.sys.mac.hardware, misc.forsale, rec.autos, talks.politics.gun, sci.med, alt.atheism, sci.crypt, sci.space, sci.electronics, rec.sport.hockey	50	500

Table 3.1.: Overview of the different subsets drawn from NG20

co-clustering algorithm for $|\overline{\mathcal{W}}|$ word clusters with the result obtained for $|\overline{\mathcal{W}}|/2$ word clusters, where each word cluster is split randomly. In [10], the authors introduce an additional correction parameter which leads to clusters of approximately the same size (which matches the evenly distributed classes in the NG20 dataset). Therefore, even for those values of β for which we obtain the same cost functions, our results need not be equal to those reported in the literature.

Results and Comparison

The results obtained by AnnITCC - averaged over 20 runs - for the different subsets of NG20 are visualized in Fig. 3.4. As can be seen, AnnITCC can discover the true document labels with high accuracy. For the Binary dataset, AnnITCC was able to achieve a micro-averaged precision of approximately 90%, for the Multi5 dataset 60% and for the Multi10 dataset approximately $60 - 65\%$. In comparison, experiments with sGITCC confirm the observations made in Sec. 3.6 and Sec. 3.7 that small $\beta \in [0, 0.4]$ lead to meaningless results in the range of random clustering, while high $\beta \in [0.6, 1]$ produce results in the range of Fig. 3.4. Fig. 3.4 further shows that the stronger the word and document clustering solutions are coupled, the worse are the results for small numbers of word clusters. This is most obvious for the Multi10 dataset for $\overline{\mathcal{W}} \in \{2, 4, 8\}$ word clusters, where the MAP values increase sharply if β increases from 0.4 to 0.6 (see Fig. 3.4(c)). For small β, the document clusters are obtained from the word clusters and, e.g., two word clusters do not contain sufficient information to distinguish between ten document clusters. This agrees with our discussion in Section 3.7.1. However, for very large $|\overline{\mathcal{W}}|$, there were no further improvements. This suggests that there exists a number of word clusters that suffice to achieve the same (or better, see below) performance as document clustering based on words.



(a) Binary (b) Multi5 (c) Multi10

Figure 3.4.: Micro-averaged precision for different NG20 subsets and AnnITCC.Different curves in a plot are for different numbers of word clusters, $|\overline{\mathcal{W}}| = \{2, 4, 8, 16, 32, 64, 128\}$ (black for fewest, blue to green to red increasing number of clusters). For comparison, we added results reported in the literature. $(*)$: Taken from [9, Table 5]; $|\overline{\mathcal{W}}|$ is unclear. $(+, \circ)$: Taken from [7, Table 3]; the best results for each dataset are displayed. These results were obtained by applying aIB for different numbers of word clusters, $|\overline{\mathcal{W}}| = \{10, 20, 30, 40, 50\}$; the displayed MAP values are averages of the individual MAP values. We were not able to compare our results to those of [11] because they used different classes of the NG20 dataset. Since the cost functions from the literature are the same as ours for the respective values of β, the difference in the performance can only be attributed to preprocessing steps, the optimization heuristics, and/or the choice of favorable data subsets.

One major issue observed from Fig. 3.4 is that for the Binary and Multi5 data, the results are almost independent of β (for sufficiently many word clusters). Only for Multi10 there was a mild increase in performance for intermediate values of β. This confirms the observations from Section 3.7.2: clustering words removes noise, hence document clustering based on word clusters may be slightly more robust than document clustering based on words. Nevertheless, since the effect is small for Multi10 (and not present for Binary and Multi5), we doubt that co-clustering of words and documents is significantly superior to one-sided document clustering w.r.t. the classification results. The classification results from [10] point to similar conclusions, since also there sIB performed very well compared to the respective co-clustering methods. Still, the authors of [7,9,11] claim that their proposed algorithms and/or cost functions for co-clustering outperform one-sided clustering. We instead suggest that, for this dataset, the choice of the cost function has less effect on the performance reported in literature than algorithmic details, preprocessing steps, and additional heuristics for, e.g., initialization.

3.9.2. MovieLens100k

Dataset, Preprocessing, and Simulation Settings

The MovieLens100k dataset[3] consists of 100000 ratings of 1682 movies by 943 users [53]. The user ratings take integer values 1 (worst) to 5 (best). We construct a user-movie matrix $\mathbb{R} := [R_{ij}]$ where R_{ij} is the rating user i gave to the movie j ($R_{ij} = 0$ if user i did not rate movie j). Note that $\mathbb{R}$ is a sparse matrix with only 100000 out of approximately 1.59 million entries being nonzero.

We refer to the RV over the users as X, the set of users as $\mathcal{X}$, the RV over movies as Y, and the set of movies as $\mathcal{Y}$. The respective clustered RVs and sets are denoted by an overline. The joint distribution between X and Y is obtained by normalizing $\mathbb{R}$ to a probability distribution.

For initializing AnnITCC we ran sGITCC 25 times with random initializations for $\beta = 1$ with $\text{tol} = 10^{-3}$ and $\#\text{iter}_{\max} = 20$. We chose the best co-clustering (Φ, Ψ) among these 25 restarts w.r.t. the cost and used this as the initialization for AnnITCC. We ran AnnITCC with $\text{tol} = 10^{-3}$, $\Delta = 0.1$ and $\#\text{iter}_{\max} = 20$. We defined 10 user clusters, i.e., $|\overline{\mathcal{U}}| = 10$, as was done in [51, 52]. Furthermore, we defined $|\overline{\mathcal{M}}| = 19$ since the MovieLens100k dataset categorizes the movies into 19 different genres.

Evaluation Metrics

Evaluating co-clustering performance for the MovieLens100k dataset is difficult. The authors of [52] proposed to assess co-clustering performance based on recommendations, i.e., a portion of the dataset is used for co-clustering, based on which the "taste" (i.e., if a user will rate a movie above 3 or not) of the users is predicted. The remaining portion of the dataset (i.e., the validation set) is used to assess this prediction. We believe that such an approach is not effective. Indeed, the available ratings in $\mathbb{R}$ are skewed in the sense that approximately 82.5% of the ratings are above 3. Hence, a naive recommendation system suggesting a positive rating for every user-movie pair in the validation set matches the user's taste with approximately 82.5%. In comparison, the authors of [52] claim a match of 89% for their approach.

A second option is to compare the co-clustering results to a plausible ground truth. For the users, demographic information is available which theoretically admits constructing such a ground truth; we nevertheless refrain from doing so, since no choice can be justified without evoking critique. For the movies, genre information is available which lends itself to evaluating movie clusters. However, not every movie is assigned to a unique genre, but may belong to multiple genres. The ground truth $\Psi^\bullet$ is therefore not a function, but a distribution over the set of genres $\overline{\mathcal{Y}}$. This is problematic for (3.19), which is why we

[3] grouplens.org/datasets/movielens/100k

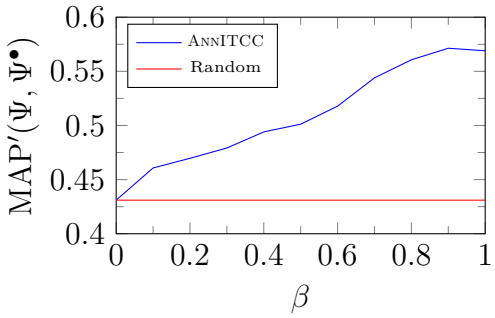


Figure 3.5.: AnnITCC performance for movie genre matching

replace it here by

$$\mathrm{MAP}'(\Psi, \Psi^\bullet) := \frac{1}{|\mathcal{Y}|} \sum_{j \in \overline{\mathcal{Y}}} \max_{i \in \overline{\mathcal{Y}}} |\Psi^{-1}(j) \cap \Psi^{\bullet -1}(i)|. \tag{3.22}$$

For each movie cluster, we look for the genre with which this cluster has the greatest overlap. Unlike for MAP, two different clusters can now be mapped to same movie genre in MAP′. Hence, MAP′, sometimes referred to as *purity*, is essentially the average of the fraction of movies in each cluster that belong to the same genre. As a side result, MAP′ gets rid of the maximum over all permutations π, which is intractable for large numbers of genres.

Results

The results are shown in Fig. 3.5. First, note that the MAP′ value for randomly generated clusters is remarkably high. This is because the number of movies in different genres varies greatly; for example, 725 movies are assigned to genre "Drama" and 505 to genre "Comedy", whereas only 24 movies belong to the genre "Film-Noir". Noting this, quantitative results based on movie genres are useful to observe trends and general behavior, but the numbers should be taken with a grain of salt. The maximum value for MAP′ in Fig. 3.5 is significantly smaller than 1. This is reasonable since co-clustering is based on a sparse matrix of user-movie rating pairs: While some users are *genre-addicts*, rating movies mainly based on their genre, other users may rate movies based on aspects unrelated to genre. Hence, one cannot expect a value MAP′ = 1 for co-clustering based on user-movie rating pairs.

We observe that MAP′ generally decreases with decreasing β and the maximum value is at $\beta = 0.9$, albeit only slightly larger than for $\beta = 1$. This shows that our algorithm is capable of outperforming ITCC ($\beta = \frac{1}{2}$), IBCC ($\beta = \frac{3}{4}$), and (albeit only slightly) IB-based ($\beta = 1$) movie clustering. For β close to 0, we obtain results which are very close to what we obtain for randomly generated movie clusters. A closer analysis revealed that the solution found for $\beta = 0$ has a lower cost than the solution found for $\beta = 1$, which means

that β-annealing was successful in escaping bad local optima, but that the genre-based "ground truth" does not coincide with the global optimum of the cost function for $\beta = 0$. We believe that, in this particular example, this phenomenon is linked to the user-movie rating matrix $\mathbf{R}$ being sparse.

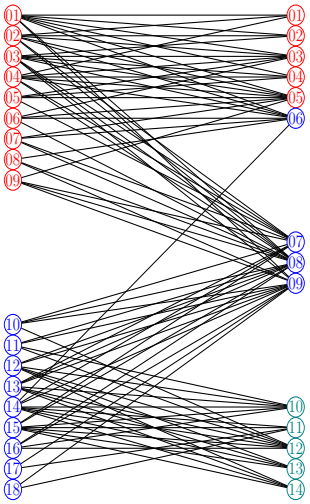
We finally complement this quantitative evaluation by a qualitative evaluation of the movie clusters. Again, we observe meaningful results for higher values of β when compared to smaller values of β. For example, looking at movie clusters for $\beta = 0.9$, we notice that many classics are clustered into one group, including *Gone With The Wind*, *Breakfast at Tiffany's (1961)*, *12 Angry Men*, *The Graduate*, *The Bridge on River Kwai*, *Citizen Kane*, *Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb*, *Vertigo*, *Casablanca*, *His Girl Friday (1940)*, *A Street Car Named Desire*, *It Happened One Night*, *The Great Dictator*, *The Great Escape*, *Philadelphia Story*. Similarly, many animated/kids movies have been assigned to a cluster, including *The Lion King*, *Alladin*, *Snow White and the Seven Dwarfs*, *Homeward Bound*, *Pinocchio*, *Turbo: A Power Rangers Movie*, *Mighty Morphin Power Rangers: The Movie*, *Cinderella*, *Alice in Wonderland (1951)*, *Dumbo (1941)*, *Beauty and the Beast*, *Winnie the Pooh and the Blustery Day*, *The Jungle Book*, *The Fox and the Hound*, *Parent Trap*, *Jumanji*, *Casper*, etc. Furthermore, our approach clustered various sequences of movies, e.g., 6 out of 8 Star Trek movies and all 7 Amityville movies have been assigned to one cluster each. In contrast, the results for $\beta = 0$ did not yield clusters one would consider meaningful.

3.9.3. Community Detection in Bipartite Graphs

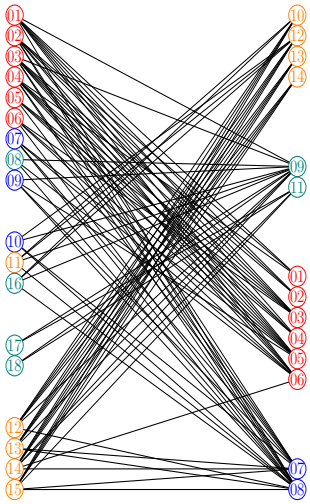
Community detection is a common problem in social network analysis and is usually concerned with (random) unipartite graphs, see [54]. In this section, we look at the related problem for bipartite graphs. There, the two sets of vertices could be the characters and the scenes of a play, and the goal could be to group characters in a meaningful way.

We apply our algorithm to the Southern Women Event Participation Dataset [48, 54]. The dataset consists of 18 women ($|\mathcal{X}| = 18$) and 14 events ($|\mathcal{Y}| = 14$), and the weight matrix $\mathbb{W}$ contains a one if the corresponding woman participated in the corresponding event and a zero otherwise. We restarted AnnITCC 50 times for $\beta = 1$ to obtain a good initial co-clustering for the annealing process. To get results comparable to those in the literature, we chose $|\overline{\mathcal{X}}| = 2, |\overline{\mathcal{Y}}| = 3$ and $|\overline{\mathcal{X}}| = |\overline{\mathcal{Y}}| = 4$. The results are displayed in Fig. 3.6 for $\beta = 0.7$.

The two women communities we obtained match with those communities reported in the literature [49, 54]. The authors of [49] also clustered the events into three clusters: The events are clustered into a group in which only women of the first women community participated, a group in which only women of the second women community participated, and a group in which women from both communities participated. Our result in Fig. 3.6(a) is remarkably similar to theirs, with the exception that the event with label 6 is put in a different group. Note, however, that in this event only one woman of the opposite community participated. Remarkably, we obtained the same co-clustering for all values of β.



(a) Two women communities (left), three event clusters (right). Colors taken from [49].



(b) Four women communities (left), four event clusters (right). Colors taken from [48].

Figure 3.6.: Community Structure of the Southern Women Event Participation Dataset. The separation between nodes indicates the clustering obtained from AnnITCC with $\beta = 0.7$, the color of the nodes is taken from reference clusterings from the literature.

For four women communities and four event clusters, we compared our results with those of Barber [48], who employed a modularity-based approach. Our event clusters in Fig. 3.6(b) are identical to those of [48], and our women communities are largely consistent. We found in a separate set of experiments that the women communities show a greater agreement for $\beta = 1$, and less agreement for $\beta = \frac{1}{2}$; the MAP values for the chosen value of $\beta = 0.7$ lie in between. Thus, community detection via ITCC can be outperformed by our algorithm for larger values of β.

Part II.

Cluster Analysis via Factor Graphs

4

Cluster Analysis with Simultaneous Global and Local View of Data

In this chapter we propose a new clustering algorithm which utilizes the mathematical framework of factor graphs. We show the efficacy of this algorithm in discovering global structure in data as well as in providing useful local information about clusters.

Cluster analysis, also known as clustering, is one of the most important tools in exploratory data analysis and unsupervised learning. It refers to the task of grouping data points so that the points belonging to the same group are more similar to one another than to the points belonging to other groups. The data points in cluster analysis can represent a wide variety of objects depending upon the application. For example each data point may represent a document (in document clustering) or a customer (in market research) or a gene (in genomics). Beyond this notion there are no universally accepted definitions of what a "cluster" means. The "goodness" of obtained results is determined by the human analyst's interpretation and the effective application of these results for further decision making. Furthermore there is no standard format of providing input information to the clustering algorithms about the data points. Due to these two reasons there is a plethora of clustering algorithms that accept input information in different formats and generate outputs conforming to different notions of what a cluster should be.

Clustering has wide spread applications. In business intelligence and market research it has been used, among other purposes, for market segmentation [55]. In biology it has been used to group similar genes [56]. In social sciences cluster analysis has been applied to discover community structures [57] and for detecting crime hotspots [58]. There are many more domains where cluster analysis has been successfully applied. Furthermore, even in many supervised machine learning tasks, cluster analysis is used as a preprocessing step.

In this chapter we focus on pairwise similarities between data points as the input information to the clustering algorithm. Note that graph clustering is a special case of this scenario as one can interpret the weights on the edges between the nodes (aka data points)

Algorithm	Optimization Aspects		Global View		Local View	
	Requires No of Clusters	Complexity	Global Structure Discovery	Outlier Detection	Exemplars	Local Structure
Desired	No	Low	Yes	Yes	Yes	Yes
PAM	Yes	$\mathcal{O}(\lvert\mathcal{X}\rvert^2)$	No	No	Yes	No
AP	No	$\mathcal{O}(\lvert\mathcal{X}\rvert^2)$	No	Yes	Yes	No
MCL	No	$\mathcal{O}(\lvert\mathcal{X}\rvert^3)$	Yes	Yes	No	No
DBSCAN	No	$\mathcal{O}(\lvert\mathcal{X}\rvert^2)$	$\sim$	$\sim$	No	No
Spectral	Yes	$\mathcal{O}(\lvert\mathcal{X}\rvert^3)$	$\sim$	No	No	No
Heirarchical	No	$\mathcal{O}(\lvert\mathcal{X}\rvert^2 \log \lvert\mathcal{X}\rvert)$	$\sim$	$\sim$	No	$\sim$
EAP	No	$\mathcal{O}(\lvert\mathcal{X}\rvert^2)$	Yes	Yes	Yes	Yes

Table 4.1.: Qualitative comparison of famous pairwise similarity based clustering algorithms w.r.t. various desired characteristics. $|\mathcal{X}|$ denotes the cardinality of the dataset and $\sim$ implies that the algorithm is not too good but also not bad for the desired trait.

as a specification of the pairwise similarity between the nodes. Furthermore, we focus mainly on hard clustering, i.e., we assign each data point to only one cluster rather than providing a vector with probabilities of a data point belonging to different clusters, as is the case in soft clustering. We do, however, show how our proposed algorithm can efficiently indicate the confidence level for each data point about joining different discovered clusters. This information can be used to transform the proposed algorithm into a soft clustering technique, but we do not explore this topic further in this work.

Table 4.1 outlines the properties that we believe are desired in a clustering algorithm from the perspective of an analyst. We will discuss these and other desired traits in more detail in Sec. 4.2. In Table 4.1 we also provide a qualitative analysis of the well known pairwise similarity based algorithms w.r.t. to these traits. Different algorithms have different strengths and shortcomings. Markov Clustering (MCL) [14] can recognize clusters, but it suffers from high computational complexity. On the other hand DBSCAN [15] has only quadratic complexity and can also identify clusters of various shapes, but it suffers from the possibility of not allocating many data points to any cluster. MCL and DBSCAN also do not provide any local view of the data. Affinity Propagation (AP) has been proposed as an alternative to K-medoids (PAM) which does not require the number of clusters as an explicit input. Like K-Medoids (PAM), AP also provides exemplars for clusters which reveal meaningful information about the typical characteristics of the data points in a cluster. However, as is the case for PAM, AP is restricted to discovering only globular clusters, which severely limits its applicability.

In this work we modify AP to develop a new clustering technique that incorporates the desired features mentioned in Table 4.1. We choose AP as our starting point due the

following two reasons:

- ▷ AP already possesses many desirable properties such as no need for a (hard) specification of the number of clusters, no initialization issues and low complexity.
- ▷ The mathematical framework of factor graphs and message passing, on which AP is based, is very flexible and provides a natural way to incorporate new information and requirements into the framework.

We start the chapter by formulating the clustering problem in Sec. 4.1. In Sec. 4.2 we present some characteristic that, we believe, are desired in a clustering algorithm. In Sec. 4.4 we introduce the original Affinity propagation algorithm (AP). Sec. 4.5 briefly discusses other extensions to AP. In Sec. 4.6 we propose the new algorithm called Extened Affinity Propagation (EAP) and in Sec. 4.7 we discuss it's different characteristics in detail. Sec. 4.8 elaborates on the hyperparameters involved in EAP and how they can be efficiently tuned for a given dataset. Sec. 4.9 and Sec. 4.10 conclude the chapter by discussing the effective application of EAP to synthetic and real datasets, respectively.

This work was done in close collaboration with Rayyan Ahmad Khan.

4.1. Problem Formulation

We are given a set of data points, denoted by $\mathcal{X} = \{x_1, x_2, \ldots, x_{|\mathcal{X}|}\}$ and a pairwise similarity matrix $\mathbb{S}$ where $\mathbb{S}(i,j) \in \mathbb{R}$ denotes the similarity of point x_i to point x_j. We use the shorthand notation $s_{ij} = \mathbb{S}(i,j)$ for convenience. Note that we do not require $\mathbb{S}$ to be symmetric or impose any other constraints on $\mathbb{S}$ beyond the obvious assumption that a data point exhibits higher pairwise similarity to another "closer" data point when compared to a data point that is further away. The range of values s_{ij} is allowed to take depends on the particular clustering algorithm that processes $\mathbb{S}$.

Cluster analysis deals with the problem of finding a suitable mapping $\ell : \mathcal{X} \to \overline{\mathcal{X}}$ where $|\overline{\mathcal{X}}| < |\mathcal{X}|$. The actual alphabet $\overline{\mathcal{X}}$ is not important for clustering, only $|\overline{\mathcal{X}}|$ is important. Depending upon the clustering algorithm $|\overline{\mathcal{X}}|$ may need to be explicitly specified by the user or it may be automatically estimated by the clustering algorithm. We define $c_i = \ell(x_i)$ to denote the cluster to which the data point x_i belongs to. Hence $c_i \in \overline{\mathcal{X}}$ for all $i \in \{1, 2, \ldots, |\mathcal{X}|\}$. Clustering algorithms normally solve the problem by defining a suitable cost function C, involving pairwise similarities s_{ij} and cluster assignments c_i, and then optimizing the cost function w.r.t. the cluster assignment:

$$[c^*] = \underset{[c]}{\operatorname{argmax}}\, \mathrm{C}\left([c], \mathbb{S}\right) \tag{4.1}$$

where the notation $[c]$ represents the set of variables c_i for all $i \in \{1, \ldots, |\mathcal{X}|\}$. For most suitable cost functions used in cluster analysis, the optimization defined in (4.1) is computationally too complex. Hence, often the clustering algorithm consists of a heuristic trying to find a "good" sub-optimal solution to the optimization problem with acceptable computational complexity.

Graph clustering [59], namely the task of clustering the nodes of a (directed) graph based on the weighted edge based interactions between pairs of nodes, can be considered an equivalent formulation of the pairwise clustering problem described above where data points are replaced by nodes and pairwise similarities are replaced by weighted edges between node pairs in the graph. Hence a clustering technique developed for the problem stated above is automatically applicable to graph clustering problems and vice versa.

The choice of how pairwise similarities are computed between data points is one of the most important factors that influences the effectiveness of the clustering results and this is true for any clustering algorithm. However this choice is highly application specific. It depends on the type of data encountered as well as on the ultimate goal of the cluster analysis. On the other hand a good clustering algorithm should be able to discover the hidden structure in $\mathbb{S}$ regardless of how the pairwise similarities were computed. Hence we separate the task of designing a suitable application specific pairwise similarity metric from the task of designing an effective clustering algorithm which should be application agnostic. In this chapter we assume that we are provided with a suitable $\mathbb{S}$ which captures the structure in the underlying dataset and our aim is to design an algorithm that can discover this hidden structure in $\mathbb{S}$.

4.2. Desired Traits in a Clustering Algorithm

In this section we elaborate on the desired properties that we are interested to see in a clustering algorithm, some of which were mentioned in Table. 4.1. These properties are mainly related to three different aspects: the ability to discover global structure of the dataset, the ability to provide useful local information for each cluster and the ease and complexity of optimization. The following characterization of desired traits is by no means a universally accepted list. Furthermore, depending on the use case some of these characteristics may not be important whereas some other characteristics not discussed here may be of interest.

4.2.1. Global View

Global Structure Discovery: The main task of cluster analysis, as discussed in Sec. 4.1, is to discover global structure in the provided dataset. Hence it is important that a clustering algorithm can discover clusters of a wide variety of "shapes". Algorithms such as K-means, K-medoids (PAM) and AP lack this property as they only discover globular clusters. On the other end of the spectrum, MCL is normally good at discovering clusters of a wide variety of shapes. We use globular cluster to refer to clusters having shapes analogous to spherical clusters in euclidean space and star shaped graph components (when the edges of the graph have lengths that are inversely proportional to pairwise similarities).

Outlier Detection and Robustness to Noise: Another useful characteristic in a clustering algorithm is the ability to identify outliers, i.e., data points that clearly do not belong to any cluster. Furthermore, the algorithm should minimize the effect of the outliers on the rest of the clustering results. Similarly, the clustering algorithm should be able to handle noisy datasets with possibly blurry separation between clusters in small subregions of the clusters.

4.2.2. Local view

These traits relate to the information captured by a clustering algorithm about the internal structure of each discovered cluster.

Local Exemplars and Subclusters: Exemplars constitute a subset of the dataset, where each exemplar represents the typical characteristics of a (sub)cluster. Both K-medoids (PAM) and AP provide such exemplars, one for each cluster. Since AP and PAM are restricted to discovering globular clusters, one exemplar per cluster may be enough to represent the typical characteristics of the cluster, but in the case of more complicated cluster structures, multiple exemplars may be needed per cluster where each exemplar represents the typical characteristics of only a surrounding subregion of the cluster. In this case we call them local exemplars. The subregion associated to any local exemplar can be thought of as a subcluster.

Local Structure of Each Cluster: Besides local exemplars and associated subclusters, the algorithm may also provide other useful information about the internal structure of each cluster. Some examples of such information are an indicator of relative densities of different clusters or how strongly neighbouring subregions of a cluster are connected with one another.

Indication of Potential Inconsistencies: An important aspect of local information, which can also be used to improve the global structure discovery, is the ability of an algorithm to indicate the data points or subregions for which the algorithm is not confident about the cluster assignment. If the number of such points is small enough, these can be dealt with manually by an analyst. Such inconsistencies can also be studied further to recognize if they are caused by the way pairwise similarity is computed or if they are caused by the algorithm and the gained insights can be used to improve either. It is also possible that these inconsistencies are due to inherent uncertainty and noise in the dataset.

4.2.3. Algorithmic and Optimization Aspects

Number of Clusters: In most practical scenarios we are a priori unaware of the number of clusters present in a dataset and may only have a vague estimate. Hence it is beneficial if the clustering algorithm does not require the exact number of clusters as an input, but only a soft estimate to provide initial guidance to the algorithm.

Hyperparameter Tuning: Every clustering algorithm requires hyperparameters that need to be provided and tuned by an analyst or another heuristic. For example DBSCAN requires two parameters, minimum number of neighbourhood points and the neighbourhood size. AP requires the self preference parameter which is a soft indicator of the number of clusters. For a good clustering algorithm it is important that the hyperparameters can be tuned relatively easily to fit to a wide range of datasets and discover suitable results.

Low Complexity: One of the most important aspects of a clustering algorithm is that it has an acceptable computational complexity. We use the Big O notation [60] to present asymptotic complexity results.

Flexibility: An often ignored, but highly useful, property in practical applications is the flexibility to easily incorporate additional a priori information and constraints in the existing algorithm.

The characterization of most of these traits is vague. This is because, as mentioned earlier, cluster analysis itself is not a precisely defined problem. Different algorithms may strive to fulfill these traits in different ways. The main aim of this section was to define a broad characterization of the desired characteristics so that the readers can understand, in a broader sense, what these traits mean when they are referred to in the following sections.

4.3. Preliminaries

We encourage the readers to check out the brief refreshers on factor graphs and Max-Product algorithm (MP) in Appendix B.1 and B.2 before moving to the next sections. These appendices introduce the appropriate notation and important concepts needed in the following sections.

4.4. Affinity Propagation

In this section we review AP before proposing our modifications in Sec. 4.6. AP was introduced in [16]. It can be seen as an alternative optimization procedure to PAM [61] for solving the K-medoids exemplar based clustering problem. In K-medoids based clustering, each cluster is represented using an exemplar. The exemplar itself is a data point belonging to the respective cluster. AP has some clear advantages over PAM as we will see later. Given the dataset $\mathcal{X}$ and the pairwise similarity matrix $\mathbb{S}$, the aim is to find exemplars and cluster assignments such that the product of pairwise similarities between the data points and the exemplars associated with the respective clusters assigned to the data points is maximized:

$$\underset{[c]}{\operatorname{argmax}} \prod_i s_{i,c_i} \quad s.t. \quad c_{c_j} = c_j \ \forall j \tag{4.2}$$

Hence the cost function in this case is

$$\mathrm{C}\left([c], \mathbb{S}\right) = \prod_i s_{i,c_i} \tag{4.3}$$

Instead of dealing with the product in (4.2) we convert the optimization problem into log domain to deal with the sum:

$$\underset{[c]}{\operatorname{argmax}} \left(\prod_i s_{i,c_i} \right) = \underset{[c]}{\operatorname{argmax}} \sum_i \bar{s}_{i,c_i} \quad s.t. \quad c_{c_j} = c_j \ \forall j \tag{4.4}$$

where $\bar{s}_{i,c_i} = \log s_{i,c_i}$ and c_i refers to the exemplar selected by the data point x_i, which also determines the cluster that the data point belongs to since each cluster has only one exemplar. Note that the s_{ij} need to be positive for (4.2) to be well behaved and (4.4) to be well defined, therefore leading to $\bar{s}_{ij}$ being any real number. Beyond this we do not impose any constraints on the pairwise similarities s_{ij} for $i \neq j$. s_{jj} represents the preference of data point x_j to become an exemplar. It is not computed the same way as the other similarity values s_{ij} for $i \neq j$, rather it is considered a hyperparameter that needs to be specified by the analyst. In case of no additional a priori information s_{jj} is assigned the same value p for all points where p is a parameter of the algorithm. p then represents the initial desire of each point to become an exemplar. Hence in AP, initially each datapoint is treated as a potential exemplar and there are no initialization issues like the ones faced in PAM, K-means and other exemplar (or centroid) based algorithms. In the absence of any additional information p is normally set to the median of $\mathbb{S}$ [16]. One has to be careful

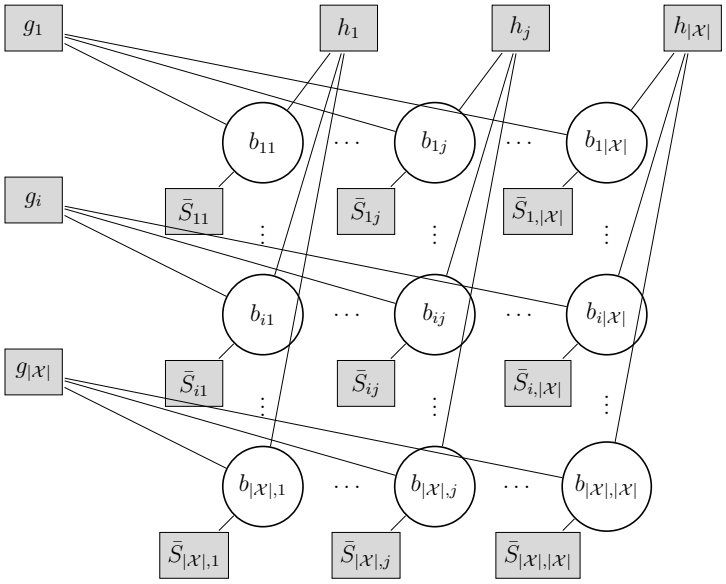


Figure 4.1.: Factor graph of (4.6) [63].

not to set p to a too high value, for example if $p > s_{ij}$ for all $i \neq j$, then (4.2) leads to a trivial solution of every point declaring itself as an exemplar and forming a single point cluster.

The constraint $c_{c_j} = c_j$, known as the consistency constraint, forces that if a data point x_j is chosen as exemplar by some other point(s), it must choose itself as it's exemplar too, hence promoting compact clusters. In the absence of this constraint, it is possible for every data point to choose any other data point as it's representative c_i and therefore every point will end up choosing the neighbour with the highest pairwise similarity as it's representative. The resulting solution will not be a good clustering solution and it will not lead to exemplars, each of which represents the typical characteristics of a subset of the dataset. Hence this consistency constraint is important to obtain a meaningful clustering result. Unlike conventional K-medoids optimization, we do not need to specify the number of clusters $|\overline{\mathcal{X}}|$ before solving the optimization problem. The self preference p in conjunction with the self consistency constraint, which motivates compact clusters, acts as a soft initial guidance for AP to determine the number of clusters needed for the given dataset.

(4.2) is an NP-Hard combinatorial optimization problem [62]. AP solves it sub-optimally in $\mathcal{O}(|\mathcal{X}|^2)$ computations using MS. For this purpose we present the reformulated version of AP, described in [63], involving binary optimization variables: Let us define a matrix $\mathbb{B} \in \{0,1\}^{|\mathcal{X}|\times|\mathcal{X}|}$ of binary variables $b_{ij} = \mathbb{B}(i,j)$, where $b_{ij} = 1 \iff c_i = j$. If we define $\mathbb{B}(i,:)$ to represent the ith row of $\mathbb{B}$ then $\mathbb{B}(i,:)$ can be considered as a one hot encoding of c_i. Similarly we use $\mathbb{B}(:,i)$ to represent the ith column in $\mathbb{B}$. Since every data point can choose only one exemplar, the objective function in (4.4) can be written as:

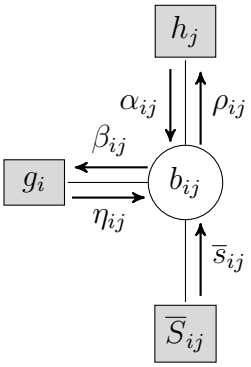


Figure 4.2.: Message passing in factor graph of Fig. 4.1 [63].

$$\underset{\mathbb{B}}{\operatorname{argmax}} \left(\sum_{i,j} \bar{s}_{ij} b_{ij} \right) \quad s.t. \quad \begin{cases} \sum\limits_{j} b_{ij} = 1 & \forall i \\ b_{jj} = \max\limits_{i} b_{ij} & \forall j. \end{cases} \tag{4.5}$$

We can reformulate (4.5) as the following unconstrained optimization problem:

$$\underset{\mathbb{B}}{\operatorname{argmax}} \left(\sum_{i,j} \bar{s}_{ij} b_{ij} + \sum_{i} g_i(\mathbb{B}(i,:)) + \sum_{j} h_j(\mathbb{B}(:,j)) \right) \tag{4.6}$$

where

$$g_i(\mathbb{B}(i,:)) = \begin{cases} 0 & if \ \sum\limits_{j} b_{ij} = 1 \\ -\infty & otherwise \end{cases} \tag{4.7}$$

$$h_j(\mathbb{B}(:,j)) = \begin{cases} 0 & if \ b_{jj} = \max\limits_{i} b_{ij} \\ -\infty & otherwise \end{cases} \tag{4.8}$$

where h_j and g_i correspond to indicator functions for the optimization problem in the product domain. AP solves (4.6) by mapping it to the *factor graph* shown in Fig. 4.1 where the factor nodes $\overline{S}_{ij}$ correspond to $\bar{s}_{ij} b_{ij}$. It involves two steps: message passing and decision mechanism.

4.4.1. Message Passing

We can find a sub-optimal solution of (4.6) by applying Max-Sum algorithm (MS) to the factor graph in Fig. 4.1. The messages exchanged between the variable node b_{ij} and the factor nodes g_i and h_j are shown in Fig. 4.2, where we have used the following shorthand

notation for convenience:

$$\beta_{ij} = \mu_{b_{ij}\to g_i}(1) - \mu_{b_{ij}\to g_i}(0) \tag{4.9}$$
$$\eta_{ij} = \nu_{g_i\to b_{ij}}(1) - \nu_{g_i\to b_{ij}}(0) \tag{4.10}$$
$$\rho_{ij} = \mu_{b_{ij}\to h_j}(1) - \mu_{b_{ij}\to h_j}(0) \tag{4.11}$$
$$\alpha_{ij} = \nu_{h_j\to b_{ij}}(1) - \nu_{h_j\to b_{ij}}(0) \tag{4.12}$$

For efficiency, we only exchange a scalar instead of a vector, constituting the difference of the message values for $b_{ij} = 1$ and $b_{ij} = 0$. This is sufficient for clustering problem, as we are only interested in determining the maximizer. This difference represents the relative gain to the overall cost function according to a given variable or factor node for choosing $b_{ij} = 1$ instead of $b_{ij} = 0$, based on the current information available to the node. The final equations used for computing these difference messages are:

$$\beta_{ij} = \bar{s}_{ij} + \alpha_{ij} \tag{4.13}$$
$$\rho_{ij} = \bar{s}_{ij} + \eta_{ij} \tag{4.14}$$
$$\eta_{ij} = -\max_{k\neq j} \beta_{ik} \tag{4.15}$$
$$\alpha_{ij} = \begin{cases} \sum\limits_{k\neq j} max(0, \rho_{kj}) & i = j \\ \min[0, \rho_{jj} + \sum\limits_{k\notin\{i,j\}} \max(0, \rho_{kj})] & i \neq j \end{cases} \tag{4.16}$$

Since g_i and h_j represent constraints in the original optimization problem, we also refer to the corresponding factor nodes sometimes as constraint nodes.

The detailed derivations of (4.16) and (4.15) are given in [62]. η_{ij} and α_{ij} lend themselves to intuitive meanings and insights into the inner workings of AP. g_i constraint focuses on satisfying the mutually exclusive assignment of i to only one exemplar. This can be seen by looking at η_{ij}, which corresponds to the maximum penalty that would be incurred by one of the other b_{ik} being switched to 0 so that b_{ij} could be switched to 1, i.e., the penalty of assigning the point i to exemplar j instead of exemplar k.

For α_{ij} we have two cases. For $i = j$, α_{ii} basically represents the motivation provided by the local neighbourhood to the data point x_i to become an exemplar. We only consider the positive motivation provided by the local neighbourhood since we are interested in checking how many points are interested in declaring x_i as their exemplar but we do not care about how far are the other points which do not want to connect to x_i and are interested in declaring some other data point as their exemplar. For $i \neq j$ we basically pass to point x_i, how motivated x_j is to become an exemplar. If x_j is already motivated to be an exemplar because of other points in it's local neighbourood, x_i is free to choose x_j as a local exemplar, indicated by 0 penalty where as if x_j is not so motivated to be an exemplar, then x_i is discouraged to declare x_j as it's exemplar by the penalty $\rho_{jj} + \sum\limits_{k\notin\{i,j\}} \max(0, \rho_{kj}) < 0$.

4.4.2. Decision Phase

During the iterations, we can approximate the difference of the corresponding max-marginals (in the log domain) for the two values of b_{ij} via accumulated belief a_{ij} given by the sum of all incoming messages to b_{ij}, i.e., $a_{ij} = s_{ij} + \eta_{ij} + \alpha_{ij}$. This sum corresponds to the evaluation of (B.19) and then taking the difference for the two values of b_{ij}. Current cost maximizing value of b_{ij} is decided by thresholding a_{ij} at 0, i.e., $b_{ij} = 1$ if $a_{ij} > 0$ and $b_{ij} = 0$ otherwise. Let $\mathbb{A}$ be the matrix of accumulated beliefs a_{ij}. Convergence is achieved in AP if the values of diagonal elements of $\mathbb{A}$ do not change over a specified number of iterations. Once message passing converges, we choose the set of exemplars as follows: $\mathcal{E} = \{k \mid a_{kk} > 0\}$. Each non exemplar point is then assigned to the exemplar most similar to it [62]

$$c_i = \underset{k \in \mathcal{E}}{\operatorname{argmax}}\, s_{ik} \qquad \forall i \notin \mathcal{E}. \tag{4.17}$$

4.5. Related Work

Since it was first published in [16], AP has been modified in different ways. Hierarchical Affinity Propagation(HAP), introduced in [64], proposes a layered structure where the exemplars of previous optimization layer are considered as the data points for the next layer. HAP tries to cluster the data hierarchically without making hard decisions at each hierarchical layer. Although the local exemplars obtained so are more meaningful than AP, the clusters obtained by HAP are still globular at each layer and there is limited information about the local structure of the clusters beyond local exemplars. Multi-Exemplar Affinity Propagation (MEAP) [65] is another approach, closely related to HAP with two layers, where the authors propose the use of exemplars and super-exemplars. Exemplars can select super-exemplars as representatives but super-exemplars are forced to select themselves. MEAP has similar drawbacks as HAP.

Soft-Constraint Affinity Propagation(SCAP) [66] is another relevant extension. Unlike other variants, SCAP allows exemplars to select other exemplars as their representative by relaxing the consistency constraint. As a result SCAP can discover a wider variety of cluster shapes. On the other hand since SCAP tries to identify global structure based on the pairwise similarities between exemplars, this corresponds to expanding a cluster by establishing direct links between the local exemplars. Hence it often leads to sub-optimal clustering. Furthermore, the only local information available is the local exemplars and direct connections between them.

A more natural approach to identify arbitrarily shaped clusters is to combine subclusters corresponding to each local exemplar by exploring the connections between subcluster boundaries. This is the approach we will take in this work. It not only has better and more robust clustering results but also provides us with additional information about relative cluster densities and strength along with the local exemplars.

4.6. Extended Affinity Propagation (EAP)

AP possesses many of the desired traits mentioned in Sec. 4.2. It does not require any cluster initializations, hence making it impervious to initialization issues suffered by many other clustering algorithms. It only requires a soft estimate of the number of clusters fed as an input in the form of self preference p. The computational complexity of AP is $\mathcal{O}(|\mathcal{X}|^2)$. Furthermore it provides some local information about each cluster in the form of an exemplar.

Unfortunately AP also suffers from the following issues: It can only discover globular clusters which seriously limits it's applicability. Besides the only local information it provides about each of these globular clusters is an exemplar.

In this section we will modify AP to develop a new algorithm which inherits the positive aspects of AP while alleviating its' shortcomings. The fact that AP is developed based on the flexible framework of factor graphs and the aforementioned positive aspects make it a suitable candidate to modify and develop a new algorithm.

In order to include all the desired characteristics discussed in Sec. 4.2, we need to modify the cost function (consequently also the message passing phase) and the decision mechanism for AP. The basic principle driving our approach is to allow multiple local exemplars in a cluster and to permit a data point to connect to multiple local exemplars if it is close enough to them. These data points that form "boundary" connections between local exemplars by connecting to multiple local exemplars then enable the decision mechanism, which looks for connected components in a graph, to discover the wide variety global structures. Moreover, the local exemplars together with these boundary connections between them provide meaningful insights into the internal structure of a cluster as we will see in the later sections. We call our proposed algorithm Extended Affinity Propagation (EAP).

4.6.1. Modified Cost Function and Message Passing

Improved Global View: The $g_i(.)$ constraint in standard AP forces each data point to choose only one exemplar. Combined with this, the decision process described in Sec. 4.4.2 forces globular clusters. For better global structure discovery we allow each data point to connect to possibly multiple local exemplars and hence form boundary connections between local exemplars. This is achieved by modifying $g_i(\cdot)$ as follows:

$$\overline{g}_i(\mathbb{B}(i,:)) = \begin{cases} u_i(\mathbb{B}(i,:)) & \text{if } \sum_j b_{ij} \geq 1 \\ -\infty & \text{otherwise} \end{cases} \tag{4.18}$$

where

$$u(\mathbb{B}(i,:)) := \sum_j b_{ij} q \tag{4.19}$$

where q is a new hyperparameter denoting the penalty incurred when a data point connects to a new exemplar. This leads to the following unconstrained optimization

$$\underset{\mathbb{B}}{\operatorname{argmax}} \left(\sum_{i,j} b_{ij} \Big(\overline{s}_{ij} + q \Big) + \sum_{j} h_j(\mathbb{B}(:,j)) + \sum_{i} \log \Big(\mathbb{1} \Big(\sum_{j} b_{ij} \geq 1 \Big) \Big) \right). \tag{4.20}$$

Our choice of $u(.)$ is motivated by the following two reasons:

- ▷ It can be seen from (4.18) and (4.20) that a data point x_i is motivated to connect to a potential local exemplar x_j only if it has a strong enough pairwise similarity with it. $\overline{g}_i(.)$ penalizes the global cost function linearly with a penalty q for each potential local exemplar that x_i chooses to connect to. However q does not affect the selection of the first exemplar as this is still a hard requirement of $\overline{g}_i(.)$. Every data point is bound to select at least one potential local exemplar. So the new objective function is bound to have a minimum penalty of $q \times |\mathcal{X}|$. This addition of a constant will not affect our calculation of argmax but allows for a bit simpler message derivations with the same final outcome.

- ▷ The computations to determine all the outgoing messages from factor node $\overline{g}_i(\cdot)$ still have $\mathcal{O}(N)$ complexity, same as the factor node $g_i(.)$ in AP. The complexity of the algorithm thus stays $\mathcal{O}(|\mathcal{X}|^2)$. Unless chosen wisely, an arbitrary choice of $u(\cdot)$ may lead to message computations with exponential complexity in $|\mathcal{X}|$ [67]. Even for an $u(\cdot)$ that is a non-linear function depending only on the number of exemplars x_i has chosen, i.e., $\sum_j b_{ij}$, the complexity of messages can increase to $\mathcal{O}(|\mathcal{X}| \log |\mathcal{X}|)$, resulting in the overall algorithm complexity of $\mathcal{O}(N^2 log(N))$.

The outgoing messages from the new constraint node $\overline{g}_i(.)$ are

$$\eta_{ij} = \max \Big(- \max_{m \neq j} \beta_{im} \ , \ q \Big). \tag{4.21}$$

The derivation of (4.21) is given in Appendix B.3. From (4.21) it can be seen that the complexity of computing all the outgoing messages simultaneously remains the same as in the case of AP. We can also look at (4.21) from an intuitive perspective by comparing it with (4.15). As discussed in Sec. 4.4.1, for AP η_{ij} corresponds to the maximum penalty due to other potential local exemplars for data point x_i. In (4.21) this penalty is limited by q. Thus the negative effect of other potential local exemplars for i on b_{ij} being 1 is limited. This allows a point to connect to more than one potential local exemplar if it is "close" enough to them. We can recover the η_{ij} of AP by setting $q = -\infty$ (and ignoring the $q \times |\mathcal{X}|$ factor in optimization in order to avoid the optimization problem being ill-posed).

Improved Local View: The new relaxed constraint $\overline{g}_i(\cdot)$ along with the new decision mechanism that we will explain in Sec. 4.6.2 solves the problem of global structure discovery by allowing the subclusters to be merged together using the "boundary" connections.

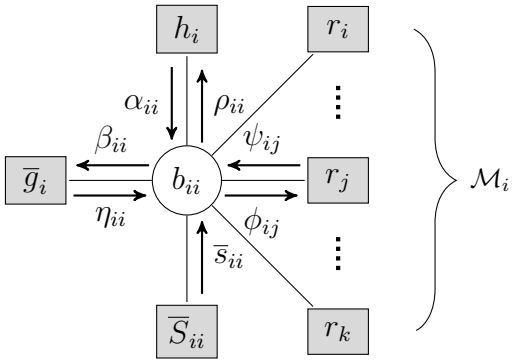
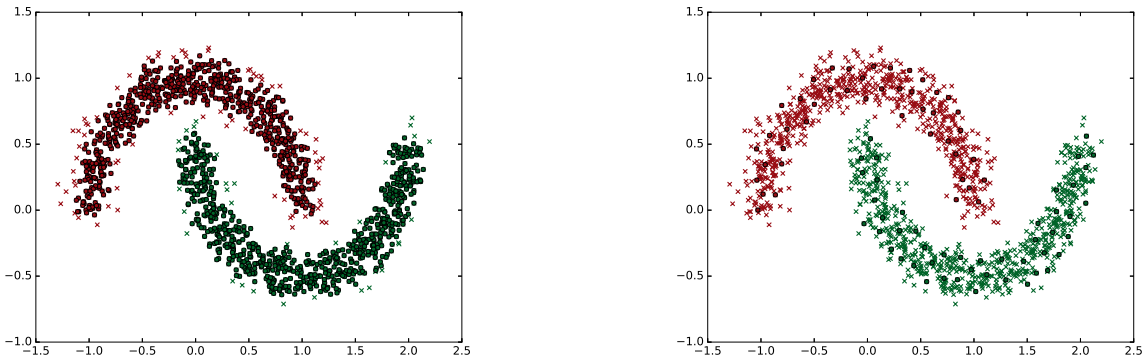


Figure 4.3.: Messages passed between a diagonal variable node b_{ii} and it's adjacent constraint nodes in EAP.

However, the relaxed constraint $\overline{g}_i(\cdot)$ obscures the local information. To understand why, suppose a dataset that lies in a metric space and the pairwise similarities depend inversely on the pairwise distances in the sense that an increase in the pairwise distance leads to a reduced pairwise similarity. In AP, if a point x_j is selected as an exemplar, the points close to x_j also have a high motivation of becoming an exemplar, during the message passing phase, as they also experience incoming messages from the local neighbourhood that are similar to incoming messages for x_j. However $g_i(\cdot)$ forces only one exemplar to appear and suppresses others. When g_i is relaxed to $\overline{g}_i$, the points close to a local exemplar also tend to become local exemplars. This leads to many local exemplars appearing very close by in dense regions of the dataset, which may obscure the local information. This phenomenon of close by local exemplars is shown in Fig. 4.4(a) where data points belonging to one cluster are marked with the same colour, circles show local exemplars and the remaining points are shown by the marker 'x'. In order to obtain well separated local exemplars again while retaining global structure discovery as well as $\mathcal{O}(|\mathcal{X}|^2)$ computational complexity, we introduce a new set of constraints on the diagonal elements of $\mathbb{B}$, i.e., the elements signifying potential local exemplars. For every data point x_j, let $\mathcal{N}_j = \{j\} \bigcup \{k | s_{jk} > \Delta\}$, i.e., the Δ-neighborhood around x_j. Well separated exemplars can be obtained by enforcing a maximum of one exemplar in each neighbourhood $\mathcal{N}_j$ by introducing the following constraint in (4.20) for each x_j.

$$r_j(\mathcal{N}_j) = \begin{cases} 0 & if \sum\limits_{k \in \mathcal{N}_j} b_{kk} \leq 1 \\ -\infty & otherwise. \end{cases} \tag{4.22}$$

Since a point x_j can belong to Δ-neighbourhood of multiple points, each b_{jj} can have multiple adjacent $r_k(\cdot)$ constraints. $\mathcal{M}_j = \{k | j \in \mathcal{N}_k\}$ denotes the set of all points whose Δ-neighbourhoods contain x_j. If x_j is a local exemplar then all the points in $\mathcal{N}_j \bigcup \mathcal{M}_j$ should not be local exemplars. Fig. 4.3 shows the messages exchanged between the diagonal variable nodes and the factor nodes for EAP where we defined the following shorthand



(a) Without neighbourhood constraint r_j

(b) With neighbourhood constraints r_j

Figure 4.4.: The figures show the impact of neighbourhood constraints, defined in (4.22), on the discovered local exemplars. It is also clear from the figure that the global structure discovery has not been impacted by the neighbourhood constraints. The two figures have been generated for the same $p = 0.6$ and $q = -0.97$, whereas the figure on the right has an $\Delta = 0.99$.

notation:

$$\phi_{ij} = \mu_{b_{ii}\to r_j}(1) - \mu_{b_{ii}\to r_j}(0) \tag{4.23}$$

$$\psi_{ij} = \nu_{r_j\to b_{ii}}(1) - \nu_{r_j\to b_{ii}}(0). \tag{4.24}$$

The final equation to compute ψ_{ij} is:

$$\psi_{ij} = -\max(0, \max_{\substack{l\in\mathcal{N}_i \\ l\neq i}} \phi_{lj}). \tag{4.25}$$

The derivation of (4.25) is given in Appendix B.4. The complexity of computing all the outgoing messages ψ_{ij} from r_j is $\mathcal{O}(|\mathcal{X}|)$, hence the overall complexity of the algorithm still stays $\mathcal{O}(|\mathcal{X}|^2)$. (4.25) provides an intuitive insight into the effect of constraint $r_j(\cdot)$. If we have more than one strong contenders for being a local exemplar in a neighborhood $\mathcal{N}_j$, they will all try to suppress one another. In such a scenario most of the potential local exemplars in the neighbourhood will get suppressed, leading to well separated local exemplars in the associated subregion. Fig. 4.4(b) shows impact of introducing the neighbourhood constraints when compared to Fig. 4.4(a).

The messages exchanged between the variable and the factor nodes for the new factor graph corresponding to EAP can be summarized as follows:

$$\beta_{ij} = s_{ij} + \alpha_{ij} + \mathbb{1}(i = j) \sum_{k\in\mathcal{M}_i} \psi_{ik} \tag{4.26}$$

$$\eta_{ij} = \max\left(-\max_{k\neq j}\beta_{ik}\ , q\right) \tag{4.27}$$

$$\phi_{ij} = s_{ii} + \alpha_{ii} + \eta_{ii} + \sum_{k\in\mathcal{M}_i\setminus j}\psi_{ij} \tag{4.28}$$

$$\psi_{ij} = -\max(0, \max_{\substack{l\in\mathcal{N}_i\\ l\neq i}}\phi_{lj}) \tag{4.29}$$

$$\rho_{ij} = s_{ij} + \eta_{ij} + \mathbb{1}(i=j)\sum_{k\in\mathcal{M}_i}\psi_{ij}. \tag{4.30}$$

$$\alpha_{ij} = \begin{cases} \sum\limits_{k\neq j} max(0,\rho_{kj}) & i=j \\ \min[0, \rho_{jj} + \sum\limits_{k\notin\{i,j\}}\max(0,\rho_{kj})] & i\neq j. \end{cases} \tag{4.31}$$

4.6.2. Decision Mechanism

Even with the modified cost function and messages, the decision mechanism specified in Sec. 4.4.2 will lead to globular clusters. The reason is because that (4.17) does not utilize the "boundary" connections, where some data points want to connect to more than one local exemplar. Rather it connects each data point only to the closest local exemplar only and considers the globular subcluster associated with each local exemplar as a separate cluster. Basically it ignores the information provided by the off diagonal elements of $\mathbb{A}$. Hence we need to modify the decision mechanism to better utilize the information present in the new $\mathbb{A}$. Like AP,after every iteration we check for the sum of all incoming messages to b_{ii}, i.e., $a_{ii} = s_{ii}+\eta_{ii}+\alpha_{ii}+\sum\limits_{k:i\in\mathcal{M}_i}\psi_{ik}$ to check for convergence. The decision phase that follows convergence of these messages is described in Alg. 4.1. Clusters are discovered by extracting connected components of the graph for adjacency matrix $\mathbb{B}_s$, the symmetrized version of $\mathbb{B}$ [68]. $\mathcal{E}$, $\mathbb{B}$ and $\mathcal{L}_i$ now contain local information about the clusters. The complexity of the decision phase is also upper bounded by $\mathcal{O}(|\mathcal{X}|^2)$.

4.6.3. Complexity

In Sec. 4.6.1 and Sec. 4.6.2, we have discussed the associated computational complexity with each modification. The complexity of none of the modifications exceeds $\mathcal{O}(|\mathcal{X}|^2)$. Hence the complexity of the overall algorithm also stays $\mathcal{O}(|\mathcal{X}|^2)$. Since we have not imposed any structural restrictions on the pairwise similarity matrix $\mathbb{S}$ (such as symmetry or sparsity etc), $\mathcal{O}(|\mathcal{X}|^2)$ is the best one can achieve. Like AP, for many problems the factor graph can be sparsified by reducing the number of variables involved in $\overline{g}$ and h constraints. g can be sparsified by considering that a data point will only consider to connect with potential local exemplars that are close enough, hence the corresponding $\overline{g}_i$ constraint may involve only other data points which are in a small enough neighbourhood of data point i based on the pairwise similarities. Similarly h_j constraints can be sparsified by considering that only those points that are near enough to the data point x_j would potentially consider to make x_j their local exemplar. For sparsifying neighbourhood constraints, we can check

Algorithm 4.1 EAP Decision Mechanism

function EAPdecision($\mathbb{A}$)
  $\mathbb{B} \leftarrow \mathbb{1}(\mathbb{A} > 0)$ ▷ Element-wise thresholding of $\mathbb{A}$
  $\mathcal{E} \leftarrow \{k \mid b_{kk} = 1\}$ ▷ Set of local exemplars
  for $j \notin \mathcal{E}$ **do** ▷ Ensure local consistency constraint
    $\mathbb{B}(:, j) \leftarrow 0$
  end for
  for $i \leftarrow 1$ to N **do**
    $\mathcal{L}_i \leftarrow \{k | k \in \mathcal{E}$ and $h_{ik} = 1\}$ ▷ Local exemplars connected to x_i
    if $\mathcal{L}_i == \emptyset$ **then** ▷ Isolated points
      $\mathcal{L}_i \leftarrow \{i\}$
      $b_{ii} \leftarrow 1$
    end if
  end for
  $\mathbb{B}_s \leftarrow \mathbb{B} \wedge \mathbb{B}^T$ ▷ Symmetrize $\mathbb{B}$
  $\ell(\cdot) \leftarrow$ ConnectedComponents($\mathbb{B}_s$) ▷ Discover connected components [68]
  return $\mathcal{E}$, $\mathbb{B}$, $\mathcal{L}$, ℓ
end function

that if a pair of data points x_i and x_k is involved in one neighbourhood constraint r_j, then we do not need to involved the same pair in another neighbourhood constraint.

Since we relax one set of hard constraints involved in AP, i.e., the g_i constraint, in practice this means that our algorithm requires less number of message iterations for the diagonal messages to converge because the decoupling in $\overline{g}_i$ avoids the equally strong cyclic negative messages that are exchanged in AP between two (or more) equally well suited potential exemplars for a data point.

4.7. Global and Local View of EAP

In this section we will highlight some important features of EAP. Many of these features are not present in AP, hence this also highlights the improvements EAP offers. Some of these features can also be associated with AP but are not handled as well by AP as EAP does. Furthermore, there are other features such as flexibility, reasonable complexity and the need for only a soft estimate of the number of clusters which are directly inherited from AP or the underlying framework of factor graphs and message passing algorithms. Hence they will not be discussed here.

4.7.1. Global Structure Discovery

One of the primary goals of modifying AP, as mentioned in Sec. 4.6, was to allow the discovery of clusters with widely varying characteristics such as various shapes and varying densities within the same dataset. In Sec. 4.9 we use synthetic datasets to show that EAP

is able to achieve this goal very effectively. In Sec. 4.10 we observe similar behaviour on real world datasets. Furthermore, as discussed in more detail in Sec. 4.8, we do not perform parameter sweeps over the hyperparameters of EAP to find these results (which would be in most practical cases computationally impractical and we will need some external information to choose the hyperparameters corresponding to the "best" results). Instead, the results presented from here onwards are obtained via successive tuning heuristic proposed in Sec. 4.8 unless stated otherwise. Therefore, one can expect to obtain similar performance in other practical applications of EAP to cluster other datasets without any additional knowledge beyond pairwise similarities. It is also interesting to note that for the datasets tested in Sec. 4.9 and Sec. 4.10, the values of the hyperparameters obtained via successive tuning, which correspond to the results shown lie in a narrow range for all the datasets. This implies that for a large variety of datasets we can find suitable hyperparamaters by tuning them in a narrow range, at least when using similar pairwise similarity metrics.

4.7.2. Local Exemplars

As opposed to AP, EAP provides multiple local exemplars per cluster. As explained in Sec. 4.6, r_j constraints lead to well separated local exemplars, each exemplar representing the typical characteristics of a subcluster. In Sec. 4.9 we will see how local exemplars are distributed among the dataset for different datasets whereas in Sec. 4.10 we will look at an example of what these local exemplars may represent for real world datasets.

The obtained local exemplars can be used to efficiently adapt the clustering results for evolving datasets. For example, they can be used to efficiently cluster a new data point x_{new} after you have already clustered the original dataset $\mathcal{X}$ using EAP. We can compare x_{new} to the already found local exemplars ($\mathcal{E}$) and assign it to the cluster which contains the closest local exemplar. This usually lowers the complexity by orders of magnitude when compared to assigning a cluster by finding the closest neighbour in the already clustered dataset, as would be the case for algorithms that do not provide any local information, and also provides further information as follows: If there are two or more local exemplars belonging to the same cluster which are almost equally close to the data point x_{new} we can connect x_{new} to all of them. If there are two exemplars belonging to different clusters that are almost equally close to x_{new}, then this new data point can be passed on to a human analyst as a potential inconsistency that needs to resolved by expert opinion. If the new data point has low enough similarity to all of the local exemplars, it can be treated as an outlier. We can apply this procedure to more than one new data points as well. In that case we should just be more careful that if we notice a sufficiently large number of new data points being declared as outliers this may be an indication of a new cluster being formed which was not present in the original clustering of $\mathcal{X}$ by EAP. Also if we notice enough data points having close enough similarities to two exemplars from different clusters, they may form a strong enough bridge between the two clusters to merge them to one. In these cases it is a good idea to either recluster whole dataset or a partial subset of the dataset.

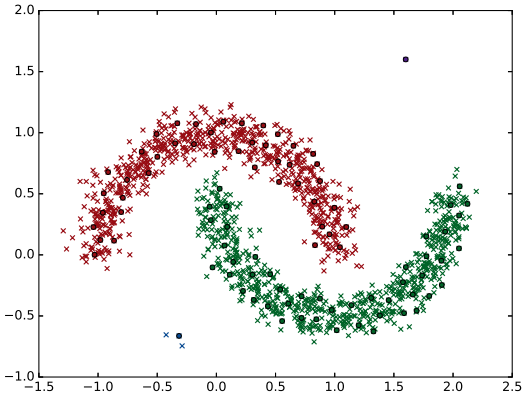


Figure 4.5.: Half-moons dataset with outliers. These results are obtained using the same parameters as Fig. 4.4(b), i.e., $p = 0.6$, $q = -0.97$ and $\Delta = 0.99$.

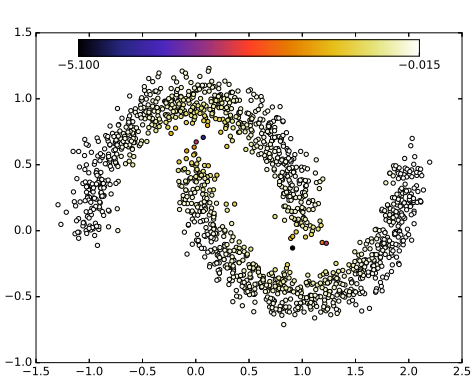
4.7.3. Outliers and Noisy Data

EAP starts by considering all data points as potential local exemplars and then explores the neighbourhood of each data point to later decide if a data point should become a local exemplar or not. This provides EAP the ability to discover outliers as they do not tend to connect to other local exemplars that belong to well formed bigger far away clusters. This is depicted in Fig. 4.5, where we again have the half-moons dataset with 2 clusters and we have now added some outliers to it. Specifically, we have a lone outlier on the top right side of the figure and a small cluster of three outliers on the bottom left of the figure. EAP recognizes the lone outlier as a single point cluster, represented by purple colour. Similarly, EAP also put together the three outliers on the lower left corner as one small cluster, designated by blue colour. It is also important to note that the presence of these outliers does not impact the clustering of the two well formed bigger clusters. This can be observed by comparing Fig. 4.5 to Fig. 4.4(b), we we have the same dataset but without the outliers.

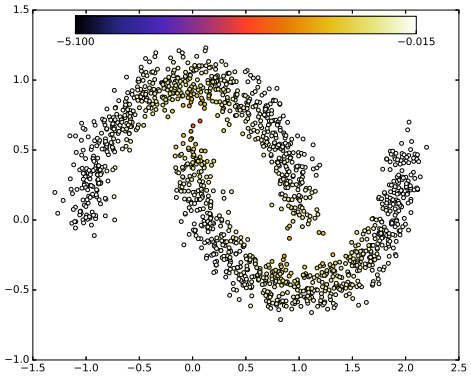
In Sec. 4.9 we will see that EAP is also able to handle noisy data with partially blurry boundaries between clusters.

4.7.4. Confidence

We can also easily and efficiently provide confidence values about EAP's decision to put a specific data point x_i into the assigned cluster c_i by using the local information provided by EAP. This can be done by defining a metric based on the comparison of the x_i's similarity to the nearest local exemplar in the assigned cluster c_i and it's similarity to the nearest



(a) Confidence via closest local exemplars



(b) Confidence via closest data points.

Figure 4.6.: The figures depict the confidence values for data points in the half-moons dataset, obtained by (a) using (4.32), (b) using (4.33). The clustering results used in either case are the same and correspond to the results in Fig. 4.4(b), only the confidence metric used is different. Note that we have used the same colour gradient scale in both figures for comparison.

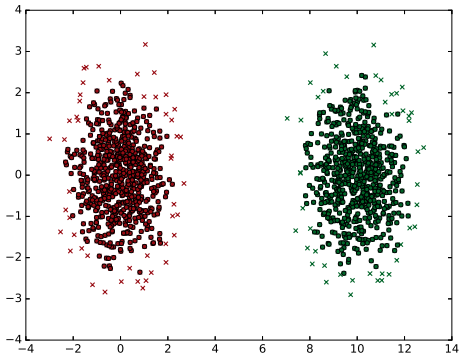
local exemplar among all other clusters. For example, the metric

$$w(x_i) = \log\left(\max_{\substack{k\in\mathcal{E}\\ \ell(x_k)=\ell(x_i)}} s_{ik} - \max_{\substack{j\in\mathcal{E}\\ \ell(x_j)\neq\ell(x_i)}} s_{ij}\right) \tag{4.32}$$

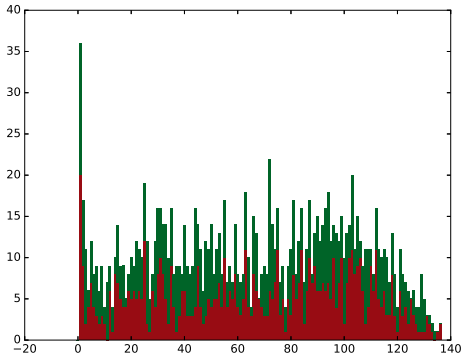
uses a logarithmic measure to evaluate the confidence. (4.32) is just an example, one can use any other way suitable way to compare the two pairwise similarity values. Fig. 4.6(a) shows the confidence of decision for each data point. As desired, the confidence is low for the points which lie on a boundary of the assigned cluster which is close to elements of another cluster. Such a confidence measure is computationally less expensive and more robust to noisy data than a confidence measure obtained by the comparison of similarity to the closest point in the assigned cluster vs the similarity to the closest point among other clusters. A confidence measure analogous to (4.32) but based on closest data points is

$$\overline{w}(x_i) = \log\left(\max_{\substack{x_k\in\mathcal{X}\\ \ell(x_k)=\ell(x_i)}} s_{ik} - \max_{\substack{x_j\in\mathcal{X}\\ \ell(x_j)\neq\ell(x_i)}} s_{ij}\right). \tag{4.33}$$

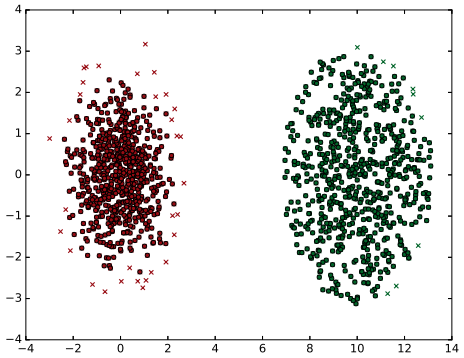
The results using (4.33) are shown in Fig. 4.6(b). We can clearly notice a sharper gradient of colour, representing the change in confidence values, in Fig. 4.6(a) vs Fig. 4.6(b). In Fig. 4.6(b), many of the data points which constitute the noisy boundary have similar confidence values to the data points which lie inside the cluster.



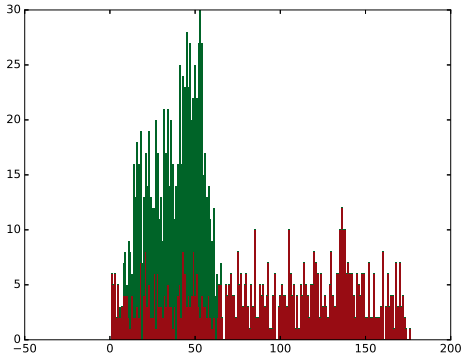
(a) Equal density clusters



(b) LEC histogram for equal density clusters



(c) Unequal density clusters



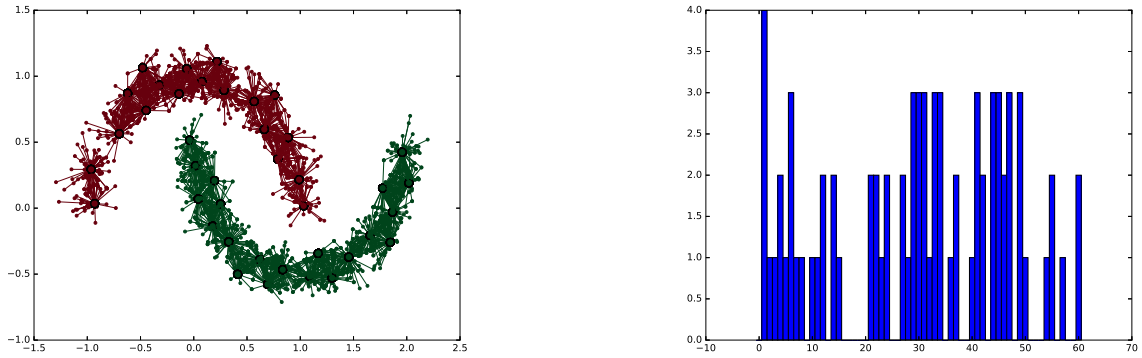
(d) LEC histogram for unequal density clusters

Figure 4.7.: The figure demonstrates how LEC histogram can be used to identify different relative densities of the clusters when the r_j constraints are inactive. These results were obtained for $p = 0.6$, $q = 0.95$ and $\Delta > 1$.

The confidence measure can be utilized to highlight a small set of points for which an analyst can try to decide which cluster they belong to.

4.7.5. Local Exemplars Connnected to a Data Point

When a data point is connected to multiple local exemplars that are well separated, this implies it is relatively close to all of these local exemplars and shares a mix of their properties. We will show examples of such mix of properties in Sec. 4.10 for real world datasets. The local exemplars connected to a data point x_i are given in $\mathcal{L}_i$ which can be computed in $\mathcal{O}(|\mathcal{E}|)$. Hence we can find the number of local exemplars connected to each data point in $\mathcal{O}(|\mathcal{E}||\mathcal{X}|)$. This information can be visualized using a histogram where the x-axis represents the number of local exemplars and the height of each bar represents



(a) Clustering results

(b) IES histogram

Figure 4.8.: The figures depict the clustering results for half-moons dataset with a weakly connected region in one cluster and the corresponding IES histogram.

the number of data points connected to these many local exemplars. Note that for two different points in the same histogram bar, the local exemplars that they are connected to may differ but the number of local exemplars they are connected to is the same. We call this the LEC histogram. We will see in Sec. 4.8 how LEC histogram can help in hyperparameter tuning.

In our development of EAP in Sec. 4.6, we introduced r_j constraints to obtain well separated local exemplars. As we mentioned in Sec. 4.6 and discussed in Sec. 4.8, inclusion of these constraints often does not impact the global structure discovery (with proper tuning of q) but it allows for a cleaner internal look of the clusters. However, looking at the clustering results and LEC histogram when $\Delta > 1$, i.e., r_j constraints are inactive, also leads to interesting insight into the dataset. It lets us recognize the relative densities of different clusters. For example, in Fig. 4.7(b) (the LEC histogram for equal density blobs shown in Fig. 4.7(a)) the bars for the two clusters are co-located whereas in Fig. 4.7(d) (the LEC histogram for unequal density blobs shown in Fig. 4.7(c)) bars for red cluster are more to the right clearly indicating a higher density of red cluster relative to the green cluster. This is because now that we have no neighbourhood constraint on local exemplars, more local exemplars appear in dense regions and lead to more local exemplar connections per data point in that region.

However, these density results should be considered with a grain of salt as LEC histogram only indicates the relative densities and for more complicated clusters with density varying significantly inside a cluster, one may no longer be able to draw conclusions about relative densities so easily. Note that for normal usage of EAP, the recommended setting is to have active r_j constraints to avoid very crowded and ultimately not so useful local information.

4.7.6. Inter Exemplar Connection Strength and Pruning

Another aspect of studying the local information is to count the number of data points that form boundary connections between two local exemplars in the same cluster. This count can be considered as an indicator of local cluster strength and for neighbouring local exemplars this can indicate the confidence of EAP to merge the associated regions into one cluster. This information can then be used to highlight potential inconsistencies where the algorithm is not too certain about it's decision to merge the regions into one cluster. These few cases can then be evaluated by an analyst. We can again form a histogram from this information where, for each exemplar we can check it's connection strength to the closest m local exemplars in the same cluster. Such a histogram is shown in Fig. 4.8(b) for the clustering results of modified half-moons dataset in Fig. 4.8(a). The half-moons dataset has been modified in Fig. 4.8(a) to contain a weakly connected region. The x-axis of the histogram represents the connection strength (i.e., number of shared data points) and the height of the bar indicates the number of local exemplar pairs with the given connection strength. We call this the IES histogram. If there is a relatively low fraction of exemplar pairs in the left-most bars, these left-most bars indicate potential inconsistencies which may need to be examined by an analyst to decide if the exemplar pairs should be connected or the links between them should be pruned. For example, in Fig. 4.8(b), there are only 4 pairs that have a single boundary connection between them. By inspecting these cases individually, we realize that 3 of them correspond to connections between relatively distant local exemplars and pruning the boundary connections between them does not have an impact on the clusters. The 4th boundary connection is highlighted in the Fig. 4.8(a). This data points provides the only link between the two subregions of the red cluster with a sparse region between them. An analyst can decide whether this data point and the corresponding local exemplars pair provide enough merit to connect otherwise two separate subsets of the dataset into one cluster. The computational complexity of constructing this histogram is $\mathcal{O}(|\mathcal{X}||\mathcal{E}|m)$. Note that we can also automate this process of pruning. The connections between all the pairs of local exemplars on the L.H.S. of a threshold, denoted by n_t in the histogram can be pruned before finding the connected components in Alg. 4.1. n_t can either be chosen by an analyst after having a look at the IES histogram or can be determined automatically, for example, by assigning n_t a value such that, for example, 99%, of the local exemplar pairs in the IES histogram have higher number of boundary connections connecting them. For our synthetic and realworld experiments in Sec. 4.10 and Sec. 4.9 we use $n_t = 3$ independent of the dataset.

4.8. Parameter Tuning and Sensitivity

EAP involves three hyperparameters: p, q and Δ. Despite the relatively high number of hyperparameters, successive tuning of these parameters to obtain desirable results is intuitive and computationally efficient in the sense that we only need a couple of steps at most before we reach the desired results. This because of the following two reasons:

- ▷ Due to the way we defined the optimization problem, the three parameters have intuitive meanings and it is easy to recognize how clustering results will change due to a change in one of the parameters.
- ▷ When we cluster the dataset using some specific values of the hyperparameters, the local information obtained indicates how the hyperparameters should be changed to improve the results.

We start by elaborating on the intuitive meaning and the suggested parameter ranges for the hyperparameters.

Self Preference: The self preference p serves a similar purpose as in AP, i.e., it indicates the interest of a data point to become a local exemplar, forming a globular subcluster around it. A higher value of p motivates more points to become local exemplars whereas a low value works to suppress potential local exemplars. For EAP, p takes normally a higher value, i.e., often between 55 and 65 percentile of the pairwise similarity values in $\mathbb{S}$, than AP where p is normally equal to the median of $\mathbb{S}$. This is because we want to motivate enough local exemplars to be distributed all over the dataset, forming subclusters which are later merged using Alg. 4.1 based on the boundary connections between the local exemplars.

Linkage penalty: The linkage penalty q defines the maximum penalty (per extra local exemplar) that a data point has to pay for connecting to more than one local exemplar. A higher penalty implies fewer boundary connections being formed between local exemplars. Since we would like boundary connections to be formed between only very close by local exemplars so we would like a data point to connect to multiple local exemplars only if it is close enough to all of them and it helps in global cluster discovery, so we use a quite high penalty. q is normally chosen to be in the range of negative of 96 to 98 percentile of $\mathbb{S}$.

Separation Radius: The separation radius Δ basically defines a neighbourhood around each local exemplar such that there should be no other local exemplar appearing in this neighbourhood. The lower the value of Δ, the bigger are the neighbourhoods $\mathcal{N}_i$ since more datapoints x_j have $s_{ij} > \Delta$ and hence the bigger is the separation between "neighbouring" local exemplars. Since the aim of introducing r_j is just to counteract the effect of loosened g_i constraint on the local information by again providing us well separated local exemplars, Δ is normally chosen to be a high value, i.e., in range from 98.5 to 99.5 percentile of $\mathbb{S}$, hence suppressing other potential local exemplars in a very close by neighbourhood. However, as discussed in Sec. 4.7.5, a $\Delta > 100$ percentile of $\mathbb{S}$, i.e., inactive r_j constraints provides us the information about relative densities of the clusters without affecting the global structure discovery.

We always specify the values of p, q and Δ in terms of percentiles of $\mathbb{S}$. Hence we define the following short hand notation: we specify the values of these parameters as real

numbers in the range $[0, 1]$ where the number represents the percentile divided by 100, for example $p = 0.65$ implies that p takes a value equal to 65 percentile of $\mathbb{S}$.

For a given dataset, we normally start with predefined hyperparameter values, for example $p = 0.6$, $q = -0.97$ and $\Delta = 0.99$. These predefined values as the typical parameter ranges specified earlier may depend on the size of the dataset, for example for larger datasets one may be interested in choosing more negative values of q and higher values of Δ. Similarly these typical values are also highly dependent of the pairwise similarity metric, the ones mentioned here are often suitable for pairwise similarity metric based on euclidean distance or a probabilistic notion of an edge between two nodes in a graph.

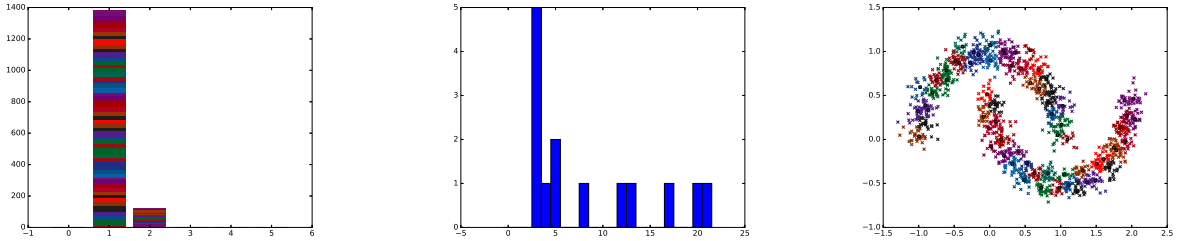
By looking at the local information for the obtained results we can then often successively tune the three parameters, starting from p, then q and finally Δ, as follows:

- ▷ **Too few local exemplars:** Assuming that Δ is in an acceptable range, this implies that we have set p too low and we should increase it such that we obtain enough local exemplars that are spread throughout the dataset.

- ▷ **Boundary connections:** Once we have adequate number of local exemplars available, we focus on building boundary connections between the local exemplars for global structure discovery. For this we use LEC and IES histograms to tune q as follows

  1. **LEC and IES shifted too much to the left:** This configuration of histograms implies that the linkage penalty is too high and hence not allowing sufficient boundary connections to form between the local exemplars. This will also manifest itself in terms of many more clusters than expected. In this scenario we should increase q to decrease the linkage penalty so that more boundary connections can appear.
  2. **LEC and IES shifted too much to the right:** This scenario is the opposite of the previous one. Due to the low linkage penalty data points also get connected to far off local exemplars (which possibly may belong to a different "ground truth" cluster). This can result in too few clusters than expected and violates the basic principle of introducing the penalty q that we want to form only boundary connections between close enough local exemplars to facilitate global cluster discovery. Allowing points to connect to even far off local exemplars rather distorts the global cluster discovery by even merging well separated clusters. In this scenario we should increase the linkage penalty (decrease q).

- ▷ **Local information:** Finally we can tune Δ to obtain the desired type of local information. If we believe that the discovered local exemplars are too close, we can decrease Δ, increasing the neighbourhood radius around each local exemplar. This will results in fewer and farther away local exemplars and a shift of both LEC and IES towards left but as long as Δ is not decreased too much the global clusters discovered remain unaffected, as was also illustrated in Fig. 4.4. The sole purpose of Δ should be to adapt the local information, hence normally, as long as one starts the



(a) LEC histogram (b) IES histogram (c) Clusters with Exemplars

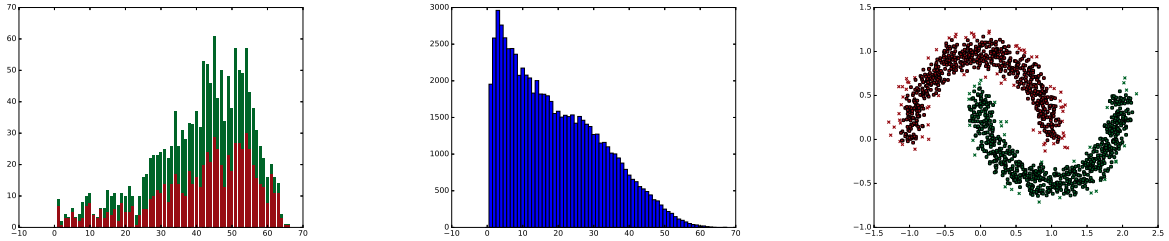
Figure 4.9.: The figure shows the results obtained in the first step of successive tuning corresponding to $p = 0.6$, $q = -0.99$, $\Delta > 1$.

tuning process with a reasonable value of Δ (such as 0.99), there is often no need to change it later in the successive tuning process.

While tuning for the boundary connections, we have not discussed the scenarios where LEC is shifted too much to the left and IES is shifted too much to the right or vice versa as these scenarios are unusual. An LEC towards right means that most points are connected to many local exemplars hence leading to strong bridges between local exemplars which in turn implies IES towards right. Similarly, an IES shifted towards right normally implies that LEC should also be shifted towards right.

The confidence values discussed in Sec. 4.7.4 can also be used to determine if the results obtained for a specific choice of hyperparameters are satisfactory. If there are too many data points with low enough confidence, this often implies that the algorithm has not been able to form boundary connections to merge subclusters where it should have. In this case we need to reduce the linkage penalty to merge these subclusters. On the other hand, when there is a relatively small number of points that exhibit a low enough confidence, these points may also be manually analyzed to see if they should be considered to form a bridge to connect two currently separate clusters to merge them

We will now present a step by step example of using the above insights to do hyperparameter tuning for the half-moons dataset earlier used in Fig. 4.4(b). As the first step, we apply EAP to the dataset for $p = 0.6$, $q = -0.99$, $\Delta > 1$. The results obtained contain 47 clusters. None of these clusters correspond to outliers (i.e., none of them are clusters consisting of only a couple of points far away from the rest of the dataset), hence all of them represent regular clusters. The number of detected regular clusters are significantly higher than expected. This is also evident by looking at the LEC histogram in Fig. 4.9(a), where almost all the data points are connected to only one local exemplar. It is also clearly visible in the IES histogram in Fig. 4.9(b), where there are very few local exemplars which share any boundary connections. These results are similar to what one can expect from AP (for the same p, AP will lead to even more clusters as it allows no boundary connections). The resulting clusters along with the local exemplars, many of which form separate clusters around them due to the lack of boundary connections, are shown in Fig. 4.9(c)



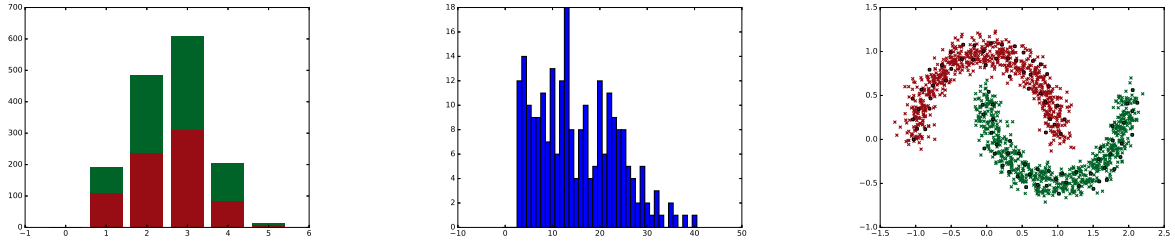
(a) LEC histogram (b) IES histogram (c) Clusters with Exemplars

Figure 4.10.: The figure shows the results obtained in the second step of successive tuning corresponding to $p = 0.6$, $q = -0.97$, $\Delta > 1$.

(some colours are reused to represent different clusters due to a limited number of clearly distinguishable colours in a small figure but in most cases it is clearly from the spatial locations of different clusters with same colour that they do not belong together). Note that Fig. 4.9(c) is shown here just for illustration purposes and is not needed to decide the next step in successive tuning. One can easily infer from the number of obtained clusters, the LEC histogram and the IES histogram that we need to reduce the linkage penalty. For the second step we reduce the linkage penalty while keeping the other hyperparameters same. The new values of the parameters are $p = 0.6$, $q = -0.97$ and $\Delta > 1$. In this case we discover two clusters and the LEC, shown in Fig. 4.10(a), as well as the IES histogram, shown in Fig. 4.10(b), is no longer shifted either too much towards left or right (taking into account that we still have $\Delta > 1$, hence there are a lot more boundary connections formed than needed). The resulting clusters along with the local exemplars are shown in Fig. 4.10(c). All three factors, number of cluster, LEC histogram and IES histogram, indicate that we have discovered the global structure well enough and now we need to decrease Δ if we want to obtain local information corresponding to well separated local exemplars.

As the final step, we reduce Δ while keeping the other hyperparameters same as the previous step. The new values are $p = 0.6$, $q = -0.97$ and $\Delta = 0.99$. We again discover two clusters as the number of global clusters is not impacted by a reasonable change in Δ for appropriately chosen p and q. Now we can also see clearly from the LEC and the number of local exemplars discovered that we no longer have very close by local exemplars since the data points otherwise would connect to all the close by exemplars whereas now the data points only connect to few local exemplars. Similarly, we also notice that whole IES histogram is now scaled down, signifying that now the local exemplars are not so close as to share almost all points they are connected to but fewer points that lie on the boundary between the two well separated local exemplars. The sole purpose of Δ is to help in discovering cleaner local information, hence one should not try to manipulate global results using Δ and it should be varied in a very narrow range.

Note that at every step we only used the available local information to determine the



(a) LEC histogram (b) IES histogram (c) Clusters with Exemplars

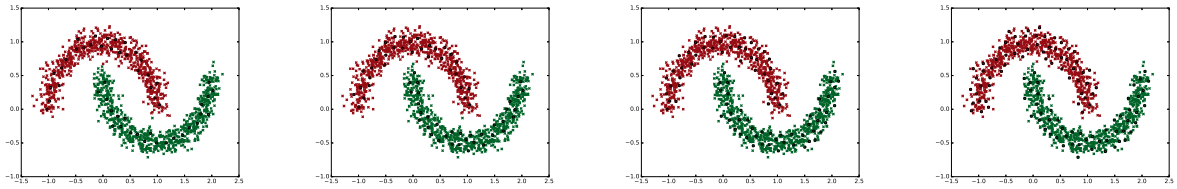
Figure 4.11.: The figure shows the results obtained in the final step of successive tuning corresponding to $p = 0.6$, $q = -0.97$, $\Delta = 0.99$.

next step for hyperparameter tuning. Hyperparameter tuning for EAP normally requires only a couple of steps before discovering the results with both suitable global clusters as well as appropriate local information. In the above example we started successive tuning with such high values of q and Δ just to illustrate the use of local information for successive tuning. For normal application, one would start with more reasonable values of q and Δ, for example $q = -0.97$ and $\Delta = 0.99$. Furthermore, in the example we did not use the confidence plots in each step but we can also use them to further help us in choosing the next step.

The results shown in Sec. 4.9 and Sec. 4.10 are all obtained using this successive tuning procedure rather than any kind of parameter sweeps. Hence these results are a good representation of what one can expect in normal application of the algorithm instead of the ceiling performance of EAP. Furthermore we also notice that the final results shown in Sec. 4.9 and Sec. 4.10 correspond to hyperparameter values that are very similar for different datasets although the datasets have very different characteristics. This also implies the ease of finding suitable hyperparameters for a wide variety of datasets by exploring a very narrow range of hyperparameter values, atleast for datasets based on similar pairwise similarity metric.

Finally we look at how global structure discovery is impacted by the variation of individual hyperparameters.

- ▷ For a fixed q and Δ, we show how the global structure discovery is relatively robust to variations in p in Fig. 4.12. In general as long as p is not too low the global cluster results don't get impacted. This is unlike AP where variations in p have strong impact on the number of clusters obtained and the cluster assignments. In the case of EAP, as long as p is varied in a reasonable range, this only changes the number of local exemplars appearing in any region but as long as the number is not too small and they can get connected via boundary connections, the number of globally discovered clusters does not change. Only when p is too low, we are no longer able to form bridge connections between the appropriate exemplars and we lose the ability to discover the global structure. In Fig. 4.12 even for $p = 0.1$, which is far outside



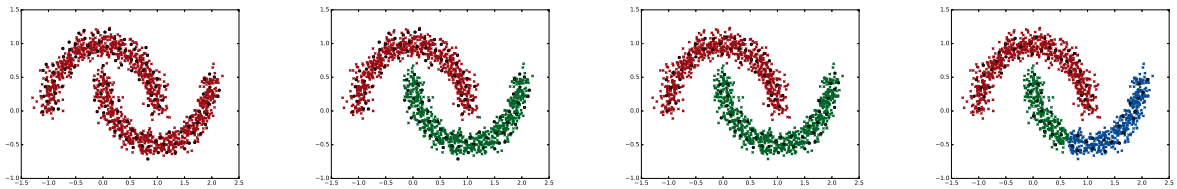
(a) $p = 0.1$ (b) $p = 0.3$ (c) $p = 0.6$ (d) $p = 0.8$

Figure 4.12.: The figure shows the robustness of the global structure discovery to variations in p where $q = -0.97$ and $\Delta = 0.99$ for all the subfigures.

the range in which we vary p during successive tuning, EAP discovers the correct global structure.

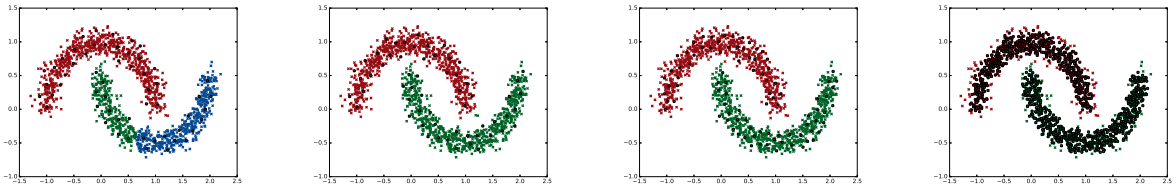
- ▷ For a fixed p and Δ, we show how the global structure discovery is impacted by the variation of q in Fig. 4.13. q is arguably the hyperparameter which needs most care but there is a reasonable range where the global clusters obtained are not affected and successive tuning is able to discover the a suitable value of q in this range quickly. When we have a too high linkage penalty, clusters start to break at their weakly connected regions as seen in Fig. 4.13(d). Note that still in Fig. 4.13(d) there have not been drastic changes in the results beyond the green cluster being broken where is was most weakly connected. On the other hand when we have too low linkage penalty very far off local exemplars also get connected by boundary connections as shown in Fig. 4.13(a).

- ▷ For a fixed p and q, the global clusters do not change when we start lowering Δ from 1. Too small a value of Δ can have a similar impact as too small p or too high q where not enough local exemplars appear in the dataset such that they can be connected via boundary connections for global structure discovery. This can be seen in Fig. 4.14(a) which leads to similar results as Fig. 4.13(d) but as we mentioned earlier, Δ should only be used to adapt the local information, hence one does not



(a) $q = -0.955$. (b) $q = -0.96$ (c) $q = -0.97$ (d) $q = -0.975$

Figure 4.13.: The figure shows the variation of the discovered clusters w.r.t. variations in q where $p = 0.6$ and $\Delta = 0.99$ for all the subfigures.



(a) $\Delta = 0.985$ (b) $\Delta = 0.986$ (c) $\Delta = 0.99$ (d) $\Delta > 1$

Figure 4.14.: The figure shows the variation of local and global clustering results due to changes in Δ where $p = 0.6$ and $q = -0.97$ for all the subfigures.

need to lower it too much as to change the global results.

4.9. Synthetic Experiments

We will use various synthetic datasets to analyze how EAP performs on them and how it captures various aspects of the datasets. As all of these datasets lie in $\mathbb{R}^2$ and the clusters are distinguishable in a Euclidean sense, we use the following pairwise similarity metric based on Euclidean distance between the points. For any two data points $x_i, x_j \in \mathcal{X}$, we have

$$s_{ij} = -\|x_i - x_j\| \tag{4.34}$$

where $\|x_i - x_j\|$ is the Euclidean distance between the two dimensional vectors x_i and x_j.

Note that since the aim of EAP is to discover the structure in the given pairwise similarity matrix, discussing synthetic datasets that are all separable in a Euclidean sense does not limit the generality of our experiments. The reason is because these datasets lead to different structures in the pairwise similarity matrix $\mathbb{S}$ due to the varied nature of the clusters in different datasets. If the same pairwise similarity matrix was computed based on some other pairwise similarity metric for some other dataset (possibly not lying in a euclidean space), EAP will discover the same structure in the data. Datasets in euclidean space that are separable in a euclidean sense just allow us to illustrate our observations in a more lucid way without invoking special domain knowledge to intrepret the results.

For quantitative global performance evaluation we will present the values for Sensitivity (Sn), Positive Predictive Value (PPV), Accuracy (Acc, goemetric mean of Sn and PPV), Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). We are using multiple metrics for quantitative evaluation of the global results due to the following two reasons:

- ▷ Since EAP (as well as AP) does not take the number of desired clusters as a hard input, it is important to use metrics which can capture the variability introduced due to potentially different number of clusters in the ground truth and the results obtained by EAP.

Dataset	Points	GT clusters	Params	Sn	PPV	Acc	NMI	ARI
Aggregation	788	7	$q = -0.97$ $p = 0.5$	0.995	0.995	0.995	0.985	0.990
Flame	240	2	$q = -0.95$ $p = 0.6$	0.983	0.991	0.987	0.9	0.955
R15	600	15	$q = -0.97$ $p = 0.6$	0.992	0.992	0.992	0.988	0.982
Circles	1500	2	$q = -0.97$ $p = 0.6$	1	1	1	1	1
Spiral	312	3	$q = -0.96$ $p = 0.6$	1	1	1	1	1

Table 4.2.: Quantitative global performance evaluation of EAP on synthetic datasets along with the final hyperparameters obtained via successive tuning.

- ▷ We are evaluating performance for a variety of datasets with diverse characteristics. For example some datasets have nearly equal sized clusters whereas others have higher imbalance in cluster sizes. Similarly some datasets have all clusters with similar densities whereas other datasets have clusters with different densities.

Performance evaluation based on one metric cannot capture all these aspects. For example, Sn tells us how well the ground truth clusters are represented by their best matching EAP generated clusters and PPV shows how well the EAP generated clusters represent their best matching ground truth clusters. Consider the following examples to understand the need for using both metrics. If an algorithm clusters everything into one cluster whereas the ground truth contains more than one clusters, then we will have an $Sn = 1$ but PPV value will be lower. On the other hand, if EAP splits a ground truth cluster into multiple clusters, then corresponding PPV values for each of these EAP generated clusters will be 1 but the Sn for the ground truth cluster will be low.

For synthetic datasets we do not compare the performance of EAP with other clustering algorithms mainly due to the following two reasons:

- ▷ For the global discovery, EAP obtains nearly perfect results on these synthetic datasets. Note that these results are obtained by successive tuning procedure described in Sec. 4.8, not by hyperparameter sweeps. Hence a comparison with other algorithms can only reveal their weaknesses in discovering certain global structures when compared to EAP. These weaknesses are well documented for the well known algorithms and hence we do not want to reiterate them but rather focus on the findings of EAP.

- ▷ The algorithms that provide local information, such as K-medoids (PAM) and AP mostly only provide local exemplars. These algorithms also suffer from a lack of

global structure discovery, rendering them not too effective for clustering in general unless the dataset only has globular clusters. Therefore comparing them to EAP in terms of local information when they cannot discover the global structure in the dataset does not lead to an interesting comparison.

Table 4.2 presents the Sn, PPV, Acc, ARI and NMI results for synthetic datasets, along with the hyperparameters corresponding to the results, obtained using successive tuning. Table 4.2 and the figures in the following subsections clearly show that EAP discovers the global structure easily for all these settings. Furthermore, although we don't show here explicitly, adding some outliers similar to how we did in Fig. 4.5 does not impact the global structure discovery and the algorithm detects the outliers.

Now we will discuss each dataset individually. We will highlight the interesting aspects of the dataset as well as what EAP discovers. For each dataset, the figures we show contain results for both a $\Delta < 1$ discovered via successive tuning and for $\Delta > 1$, for the same p and q. The results for $\Delta > 1$ are shown just to make statements about the relative densities of different discovered clusters in a dataset. In practice, if one is not interested in clearly highlighting relative densities, it is never desirable to run EAP for $\Delta > 1$. It is also important to note that the scale of colour variation for the confidence plot is automatically adapted for each dataset based on the variability of the confidence values. Therefore when looking at the confidence plots, one should also account for the colour scale.

4.9.1. Aggregation

The Aggregation dataset is taken from [69]. The important aspects of the dataset include the ground truth clusters of varying sizes as well different densities. Some of the ground truth clusters are also linked by small "noise bridges". EAP is able to handle all the three issues. The LEC histogram for $\Delta > 1$, shown in Fig. 4.15(f), clearly shows that the blue cluster has higher density when compared to other discovered clusters as the data points corresponding to the blue clusters dominate the R.H.S. of the LEC histogram. Furthermore, the noise bridges between different clusters clearly get highlighted as potential inconsistencies with low confidence values in Fig. 4.15(d). Note that we have used $p = 0.5$ which is a bit removed from what we normally employ for EAP (although as we illustrated in Sec. 4.8, the results are often not too sensitive to p for the appropriate q and Δ). This is because for consistency purposes we wanted use the same p for both $\Delta < 1$ and $\Delta > 1$. Otherwise, using $p = 0.6$, $q = 0.97$ and $\Delta = 0.99$ also yields very similar results to the ones shown in Fig. 4.15 for $\Delta < 1$.

4.9.2. Flame

The Flame dataset is taken from [70]. The two important aspects of Flame are: First, there is relatively long noisy boundary between the two non linearly separable unequal sized clusters. Second, it contains two outliers (visible in the upper left corner of Fig. 4.16(a)). EAP not only discovers the clusters correctly but also highlights the noisy boundary between

the two clusters as potential inconsistencies in the confidence plot Fig. 4.16(d). EAP also discovers the outliers and creates a separate cluster for these two nearby outliers (indicated by red colour in Fig. 4.16(a)). Note that the these two datapoints are considered a part of the upper cluster in the ground truth (hence not marked as outliers), again reflecting on the subjective nature of clustering problem, as from a visual perspective they appear to be outliers.

4.9.3. R15

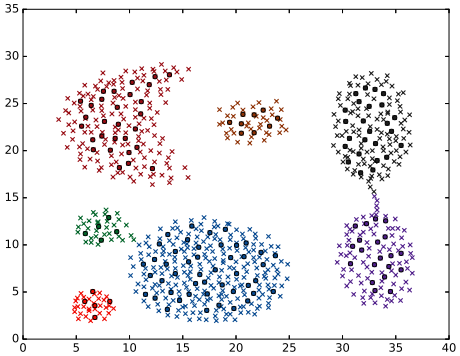
The R15 dataset is taken from [71]. It consists of spherical clusters of equal size but with varying separations between the clusters. Since all the clusters are approximately globular, most of them are adequately representable by a single local exemplar. This is clearly visible in LEC histogram, shown in Fig. 4.17(b), where most of the points are connected with only one local exemplar and also in IES, shown in Fig. 4.17(c), where only there are only two local exemplar pairs that share boundary connections. Hence LEC histogram in Fig. 4.17(b) and IES histogram in Fig. 4.17(c) indicate the presence of mainly relatively small globular clusters in the dataset. Looking at the confidence values, we can also infer that for some clusters (the ones in the outer radius), the "neighbouring" clusters are far enough as the confidence values for all data points in these clusters are high, whereas for some other clusters (the ones in the inner radii), the "neighbouring" clusters are much closer, indicated by the lower confidence values of some of the points in these clusters.

4.9.4. Concentric Circles

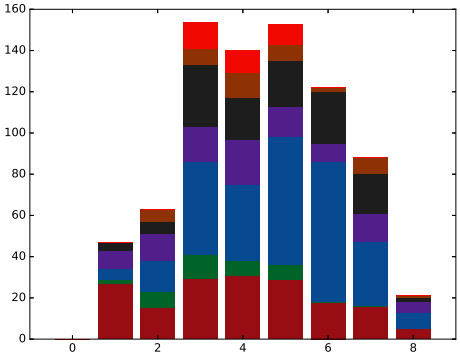
The concentric circles dataset consists of two concentric ring shaped equal sized clusters. Two key aspects of the dataset are that the clusters are non-linearly separable and they have different densities. EAP discovers the clusters and the LEC histogram for $\Delta > 1$ in Fig. 4.18(f) highlights the fact that the green cluster has a higher density.

4.9.5. Spirals

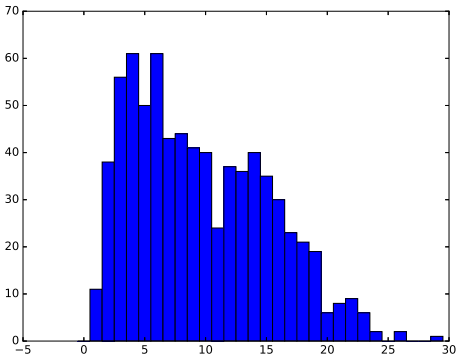
The last synthetic dataset we look at is taken from [72]. The dataset consists of three intertwined spiraling clusters of almost equal size. Fig. 4.19 shows how EAP neatly used the distributed local exemplars and the boundary connections between them to form chains and discovers the global spiral structures. Note that IES shows that some of the local exemplars are strongly connected. These are the local exemplars near the inner edge of the spirals where the density is significantly higher than the outer edge. This shows how IES can also be used to indicate the varying densities inside a cluster.



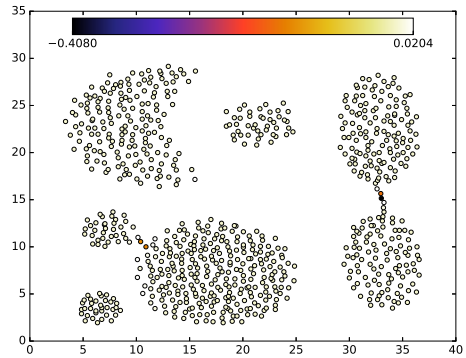
(a) Clustering results with exemplars for $\Delta = 0.995$



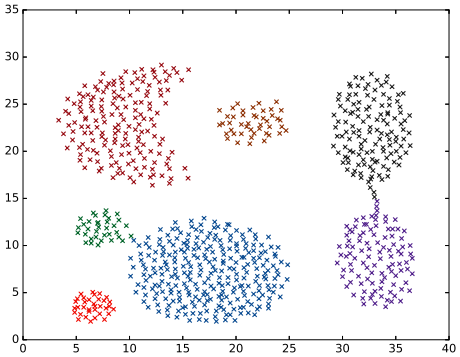
(b) LEC histogram for $\Delta = 0.995$



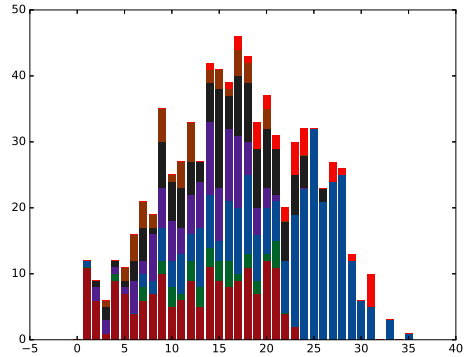
(c) IES histogram for $\Delta = 0.995$



(d) Confidence values for $\Delta = 0.995$

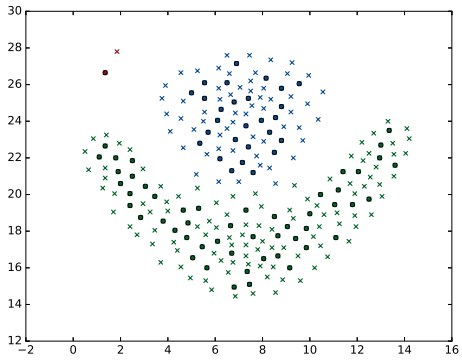


(e) Clustering results for $\Delta > 1$

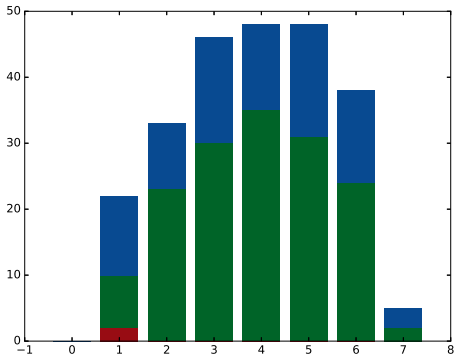


(f) LEC histogram for $\Delta > 1$

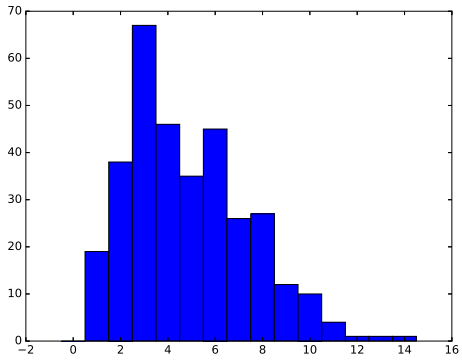
Figure 4.15.: EAP results for Aggregation dataset where $p = 0.5$ and $q = 0.97$.



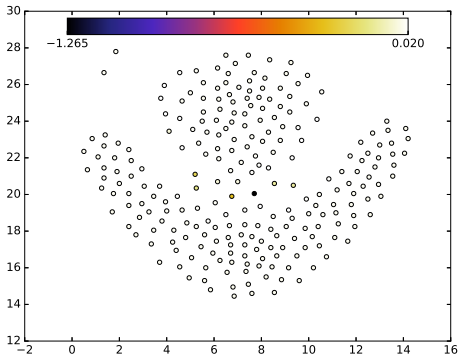
(a) Clustering results with exemplars for $\Delta = 0.99$



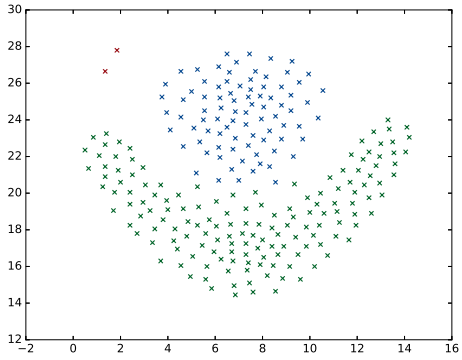
(b) LEC histogram for $\Delta = 0.99$



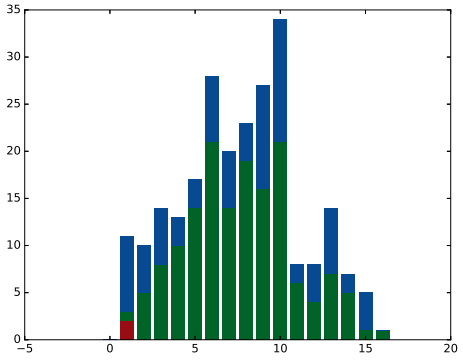
(c) IES histogram for $\Delta = 0.99$



(d) Confidence values for $\Delta = 0.99$

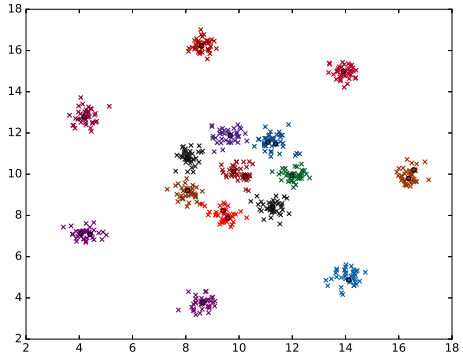


(e) Clustering results for $\Delta > 1$

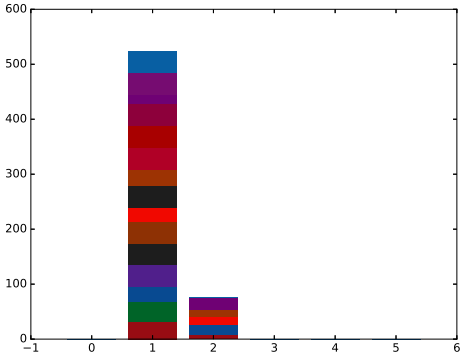


(f) LEC histogram for $\Delta > 1$

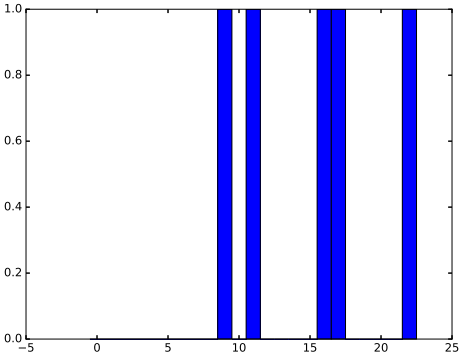
Figure 4.16.: EAP results for Flame dataset where $p = 0.6$ and $q = -0.95$



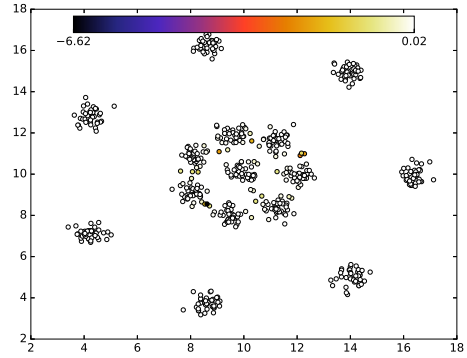
(a) Clustering results with exemplars for $\Delta = 0.99$



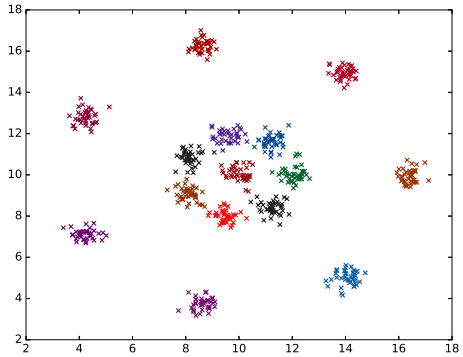
(b) LEC histogram for $\Delta = 0.99$



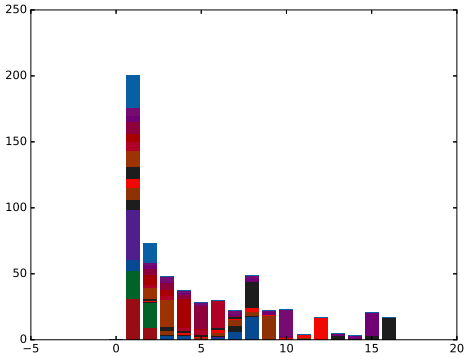
(c) IES histogram for $\Delta = 0.99$



(d) Confidence values for $\Delta = 0.99$

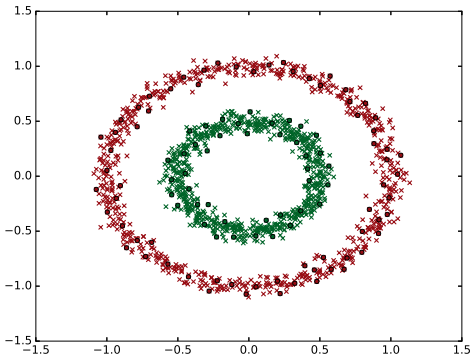


(e) Clustering results for $\Delta > 1$

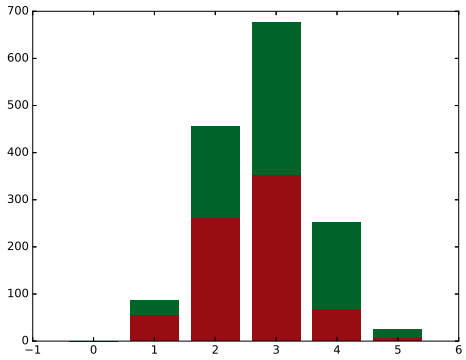


(f) LEC histogram for $\Delta > 1$

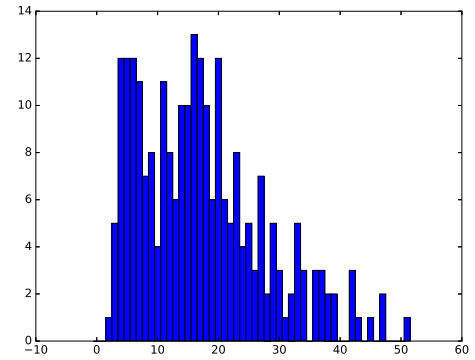
Figure 4.17.: EAP results for R15 dataset where $p = 0.6$ and $q = -0.97$



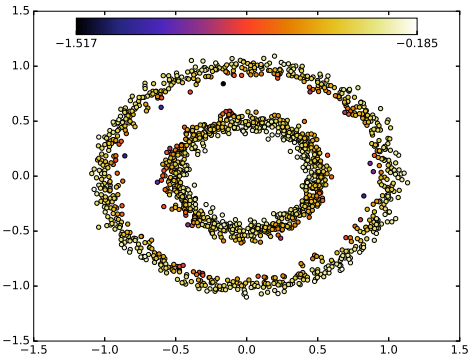
(a) Clustering results with exemplars for $\Delta = 0.99$



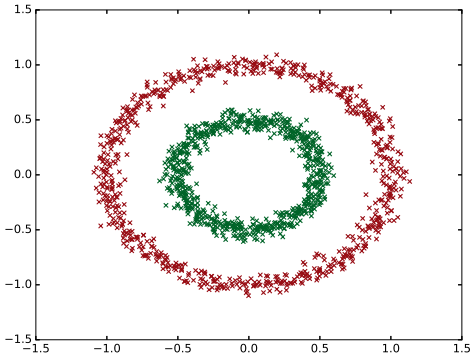
(b) LEC histogram for $\Delta = 0.99$



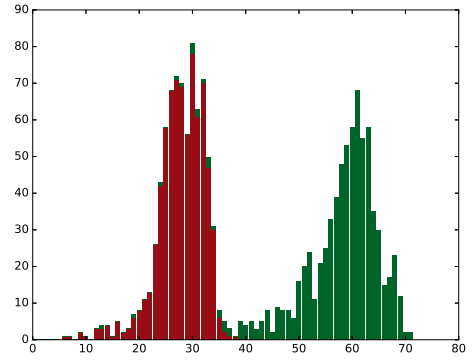
(c) IES histogram for $\Delta = 0.99$



(d) Confidence values for $\Delta = 0.99$

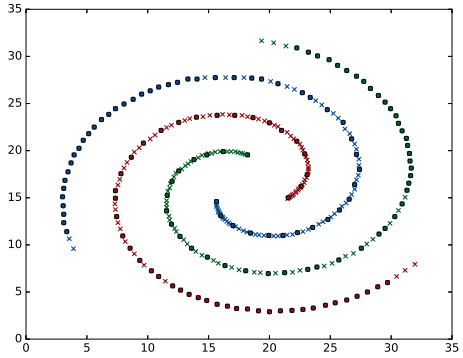


(e) Clustering results for $\Delta > 1$

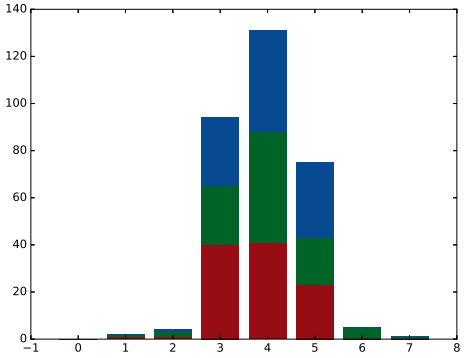


(f) LEC histogram for $\Delta > 1$

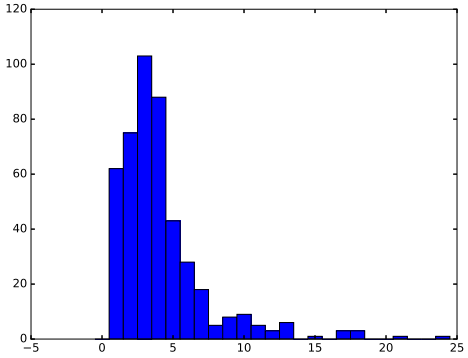
Figure 4.18.: EAP results for concentric circles dataset where $p = 0.6$ and $q = -0.97$



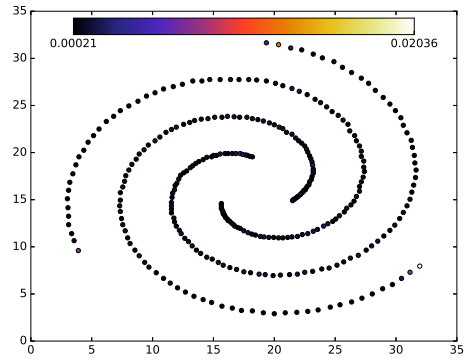
(a) Clustering results with exemplars for $\Delta = 0.99$



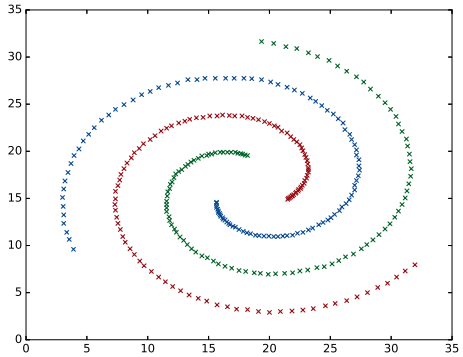
(b) LEC histogram for $\Delta = 0.99$



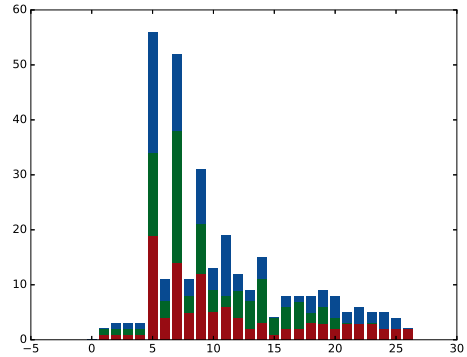
(c) IES histogram for $\Delta = 0.99$



(d) Confidence values for $\Delta = 0.99$



(e) Clustering results for $\Delta > 1$



(f) LEC histogram for $\Delta > 1$

Figure 4.19.: EAP results for Spirals dataset where $p = 0.6$ and $q = -0.96$

4.10. Real World Datasets

Now we apply EAP to three real datasets, Optdigits [73], MNIST [74] and protein interactions dataset [75]. We will present both the performance obtained via successive tuning as well as the ceiling performance obtained by sweeping all 3 hyperparameters over a broad enough range. The comparison between the two depicts the performance gap one may experience in practice w.r.t. a plausible ground truth and it also signifies the effectiveness of the successive tuning heuristic discussed in Sec. 4.8. We also compare the global performance of EAP with it's parent algorithm AP. We provide a comparison for both the heuristic performance and the ceiling performance of the two algorithms. The heuristic performance of AP is obtained by setting $p = 0.5$ [16]. We do not provide comparisons for other algorithms such as MCL and DBSCAN as they do not provide local information, hence they do not serve similar purpose. Interested readers can, however, compare our results for MNIST dataset to the ones discussed in [65]. We have not included their results here because we have not verified their simulations for all the algorithms that they present results for and their approach for comparison differs in the sense that they compare the algorithms for fixed number of discovered clusters. Similarly, for protein interactions dataset, interested readers can compare our results to [76] which provides a detailed comparison between AP and MCL. The performance comparison between AP and EAP is presented in Table 4.3 for all three datasets. We have only presented Acc values in the table to present the results in a compact way for comparison. NMI and ARI paint a very similar picture in terms of comparison, hence they are not included here for the sake of brevity. We have also included the number of clusters discovered by AP and EAP for the heuristic results. Finally, for Optdigits and MNIST, we also show examples of what local exemplars may represent in real datasets and what does it imply for the data points that are connected to more than one well separated local exemplars.

4.10.1. Optdigits

Optdigits [73], is a dataset consisting of grey scale 8×8 images of handwritten digits. We only use the test set, consisting of 1797 images, for clustering. We use (4.34) to compute the pairwise similarities based on euclidean distance. We do recognize that for optical character recognition applications one can use a better adapted pairwise similarity metric, such as one based on SIFT features [77] but we want to show what EAP is able to extract using this crude measure and not rely on the power of a strong metric which already simplifies the clustering problem significantly for the algorithm. Besides, one can use the training set to extract the right scaling or other factors for the pairwise similarity metric but as our focus is not on how to design pairwise similarity metrics, we do not do employ any such techniques. For the ground truth we separate the dataset into 10 clusters each corresponding to a different digit. Table 4.3 shows the global performance of AP and EAP as well the corresponding hyperparameters and number of clusters discovered. We can see that EAP outperforms AP in both ceiling as well as heuristic performance. Furthermore, EAP is able to achieve good accuracy and the gap between heuristic and ceiling perfor-

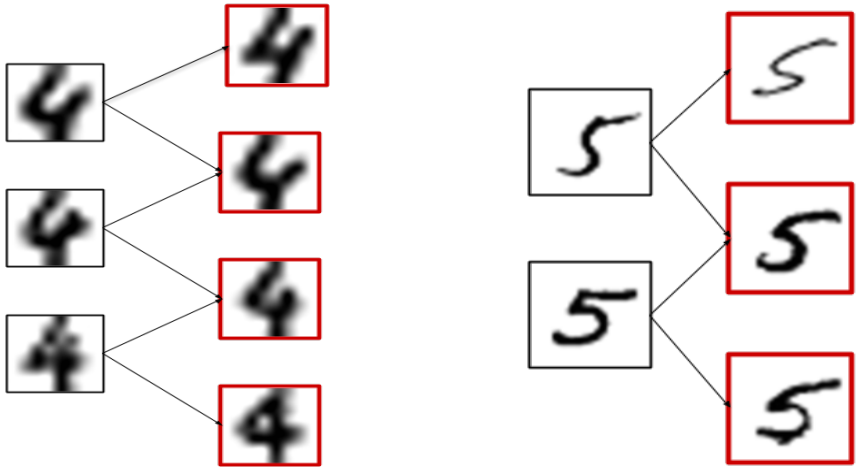
	AP			EAP			
Dataset	H: Acc	H: Clusters	C: Acc	H: Params	H: Acc	H: Clusters	C: Acc
Optdigits	0.354	137	0.695	$p = 0.6$ $q = -0.98$	0.86	26	0.871
MNIST	0.339	113	0.456	$p = 0.5$ $q = -0.98$	0.562	53	0.583
Proteins	0.824	405	0.875	$p = 0.5$ $q = -0.97$	0.864	473	0.909

Table 4.3.: Comparison of both heuristic (denoted by H:) and ceiling (denoted by C:) performance, in terms of Acc, of AP and EAP on real datasets. For all three datasets successive tuning results into $\Delta = 0.99$. For Proteins dataset the hyperparameter values are mentioned in terms of the percentile of non-zero values in the pairwise similarity matrix since $\mathbb{S}$ is sparse. In case of sparse matrices, the parameter tuning should be done in terms of the non-zero pairwise similarities.

mance is not big. Fig. 4.20(a) shows an example of the discussion in Sec. 4.7.5. Each image represents a data point and the local exemplars are marked with the red boundaries. We can see that the different local exemplars correspond to different handwriting styles and that the data points connected to two local exemplars exhibit a mix of the two handwriting styles.

4.10.2. MNIST

MNIST is also a similar dataset of 28×28 gray scale images of handwritten digits. We choose 1000 images from the dataset at random to define $\mathcal{X}$. We calculate the pairwise similarities again via (4.34) and the same discussion is valid for MNIST regarding the computation for better pairwise similarity metrics as for Optdigits. We consider all the examples of a specific digit as belonging to one ground truth cluster. Table 4.3 shows the performance. We can again see that EAP provides a significantly better heuristic performance as compared to AP. The absolute performance itself is not very good but when one compares it to the performance of other algorithms reported in [65], it is at par or better than the algorithms reported there. We believe that all these algorithms suffer in terms of absolute performance due to the crude pairwise similarity metric used. As the images get bigger in size, there is a higher chance of pixels corresponding to the digits not being aligned in different images and this leads to a lower pairwise similarity between different images of same digit as well. This again signifies the importance of finding a problem dependent suitable method to calculate pairwise similarities. We again notice that the gap between the heuristic and the ceiling performance of EAP is not big. Fig. 4.20(b) shows another example of the discussion in Sec. 4.7.5 where different local exemplars in the same cluster correspond to different handwriting styles and the data points connected



(a) Optdigits (b) MNIST

Figure 4.20.: Examples of some local exemplars, corresponding to different hand writing styles for the same digit. The figure also shows some data points connected to two local exemplars, exhibiting a mix of the two hand writing styles.

to two local exemplars show a mix of the properties of both local exemplars.

4.10.3. Proteins Interactions Dataset

Finally, we apply EAP to the protein interactions dataset, taken from [75]. The pairwise similarity values for this dataset basically represent protein-protein interactions. [78] proposes a probabilistic measure, which takes value between 0 and 1, to specify the pairwise interactions. We use this to define our pairwise similarity matrix. This dataset is different from the previously discussed synthetic and real datasets in the sense that the available ground truth does not necessarily assign each data point to only one ground truth cluster. Furthermore, some ground truth clusters are a subset of other ground truth clusters. In order to reduce such overlap in the discovered clusters, we threshold the protein-protein interactions such that the interactions below 0.4 are set to 0[1]. Furthermore, we only consider proteins that are common with CYC2008. This leaves us with 1171 proteins (data points) assigned to 276 ground truth clusters. Table 4.3 shows the global performance. EAP outperforms AP but only by a small margin, both in terms of heuristic as well as ceiling performance, since AP already performs well on this dataset. Besides, the heuristic and ceiling performance of EAP are again close. For this dataset we cannot provide any insights into the meaning of the obtained local information since we don't have the domain knowledge to interpret the results and since most of the discovered clusters consist of only a couple of proteins.

[1]This approach has been adopted in previous studies too e.g. [76] and [79], although different thresholds have been used there.

Part III.

Information-Theoretic Cost Functions for Training Deep Neural Networks

5

Learning Representations for Neural Network-Based Classification Using the Information Bottleneck Principle

In this chapter we investigate training Deep Neural Networks (DNNs) for classification via the Information Bottleneck (IB) principle. We point out two severe issues in this regard and discuss various ways to remedy them. We also elaborate on representation learning and regularizing intermediate representations in this context.

Classification is one of the two key categories of supervised learning problems, the other being regression. Classification tasks arise in various domains, such as image classification, text classification or speech processing and have widespread applications ranging from spam detection to medical imaging and logistics. In recent years DNNs have become the dominant method for dealing with complex classification tasks in many different domains, especially in computer vision and natural language processing [18]. DNNs have been shown to achieve human-level or even better performance for various classification tasks and are therefore rapidly being adapted by various industries for performing different tasks.

One of the main themes in supervised learning problems is to devise and train algorithms in such a way that they generalize well and avoid overfitting to the training data. Achieving this is considered especially difficult for modern state-of-the-art deeper architectures [80] as their flexibility and the sheer number of learnable parameters means that they can "memorize" the training set [81], especially for smaller datasets. This makes them highly prone to overfitting. In order to avoid overfitting, DNNs are often trained via cost functions which not only focus on their accuracy performance, traditionally via cross-entropy loss, but also introduce a regularization term to avoid overfitting. Some of the traditional regularization mechanisms focused on explicitly limiting the model complexity whereas the modern regularization methods, such as Dropout [82], have broadened the concept significantly and no longer focus on explicitly limiting the model complexity. Besides,

in recent years additional terms have been introduced in cost functions to incorporate other desired operational goals, such as fairness [83] and privacy [84], in machine learning systems. We also refer to this as regularization.

The paper [19] sparked a new area of research by proposing to train DNNs by minimizing the IB functional. This approach aims at balancing two objectives: maximizing the information extracted about the class label, which is analogous to accuracy in traditional cost functions, and minimizing the information preserved about the input, which is analogous to regularization in traditional cost functions. Hence the IB functional focuses on preserving all the information about the class label in the input while getting rid of the rest of the information. Furthermore, it has been suggested that the IB functional has the potential to directly train latent representations of a DNN instead of influencing them only via cost functions acting on the output and the network parameters. This connects the use of IB functional for training to representation learning, where one aims to enforce and analyze different characteristics of the latent representation of a DNN.

Subsequently, the IB framework has been used to train DNNs for discrete or continuous inputs [85–89]. These works report remarkable performance in classification tasks (see also Sec. 5.5), but only after slightly departing from the IB framework by claiming that the IB functional is hard to compute. As a remedy, they replace mutual information terms with bounds in order to obtain cost functions that can be computed and optimized using gradient-based methods.

In this chapter, we present a thorough analysis of using the IB functional for training DNNs. We start by discussing representation learning in the context of classification, i.e., we identify some desired characteristics of a DNN based classifier (Sec. 5.2). We then show that in deterministic DNNs the IB functional leads to an ill-posed optimization problem by either being infinite for almost all parameter choices or by being a piecewise constant function of the parameters (Sec. 5.3.1 and Sec. 5.3.2). Moreover, we show in Sec. 5.3.3 that the IB functional captures only a small subset of properties desirable for the intermediate representation L when performing classification, and hence it is not suitable as a cost function for training deterministic DNNs. In Sec. 5.4 we then show how the utility of IB can be partly recovered in several ways, e.g., by including a decision rule, replacing the IB functional with a better behaved cost function, and training stochastic neural networks. To utilize these considerations, we postpone discussing related work until Sec. 5.5. We argue that the successes of [85–89] must be attributed to, and provide experimental evidence for the validity of these steps – replacing the functional, making the DNN stochastic, data augmentation, including the decision rule – and not on the fact that these works are based on the IB principle. We then turn our attention in Sec. 5.6 to showing how some of the remedies proposed in Sec. 5.4, as well some recent works [90–92], point to a new promising paradigm where one regularizes the learned latent representations to enforce desired properties.

Finally, we discuss two topics that are not directly related to training DNNs via the IB principle but are related to our earlier discussion in this chapter. In Sec. 5.7, we briefly comment, based on our observations in Sec. 5.3, about another related active area of interest: Analyzing the standard training mechanisms as well as already trained DNNs

by studying their latent representations in terms of the IB functional. In Sec. 5.8, we then investigate the relation between the cross-entropy loss and its information-theoretic analog used in the IB functional. Delving into this topic gives important insights about the relation between cross-entropy loss and different mutual information terms in the context of training DNNs. Cross-entropy is very often used to the track accuracy of a DNN during training for classification tasks and has also been used as a surrogate for the precision term in the IB functional [85, 86].

This research was conducted in close collaboration with Dr. Bernhard Geiger.

5.1. Preliminaries

Consider two dependent RVs X and Y, where $X \in \mathbb{R}^n$ is the input and $Y \in \mathcal{Y}$, for some finite set $\mathcal{Y}$, is the class variable. In our analysis, we make the uncommon assumption that the joint distribution $P_{X,Y}$ between the features X and the class Y is known. P_Y is a pmf as $\mathcal{Y}$ is finite but P_X can be discrete, continuous or singular. Assuming that we know $P_{X,Y}$ not only admits more rigorous statements, but makes them independent of the optimization heuristic used for training and corresponds to a best-case scenario for training. Nevertheless, we make regular comments on how our analysis changes in case only a finite dataset is available.

A classifier takes X as input and tries to predict the class label Y for the given X. For example, X may represent an image and the aim of the classifier may be to predict the object present in the image, from a list of possible objects $\mathcal{Y}$. The output of a classifier can take different forms. For example the classifier may output a hard decision, predicting one specific $y \in \mathcal{Y}$ as the predicted class label for a given $x \in \mathbb{R}^n$. In this case the classifier is a (possibly stochastic) mapping from $\mathbb{R}^n$ to $\mathcal{Y}$. Another possible form of classifier output is a probability vector over $\mathcal{Y}$, denoting the estimated likelihoods according to the classifier, of different $y \in \mathcal{Y}$ being the class label for a given $x \in \mathbb{R}^n$.

In this chapter, we focus on classifiers based on feed-forward DNNs [18, Chap. 6]. Since we study only feed-forward DNNs, we use the term DNN to refer exclusively to feed-forward DNNs. The DNN accepts the RV X at the input and responds with the transformed RV $\hat{Y}$ (not necessarily defined over $\mathcal{Y}$). This transformed RV $\hat{Y}$ is then fed to a decision rule (see [18, Chap. 6.2] for a discussion on decision rules used in conjunction with DNNs for classification) to produce an estimate of the class label Y. If the DNN has m hidden layers, then the column vector collecting all neuron outputs in the i-th hidden layer is the *intermediate representation* denoted by L_i. We call $L_0 \triangleq X$ the input and $L_{m+1} \triangleq \hat{Y}$ the output of the DNN, while for $i \in \{1 \dots m\}$, we call L_i a latent representation. Abusing notation, $|L_i|$ denotes the number of neurons in the i-th layer, e.g., $|L_0| = n$. Whenever we refer to intermediate representations of which the layer number is immaterial we write L instead of L_i.

If the DNN is deterministic, then L_i and L_{i+1} are related by a function $g_i \colon \mathbb{R}^{|L_i|} \to \mathbb{R}^{|L_{i+1}|}$ that maps the former to the latter and that depends on a set Θ_i of (weight and bias) parameters. E.g., if $\mathbb{W}_i$ is the matrix of weights between the i-th and $(i+1)$-th layer, $\boldsymbol{b}_{i+1}$ the vector of biases for the $(i+1)$-th layer, and $\sigma \colon \mathbb{R} \to \mathbb{R}$ an activation function, then $\Theta_i = \{\mathbb{W}_i, \boldsymbol{b}_{i+1}\}$ and

$$L_{i+1} = g_i(L_i) = \sigma\left(\mathbb{W}_i^T L_i + \boldsymbol{b}_{i+1}\right) \tag{5.1}$$

where the activation function is applied coordinate-wise. Whether the activation function is sigmoidal, ReLU, leaky ReLU, tanh, or softplus is immaterial for the results that follow, unless stated otherwise. We define $L_{i,j} \in \mathbb{R}$ to be the j-th component of the vector $L_i \in \mathbb{R}^{|L_i|}$, $X_j = L_{0,j}$ and $\hat{Y}_j = L_{m+1,j}$. We have $L_i = f_i(X)$, where $f_i \triangleq g_{i-1} \circ \cdots \circ g_1 \circ g_0$ shall be called the *encoder for* L_i. Similarly, $\hat{Y} = h_i(L_i)$, where $h_i \triangleq g_m \circ \cdots \circ g_i$ is called the *decoder of* L_i. If the DNN is stochastic, then g_i is a stochastic map parameterized by

Θ_i, and the encoder f_i and decoder h_i are obtained in the same way as for deterministic DNNs by appropriately concatenating the stochastic maps $\{g_i\}$. When we discuss an intermediate representation L, where the layer number is immaterial, we will denote the corresponding encoder and decoder for L by f and h. Note that this setup subsumes concepts like convolution layers and skip connections by imposing further restrictions on the weight matrices $\mathbb{W}_i$ or the intermediate representations L_i.

5.2. Learning Representations for Classification

Training DNNs via the IB functional for supervised learning tasks focuses on learning "suitable" intermediate representations $\{L_i\}$. Hence, in order to critically study IB functional based training, we first present a subset of properties of an intermediate representation L desirable for a classification task. These properties are either motivated by a taking a signal processing perspective of the classifier, or are linked to the primary operational goal of good accuracy and generalization. Operational goals such as generalization are often not directly enforcable while training. Desired properties for intermediate representations provide one mechanism to formulate and to indirectly enforce such operational goals. We do not accompany these properties with precise mathematical definitions – this is out of the scope of this work and left for future research. Nevertheless, taken as guiding principles, these properties suffice to point out the shortcomings of the IB principle for training DNNs and to discuss ways to remedy them in the later sections. For classification, the representation L should:

P1 **inform about Y.** This means that the representation should contain as much information about the class variable Y as was contained in the input X, i.e., L should be a *sufficient statistic* for Y.

P2 **be compressed.** The representation L should not tell more about X than is necessary to correctly estimate Y, i.e., successive representations in a DNN should attain higher invariance, in some sense, to nuisance factors which are not relevant to the class label Y. Compression, and consequently invariance, can, for example, be quantified statistically (e.g., L is a *minimal* sufficient statistic for Y) or geometrically (e.g., data points from different classes are mapped to different dense clusters in $\mathbb{R}^{|L|}$). The minimal sufficient statistic perspective is an information-theoretic approach of looking at L, whereas the geometric perspective is closer to a signal processing view of the system, taking into account that the layers of a DNN define transformations from one Euclidean space $\mathbb{R}^{|L_i|}$ to another Euclidean space $\mathbb{R}^{|L_{i+1}|}$.

P3 **admit a simple decision function.** The successive intermediate representations should be such that the class Y can be estimated from them by using successively "simpler" functions. The term "simple" here has to be taken relative to the capabilities of the information sink or the system processing L. E.g., in DNN, decisions are often made by searching for the output neuron with the maximum activation

(arg max) or by binary quantization (for $|\mathcal{Y}| = 2$) so the intermediate representation $\hat{Y}$ should be such that these simple decision functions suffice to predict the class label from the $\mathbb{R}^{|\hat{Y}|}$-dimensional RV $\hat{Y}$. P3 basically caters to a signal processing view of the DNN, i.e., for a classification task, the aim of the DNN is to process X to not only extract but also simplify the information about Y available in X. The information about Y should be presented in an increasingly simplified manner in successive deeper intermediate representations so that ultimately at the output of the DNN it can be effectively utilized by a simple decision mechanism.

P4 **be robust.** This means that adding a small amount of noise to X or transforming it with a well-behaved transform (e.g., small deformations that do not impact the information about the associated class label) should not lead to big differences in the intermediate representation. E.g., the dense clusters in $\mathbb{R}^{|L|}$ corresponding to different classes should be far apart and the small deformations should not change the cluster in L that a data point is mapped to.

Historically, the primary goals of training have been to extract information about Y from the input X such that this simplified extracted information can be effectively used by a simple decision mechanism to estimate Y (P1 and P3). Traditionally, these goals have been achieved by using mean-squared error or cross-entropy as a cost function. As these cost functions are defined based on the output of the DNN or the subsequent decision rule, they enforce P3 in latent representations only indirectly via the output. P4, i.e., robustness, has been linked to improved generalization capabilities of learning algorithms [93–95]; regularization measures such as dropout have been shown to instill robustness and improved generalization. Reference [19] has additionally introduced the idea of having maximally compressed intermediate representations (P2). The intuition behind this requirement is that this should avoid overfitting, and hence improve generalization, by making the network forget about the specific details of the individual examples and by making the intermediate representations invariant to nuisances in the input that are not relevant for the classification task. In this sense, P2 and P4 may have overlapping nature as noise and deformations (P4) may also be considered as nuisance factors (P2). The exact nature of overlap depends on the precise definitions. In this work we differentiate the two on an abstract level as follows: On the one hand, P2 tries to make intermediate representations invariant to the changes in the input while moving along a manifold in $supp(P_X)$ that belongs to a specific class label. For example, when classifying MNIST digits [74], one feature that varies while moving along the manifold is the handwriting style and for digit classification tasks we would like to be able identify the correct digit in the picture regardless of the handwriting style. On the other hand, P4 tries to make the intermediate representations invariant to small deformations or noise that cause the deformed input to most likely no longer lie on the manifold, to which the original input sample belonged to, but in close vicinity of the manifold. Again considering the example of classifying MNIST digits, this may correspond to sensor noise introducing unnatural color or intensity variations, or it may correspond to a deformation such as stretching the digit so that is still recognizable but unlikely to be

written so by a human. In order to study robustness (P4), we need to study the behaviour of DNN outside, but in relatively close neighbourhood, of $supp(P_X)$.

In addition to achieving P1 through P4, one may wish that the DNN producing these intermediate representations is architecturally economical. E.g., the DNN should consist of few hidden layers, few neurons per layer, few convolution filters or sparse weight matrices $\{\mathbb{W}_i\}$, or the inference process based on the DNN should be computationally economical. This goal becomes particularly important when deploying these DNNs on embedded devices with limited computational resources and real-time processing constraints. While currently the network architectures leading to state-of-the-art performance in various classification tasks are highly over-parameterized, a major portion of the network parameters can often be pruned without significant deterioration in performance [96–98]. It has also been suggested that the over-parameterization of the network just provides ease of optimization during training [99]. Hence, one may wish to obtain certain desired characteristics in intermediate representations L that either help in training architecturally or computationally economical DNNs to achieve state-of-the-art performance, or that admit significant pruning after training without performance degradation. Besides, for specific applications it may be of interest that the classifier not only generalizes well but also embodies other operational goals such as fairness. Such goals may be achieved by enforcing additional desired properties for the learned intermediate representations.

Of course, these goals are not completely independent. For example, if a representation is robust and compressed, e.g., if the different regions in input domain $\mathbb{R}^N$ corresponding to different classes are mapped to clusters dense and far apart in the intermediate representation domain $R^{|L|}$, then it may be easier to find a simple decision rule to estimate Y from L. Such a representation L, however, may require an encoder f with significant architectural or computational complexity.

Since goals P1-P4 are formulated as properties of the intermediate representation, achieving them can be accomplished by designing regularizers for L based on, e.g., the joint distribution between X, Y, and L. Such regularization departs from classical regularization that depends on the parameters $\{\Theta_i\}$ of the DNN and relates closely to representation learning. Representation learning is an active field of research and various sets of desired properties for representations have previously been proposed. These are similar to our proposal but differ in subtle and key aspects.

In [100], Bengio et al. discussed desired characteristics of representations of the input in terms of invariant, disentangled and smoothly varying factors whereas our focus is on learning representations for a specific classification task. Nevertheless, P1-P4 have similarities to the properties discussed in [100]. For example, the hierarchical organization of explanatory factors discussed in [100] can lead to more abstract concepts at deeper layers. This subsequently may imply successively simpler decision functions required to estimate Y from L (P3). Similarly, [100] discusses invariance and manifold learning mainly in the context of auto-encoders, focusing primarily on X. Our P2 goes one step further by including Y in the picture, i.e., it aims to remove all information from L that is not useful for determining Y. In a geometric understanding of compression this could mean to collapse the input manifolds corresponding to different class labels to, for example,

separate dense clusters in $\mathbb{R}^{|L|}$ (as observed in [86, Fig. 2] and [85, Fig. 2]). This conforms to our earlier discussion about the relation between P2 and generalization via invariance to input variations on a manifold. Furthermore, in the context of representation learning, robustness is often related to denoising and contractive auto-encoders [100]. However, our P4 aims to learn representations that are robust for the classification task, whereas for auto-encoders the aim is to learn robust representations to recover the input.

The authors of [101] focused on formulating a similar set of desired properties in terms of information-theoretic objectives. Their approach involved considering also network parameters as RVs, unlike [19, 85, 86] and our work where only X, Y, and latent representations (which are transformations of X) are RVs. Their definitions share similar intuitive meaning as ours; e.g., sufficiency is equivalent to P1, minimality and invariance follow the same spirit as P2, and invariance can also be partially linked to robustness (P4). However, as we discuss in Sec. 5.3 (at least for the case when only X and Y are RVs, but not the network parameters), defining P2-P4 in terms of information-theoretic quantities may not imply characteristics in DNNs that are desired for a classification task.

Both [100] and [101] have introduced an additional desired property of representations that they call disentanglement. In the context of classification, disentanglement is meant to complement invariance (P2 and P4 in our case). Invariance is achieved by keeping the robust features which are relevant to the classification task, whereas disentanglement requires making the extracted relevant features independent (in the sense of total correlation [101] or some other metric). We have not included this property in our list for the following reasons: First, disentangling latent representations does not necessarily improve classification or generalization performance. Disentanglement also does not imply lower encoder or decoder computational or architectural complexity. Finally in order for disentanglement to lead to features that are more interpretable for humans, it would imply that the explanatory factors understandable by humans are independent. However, when dealing with the P_X for a specific classification task, features that are understandable for humans are not necessarily statistically independent (such as, e.g., size and weight of an object). We believe that more experiments are necessary to determine whether (and when) disentanglement, separated from other desirable properties, improves performance or human understandability of the latent representations for classification tasks. However, for generative models, aiming to fit a probabilistic model to the latent representation with independent features can be advantageous in terms of the computational ease to generate samples later from the latent representation.

5.3. Why and How IB Fails for Training Deterministic DNNs

In this section, we investigate the problem of learning an intermediate representation L_i (which can also be $L_{m+1} = \hat{Y}$) by a deterministic DNN with a given structure via

minimizing the IB functional, i.e., we consider[1]

$$\min_{\Theta_0,\dots,\Theta_{i-1}} I(X;L_i) - \beta I(Y;L_i). \tag{5.2}$$

The minimization is only over $\{\Theta_0 \dots \Theta_{i-1}\}$ as these are the only parameters that influence L_i and hence the objective, i.e., only the encoder f_i influences the objective, not the decoder h_i. The IB functional applied to DNNs focuses on P1 and P2, defining them via the mutual information terms $I(Y;L)$ and $I(X;L)$, respectively. Such an approach has been proposed by [19,20] and, subsequently, the IB framework has been suggested as a possible design principle for DNNs [19, 85–88]. It was claimed that on this basis compressed, simple, and robust representations can be obtained (see [86, Fig. 2] and [85, Fig. 2]).

Indeed, the intermediate layers of a DNN with good performance are characterized by a high $I(Y;L)$. However, they do not need to have small $I(X;L)$. For example, [92] propose invertible architectures which achieves state of the art performance, indicating that a small value of the IB functional is not necessary for good classification performance. Furthermore, since the IB framework was introduced to regularize intermediate representations rather than DNN parameters, a small value of $I(X;L)$ does not imply low architectural or computational complexity. Finally, small values of $I(X;L)$ do not relate causally to improved generalization performance, as it has been observed based on empirical evidence in [102].

In the following discussion, we show that applying the IB framework for training DNNs in this way suffers from two more major issues: The first issue is that, in many practically relevant cases, the IB functional is either equal to infinity or a piecewise constant function of the set of parameters $\{\Theta_i\}$. This either makes the optimization problem ill-posed or makes solving it difficult. We investigate these issues in Sec. 5.3.1 and Sec. 5.3.2. The second issue, which we investigate in Sec. 5.3.3, is connected to the invariance of mutual information under bijections and shows that focusing on goals P1 and P2 is not sufficient for a good classification system, at least when capturing P1 and P2 within the IB functional (5.2). Specifically, we show that minimizing the IB functional (5.2) does not necessarily lead to classifiers that are robust (P4) or that allow using simple decision functions (P3).

5.3.1. Continuous Features: The IB Functional is Infinite

Solving (5.2) requires that the IB functional can be evaluated for a set of parameters $\{\Theta_i\}$. Since Y is a discrete RV with finite support, the precision term $I(Y;L) \leq H(Y)$ is finite and can be computed (at least in principle). Suppose now that the distribution of the features X has an absolutely continuous component. The following theorem shows that, for almost every non-trivial choice of $\{\Theta_i\}$, the IB functional is infinite and, hence, its optimization is ill-posed. The proof is deferred to Appendix C.1.

[1] If the training is approached in a greedy manner, then Θ_0 to Θ_{i-2} are already fixed by training the previous latent representations. Then the minimization in (5.2) is only over Θ_{i-1}. The discussion in this section is independent of whether the training is based on a greedy approach, a joint approach or a combination thereof.

Theorem 5.1. Let $X = L_0$ be an n-dimensional RV, the distribution of which has an absolutely continuous component with a probability density function p_X that is continuous on a compact set $\mathcal{X}$ in $\mathbb{R}^n$. Consider a DNN as in the setup of Sec. 5.1. Suppose that the activation function σ is either bi-Lipschitz or continuously differentiable with strictly positive derivative. Then, for every $i = 1, \ldots, m$ and almost every choice of weight matrices $\mathbb{W}_0, \ldots, \mathbb{W}_i$, we have

$$I(X; L_{i+1}) = \infty. \tag{5.3}$$

In [102, Appendix C] it has been observed that the mutual information between the continuously distributed input X and an intermediate representation L becomes infinite if L has a continuous distribution. This assumption is often not satisfied: For example, the output of a ReLU activation function is, in general, the mixture of a continuous and a discrete distribution. Also, if the number of neurons $|L_i|$ of some layer exceeds the number of neurons of any preceding layer, or the dimension of the input X, then L_i cannot have a continuous distribution on $\mathbb{R}^{|L_i|}$ if the activation functions satisfy the conditions of Theorem 5.1. Therefore, our Theorem 5.1 is more general than [102, Appendix C] in the sense that continuity of the distribution L is not required.

Theorem 5.1 shows that the IB functional leads to an ill-posed optimization problem for, e.g., sigmoidal and tanh activation functions (which are continuously differentiable with strictly positive derivative) as well as for leaky ReLU activation functions (which are bi-Lipschitz). The situation is different for ReLU or step activation functions. For these activation functions, the intermediate representations L may have purely discrete distributions, from which follows that the IB functional is finite (at least for a non-vanishing set of parameters). As we discuss in Sec. 5.3.2, in such cases other issues dominate, such as the IB functional being piecewise constant.

Note that the issue discussed in this section is not that the IB functional is difficult to compute, as was implied in [85, 86]. Indeed, Theorem 5.1 provides us with the correct value of the IB functional, i.e., infinity, for almost every choice of weight matrices. At the same time, Theorem 5.1 shows that in such a scenario it is ill-advised to *estimate* mutual information from a data sample, as such estimators are valid only if the true mutual information determined by the assumed underlying distribution is finite. Indeed, the estimate $\hat{I}(X; L)$ reveals more about the estimator and the dataset than it does about the true mutual information, as the latter is always infinite by Theorem 5.1; see also the discussion in [102, Sec. 2 & Appendix C]

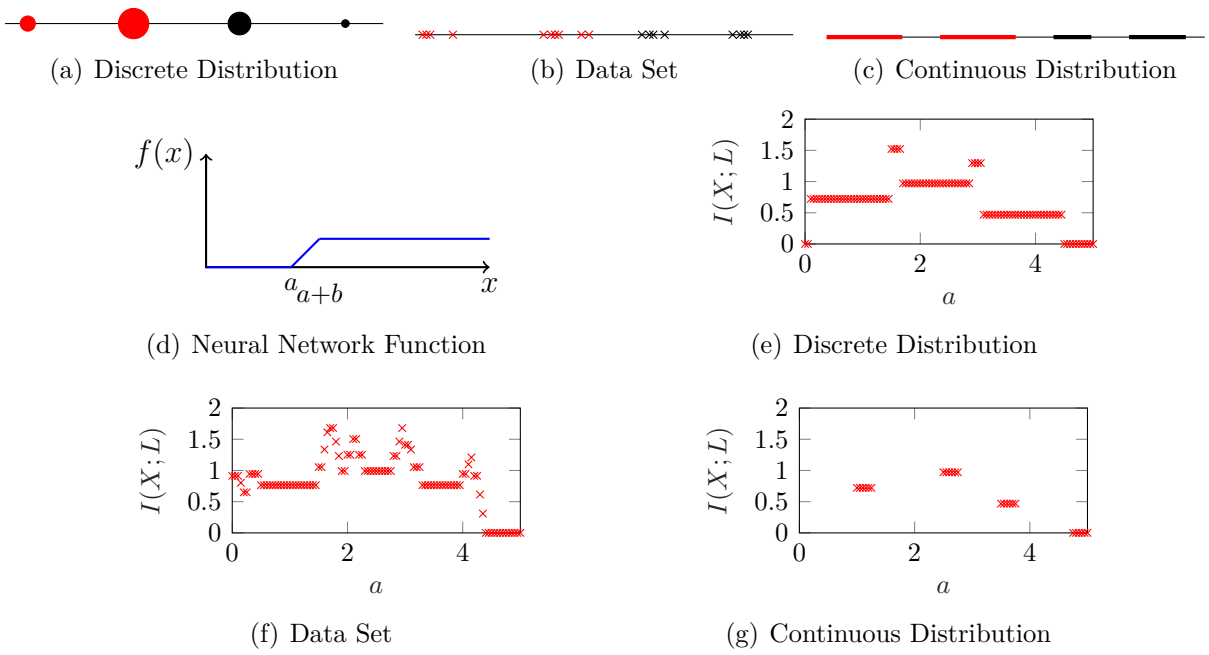


Figure 5.1.: (a)-(c): The line segment depicts the set $[0, 1]$, from which the feature RV X takes its values. Red (black) color indicates feature values corresponding to class $Y = 0$ ($Y = 1$). (a): The one-dimensional feature variable has a discrete distribution with mass points as indicated by the circles. The size of the circles is proportional to the probability mass. (b): Training based on a data set $\mathcal{D}$. Crosses indicate data points. (c): The one-dimensional feature variable has a continuous distribution with support indicated by the thick lines, the probability masses on each interval are identical to the probability masses of the points in (a). (d): The function f implemented by a DNN with a one hidden layer with two neurons, ReLU activation functions, and a single output neuron. The parameters leading to this function are $\Theta_0 = \{[1; 1], [-a; -a-b]\}$ and $\Theta_1 = \{[1, -1], 0\}$. (e)-(g) show the mutual information $I(X; L)$ as a function of the parameter a, for $b = 0.25$, evaluated on a grid of a ranging from 0 to 5 in steps of 0.05. It can be seen that the mutual information is piecewise constant. The missing values in (g) indicate that the mutual information is infinite at the respective positions.

5.3.2. Discrete Features or Learning from Data: The IB Functional is Piecewise Constant

We next assume that the features have a discrete distribution, i.e., X can take on only a finite number of different points in $\mathcal{X} \subset \mathbb{R}^n$. For example, consider a RV X over black-and-white images with N pixels, in which case the distribution of X is supported on $\mathcal{X} = \{0,1\}^N$. In such a case, the entropy of X is finite and so is the entropy of every intermediate representation L. More precisely, since the DNN is deterministic, the distribution of L is discrete as well, from which it follows that $I(X;L) = H(L)$ can assume only finitely many values. Similarly, since both L and Y are discrete, also $I(Y;L)$ can assume only finitely many different values. Indeed, $I(X;L)$ and $I(Y;L)$ may change only when two different $x \in \mathcal{X}$ that were previously mapped to different values of intermediate representation now get mapped to the same value or vice-versa. As a consequence, the IB functional is a piecewise constant function of the parameters $\{\Theta_i\}$ and, as such, difficult to optimize. Specifically, the gradient of the IB functional w.r.t. the parameter values is zero almost everywhere, and one has to resort to other optimization heuristics that are not gradient-based.

The problem of piecewise continuity persists if the empirical joint distribution of X and Y based on a dataset $\mathcal{D}$ with finitely many data points is used to optimize the IB objective. The entropy of X equals $\log|\mathcal{D}|$, and the IB functional remains piecewise constant. Indeed, $I(X;L)$ and $I(Y;L)$ may change only when two different data points that were previously mapped to different values of intermediate representation now get mapped to the same value or vice-versa. It was shown empirically in [102, Fig. 15] that $I(X;L) = \log|\mathcal{D}|$ throughout training, i.e., for a large selection of weight matrices.

Finally, the IB functional can be piecewise constant also for a continuously distributed feature RV X if step or ReLU activation functions are used. This can happen, for example, if the distribution of X is supported on a disconnected set $\mathcal{X} \subset \mathbb{R}^n$. Such a situation is depicted in Fig. 5.1 together with the scenarios of a discretely distributed feature RV X and a dataset $\mathcal{D}$.

5.3.3. Invariance under Bijections: The IB Functional is Insufficient

Leaving aside the fundamental problems discussed in Sec. 5.3.1 and Sec. 5.3.2, we now show that the IB functional is insufficient to fully characterize classification problems using DNNs. Specifically, we show that training a DNN by minimizing (5.2) does not lead to representations that admit simple decision functions (P3) or are robust to noise, well behaved transformations or small distortions (P4). To this end, we give several examples comparing two DNNs whose intermediate representations are equivalent in terms of the IB functional, but where one of them is clearly a more desirable solution. Since the IB functional does not give preference to any of the two solutions, we conclude that it is insufficient to achieve intermediate representations satisfying the requirements stated in Sec. 5.2. For the sake of argument, we present simple, synthetic examples instead of

empirical evidence on real-world datasets to illustrate these shortcomings. These examples not only have all the essential aspects associated with training a DNN for a practical classification task but also, because of the simplicity of the examples, they lend themselves to clearly highlighting and explaining different shortcomings in isolation. One can then easily extrapolate how one may encounter these issues in practical scenarios.

We consider a binary classification problem (i.e., $Y \in \{0, 1\}$) based on two-dimensional input X shown in Fig 5.2. The input RV (or samples) takes values in the four disjoint compact sets $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$, $\mathcal{S}_1$ and $\mathcal{S}_3$ (marked red) corresponding to $Y = 0$ and $\mathcal{S}_2$ and $\mathcal{S}_4$ (marked black) corresponding to $Y = 1$. We also define $\mathcal{T} = \mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3 \bigcup \mathcal{S}_4$. Perfect classification is possible in principle, i.e., the distributions of X given different classes have disjoint support[2] and $I(X;Y) = H(Y)$. The DNNs required to obtain the intermediate representations discussed in the examples can be easily implemented using ReLU activation functions (the examples can be modified to work with other activation functions). For the sake of simplicity, in this example the class label depends only on one dimension of the input.

Moreover, in the examples we evaluate the IB functional for the output $\hat{Y}$. The considerations are equally valid if the presented functions are encoders for a latent representation instead of $\hat{Y}$. Finally, one can extend the examples where the intermediate representation has more than one dimension.

We start by presenting examples to study the behaviour of the IB functional w.r.t. P3. consider the two functions $f'_{disc}(\cdot)$ and $f''_{disc}(\cdot)$ on the left-hand side (L.H.S.) in Fig. 5.2, implemented by two DNNs. The corresponding figures on the right-hand side (R.H.S.) show the support of the distribution of $\hat{Y}'_{disc}$ and $\hat{Y}''_{disc}$, i.e., to which X is mapped by f'_{disc} and f''_{disc}, respectively. It is easy to see that the IB functional evaluates to the same value for both DNNs. Indeed, both DNNs have identical compression terms, i.e., $I(X;\hat{Y}'_{disc}) = I(X;\hat{Y}''_{disc})$ and perfect precision, i.e., $I(Y;\hat{Y}'_{disc}) = I(Y;\hat{Y}''_{disc}) = H(Y)$. However, while f'_{disc} admits a simple decision by thresholding $\hat{Y}'_{disc}$ at $1/2$, the representation $\hat{Y}''_{disc}$ requires a more elaborate decision rule. This is true regardless of whether the input X has a continuous or discrete distribution supported on a subset of $\mathcal{T}$. It is also true if the computations are done based on a dataset with input samples lying in $\mathcal{T}$.

The same phenomenon can be observed when we compare the two functions $\hat{Y}'_{cont} = f'_{cont}(X_1)$ and $\hat{Y}''_{cont} = f''_{cont}(X_1)$ in Fig. 5.2, implemented by two DNNs. If the input has a continuous distribution supported on $\mathcal{T}$, this leads to the continuous output RVs $\hat{Y}'_{cont}$ and $\hat{Y}''_{cont}$ shown on the R.H.S. Again both DNNs have perfect precision and identical compression terms, where $I(X;\hat{Y}'_{cont})$ and $I(X;\hat{Y}''_{cont})$ are both infinite in this case due to a continuously distributed $\hat{Y}'_{cont}$ and $\hat{Y}''_{cont}$. However, f'_{cont} admits a simple decision by thresholding $\hat{Y}'_{cont}$ at $1/2$, whereas $\hat{Y}''_{cont}$ requires a more elaborate decision rule.

Next we turn to the question of robustness against noisy inputs. This, in general, cannot be answered by looking at the intermediate representations alone as we show in the following two examples. To this end, consider first the situation depicted in Fig. 5.3. As it

[2]We refer the reader to [103] regarding additional issues under which the IB framework suffers in this scenario.

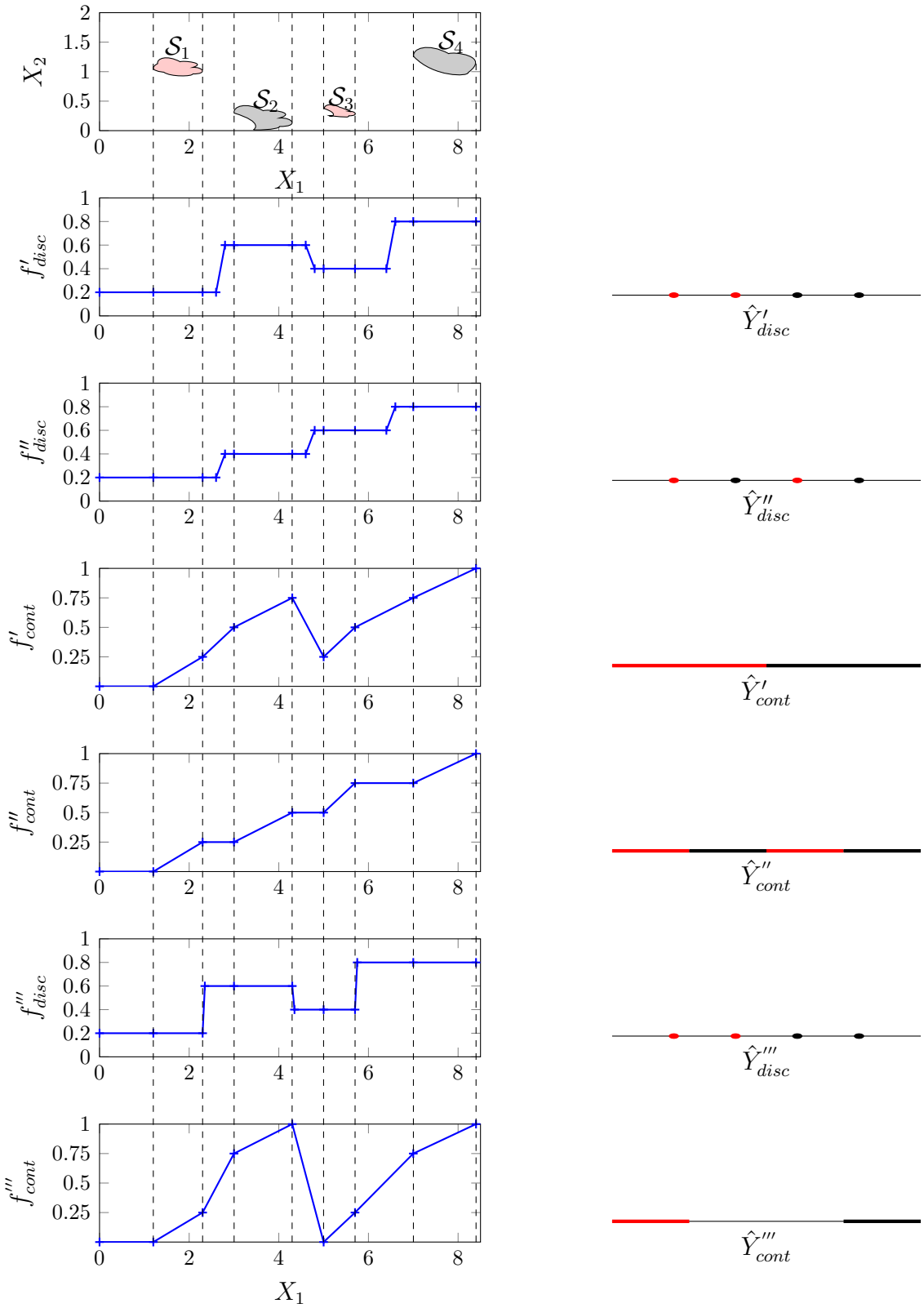


Figure 5.2.: Representational simplicity and robustness in binary classification: The top figure on the L.H.S. illustrates the two-dimensional input space and the support of the input X in $\mathbb{R}^2$. The rest of the figures on the L.H.S. show various functions of X_1 (since Y only depends on X_1) implementable using a ReLU-based DNN. The figures on R.H.S. show the output RVs $\hat{Y}$ when X is transformed via the corresponding functions on the L.H.S. Red (black) color indicates feature values corresponding to class $Y = 0$ ($Y = 1$).

Figure 5.3.: Robustness in binary classification: The L.H.S. shows the feature space, with X_1 on the horizontal and X_2 on the vertical axis. One can see that X is distributed on $[0, 1/2] \times [0, \varepsilon]$ if $Y = 0$ and on $[0, \varepsilon] \times [1/2, 1]$ if $Y = 1$, for $0 < \varepsilon \ll 1$. The R.H.S. shows the supports of the distributions $P_{\hat{Y}|Y=0}$ and $P_{\hat{Y}|Y=1}$, obtained by two different DNNs with identical IB functionals. The blue dot represents a noisy feature or a data point not in the training set. See the text for details.

can be seen, perfect classification is possible with a single neuron with a ReLU activation function. We consider two different DNNs with no hidden layers and single output neuron with parameterizations $\Theta_0^I = \{[1, 1], 0\}$ and $\Theta_0^{II} = \{[0, 1], 0\}$. Both parameterizations are equivalent in terms of the IB functional, leading to identical precision and compression terms. Note, however, that the DNN f^{II} parameterized by Θ_0^{II} is more robust to small amounts of noise or distortion than f^I. This can be seen by the blue dot in Fig. 5.3 indicating a noisy input generated (with high probability) by class label $Y = 1$. While f^I does not admit distinguishing this point from features generated by class label $Y = 0$, f^{II} does (see R.H.S. of Fig. 5.3). Indeed, thresholding $\hat{Y}^I$ at 0.5 and $\hat{Y}^{II}$ at 0.25 yields the decision regions indicated by dashed and dotted lines on the left of Fig. 5.3.

As a second example, consider the two DNNs implementing the functions f'_{disc} and f'''_{disc} in Fig. 5.2. The corresponding RVs $\hat{Y}'_{disc}$ and $\hat{Y}'''_{disc}$ for the given X are shown on the R.H.S. respectively. For the given distribution of X, we notice that $\hat{Y}'_{disc} = \hat{Y}'''_{disc}$, i.e., the two DNNs implement the same function over the support of X. Adding noise to X or distorting X has the potential to enlarge the support of its distribution, but for small amounts of noise we still have disjoint support sets given different classes, hence implying that perfect classification is still possible in principle. In this case, f'_{disc} will be more robust to such noise and distortions when compared to f'''_{disc}, due to the sharp transitions of f'''_{disc} on the borders of current support of the distribution of X. This holds true whether the input X is continuous or discrete with support over a subset of $\mathcal{T}$. It also holds if the computations are done based on a dataset with input samples lying in $\mathcal{T}$. These two examples show that in order to study an intermediate representation in terms of robustness we have to study not only the representation itself but also the it's encoding function and the behaviour of the encoding function/intermediate representation in the neighbourhood of the current support of X.

In conclusion, the IB framework may serve to train a DNN whose output is maximally

compressed (in an information-theoretic sense) and maximally informative about the class. However, one cannot expect that the DNN output admits taking a decision with a simple function, or that the DNN is robust against noisy or distorted features. This is true also for DNNs with discrete-valued features X, such as those discussed in [19], for which the issue of Sec. 5.3.1 does not appear. Based on the link between robustness and generalization, this suggests that the IB functional cannot be used to quantify generalization of a DNN during training, which is in line with the experimental observations in [102].

5.4. How to Use IB-Like Cost Functions for Training DNNs

The issues we discussed in Sec. 5.3 apply to training deterministic DNNs. In this section we discuss possible remedies for these problems that can guarantee that the IB functional is finite and that specific pairs of intermediate representations, related by invertible transforms, are not equivalent anymore, which subsequently implies that these approaches may be able to capture robustness (P3) and simplicity (P4) better. These approaches are inspired either by how such DNN based classification systems are used in practice, and by our observations in Sec. 5.3. However, some of the proposed approaches lead to the IB functional being piecewise constant, similar to the scenario in Sec. 5.3.2.

5.4.1. Including a Hard Decision Rule

One approach to successfully apply the IB framework for DNN training is to include the hard decision rule, that is used to translate the DNN output $\hat{Y}$ to a predicted class label, into the IB functional. Specifically, when applied to the output representation $\hat{Y}$, for a given hard decision rule, represented by the deterministic function $\delta\colon \mathbb{R}^{|\hat{\mathcal{Y}}|} \to \mathcal{Y}$ and $\tilde{Y} = \delta(\hat{Y})$, the goal shall be to solve

$$\min_{[\Theta]} I(X;\tilde{Y}) - \beta I(Y;\tilde{Y}). \tag{5.4}$$

For example, for binary Y and a single output neuron, i.e., one dimensional $\hat{Y}$, one could set $\tilde{Y} = \mathbb{1}[\hat{Y} > 0.5]$, where $\mathbb{1}[\cdot]$ is the indicator function; for $|\mathcal{Y}| > 2$ output neurons, $\tilde{Y}$ could be the index of the output neuron (component of $\hat{Y}$) with the maximum value.

Since $\tilde{Y}$ has a discrete distribution with as many mass points as the class variable, the compression term $I(X;\tilde{Y})$ appears useless: compression is enforced by including the decision rule δ. Similarly, the simplicity of the representation is automatically enforced by the simplicity of the decision function δ when solving (5.4). Moreover, the IB functional becomes computable for the output layer because we have $I(X;\tilde{Y}) \leq H(\tilde{Y}) \leq \log|\mathcal{Y}|$. Finally, for the example depicted in Fig. 5.2, assuming that $\mathbb{P}(X \in \mathcal{S}_i) = \frac{1}{4}$ for $i \in \{1,2,3,4\}$ and setting $\tilde{Y}'_{cont} = \mathbb{1}[\hat{Y}'_{cont} > 0.5]$ and $\tilde{Y}''_{cont} = \mathbb{1}[\hat{Y}''_{cont} > 0.5]$ clearly favors the first option over the latter: While we still get $I(X;\tilde{Y}'_{cont}) = I(X;\tilde{Y}''_{cont}) = 1$ (finite now due

to quantization), we have $I(Y;\tilde{Y}'_{cont}) = H(Y)$ but $I(Y;\tilde{Y}''_{cont}) = 0$.

Including the decision rule, however, does not lead to improved robustness. Indeed, consider Fig. 5.3. Then, the resulting RVs $\tilde{Y}^I = \mathbb{1}[\hat{Y}^I > 0.5]$ and $\tilde{Y}^{II} = \mathbb{1}[\hat{Y}^{II} > 3/8]$ are identical (and, thus, so are the IB functionals), with the former suffering from reduced robustness. The same can be observed by looking at $\tilde{Y}'_{disc} = \mathbb{1}[\hat{Y}'_{disc} > 0.5]$ and $\tilde{Y}'''_{disc} = \mathbb{1}[\hat{Y}'''_{disc} > 0.5]$. Furthermore, this shows that including a decision rule, due to the coarse quantization of $\hat{Y}$, leads to large equivalence classes of DNNs that evaluate to the same value in (5.4), which is conceptually similar to the IB functional being piecewise constant (cf. Sec. 5.3.2 and [20, Sec. 3.5]).

To apply this method to train intermediate representations other than $L_{m+1} = \hat{Y}$, one possible approach is to feed the latent representation L to an auxiliary decision rule $\delta\colon \mathbb{R}^{|L|} \to \mathcal{Y}$ and minimize (5.4) for $\tilde{Y} = \delta(L)$. Auxiliary decision rules have been utilized in the past, for example in GoogLeNet [104], to assist in training latent layers of very deep architectures, albeit using traditional cost functions.

5.4.2. Including a Soft Decision Rule

Another option related to Sec. 5.4.1 is to introduce a soft decision rule, that takes $\hat{Y}$ as input and outputs a probability distribution over $\mathcal{Y}$, into the IB functional. The soft decision rule can be considered as a simple mechanism to generate a variational approximation $Q_{Y|\hat{Y}}$ of the true posterior $P_{Y|\hat{Y}}$. $Q_{Y|\hat{Y}}$ can then be used to define $\tilde{Y} \sim Q_{Y|\hat{Y}}$, which is a discrete RV with alphabet $\mathcal{Y}$ that depends stochastically (and not deterministically) on the feature vector X. For example, in a one-vs-all classification problem with $\hat{Y}$ generated by softmax layer with $|\mathcal{Y}|$ neurons, the i-th entry of $\hat{Y}$ can be interpreted as the probability that $\tilde{Y} = i$. This $\tilde{Y}$ can then be used to compute (5.4).

Using this approach not only guarantees that the functional in (5.4) is finite but also, unlike in Sec. 5.4.1, admits applying gradient-based optimization techniques even for finite datasets. Moreover, using a simple soft decision rule, e.g., the one discussed earlier, makes the precision term sensitive to simplification, encouraging this property in $\hat{Y}$. The precision term also promotes P2 in a geometric sense. In the above example of softmax output $\hat{Y}$, in the case when Y is a deterministic function of X, the precision term will encourage $|\mathcal{Y}|$ dense clusters on the corners of a $|\mathcal{Y}| - 1$ probability simplex. These claims can be verified, for example, by looking at $\hat{Y}'_{cont}$, $\hat{Y}''_{cont}$, and $\hat{Y}'''_{cont}$ in Fig. 5.2 (assuming $\hat{Y}'_{cont}$, $\hat{Y}''_{cont}$, and $\hat{Y}'''_{cont}$ are uniform over their support), and identifying values of $\hat{Y}$ as probabilities that $\tilde{Y} = 1$. The role of the compression term $I(X;\tilde{Y})$ remains questionable and becomes more complicated than in Sec. 5.4.1. On the one hand, $\tilde{Y}$ is discrete which automatically enforces implicit compression. Moreover, $I(X;\tilde{Y}) = H(\tilde{Y}) - H(\tilde{Y}|X) = H(\tilde{Y}) - H(\tilde{Y}|\hat{Y})$ is smaller for $\hat{Y} = \hat{Y}'_{cont}$ than for $\hat{Y} = \hat{Y}'''_{cont}$ in Fig. 5.2, hence minimizing $I(X;\tilde{Y})$ now prefers f'_{cont} over f'''_{cont} rather than evaluating them equally. $I(X;\tilde{Y})$ prefers more uniform $Q_{Y|\hat{Y}}$, hence less confident predictions just to reduce $I(X;\tilde{Y})$, even when Y is a deterministic function of X and the classes are well separated at $\hat{Y}$. On the other hand, this inclination towards less confidence predictions makes $I(X;\tilde{Y})$ plays a role similar to label smoothing [18, Sec. 7.5],

which can be beneficial to avoid overfitting in the case of a small dataset if carefully done. Therefore, if one decides to keep $I(X;\tilde{Y})$ in (5.4), the choice of β and the effect of the compression term should be carefully monitored.

Note that during training we do need to define and deal with RV $\tilde{Y}$ based on $Q_{Y|\hat{Y}}$ in order to not only have a finite IB functional, but also to ensure desirable properties such as simplification. However, during inference we can directly utilize the variational approximation $Q_{\hat{Y}|Y}$ as our soft estimate and there is no need for $\tilde{Y}$.

To apply this method to a latent representation, one possible approach is to feed the latent representation L to a linear layer of size $\mathcal{Y}$ followed by a softmax layer to generate $\tilde{Y}$.

5.4.3. Stochastic DNNs

A further approach is to use the IB functional to train stochastic DNNs rather than deterministic ones. A DNN can be made stochastic by, e.g., introducing noise to the intermediate representation(s). The statistics of the noise can themselves be considered trainable parameters or adapted to (the statistics of) the intermediate representation(s). The objective function to be optimized remains (5.2). For $I(X;L)$ to be finite, it suffices to add noise with an absolutely continuous distribution to L. Depending upon where and what type of noise is introduced, $I(Y;L)$ can encourage different notions of robust representations, for which $I(Y;L)$ does not degrade by introduction of noise or deformations. Similarly $I(Y;L)$ may also promote an intermediate representation with well separated (sub)regions corresponding to different labels, which in turn admits simpler decision functions for the stochastic representations. For example, a small amount of uniform noise added to the intermediate representation leads to a better IB functional for f'_{cont} than f''_{cont} and for f'''_{cont} than f'_{cont} in Fig. 5.2. For stochastic DNNs, the compression term $I(X;L)$ can encourage more compact representations. For example, again in Fig. 5.2, adding a small amount of noise N to the output makes $I(X; f'_{cont}(X_1)+N)$ larger than $I(X; f'''_{cont}(X_1)+N)$, making the latter representation more desirable. Note that in case of stochastic DNNs, the noisy intermediate representation, such as $f'_{cont}(X_1)+N$ or $f''_{cont}(X_1)+N$, is considered L and fed as input to the next layer of the DNN.

In addition to resolving the issues associated with IB functional mentioned in Sec. 5.3, training a stochastic DNN in such a way also provides a novel way of data augmentation. Sampling the intermediate representation L multiple times during training for each input sample can be viewed as a way of dataset augmentation, which may lead to improved robustness. Introducing noise in a latent (bottleneck) representation thus presents an alternative to the data augmentation approach proposed in [105], which requires training a separate auto-encoder to obtain latent representations to be perturbed by noise.

5.4.4. Replacing the IB Functional

A final approach is to replace the IB functional by a cost function that is more well-behaved, but motivated by the IB framework. Specifically, by replacing mutual information by (not

necessarily symmetric) quantities $\bar{I}_C(\cdot;\cdot)$ and $\bar{I}_P(\cdot;\cdot)$, we replace (5.2) by

$$\min_{\{\Theta_i\}} \bar{I}_C(X;L) - \beta \bar{I}_P(Y;L). \tag{5.5}$$

This approach can be used for training the DNN both as a whole and layer-wise.

We first consider setting $\bar{I}_C(X;L) = I(\nu_X(X);\nu_L(L))$ and $\bar{I}_P(Y;L) = I(Y;\nu'_L(L))$, where ν_X, ν_L, and ν'_L are quantizers, that are adapted according to the statistics of the latent representation L and w.r.t. one another[3]. It is important to note that the quantization is not performed inside the DNN, but only for computing the function in (5.5). This is the typical approach performed when mutual information is estimated from finite datasets using histogram-based methods. Unlike [20], we argue that the design of ν_X, ν_L, and ν'_L should not be guided by the goal of estimating the true mutual information (which is bound to fail according to our analysis in Sec. 5.3.1), but by the aim to instill the desired properties from Sec. 5.2 into the cost function (5.5).

The effect of quantization is that $\bar{I}_C(X;L)$ becomes finite. Moreover, if ν'_L is set appropriately, solving (5.5) leads to simpler representations. Considering again Fig 5.2, setting $\nu'_L(\cdot) = \mathbb{1}[\cdot > 0.5]$ prefers f'_{disc} over f''_{disc} (and similarly f'_{cont} over f''_{cont}); however, the finer the quantization ν'_L is, the less sensitive is (5.5) to the simplicity of the intermediate representation L.

The fact that ν_L and ν'_L need not coincide yields an advantage over the solution proposed in Sec. 5.4.1 in the sense that the compression term can become useful now. With the above choice of ν'_L we see that the precision term is the same for f'_{cont} and f'''_{cont}. However, if ν_X is the identity function and ν_L a uniform quantizer with four quantization levels in $[0,1]$, then f'_{cont} leads to a larger compression term than f'''_{cont}, thus favoring f'''_{cont}. However, as we observed in Sec. 5.4.1, the quantized IB functional partitions DNNs into large equivalence classes that do not necessarily distinguish according to robustness. Additionally, the quantized IB functional is piecewise constant when used for finite datasets. Finally, choosing appropriate quantizers is not trivial; e.g., the effect of this choice has been empirically evaluated in [102], with a focus only on the compression term. For the quantized IB functional, choosing quantizers becomes even more complicated.

Without going into details, we note that computing a noisy IB functional, for example by setting $\bar{I}_C(X;L) = I(X;L+N)$ and $\bar{I}_P(Y;L) = I(Y;L+N')$ for noise variables η and η' that are adapted according to (the statistics of) L and w.r.t. each other[3], can lead to simplified and compact representations. In contrast to the quantized IB functional, the noisy IB functional can even lead to robust representations and, for appropriately chosen noise models, is not piecewise constant for finite datasets, hence admitting efficient optimization using gradient-based methods. Again, we note that noise is not introduced inside the DNN, but only in the computation of (5.5); hence, the DNN is still deterministic.

Other than quantization and introducing noise in the computation of the mutual infor-

[3]It is important to adapt the quantizers (or the noise) to (the statistics of) the latent representation L in order to rule out ways to decrease the cost without fundamentally changing the characteristics of L, e.g., by simple scaling.

mation terms, one may go one step further and replace these terms with different quantities. For example, it is common to replace the precision term by the cross-entropy loss between the true posterior $P_{Y|L}$ and a variational approximation. Moreover, also the compression term can be replaced by terms that are inspired by $I(X;L)$, but differ in essential details. These changes to the optimization problem often directly enforce goals such as P2-P4, even though they have not been captured by the original IB functional based optimization.

Finally, when replacing the two terms in (5.2) with different quantities, one may even choose to select different intermediate representations for the precision term and for the compression term. For example the compression term can be defined based on a latent representation and the precision term can be defined based on $\hat{Y}$. The compression term can then enforce desired properties on the latent representation whereas the precision term can ensure that the output of the DNN $\hat{Y}$ admits simple decisions and predicts Y well enough. In contrast, evaluating (5.2) only for an internal representation L trains only the encoder, failing to instill desired properties into $\hat{Y}$; evaluating (5.2) only for the output $\hat{Y}$ trains the whole DNN, but does not necessarily lead to internal representations L with the desirable properties from Sec. 5.2.

The approaches in this section are not completely independent. For example, on the one hand, a probabilistic interpretation of the output (Sec. 5.4.2) can be considered to be a special type of stochastic DNN in which the stochasticity appears only in the output neurons (or the auxiliary output neurons) and only during training. On the other hand, evaluating the IB functional for this probabilistic interpretation can be considered as replacing the IB functional with a different cost function. This is in line with the reasoning in [106], illustrating that the same problem may be solved equivalently by adapting the optimization method, the feasible set, or the cost function.

Although (5.2) and the variants suggested in this section deal with encoders of one intermediate representation at a time, leading to greedy or semi-greedy (if we do not fix the previous representations already trained by their IB functionals but let them be fine tuned by IB functionals of the deeper layers) layer-wise training, but for all of these methods including the original IB formulation, we can form a weighted sum of IB functionals (or it's variants proposed in this section) for different intermediate representations and use this weighted sum as the training objective to optimize all the layers jointly. Such an approach of weighted objectives was also employed in [104].

A common theme in our remedies from Secs. 5.4.1 to 5.4.4 is that they encourage latent representations in which data points from different classes are represented in some geometrically compact manner. In other words, the proposed remedies encourage compression (P2) in a geometric sense rather than in the sense of a minimal sufficient statistic. This is intuitive, since representing classes by clusters tight and far apart allows using simple decision rules for classification (P3). While such clustering does not immediately ensure robustness (P4), the injection of noise (either directly or only in the computation of the IB functional, cf. Sec. 5.4.3 and 5.4.4) does.

All this certainly does not imply that measuring P2 in information-theoretic terms is

inadequate. Rather, it illustrates that measuring P2 in information-theoretic terms is insufficient to instill desirable properties such as simple decision functions or robustness, while understanding P2 in geometric terms has the potential to do so. Besides, invariance and compression (P2) in a geometric sense has also been linked to generalization [94].

5.5. Critical Discussion of and Experimental Evidence from Related Work

In this section, we not only critically assess and provide insights into the related work in light of our Sec. 5.3, but we also discuss, where relevant, how some of these works provide experimental evidence to support the approaches we propose in Sec. 5.4. These works report successes in terms of different operational goals, such as generalization, adversarial robustness, and out-of-distribution data detection, that are directly relevant for applying classifiers in practice. Therefore, this shows that our proposed remedies are not only successful in instilling desired characteristics in intermediate representations, such as P3 and P4, but are also, via these characteristics of the intermediate representations, able to successfully achieve various operational goals. Hence, although our approach is mainly focused on providing analytical understanding via theory and intuitive examples, we also rely on the experimental evidence from other works to support our claims.

The idea of using the IB framework for DNNs was first introduced in [19]. They proposed the application of IB functional for both analysis as well as training DNNs. In the context of training, [19] mentioned that this approach provides information-theoretic optimization criteria for obtaining *optimal* DNN representations.

[107] proposed deterministic IB functional which replaces $I(X;L)$ by $H(L)$. Since we deal with deterministic DNNs in Sec. 5.3, we have $I(X;L_i) = H(L_i)$. The IB functional thus coincides with the deterministic IB functional proposed in [107] and the discussion in Sec. 5.3 is also applicable for deterministic IB.

The authors of [108] studied the latent representations (obtained via training a DNN using a standard loss function) in the context of $I(X;L)$. They inject Gaussian noise at the output of every neuron and show that, in this case, a geometric clustering of the latent representation is captured well by both the compression term $I(X;L)$ and the entropy of the quantized latent representation, $H(\nu_L(L))$. As we mention at the end of Sec. 5.4, encouraging geometric clustering has the potential to directly instill the desirable properties of simple decision rules (P3) and robustness (P4) into latent representations. The observations in [108] therefore support our proposal to either use a stochastic DNN (Sec. 5.4.3) or to replace the cost function (e.g., via quantized entropy) to instill desirable properties into the latent representations (Sec. 5.4.4).

Reference [109] uses $I(X;L)$ to bound the generalization gap from above, hence establishing a connection between $I(X;L)$ and one of the most important operational goals while training DNNs, i.e., generalization. This could provide a strong justification of using the IB functional for training DNNs but, although theoretically interesting, their bound

relies on strong assumptions such as X and L being discrete and the use of an "optimal" decoder for the latent representation L. Furthermore, the upper bound accounts only for the generalization gap and not the actual performance (which can be bad despite a small generalization gap). Finally, the upper bound is infinite, and hence not useful, in the setting of deterministic DNNs with continuous X and L that was discussed in Sec. 5.3.

We next turn to works that train DNNs using cost functions inspired by the IB principle. The authors of [106] proposed minimizing the IB functional based on parametric distributions for the (stochastic) encoder and decoder, i.e., combining approaches from Secs. 5.4.3 and 5.4.4. They showed that minimizing the cost function (regularized by total correlation to encourage disentangled representations) is equivalent to minimizing cross-entropy over DNNs with multiplicative noise (dubbed *information dropout*). They also discovered that, for a certain choice of the parameter β and for the goal of reconstruction, i.e., $Y \equiv X$, the regularized cost function is equivalent to the one for variational auto-encoding [110].

Reference [86] trained a stochastic DNN using a variational upper bound on the IB functional and showed that the resulting DNN has state-of-the-art generalization performance, as well as improved robustness to adversarial examples. They introduce noise at a dedicated bottleneck layer (cf. Sec. 5.4.3), leading to a stochastic DNN with finite IB functional for the bottleneck and the subsequent layers. The authors then replace the compression term $I(X;L)$ for the bottleneck layer with a variational upper bound (cf. Sec. 5.4.4) to make the compression term tractable; the resulting term is no longer invariant under bijections and encourages bottleneck representations that are compact in a geometric sense. They further replace the precision term $I(Y;L)$ by cross-entropy loss. This can be interpreted as two steps applied sequentially, namely first lower bounding $I(Y;L)$ by $I(Y;\hat{Y})$ and then lower bounding $I(Y;\hat{Y})$ by cross-entropy loss with a probabilistic interpretation of the output $\hat{Y}$ [4] (cf. Sec. 5.4.2). Combining the bounds on compression and precision terms thus instill desirable properties in both the bottleneck representation L and the output representation $\hat{Y}$ (cf. end of Sec. 5.4.4). Unlike $I(Y;L)$ and $I(Y;\hat{Y})$, cross-entropy applied to a probabilistic interpretation of the output is no longer insensitive to bijections and, in conjunction with noise introduced at the bottleneck layer, enforces simplicity of the decision rule and robustness in the trained DNN (see [86, Fig. 2]).

The authors of [85] optimized the IB functional using stochastic DNNs, which is closely related to training stochastic DNNs using the IB functional. They followed a very similar approach as the authors of [86], with the main difference that they replace the variational bound on the compression term by a non-parametric bound. They show that the intermediate representations they obtain form geometrically dense clusters as compared to the representations of DNNs trained using traditional cost functions (see [85, Fig. 2]).

The authors of [88] used the same technique as in [86] to train stochastic DNNs but they measure the performance of the DNNs in terms of classification calibration as well as the DNN's ability to detect out-of-distribution data. Since the training technique is the same as in [86], our discussion of [86] also applies here.

[89] also uses a setup similar to [86]. For example, they introduce Gaussian noise at

[4] See Sec. 5.8 for a detailed discussion of this two step perspective.

the latent representation (Sec. 5.4.3) and approximate $I(X;L)$ similarly. However, they replace the precision term $I(Y;L)$ in the IB functional with a term that quantifies channel deficiency based on Kullback-Leibler divergence and use a tractable approximation for this new precision term (cf. Sec 5.4.4).

In summary, the authors of [85, 86, 88, 89] propose cost functions motivated by the IB framework but depart from it by using a combination of techniques from Sec. 5.4. Their promising performance in terms of various operational goals is therefore experimental evidence of the success of our proposed remedies, as it cannot be attributed to the IB functional itself based on our discussion in Sec. 5.3. Note that these works do not explicitly mention that the IB functional leads to an ill-posed optimization problem the solution of which lacks desirable properties such as representational simplicity and robustness. Instead they introduced the aforementioned modifications to obtain tractable bounds on the IB functional that can be optimized using gradient based methods.

5.6. Regularizing Intermediate Representations

We believe that the idea of regularization introduced by the IB functional, i.e., to regularize the intermediate representations L rather than the parameters $\{\Theta_i\}$ of the DNN, has great potential, as we elaborate in this section[5].

The motivation to regularize parameters $\{\Theta_i\}$ of a DNN partially comes from the goal of introducing a soft limitation on the hypothesis class representable by the DNN. The motivation to restrict the complexity of the hypothesis class representable by a DNN further stems from the relation between generalization and traditional complexity measures. These measures focus on what a DNN can do based on the network architecture while ignoring the (estimated) data statistics $P_{X,Y}$ and the actual function implemented by the DNN, i.e., the network parameters after training on the actual data. For example, VC dimension [111] is independent of $P_{X,Y}$ and the learned network parameters, and the generalization error bound based on Rademacher complexity [112], only depend on (estimated) P_X, ignoring $P_{Y|X}$ and the learned network parameters. However, these traditional complexity measures and associated generalization bounds fail to explain why largely over-parameterized DNNs perform well although they are capable of memorizing the whole dataset [81, 94, 101]. The discussion in [113] suggests that a perspective of the DNN, which also involves the (estimated) relation between X and Y as well as the learned network parameters, can lead to more meaningful insights in terms of its generalization performance; the methods employed in [92, 94, 95] for explaining and understanding the success of such networks explicitly or implicitly focus on the learned representations for the specific task. This hints at the fact

[5]Dropout [82] also regularizes latent representations but it does not directly focus on instilling specific desired characteristics in the latent representation. Instead, Dropout takes an ensemble approach of getting redundant and robust latent representations by randomly eliminating neurons in latent representations during training. In this chapter we do not delve into such approaches of regularizing latent representations. We rather focus on methods that directly aim to instill specific desired characteristic(s) in intermediate representation(s).

that regularization based on properties of intermediate representations instead of solely focused on network parameters can be beneficial. The experiments in [108] also support this claim by arguing that geometric clustering of the latent representations is a valid goal for training DNNs for classification.

Several regularizers trying to instill desired characteristics directly in an intermediate representation (without necessarily being motivated by the IB principle) have been proposed in the recent literature. Reference [83] introduces $I(C;L)$, where C is some nuisance factor or discriminatory trait, as a regularizer. Minimizing it aims at making the latent representation L and the performance of the DNN invariant to C. The authors evaluate this regularization in a variational auto-encoding setup and as an additional term in the variational IB setup of [86]. Depending upon the relation between X and C, the term $I(C;L)$ may also suffer from the issues discussed in Sec. 5.3.1 or Sec. 5.3.2 for deterministic DNNs. In [114], the authors regularized the final softmax output, interpreted as a probability distribution over $\mathcal{Y}$ (cf. $\tilde{Y}$ in Sec. 5.4.2), by $H(\tilde{Y}|X)$ to penalize overly confident output estimates. The experiments in [90] suggest that back-propagation of classification error or a similar loss function from the output does not lead to latent representations with properties such as discrimination and invariance (which are intuitively similar to P1 and P3 and to P2 and P4 from Sec. 5.2, respectively). They propose a "hint penalty" that encourages latent representations being similar if they correspond to the same class. Similarly, [91] defined the regularization on latent representations as a label-based clustering objective, which is conceptually similar to defining the goal of compression (P2 in Sec. 5.2) in a geometric sense. The authors of [91] discuss the performance of such regularizers for various problems including auto-encoder design, classification, and zero shot learning.

In addition to improved generalization, representation-based regularizers also provide a flexible way to incorporate additional operational goals and sometimes lead to DNNs satisfying these goals without explicitly imposing them. For example, [86] demonstrated enhanced adversarial robustness for such networks, although the networks were not specifically trained to be adversarially robust. Such regularization also provides a more flexible data augmentation method for training and inference as compared to the fixed transformations done at the input currently in practice, e.g. rotation, translation etc. For example, the authors of [85,86] sample the noisy bottleneck representation L multiple times for each input training example. This data augmentation mechanism also yields an advantage over the one introduced in [105] by obviating the need to train a separate auto-encoder and gets automatically adapted to the classification task at hand. Regularizers can also be used to enforce privacy guarantees or to ensure insensitivity to transformations such as rotations, translations, etc. which is attributed to be one cause of the superior performance of DNNs [101].

All of these works and those discussed in Sec. 5.5 provide empirical evidence that regularizing latent representation(s) is a promising endeavor, achieving generalization, robustness, fairness, classification calibration, and data augmentation. The discussion at the end of Sec. 5.4 and in [108] along with the empirical evidence from [90–92] leads us to believe that defining a latent representation regularizer in a geometric sense in conjunction with noise/stochasticity is a promising domain of future research. Such a direct geometric de-

sign can also ensure compatibility with standard optimization tools used in deep learning, such as gradient-based training methods. On the other hand, although regularizing latent representations is a key feature of the IB framework, it fails to instill desired properties in the latent representations. The success of invertible DNNs, e.g., iRevnet [92], and our analyses suggest that the information-theoretic compression-based regularization term $I(X;L)$ either becomes obsolete or has to be replaced. Similarly, although intuitively often attractive, if one aims to define the latent regularizer via some other information-theoretic cost (e.g., as in [83]), it is important to mitigate issues including, but not limited to, the ones discussed in Sec. 5.3.1, Sec. 5.3.2, and Sec. 5.3.3. In contrast, a restriction to a specific prior distribution and approximations being used to evaluate the information-theoretic cost lead to a more direct and intuitive geometric interpretation (e.g., as observed in [85, 86]) which can be utilized. Thus, in light of the discussion in this work and in [92, 94, 108] along with the empirical evidence in [90, 91], we conclude that designing regularizers directly with the aim of instilling certain properties desirable for the intermediate representation L (such as discussed in Sec. 5.2) may be a more fruitful approach than trying to repair the problems inherent in the IB functional (or other information-theoretic cost functions) in the context of classification.

5.7. Analysing DNNs via IB principle

While the main focus of this chapter is on training DNNs via the IB functional, in this section we briefly discuss the implications of our findings on another area of research sparked by [19]: Analyzing DNNs using the IB framework. The authors of [19] proposed using the IB functional to analyze DNNs in terms of performance as well as architectural compactness, and that this can be done not only for the output but also for hidden layer representations of a DNN.

In [20], the authors applied these ideas to analyze DNNs trained using cross-entropy without regularization. They empirically observed that compression (in the sense of a small $I(X;L)$) cannot be linked to architectural simplicity. Furthermore, based on their empirical observations, they claimed that training includes a compression phase that, they believe, is causally linked to improved generalization performance of the DNN. The authors of [102] present empirical evidence contradicting this claim, which initiated a debate that is still ongoing. Moreover, the authors of [102] discussed analytically and empirically that the compression phase observed in [20] is an artifact of the quantization strategy used to approximate the compression term in connection with the activation function used. They also briefly looked at the computability issues of the compression term in the IB functional [102, Sec. 2 & Appendix C], recognized that this term is infinite if the intermediate representation is continuous, and suggested to replace the compression term by the mutual information between X and a noisy or quantized version of L (cf. Sec. 5.4.4). Besides, recently invertible DNN architectures [92, 115] have been proposed that achieve state-of-the-art performance. For such invertible networks $I(X;L)$ stays the same regardless of the learned network parameters. The success of such networks also casts doubt on the claims

in [20] that information-theoretic compression can be directly linked to generalization capabilities.

Our discussion in Sec. 5.3, although done in the context of training DNNs, also applies to analyzing DNNs using the IB framework. The discussion in Sec. 5.3.3 implies that, without additional changes such as the ones proposed in Sec. 5.4, a good result in terms of the IB functional does neither admit statements about the robustness of a deterministic DNN nor about the simplicity of the required decision function. It further means that the IB framework cannot be used to make statements about the classification performance or generalization without introducing additional constraints. Furthermore, the ill-posed or piecewise constant nature of IB functional for the classification task using deterministic DNNs (cf. Sec. 5.3.1 and Sec. 5.3.2) further complicates the situation. Hence our observations regarding generalization and compression observed in [20] are in line with the observations in [102]. To summarize, unless remedied via additional modifications, these problems make the IB functional an unfit tool for analysing deterministic DNNs.

5.8. Our Perspective on Bounding $I(Y;L)$

In this section we briefly discuss the relation between $I(Y;L)$ and cross-entropy loss, which is often used as a measure of accuracy while training DNNs. We provide an alternative view to the one presented in [85, 86] about the cross-entropy loss as a lower bound for $I(Y;L)$.

We start by restricting ourselves to the setup where, for the latent layer L under consideration, the encoder $L = f(X)$ can be a stochastic map of X but the decoder $\hat{Y} = h(L)$ is a deterministic map. Note that the scenarios in [85, 86] are subsumed in this setup. The prevalent perspective, also described in [85] and [86], of defining the lower bound is:

$$I(Y;L) \geq H(Y) - H(Y|L) - D\left(P_{Y|L} \| Q_{Y|L}\right) \tag{5.6}$$

$$= H(Y) - C\left(P_{Y|L} \| Q_{Y|L}\right) \tag{5.7}$$

where $C(\cdot\|\cdot)$ denotes cross-entropy and $Q_{Y|L}$ is any conditional probability distribution over $\mathcal{Y}$ given L (normally referred to as the variational approximation of the true posterior $P_{Y|L}$).

For deterministic h, we make the assumption that for all $y \in \mathcal{Y}$ and $\ell \in \mathbb{R}^{|L|}$ we have

$$Q_{Y|L}(y|\ell) = Q_{Y|\hat{Y}}(y|h(\ell)) \tag{5.8}$$

i.e., $Q_{Y|L}$ depends on L only via the deterministic decoder $\hat{Y} = h(L)$. This assumption defines the role that the deterministic decoder map h plays in the variational approximation and is valid for the scenarios described in [85,86]. $Q_{Y|\hat{Y}}$ is the variational approximation of the true posterior $P_{Y|\hat{Y}}$ obtained via application of a soft decision rule to $\hat{Y}$ (cf. Sec. 5.4.2). For example, in [85,86] the output $\hat{Y}$ of the last softmax layer is interpreted as a probability

distribution $Q_{Y|\hat{Y}}$ over $\mathcal{Y}$ (as in Sec. 5.4.2).

On the one hand, for $Q_{Y|L}$ satisfying (5.8), we have

$$\begin{aligned} C\left(P_{Y|L}\|Q_{Y|L}\right) &= -\mathbb{E}_{L,Y\sim P_{L,Y}}\left(\log Q_{Y|L}(Y|L)\right) \\ &\overset{(a)}{=} -\mathbb{E}_{L,Y\sim P_{L,Y}}\left(\log Q_{Y|\hat{Y}}(Y|h(L))\right) \\ &\overset{(b)}{=} -\mathbb{E}_{\hat{Y},Y\sim P_{\hat{Y},Y}}\left(\mathbb{E}_{L\sim P_{L|\hat{Y}}(\cdot|\hat{Y})}\left(\log Q_{Y|\hat{Y}}(Y|h(L))\right)\right) \\ &\overset{(c)}{=} -\mathbb{E}_{\hat{Y},Y\sim P_{\hat{Y},Y}}\left(\log Q_{Y|\hat{Y}}(Y|\hat{Y})\right) \\ &= C\left(P_{Y|\hat{Y}}\|Q_{Y|\hat{Y}}\right) \end{aligned} \tag{5.9}$$

where (a) is due to (5.8), (b) due to the law of total expectation, and (c) because $\hat{Y}=h(L)$ and the inner expectation has a constant argument for a deterministic h.

On the other hand, we have

$$\begin{aligned} I(Y;L) &= I(Y;\hat{Y}) + I(Y;L|\hat{Y}) \\ &\overset{(a)}{\geq} I(Y;\hat{Y}) \\ &\overset{(b)}{\geq} H(Y) - C\left(P_{Y|\hat{Y}}\|Q_{Y|\hat{Y}}\right) \\ &\overset{(c)}{=} H(Y) - C\left(P_{Y|L}\|Q_{Y|L}\right) \end{aligned} \tag{5.10}$$

where (a) follows from the non-negativity of mutual information, (b) is obtained by employing the same technique as in (5.7) and (c) is due to (5.9). Inequalities (a) and (b) signify our two-step perspective of lower bounding $I(Y;L)$ using the cross-entropy loss $C\left(P_{Y|L}\|Q_{Y|L}\right)$. Specifically, (a) defines the relevant information loss [116, Chap. 5] due to the transformation of L to $\hat{Y}$ via the deterministic decoding map h, while (b) represents the loss due the variational approximation $Q_{Y|L}=Q_{Y|\hat{Y}}$ obtained by applying the decision rule to $\hat{Y}$. Hence, we have divided the total approximation loss into two parts, one corresponding to the decoding map h and the other corresponding to the decision rule, in the same way in which $Q_{Y|L}$ is defined based on L in two steps via $\hat{Y}$. This also shows that the cross-entropy loss used in [85, 86] is a better surrogate for $I(Y;\hat{Y})$ than $I(Y;L)$.

We now briefly consider a stochastic decoding map h and define

$$Q_{Y|L}(y|\ell) = \mathbb{E}_{\hat{Y}\sim P_{\hat{Y}|L}(\cdot|\ell)}\left(Q_{Y|\hat{Y}}(y|\hat{Y})\right) \tag{5.11}$$

which simplifies to (5.8) in case h is deterministic. For stochastic h and $Q_{Y|L}(y|\ell)$ defined in (5.11) we have

$$C\left(P_{Y|L}\|Q_{Y|L}\right) = -\mathbb{E}_{L,Y\sim P_{L,Y}}\left(\log\left(\mathbb{E}_{\hat{Y}\sim P_{\hat{Y}|L}(\cdot|L)}\left(Q_{Y|\hat{Y}}(Y|\hat{Y})\right)\right)\right)$$

$$\begin{aligned} &\overset{(a)}{\leq} -\mathbb{E}_{L,Y,\hat{Y}\sim P_{L,Y,\hat{Y}}}\left(\log Q_{Y|\hat{Y}}(Y|\hat{Y})\right) \\ &\overset{(b)}{=} C\left(P_{Y|\hat{Y}}\|Q_{Y|\hat{Y}}\right) \end{aligned} \tag{5.12}$$

where (a) follows from Jensen's inequality and (b) is due to law of total expectation. This suggests that for stochastic encoders, $C\left(P_{Y|L}\|Q_{Y|L}\right)$ provides a tighter lower bound for $I(Y;L)$ than $C\left(P_{Y|\hat{Y}}\|Q_{Y|\hat{Y}}\right)$.

The two cross-entropy terms in (5.12) have different operational interpretations in the case of a stochastic decoding map h. To understand this better, we investigate how they are estimated in practice. For each available data sample (x, y), we generate multiple samples of the latent representation L according to the stochastic map $f = P_{L|X}$. Then, for every ℓ, we generate multiple samples of $\hat{Y}$ according to the stochastic map $h = P_{\hat{Y}|L}$. On the one hand, to compute $C\left(P_{Y|L}\|Q_{Y|L}\right)$, we average $Q_{Y|\hat{Y}}$ corresponding to different samples of $\hat{Y}$ generated for the same tuple (x, ℓ, y) to calculate $Q_{Y|L}$ (i.e., (5.11)). This is used to calculate the log loss $\left(\log Q_{Y|L}\right)$. On the other hand, for $C\left(P_{Y|\hat{Y}}\|Q_{Y|\hat{Y}}\right)$, we evaluate log loss for each sample of $\hat{Y}$ separately $\left(\log Q_{Y|\hat{Y}}\right)$. These log losses are then averaged to calculate the loss corresponding to each tuple $(x, \hat{y}, y)$.

A

Appendices for Chapter 2

A.1. Proof of Proposition 2.4

Consider the relation $\mathcal{R}_\varepsilon = \{(g(z), z)\colon z \in \mathcal{Z}\}$. It can be shown that

$$\forall T \subseteq \mathcal{Z} \cup \overline{\mathcal{Z}}\colon \mathcal{R}_\varepsilon(T) = g^{-1}(T \cap \overline{\mathcal{Z}}) \subseteq \mathcal{Z}. \tag{A.1}$$

We thus need to show that, for all z and all $B \subseteq \overline{\mathcal{Z}}$,

$$\sum_{z' \in g^{-1}(B)} \mathbb{P}(z, z') \geq \sum_{\overline{z} \in B} \mathbb{Q}(g(z), \overline{z}) - \varepsilon. \tag{A.2}$$

Now let $\mathbb{R} = \mathbb{P}\mathbb{W}$, i.e., we have

$$\mathbb{R}(z, \overline{z}) = \sum_{z' \in g^{-1}(\overline{z})} \mathbb{P}(z, z') \tag{A.3}$$

One can show along the lines of [4, (65)–(68)] that

$$\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) = \sum_{z \in \mathcal{Z}} \mu_z \underbrace{\sum_{\overline{z} \in \overline{\mathcal{Z}}} \mathbb{R}(z, \overline{z}) \log \frac{\mathbb{R}(z, \overline{z})}{\mathbb{Q}(g(z), \overline{z})}}_{=: D(\mathbb{R}(z,:) || \mathbb{Q}(g(z),:))} \tag{A.4}$$

from which we get that, for every z,

$$D\left(\mathbb{R}(z,:) || \mathbb{Q}(g(z),:)\right) \leq \frac{\mathrm{C}_L(\mathbf{Z}, \mathbb{W})}{\min\limits_z \mu_z}. \tag{A.5}$$

With Pinsker's inequality [40, Lemma 12.6.1] and [40, (12.137)] we thus get that, for every z and every $B \subseteq \overline{\mathcal{Z}}$,

$$\left| \sum_{\overline{z} \in B} \mathbb{R}(z, \overline{z}) - \mathbb{Q}(g(z), \overline{z}) \right| \leq \sqrt{\frac{\ln(2) \mathrm{C}_L(\mathbf{Z}, \mathbb{W})}{2 \min_z \mu_z}}. \tag{A.6}$$

Combining this with (A.3) thus shows that (A.2) holds for

$$\varepsilon = \sqrt{\frac{\ln(2) \mathrm{C}_L(\mathbf{Z}, \mathbb{W})}{2 \min_z \mu_z}}. \tag{A.7}$$

This completes the proof.

A.2. Proof of Lemma 2.5

We show that the derivative of $\delta_\beta(\mathbf{Z}, \mathbb{W})$ w.r.t. β is positive. Indeed,

$$\frac{\mathrm{d}}{\mathrm{d}\,\beta} \delta_\beta(\mathbf{Z}, \mathbb{W}) = \bar{R}(\mathbf{Z}) - \bar{R}(\overline{\mathbf{Z}}) - \bar{D}(\overline{\mathbf{Z}} || \tilde{\mathbf{Z}}) \tag{A.8}$$

$$= I(Z_1; Z_2) - H(\overline{Z}) + \bar{H}(\overline{\mathbf{Z}}) - H(\overline{Z}_2 | \overline{Z}_1) + \bar{H}(\mathbf{Z}). \tag{A.9}$$

The entropy rate of the reversed process equals the entropy rate of the original process, i.e.,

$$\bar{H}(\overline{\mathbf{Z}}) = \lim_{n \to \infty} H(\overline{Z}_n | \overline{Z}_1^{n-1}) = \lim_{n \to \infty} H(\overline{Z}_1 | \overline{Z}_2^n). \tag{A.10}$$

We can now apply [40, Lem. 4.4.1] to both sides to get $\bar{H}(\overline{\mathbf{Z}}) \geq H(\overline{Z}_2 | Z_1)$ and $\bar{H}(\overline{\mathbf{Z}}) \geq H(\overline{Z}_1 | Z_2)$. We use this in the derivative to get

$$\frac{\mathrm{d}}{\mathrm{d}\,\beta} \delta_\beta(\mathbf{Z}, \mathbb{W}) \geq I(Z_1; Z_2) - H(\overline{Z}) + H(\overline{Z}_1 | Z_2) - H(\overline{Z}_2 | \overline{Z}_1) + H(\overline{Z}_2 | Z_1) \tag{A.11}$$

$$= H(Z | \overline{Z}) - H(Z_1 | \overline{Z}_1, Z_2) - H(\overline{Z}_2 | \overline{Z}_1) + H(\overline{Z}_2 | Z_1) \tag{A.12}$$

$$= I(Z_1; Z_2 | \overline{Z}_1) - I(Z_1; \overline{Z}_2 | \overline{Z}_1) \geq 0 \tag{A.13}$$

by data processing.

A.3. Proof of Lemma 2.6

The first property follows by recognizing that

$$\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) = (1 - \beta) \mathrm{C}_L(\mathbf{Z}, \mathbb{W}) + \beta (\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) - \mathrm{C}_L(\mathbf{Z}, \mathbb{W})) \tag{A.14}$$

and that $\mathrm{C}_P(\mathbf{Z}, \mathbb{W}) \geq \mathrm{C}_L(\mathbf{Z}, \mathbb{W})$.

The second property follows immediately from the definition of $\delta_\beta(\mathbf{Z}, \mathbb{W})$ and $\mathrm{C}_P(\mathbf{Z}, \mathbb{W})$. For the third property, note that

$$\begin{aligned}\mathrm{C}_1(\mathbf{X}, \mathbb{W}) &= \mathrm{C}_P(\mathbf{Z}, \mathbb{W}) - \mathrm{C}_L(\mathbf{Z}, \mathbb{W}) \\ &= I(Z_1; Z_2) - H(\overline{Z}) + H(\overline{Z}_2|Z_1) \\ &= I(Z_1; Z_2) - I(Z_1, \overline{Z}_2) = I(Z_1; Z_2|\overline{Z}_2).\end{aligned}$$

The fourth property is obtained by observing that, if $\beta \leq 0.5$

$$\begin{aligned}\delta_\beta(\mathbf{Z}, \mathbb{W}) - \beta I(Z_1; Z_2) &= (1-\beta)H(\overline{Z}_2|\overline{Z}_1) - (1-2\beta)\bar{H}(\overline{\mathbf{Z}}) - \beta H(\overline{Z}) \\ &\leq (1-\beta)H(\overline{Z}_2|\overline{Z}_1) - (1-2\beta)H(\overline{Z}_2|Z_1) - \beta H(\overline{Z}) \\ &= (1-2\beta)H(\overline{Z}_2|\overline{Z}_1) - (1-2\beta)H(\overline{Z}_2|Z_1) - \beta I(\overline{Z}_1; \overline{Z}_2) \\ &= (1-2\beta)\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) - \beta I(\overline{Z}_1; \overline{Z}_2).\end{aligned}$$

The inequality is reversed for $\beta \geq 0.5$.

For the fifth property, we repeat the last steps with

$$-(1-2\beta)\bar{H}(\overline{\mathbf{Z}}) \leq -(1-2\beta)H(\overline{Z}_2|\overline{Z}_1) \tag{A.15}$$

noticing that $(1-2\beta) \leq 0$ if $\beta \geq 0.5$. Again, the inequality is reversed for $\beta \leq 0.5$.

If $\mathbf{Z}$ is reversible, then the PMFs do not change if the order of the indices is reversed. As a consequence, we have $I(Z_1; Z_2|\overline{Z}_2) = I(Z_2; Z_1|\overline{Z}_1) = \mathrm{C}_1(\mathbf{Z}, \mathbb{W})$. But $\mathrm{C}_L(\mathbf{Z}; \mathbb{W}) = I(\overline{Z}_2; Z_1|\overline{Z}_1) \leq \mathrm{C}_1(\mathbf{Z}, \mathbb{W})$ by data processing. Thus, the sixth property follows by noting that, with (A.14), $\mathrm{C}_\beta(\mathbf{Z}, \mathbb{W}) = (1-\beta)\mathrm{C}_L(\mathbf{Z}, \mathbb{W}) + \beta\mathrm{C}_1(\mathbf{Z}, \mathbb{W})$.

B

Appendices for Chapter 4

B.1. Factor Graphs

Factor graphs [117] are bipartite graphs that graphically represent factorizable functions. A function $f(\cdot)$ is said to be factorizable if it can be factored into product of "local" functions $g_j(\cdot)$:

$$f(z_1, z_2, \cdots, z_n) = \prod_j g_j(\mathcal{Z}_j) \tag{B.1}$$

where the jth factor $g_j(\cdot)$ is a function of only a subset $\mathcal{Z}_j$ of all the variables $\{z_1, z_2 \dots z_n\}$ involved in $f(\cdot)$. The notation $g_j(\mathcal{Z}_j)$ means that g_j is a function of the variables in the set $\mathcal{Z}_j$. One can construct a graph of this function where each variable is represented by a circular node and each factor is represented by a rectangular node. A variable node z_i and a factor node g_j are connected by an edge if $z_i \in \mathcal{Z}_j$. We call two nodes in graph adjacent if they are connected directly via an edge. There are no edges between two variable nodes or between two factor nodes, hence it is a bipartite graph. This graph is known as the factor graph of f.

Fig. B.1(a) shows the factor graph of the function in the following equation:

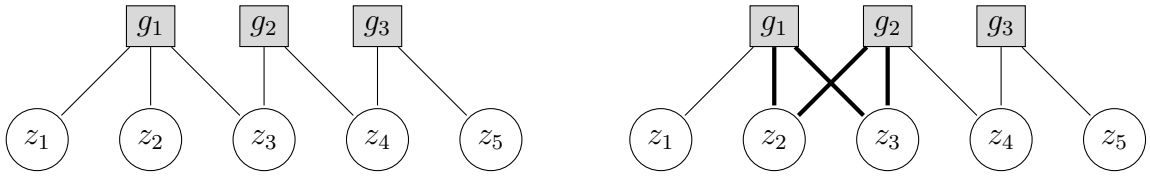
$$f(z_1, z_2, z_3, z_4, z_5) = g_1(z_1, z_2, z_3) g_2(z_3, z_4) g_3(z_4, z_5) \tag{B.2}$$

whereas Fig. B.1(b) shows the factor graph of the following function:

$$f(z_1, z_2, z_3, z_4, z_5) = g_1(z_1, z_2, z_3) g_2(z_2, z_3, z_4) g_3(z_4, z_5) \tag{B.3}$$

The factor graph in Fig. B.1(a) is acyclic while the one in Fig. B.1(b) contains a cycle, highlighted using thick edges.

Factor graphs are useful for doing maximization (or maginalization) of factorizable functions involving large number of discrete variables in a computationally tractable way. Instead of doing the maximization (or marginalization) jointly we can do maximization (or marginalization) locally for each factor and then combine the results to obtain either exact or approximate maximum (or marginal) of f. The schedule for doing the local computations and how to combine the local computations is defined by the message passing algorithm applied to the factor graph. The message passing algorithm dealing with marginalization using factor graphs is known as sum-product algorithm and the message passing algorithm dealing with maximization using factor graphs is known as max-product algorithm. These algorithms are also sometimes referred to as belief propagation. [117] provides a detailed survey of factor graphs, belief propagation and their applications.



(a) Factor graph for (B.2) (b) Factor graph for (B.3)

Figure B.1.: Examples of factor graphs with and without cycles.

B.2. Max-Product Algorithm

We are interested in computing max-marginals, hence we will focus on max-product algorithm (MP) [118]. Max marginal w.r.t. variable z_i is defined as

$$\tilde{f}_i(z_i) = \underset{\sim z_i}{\operatorname{argmax}} f([z]) \tag{B.4}$$

where the $\sim z_i$ in optimization arguments implies that we are optimizing the objective function over all variables involved except z_i. To explain MP let's start with the example in Fig. B.1(a) of an acyclic factor graph. If we want to evaluate $\tilde{f}_3(z_3)$ we can express it as follows:

$$\begin{aligned} \tilde{f}_3(x_3) &= \max_{\sim z_3} f(z_1, z_2, z_3, z_4, z_5) && \text{(B.5)} \\ &= \left(\max_{\sim z_3} g_1(z_1, z_2, z_3)\right) \left(\max_{\sim z_3} g_2(z_3, z_4) \left(\max_{\sim z_4} g_3(z_4, z_5)\right)\right). && \text{(B.6)} \end{aligned}$$

Define

$$\alpha(z_4) = \max_{\sim z_4} g_3(z_4, z_5) \tag{B.7}$$

$$\beta(z_3) = \max_{\sim z_3} g_2(z_3, z_4) \cdot \alpha(z_4) \tag{B.8}$$

$$\gamma(z_3) = \max_{\sim z_3} g_1(z_1, z_2, z_3). \tag{B.9}$$

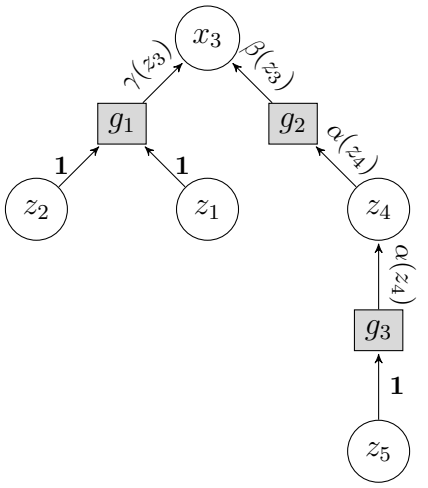


Figure B.2.: The figure represents MP applied to compute $\tilde{f}_3(z_3)$. The factor graph in Fig. B.1(a) is redrawn with z_3 as the root node. The messages shown on the edges represent the computations in (B.7),(B.8) and (B.9).

Hence

$$\tilde{f}_3(z_3) = \gamma(z_3) \cdot \left(\max_{\sim z_3} g_2(z_3, z_4)\alpha(z_4) \right) \tag{B.10}$$

$$= \gamma(z_3) \cdot \beta(z_3) \tag{B.11}$$

where $\alpha(\cdot)$, $\beta(\cdot)$ and $\gamma(\cdot)$ denote the maximization of individual factors or a subset of factors over a subset of the involved variables. If we assume $z_i \in \mathcal{Z}$ for a finite alphabet $\mathcal{Z}$ then brute force optimization of (B.5) requires $\mathcal{O}(|\mathcal{Z}|^4)$ computations whereas (B.6) represents how MP defines an efficient way to do the maximization using $\mathcal{O}(|\mathcal{Z}|^2)$ computations by utilizing the distributive law. These computations can be visualized in Fig. B.2 where we have redrawn the factor graph in Fig. B.1(a) as a tree to depict the computations in the sequence they are done. The root node, in this case z_3, is the variable node w.r.t. which we are computing max-marginal. The computations start at the leaf nodes. Each node waits until it receives an incoming message from all of it's children, where each incoming message corresponds to the aggregated computations for the corresponding subtree, and then it computes the outgoing message to it's parent node in the following way:

- ▷ Once a variable node z_i receives incoming message vectors from all of it's children nodes, in order to compute the outgoing message to the parent factor node g_j we evaluate the product of all incoming messages from the children nodes for each value

of the variable $z_i \in \mathcal{Z}$:

$$\mu_{z_i \to g_j}(z_i) = \prod_{g_k \in \mathcal{F}_i \setminus \{g_j\}} \nu_{g_k \to z_i}(z_i) \tag{B.12}$$

where $\mathcal{F}_i$ represents the set of all local functions that involve variable z_i, i.e., all factor nodes adjacent to z_i. The vector $\boldsymbol{\mu}_{z_i \to g_j}$, which is a concatenation of these values for all $z_i \in \mathcal{Z}$, is the message that variable node z_i passes on to it's parent factor node g_j. In the case when the variable node z_i is a leaf node it assigns to $\boldsymbol{\mu}_{z_i \to g_j}$ a vector of all 1s, denoted by $\mathbf{1}$.

▷ Once a factor node g_j receives incoming message vectors from all of it's children variable nodes, we compute the max-marginal w.r.t. the parent variable node z_i of the product of the factor g_j with the incoming message vectors from the children variable nodes:

$$\nu_{g_j \to z_i}(z_i) = \max_{\sim z_i} \left(g_j(\mathcal{Z}_j) \prod_{z_k \in \mathcal{Z}_j \setminus \{z_i\}} \mu_{z_k \to g_j}(z_k) \right) \tag{B.13}$$

This max-marginal, for all values of the parent variable node z_i, then constitutes the message vector $\boldsymbol{\nu}_{g_j \to z_i}$ passed from factor node g_j to the parent variable node z_i. In the case when the factor node g_j is a leaf node, since it only involves the parent variable, we assign $\nu_{g_j \to z_i}(z_i) = g_j(z_i)$.

Note that the computations performed at the variable nodes do not explicitly depend on the structure of the local functions but the computations performed at the factor nodes depend explicitly on the structure of the local functions. The messages passed between the nodes can be thought of both as a function of single discrete variable as well as a vector of length $|\mathcal{Z}|$. In our example where we compute R.H.S. of (B.5), the computations defined in (B.7), (B.8) and (B.9) correspond to the outgoing messages from different nodes as shown in Fig. B.2.

Finally the root variable node z_3 computes the max-marginal described in (B.5) by taking the element wise product of the incoming message vectors as described in (B.11). The general form of (B.11) is given as

$$\tilde{f}_i(z_i) = \prod_{g_k \in \mathcal{F}_i} \nu_{g_k \to z_i}(z_i) \tag{B.14}$$

We can similarly compute the max-marginal w.r.t. other variables z_i by rearranging Fig. B.1(a) in a similar way, considering z_i as the root node. If we want to evaluate max-marginals w.r.t. all variables, we can reduce the computations significantly by computing them simultaneously instead of considering each variable node as the root node and doing separate computations for each max-marginal. To compute all the max-marginals simultaneously we have the following procedure: We no longer have one specific root variable node. We start computing the outgoing messages from the nodes that are connected to

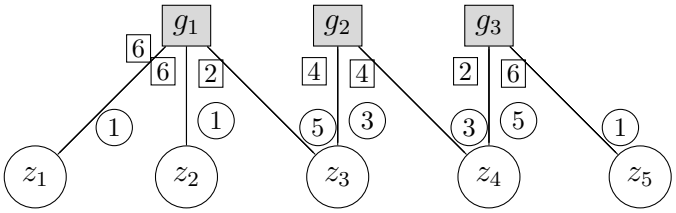


Figure B.3.: The figure shows the sequence in which messages are passed among the nodes for computing all the max-marginals simultaneously in the factor graph of Fig. B.1(a). The encircled numbers on the R.H.S. of each edge indicates the step at which the connected variable node sends a message to the connected factor node whereas the boxed numbers on the L.H.S. of each edge represent the step at which the connected factor node sends a message to the connected variable node.

only one other node. Each node now sends an outgoing message vector to all of it's adjacent nodes. For the message from a variable node z_i to an adjacent factor node g_j, z_i considers g_j as it's parent node and waits until it has received incoming messages from all the other adjacent factor nodes and then computes $\boldsymbol{\mu}_{z_i \to g_j}$ using (B.12). Similarly for the outgoing message from a factor node g_j to an adjacent variable node z_i, g_j considers z_i as it's parent node and waits until it has received messages from all the other adjacent variable nodes. Then it computes $\boldsymbol{\nu}_{g_j \to z_i}$ using (B.13). MP terminates when two messages have been send along each edge of the factor graph, one in either direction. The max-marginals are then computed at each variable node z_i using (B.14). For the factor graph in Fig. B.1(a), this sequence of the messages is shown in Fig. B.3.

In the case of an acyclic factor graph this procedure is guaranteed to compute all the required messages in a finite number of steps. It also leads to exact max-marginals once MP terminates after two messages have traversed each edge in the factor graph. However in the case of a factor graph with cycles this procedure will fail to complete since, at some point during the computations, the nodes in the cycle will keep waiting for the messages from one another. In such a case, to start the process every variable node sends a predefined message to all the the adjacent factor nodes. We also do not stop the message passing algorithm once two messages have traversed on each edge, rather we continue iterating by passing messages back and forth between variable nodes and factor nodes alternatively until the messages converge or a maximum number of iterations is reached. Finally we use (B.14) to obtain an approximation of the max-marginals.

An important property of the max-marginals is that

$$\underset{[z]}{\operatorname{argmax}}\, f(z_1, z_2, \cdots, z_n) = \left(\underset{z_1}{\operatorname{argmax}}\, \tilde{f}_1(z_1), \underset{z_2}{\operatorname{argmax}}\, \tilde{f}_2(z_2), \cdots, \underset{z_n}{\operatorname{argmax}}\, \tilde{f}_n(z_n) \right) \quad \text{(B.15)}$$

Hence to determine the joint maximizer for f we can determine the individual maximizers for the max marginals $\tilde{f}_i(z_i)$.

MP normally leads to numerical accuracy issues due to the product of a large number of factors. Since the maximizer of a function is not affected by taking logarithm, we can transform (B.4) to log domain:

$$\begin{aligned}\log \tilde{f}_i(z_i) &= \underset{\sim z_i}{\operatorname{argmax}} \log f(z_1, z_2, \cdots, z_n) \\ &= \underset{\sim z_i}{\operatorname{argmax}} \sum_j \log\left(g_j(\mathcal{Z}_j)\right) \end{aligned} \tag{B.16}$$

The corresponding message passing algorithm, known as max-sum algorithm (MS) to compute the log domain version of (B.4) has the following messages passed between variable and factor nodes:

$$\mu_{z_i \to g_j}(z_i) = \sum_{g_k \in \mathcal{F}_i \setminus \{g_j\}} \nu_{g_k \to z_i}(z_i) \tag{B.17}$$

$$\nu_{g_j \to z_i}(z_i) = \max_{\sim z_i} \left(\log g_j(\mathcal{Z}_j) + \sum_{z_k \in \mathcal{Z}_j \setminus z_i} \mu_{z_k \to g_j}(z_k) \right) \tag{B.18}$$

Note that the only two differences to the MP messages is that we are now using logarithms of the local functions for computations and the products in (B.13) and (B.12) have been replaced by sums. Finally we can obtain $\log \tilde{f}_i(z_i)$ as follows

$$\log \tilde{f}_i(z_i) = \sum_{g_k \in \mathcal{F}_i} \nu_{g_k \to z_i}(z_i) \tag{B.19}$$

B.3. Derivation of (4.21)

Using max-sum algorithm, we have

$$\nu_{\overline{g}_i \to b_{ij}}(b_{ij}) = \max_{\sim b_{ij}} \left(\overline{g}_i(\mathbb{B}(i,:)) + \sum_{l \neq j} \mu_{b_{il} \to g_i}(b_{il}) \right) \tag{B.20}$$

We will evaluate (B.20) for $b_{ij} = 0$ and $b_{ij} = 1$. Then we will obtain η_{ij} using (4.10). For $b_{ij} = 1$, we satisfy the requirement $\sum_j b_{ij} \geq 1$ regardless of the values of other variables involved. Hence $\overline{g}_i(\mathbb{B}(i,:)) = \sum_k b_{ik} q$

$$\nu_{\overline{g}_i \to b_{ij}}(1) = \max_{\sim b_{ij}} \left[\sum_m b_{im} q + \sum_{l \neq j} \mu_{b_{il} \to \overline{g}_i}(b_{il}) \right] \tag{B.21}$$

$$\overset{(a)}{=} q + \max_{\sim b_{ij}} \left[\sum_{l \neq j} \left(b_{il} q + \mu_{b_{il} \to \overline{g}_i}(b_{il}) \right) \right] \tag{B.22}$$

$$\stackrel{(b)}{=} q + \sum_{l \neq j} \max_{b_{il} \in \{0,1\}} \left(b_{il} q + \mu_{b_{il} \to \overline{g}_i}(b_{il}) \right) \tag{B.23}$$

$$= q + \sum_{l \neq j} \max \left\{ q + \mu_{b_{il} \to \overline{g}_i}(1), \mu_{b_{il} \to \overline{g}_i}(0) \right\} \tag{B.24}$$

$$\stackrel{(c)}{=} q + \sum_{k \neq j} \mu_{b_{ik} \to \overline{g}_i}(0) + \sum_{l \neq j} \max \{ q + \beta_{il}, 0 \} \tag{B.25}$$

where (a) follows by taking the penalty for $b_{ij} = 1$ outside the maximum. (b) follows from the observation that each component of the sum depends on only one variable and can be optimized independently. (c) follows from the definition of β_{il} in (4.9).

For $b_{ij} = 0$, we need to have at least one $b_{il} = 1$ in order to avoid $\overline{g}_i(\mathbb{B}(i,:)) = -\infty$. The rest of the variables involved can be chosen freely.

$$\nu_{\overline{g}_i \to b_{ij}}(0) = \max_{m \neq j} \left[q + \mu_{b_{im} \to \overline{g}_i}(1) + \max_{\{b_{il}\} \setminus \{b_{ij}, b_{im}\}} \left(\sum_{l \neq \{m,j\}} \left(b_{il} q + \mu_{b_{il} \to \overline{g}_i}(b_{il}) \right) \right) \right] \tag{B.26}$$

$$= \max_{m \neq j} \left[q + \mu_{b_{im} \to \overline{g}_i}(1) + \sum_{k \neq \{j,m\}} \mu_{b_{ik} \to \overline{g}_i}(0) + \sum_{l \neq \{m,j\}} \max \{ q + \beta_{il}, 0 \} \right] \tag{B.27}$$

where (d) follows by using same techniques as in (b) and (c)Combining (B.25) and (B.27) we get

$$\eta_{ij} = - \max_{m \neq j} \left[\mu_{b_{im} \to \overline{g}_i}(1) - \mu_{b_{im} \to \overline{g}_i}(0) + \max \{ q + \beta_{im}, 0 \} \right] \tag{B.28}$$

$$= - \max_{m \neq j} \left[\beta_{im} + \min \{ -q - \beta_{im}, 0 \} \right] \tag{B.29}$$

$$= - \max_{m \neq j} \left[\min \{ \beta_{im}, -q \} \right] \tag{B.30}$$

$$= \max \left[- \max_{m \neq j} \beta_{im} \ , \ q \right] \tag{B.31}$$

B.4. Derivation of (4.25)

For $i \in \mathcal{N}_j$ and $b_{ii} = 1$, we need all the other involved variables to be 0 in order to avoid $r_j(\mathcal{N}_j) = -\infty$. Hence

$$\nu_{r_j \to b_{ii}}(1) = \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \mu_{b_{kk} \to r_j}(0) \tag{B.32}$$

For $b_{ii} = 0$, in order to avoid $r_j(\mathcal{N}_j) = -\infty$ we can either choose all other involved variables to be also 0 or we can allow one of them to be equal to 1. Hence

$$\nu_{r_j \to b_{ii}}(0) = \max \left\{ \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \mu_{b_{kk} \to r_j}(0), \max_{\substack{l \in \mathcal{N}_j \\ l \neq i}} \left[\mu_{b_{ll} \to r_j}(1) + \sum_{\substack{k \in \mathcal{N}_j \\ k \neq \{i,l\}}} \mu_{b_{kk} \to r_j}(0) \right] \right\} \tag{B.33}$$

$$= \max \left\{ 0, \max_{\substack{l \in \mathcal{N}_j \\ l \neq i}} \phi_{lj} \right\} + \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \mu_{b_{kk} \to r_j}(0) \tag{B.34}$$

Combining (B.32) and (B.34) we obtain

$$\psi_{ij} = -\max \left\{ 0, \max_{\substack{l \in \partial_j \\ l \neq i}} \phi_{lj} \right\} \tag{B.35}$$

C

Appendices for Chapter 5

C.1. Proof of Theorem 5.1

Let $\boldsymbol{z} \triangleq (z_1, \ldots, z_N) \in \mathbb{Z}^N$. Moreover, we define the N-dimensional cube with side length a and bottom-left corner at $z \in \mathbb{R}^N$ as $\mathcal{C}(z, a) \triangleq [z_1, z_1 + a) \times \cdots [z_N, z_N + a)$. For example, the RV $[X]_m \triangleq \lfloor mX \rfloor / m$, where the floor operation is applied element-wise, is obtained by quantizing X with a quantizer that has quantization bins $\mathcal{C}(z/m, 1/m)$, $z \in \mathbb{Z}^N$.

Let $H(Z)$ denote the entropy of the discrete RV Z with probability mass function P_Z and alphabet $\mathcal{Z}$, and let

$$H_2(Z) \triangleq -\log\left(\sum_{z \in \mathcal{Z}} \left(P_Z(z)\right)^2\right). \tag{C.1}$$

denote the Rényi entropy of second order of Z. The correlation dimension of a general RV X is defined as [119]

$$d_2(X) \triangleq \lim_{m \to \infty} \frac{H_2([X]_m)}{\log m} \tag{C.2}$$

provided the limit exists. The information dimension $d(X)$ is defined accordingly, with Rényi entropy of second order replaced by entropy [120].

Proof of the Theorem: The proof consists of four ingredients. Assuming that the distribution of X has a continuous PDF supported on a compact set, we first show that the input X has positive correlation dimension. Then, we show that the correlation dimension remains positive throughout the DNN. Afterwards, we show that the output has positive information dimension, from which follows that $I(X; L) = \infty$. Finally, we relax the condition that X has a continuous PDF supported on a compact set, but require that its distribution has at least such a component.

We start by assuming that X has a continuous PDF that is supported on a compact set $\mathcal{X}$ in $\mathbb{R}^N$.

Lemma C.1. Let X be an N-dimensional RV with a PDF p_X that is continuous and supported on a compact set $\mathcal{X}$ in $\mathbb{R}^N$. Then, $d_2(X) = N$.

This result generalizes [119, Th. 3.I.c] to higher-dimensional RVs.

Proof: Since p_X is continuous, so is its square p_X^2. Since both p_X and p_X^2 are continuous and supported on a compact set, they are Riemann integrable. Hence, the differential Rényi entropy of second order,

$$h_2(X) \triangleq -\log \int_{\mathcal{X}} p_X^2(x)\mathrm{d}x \tag{C.3}$$

exists. We can sandwich $h^2(X)$ by using the upper and lower Darboux sums, i.e., we can write

$$\begin{aligned} &L_{h_2,m} \\ &\triangleq -\log\left(\sum_{z\in\mathbb{Z}^N} \frac{1}{m^N} \sup_{x\in\mathcal{C}(z/m,1/m)} p_X^2(x)\right) \\ &\leq h_2(X) \\ &\leq -\log\left(\sum_{z\in\mathbb{Z}^N} \frac{1}{m^N} \inf_{x\in\mathcal{C}(z/m,1/m)} p_X^2(x)\right) \triangleq U_{h_2,m}. \end{aligned}$$

Note further that by the mean value theorem we can find $t_z \in \mathcal{C}(z/m, 1/m)$ such that

$$P_{[X]_m}(z/m) = \int_{x\in\mathcal{C}(z/m,1/m)} p_X(x)\mathrm{d}x = \frac{1}{m^N} p_X(t_z). \tag{C.4}$$

This allows us to write $H_2([X]_m)$ as

$$H_2([X]_m) = -\log\left(\sum_{z\in\mathbb{Z}^N} \frac{1}{m^{2N}} p_X^2(t_z)\right) \tag{C.5}$$

Since $p_X^2(t_z)$ lies between the infimum and the supremum p_X can assume on the cube $\mathcal{C}(z/m, 1/m)$, we obtain

$$L_{h_2,m} \leq H_2([X]_m) - N\log m \leq U_{h^2,m} \tag{C.6}$$

where, because of Riemann integrability, the outer terms of this inequality become equal as m tends to infinity. Hence, with (C.2), $d_2(X) = N$. ∎

Now suppose that L_i has a distribution with correlation dimension $d_2(L_i)$ and compact support $\mathcal{L}_i \subset \mathbb{R}^{|L_i|}$. Then, we have by [121, Th. 1.1]

$$d_2(\mathbb{W}_i^T L_i) = \min\{|L_i|, |L_{i+1}|, d_2(L_i)\} \tag{C.7}$$

for almost every $|L_i| \times |L_{i+1}|$ matrix $\mathbb{W}_i$ (in the sense of the Lebesgue measure on the space of $|L_i| \times |L_{i+1}|$ matrices). Since $\mathcal{L}_i$ is compact, so is the support $\mathcal{L}_i'$ of the distribution

of $\mathbb{W}_i^T L_i + \boldsymbol{b}_{i+1}$. Thus, we can find a rectangle $\mathcal{L}_i'' \triangleq \mathcal{L}_{i,1}'' \times \cdots \times \mathcal{L}_{i,|L_{i+1}|}'' \subset \mathbb{R}^{|L_{i+1}|}$ that contains $\mathcal{L}_i'$. Suppose the activation function σ is continuously differentiable with strictly positive derivative. Thus, it is strictly monotonically increasing and has a strictly monotonic inverse that, by the inverse function theorem, is continuously differentiable. Let $\mathcal{L}_{i+1,j}''$ be the image of $\mathcal{L}_{i,j}''$ under σ; since $\mathcal{L}_{i,j}''$ is compact, so is $\mathcal{L}_{i+1,j}''$. It follows that the function $\sigma\colon \mathcal{L}_{i,j}'' \to \mathcal{L}_{i+1,j}''$ is bi-Lipschitz, hence so is the function mapping $\mathbb{W}_i^T L_i + \boldsymbol{b}_{i+1}$ to L_{i+1}. Since bi-Lipschitz mappings do not affect correlation dimension (see [122, Th. 2.6]), we have with (C.7)

$$d_2(L_{i+1}) = \min\{|L_i|, |L_{i+1}|, d_2(L_i)\}. \tag{C.8}$$

Furthermore, if the support of the distribution of L_i is compact, so is the support of the distribution of L_{i+1}. We can thus apply this argument recursively: Indeed, since the distribution of $L_0 = X$ has a compact support on $\mathbb{R}^N$, we get with Lemma C.1,

$$d_2(L_{i+1}) = \min_{j=0,\ldots,i+1} |L_j|. \tag{C.9}$$

Correlation dimension bounds information dimension from below (see also [123, p. 257]). Hence,

$$d(L_{i+1}) \geq \min_{j=0,\ldots,i+1} |L_j| \tag{C.10}$$

i.e., the information dimension of the output of the DNN is positive.

We finally relax the condition that X has a continuous PDF supported on a compact set. By assumption, the distribution of X has an absolutely continuous component that has a PDF p_X that is continuous on a compact set $\mathcal{X} \subset \mathbb{R}^N$. The distribution of X therefore is a mixture of an absolutely continuous distribution supported on $\mathcal{X}$ and some other distribution that may have non-compact support, be discrete, or even singular w.r.t. the N-dimensional Lebesgue measure. Let X_c denote the RV distributed according to the absolutely continuous component, and let X_s denote the RV distributed according to the remaining component. Let further Y be a binary RV controlling this mixture. In other words, we have $X = X_c$ if $Y = 0$ and $X = X_s$ if $Y = 1$. Then, the information dimension of L_{i+1} satisfies [124, Th. 2]

$$d(L_{i+1}) = \mathbb{P}(Y = 0) d(L_{i+1}|Y = 0) + \mathbb{P}(Y = 1) d(L_{i+1}|Y = 1) \tag{C.11}$$

where $\mathbb{P}(Y = 0) > 0$ by assumption. Since $X = X_c$ if $Y = 0$, we can use (C.10) to show that $d(L_{i+1}|Y = 0) > 0$. Combining this with $\mathbb{P}(Y = 0) > 0$ yields that $d(L_{i+1}) > 0$ regardless of the distribution of X_s.

In contrast, since the DNN is deterministic, the conditional distribution of L_{i+1} given X is discrete. It follows that $d(L_{i+1}|X) = 0$. Therefore,

$$d(L_{i+1}) > d(L_{i+1}|X) \tag{C.12}$$

from which we obtain $I(X; L_{i+1}) = \infty$ with [116, Prop. 4.2]. This completes the proof. ■

D

Notation and Abbreviations

Mathematical Notation

x	elements of a set or numbers
$\boldsymbol{x}$	vectors
X	random variables
$\mathcal{X}$	sets.
$\mathbb{X}$	matrices. Matrices will also be used to represent joint probability distribution between two random variables or conditional probability distribution of one random variable given another random variable. Matrix elements are indexed as $\mathbb{X}(i, j)$
$\mathbf{X}$	random processes
$g : \mathcal{X} \to \mathcal{Y}$	function from $\mathcal{X}$ to $\mathcal{Y}$
P_X	probability distribution over $\mathcal{X}$
$P_{Y\|X}$	conditional probability distribution of a random variable over $\mathcal{Y}$ given a random variable over $\mathcal{X}$. $Y = P_{Y\|X}(X)$ implies that Y is obtained by applying the conditional distribution $P_{Y\|X}$ to X
$\|\mathcal{X}\|$	cardinality of the set $\mathcal{X}$
$H(X)$	entropy of the RV X
$I(X;Y)$	mutual Information between RV X and Y
$\mathbb{R}^n$	n-dimensional space of real numbers
$\mathbf{X} \sim \text{Mar}(\mathcal{X}, \mathbb{P})$	a Markov process over alphabet $\mathcal{X}$ and transition probability matrix $\mathbb{P}$
$\mathbb{1}(\cdot)$	Indicator function. It assumes value 0 when the condition provided is false and value 1 when the provided condition is true.

Abbreviations

RV	Random Variable
KLDR	Kullback-Leibler Divergence Rate
sGITMA	Sequential Generalized Information-Theoretic Markov Aggregation
AnnITMA	β-Annealing Information-Theoretic Markov Aggregation
sGITCC	Sequential Generalized Information-Theoretic Co-Clustering
AnnITCC	β-Annealing Information-Theoretic Co-Clustering
MCL	Markov Clustering
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
AP	Affinity Propagation
MP	Max-Product algorithm
MS	Max-Sum algorithm
EAP	Extended Affinity Propagation
DNN	Deep Neural Network
ReLU	Rectified Linear Unit

Bibliography

[1] M. A. Katsoulakis and J. Trashorras, "Information loss in coarse-graining of stochastic particle dynamics," *J. Stat. Phys.*, vol. 122, pp. 115–135, Jan. 2006.

[2] K. Deng, P. G. Mehta, and S. P. Meyn, "Optimal Kullback-Leibler aggregation via spectral theory of Markov chains," *IEEE Trans. Autom. Control*, vol. 56, pp. 2793–2808, Dec. 2011.

[3] Y. Xu, S. M. Salapaka, and C. L. Beck, "Aggregation of graph models and Markov chains by deterministic annealing," *IEEE Trans. Autom. Control*, vol. 59, pp. 2807–2812, Oct. 2014.

[4] B. C. Geiger, T. Petrov, G. Kubin, and H. Koeppl, "Optimal Kullback-Leibler aggregation via Information Bottleneck," *IEEE Trans. Autom. Control*, vol. 60, pp. 1010–1022, Apr. 2015.

[5] M. Vidyasagar, "Reduced-order modeling of Markov and hidden Markov processes via aggregation," in *Proc. IEEE Conf. on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010, pp. 1810–1815.

[6] R. A. Amjad, C. Blöchl, , and B. C. Geiger, "A generalized framework for Kullback-Leibler Markov aggregation," *"(accepted at) IEEE Trans. Autom. Control"*, Sep. 2017.

[7] N. Slonim and N. Tishby, "Document clustering using word clusters via the Information Bottleneck method," in *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Athens, Greece, Jul. 2000, pp. 208–215.

[8] N. Slonim, N. Friedman, and N. Tishby, "Unsupervised document classification using sequential information maximization," in *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Tampere, Aug. 2002, pp. 129–136.

[9] I. Dhillon, S. Mallela, and D. Modha, "Information-theoretic co-clusterings," in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, Washington, D.C., Aug. 2003, pp. 89–98.

[10] R. Bekkerman, R. El-Yaniv, and A. McCallum, "Multi-way distributional clustering via pairwise interactions," in *Proc. Int. Conference on Machine Learning (ICML)*, Bonn, 2005, pp. 41–48.

[11] P. Wang, C. Domeniconi, and K. B. Laskey, "Information Bottleneck co-clustering," in *Proc. Workshop on Text Mining*, Columbus, Ohio, May 2010.

[12] C. Blöchl, R. A. Amjad, and B. C. Geiger, "Co-clustering via information-theoretic markov aggregation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, pp. 720–732, April 2019.

[13] B. C. Geiger and R. A. Amjad, "Mutual information-based clustering: Hard or soft?" in *Proc. Int. ITG Conf. on Systems, Communications and Coding (SCC)*, Hamburg, Feb. 2017, pp. 1–6.

[14] S. M. Van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, 2001.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, Aug. 1996, pp. 226–231.

[16] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[17] R. A. Amjad, R. A. Khan, and M. Kleinsteuber, "Extended Affinity Propagation: Global discovery and local insights," *(submitted to) IEEE Transactions on Knowledge and Data Engineering.*

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, www.deeplearningbook.org.

[19] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proc. IEEE Information Theory Workshop (ITW)*, Jerusalem, Apr. 2015, pp. 1–5.

[20] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," arXiv:1703.00810v3 [cs.LG], Apr. 2017.

[21] R. A. Amjad and B. C. Geiger, "Learning representations for neural network-based classification using the Information Bottleneck principle," *(accepted at) IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[22] A. Abate, "Approximation metrics based on probabilistic bisimulations for general state-space Markov processes: A survey," *Electronic Notes in Theoretical Computer Science*, vol. 297, pp. 3 – 25, Dec. 2013.

[23] B. C. Geiger and C. Temmel, "Lumpings of Markov chains, entropy rate preservation, and higher-order lumpability," *J. Appl. Probab.*, vol. 51, pp. 1114–1132, Dec. 2014.

[24] R. M. Gray, *Entropy and Information Theory*, 2nd ed. New York, NY: Springer, 1990.

[25] D. H. Wolpert, J. A. Grochow, E. Libby, and S. DeDeo, "Optimal high-level descriptions of dynamical systems," arXiv:1409.7403v2 [cs.IT], Jun. 2015.

[26] J. G. Kemeny and J. L. Snell, *Finite Markov chains*, 2nd ed. Springer, 1976.

[27] M. Rosenblatt, "Functions of a Markov process that are Markovian," *Journal of Mathematics and Mechanics*, pp. 585–596, 1959.

[28] P. Buchholz, "Exact and ordinary lumpability in finite Markov chains," *Journal of Applied Probability*, vol. 31, pp. 59–75, Mar. 1994.

[29] L. Gurvits and J. Ledoux, "Markov property for a function of a Markov chain: a linear algebra approach," *Linear Algebra Appl.*, vol. 404, pp. 85–117, Jun. 2005.

[30] O. Pfante, N. Bertschinger, E. Olbrich, N. Ay, and J. Jost, "Comparison between different methods of level identification," *Advances in Complex Systems*, vol. 17, 2014.

[31] D. Blackwell, "The entropy of functions of finite-state Markov chains," in *Trans. first Prague Conf. Inf. theory*, Prague, Nov. 1957, pp. 13–20.

[32] O. Ordentlich, "Novel lower bounds on the entropy rate of binary hidden Markov processes," in *Proc. IEEE Int. Sym. on Information Theory (ISIT)*, Barcelona, Jul. 2016, pp. 690–694.

[33] J. Desharnais, A. Edalat, and P. Panangaden, "Bisimulation for labelled Markov processes," *Information and Computation*, vol. 179, pp. 163 – 193, Dec. 2002.

[34] G. Bian and A. Abate, "On the relationship between bisimulation and trace equivalence in an approximate probabilistic context," in *Proc. Int. Conf. on Foundations of Software Science and Computation Structure (FOSSACS)*, Uppsala, Apr. 2017, pp. 321–337.

[35] N. Tishby, F. C. Pereira, and W. Bialek, "The Information Bottleneck method," in *Proc. Allerton Conf. on Communication, Control, and Computing*, Monticello, IL, Sep. 1999, pp. 368–377.

[36] N. Slonim, "The Information Bottleneck: Theory and applications," Ph.D. dissertation, Hebrew University of Jerusalem, 2002.

[37] B. M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels," *IEEE Trans. Inf. Theory*, vol. 60, pp. 4544–4552, Aug. 2014.

[38] N. Slonim and N. Tishby, "Agglomerative Information Bottleneck," in *Advances in Neural Information Processing Systems (NIPS)*, Denver, CO, Nov. 1999, pp. 617–623.

[39] A. Blake and A. Zisserman, *Visual Reconstruction*. Cambridge, MA, USA: MIT Press, 1987.

[40] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 1st ed. Wiley Interscience, 1991.

[41] B. C. Geiger and Y. Wu, "Higher-order optimal Kullback-Leibler aggregation of Markov chains," in *Proc. Int. ITG Conf. on Systems, Communications and Coding (SCC)*, Hamburg, Feb. 2017, pp. 1–6.

[42] R. El-Yaniv and O. Souroujon, "Iterative double clustering for unsupervised and semi-supervised learning," in *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Dec. 2001, pp. 1025–1032.

[43] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Francisco, California, 2001, pp. 269–274.

[44] L. Labiod and M. Nadif, "Co-clustering for binary and categorical data with maximum modularity," in *Proc. IEEE Int. Conf. on Data Mining*, Vancouver, Dec. 2011, pp. 1140–1145.

[45] M. Ailem, F. Role, and M. Nadif, "Co-clustering document-term matrices by direct maximization of graph modularity," in *Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM)*, Melbourne, 2015, pp. 1807–1810.

[46] ——, "Sparse Poisson latent block model for document clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, pp. 1563–1576, Jul. 2017.

[47] S. Sra and I. S. Dhillon, "Nonnegative matrix approximation: Algorithms and applications." Dept. of Computer Sciences, University of Texas at Austin, Tech. Rep. TR-06-27, 2006.

[48] M. J. Barber, "Modularity and community detection in bipartite networks," *Phys. Rev. E*, vol. 76, Dec. 2007.

[49] P. Doreian, V. Batagelj, and A. Ferligoj, "Generalized block modeling of two-mode network data," *Social Networks*, vol. 26, pp. 29 – 53, 2004.

[50] M. Plumbley, "Information theory and unsupervised neural networks," Cambridge University Engineering Department, Tech. Rep. CUED/F-INFENG/TR. 78, 1991.

[51] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to Bregman co-clustering and matrix approximation," *Journal of Machine Learning Research*, vol. 8, pp. 1919–1986, Aug. 2007.

[52] C. Laclau, I. Redko, B. Matei, Y. Bennani, and V. Brault, "Co-clustering through optimal transport," in *Proc. Int. Conf. on Machine Learning (ICML)*, vol. 70, Sydney, Aug. 2017, pp. 1955–1964.

[53] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, pp. 1–19, Dec. 2015.

[54] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75 – 174, Feb. 2010.

[55] P. Arabie, "Cluster analysis in marketing research," *Advanced methods of marketing research*, pp. 160–189, 1994.

[56] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, pp. 14 863–14 868, Dec. 1998.

[57] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 7821–7826, Jun. 2002.

[58] T. H. Grubesic and A. T. Murray, "Detecting hot spots using cluster analysis and gis," in *Proc. 5th Annual Int. Conf. Crime Mapping Research*, vol. 26, Dec. 2001.

[59] S. E. Schaeffer, "Survey: Graph clustering," *Comput. Sci. Rev.*, vol. 1, pp. 27–64, Aug. 2007.

[60] P. Bachmann, *Die Analytische Zahlentheorie*. Leipzig: Teubner, 1894, vol. 2.

[61] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 2009, vol. 344.

[62] I.-E. Givoni, "Beyond affinity propagation: Message passing algorithms for clustering," Ph.D. dissertation, University of Toronto, 2012.

[63] D. Dueck, "Affinity propagation: clustering data by passing messages," Ph.D. dissertation, University of Toronto, 2009.

[64] I. E. Givoni, C. Chung, and B. J. Frey, "Hierarchical affinity propagation," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, Barcelona, Jul. 2011, pp. 238–246.

[65] C. D. Wang, J. H. Lai, C. Y. Suen, and J. Y. Zhu, "Multi-exemplar affinity propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2223–2237, Sep. 2013.

[66] M. Leone, Sumedha, and M. Weigt, "Clustering by soft-constraint affinity propagation," *Bioinformatics*, vol. 23, pp. 2708–2715, Sep. 2007.

[67] D. Tarlow, I. Givoni, and R. Zemel, "Hop-map: Efficient message passing with high order potentials," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9. Chia Laguna Resort, Sardinia: PMLR, May 2010, pp. 812–819.

[68] J. Hopcroft and R. Tarjan, "Algorithm 447: efficient algorithms for graph manipulation," *Communications of the ACM*, vol. 16, pp. 372–378, 1973.

[69] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," in *Proceedings of the 21st International Conference on Data Engineering*, Tokyo, Apr. 2005, pp. 341–352.

[70] L. Fu and E. Medico, "Flame, a novel fuzzy clustering method for the analysis of dna microarray data," *BMC Bioinformatics*, vol. 8, Jan 2007.

[71] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1273–1280, Sep. 2002.

[72] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recogn.*, vol. 41, Jan. 2008.

[73] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[74] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov. 1998.

[75] S. Pu, J. C. J. Wong, B. Turner, E. Cho, and S. J. Wodak, "Up-to-date catalogues of yeast protein complexes," in *Nucleic Acids Research*, vol. 37, 2009, pp. 825–831.

[76] J. Vlasblom and S. J. Wodak, "Markov clustering versus affinity propagation for the partitioning of protein interaction graphs," *BMC Bioinformatics*, vol. 10, Mar. 2009.

[77] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[78] S. R. Collins, P. Kemmeren, X.-C. Zhao, J. F. Greenblatt, F. Spencer, F. C. P. Holstege, J. S. Weissman, and N. J. Krogan, "Toward a comprehensive atlas of the physical interactome of saccharomyces cerevisiae," *Molecular & Cellular Proteomics*, vol. 6, pp. 439–450, Jan. 2007.

[79] S. Pu, J. Vlasblom, A. Emili, J. Greenblatt, and S. J. Wodak, "Identifying functional modules in the physical interactome of saccharomyces cerevisiae," *PROTEOMICS*, vol. 7, pp. 944–960, Mar. 2007.

[80] S. Belharbi, "Neural networks regularization through representation learning," Ph.D. dissertation, Normandie Université, INSA Rouen Normandie, LITIS laboratory, 2018.

[81] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. International Conference on Learning Representations (ICLR)*, Toulon, Apr. 2017.

[82] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[83] D. Moyer, S. Gao, R. Brekelmans, A. Galstyan, and G. Ver Steeg, "Invariant representations without adversarial training," in *Proc. Advances in Neural Information Processing Systems 31*, Montreal, Dec. 2018, pp. 9102–9111.

[84] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, 2018, pp. 634–646.

[85] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear information bottleneck," arXiv:1705.02436v7 [cs.IT], Sep. 2018.

[86] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *Proc. International Conference on Learning Representations (ICLR)*, Toulon, Apr. 2017.

[87] T. T. Nguyen and J. Choi, "Parametric information bottleneck to optimize stochastic neural networks," Oct. 2018, preprint. [Online]. Available: openreview.net/forum?id=ByED-X-0W

[88] A. A. Alemi, I. Fischer, and J. V. Dillon, "Uncertainty in the variational information bottleneck," arXiv:1807.00906v1 [cs.LG], Jul. 2018.

[89] P. K. Banerjee and G. Montufar, "The variational deficiency bottleneck," arXiv:1810.11677v1 [cs.LG], Oct. 2018.

[90] S. Belharbi, C. Chatelain, R. Hérault, and S. Adam, "Neural networks regularization through class-wise invariant representation learning," arXiv:1709.01867v4 [cs.LG], Dec. 2017.

[91] R. Liao, A. Schwing, R. Zemel, and R. Urtasun, "Learning deep parsimonious representations," in *Proc. Advances in Neural Information Processing Systems 29*, Barcelona, Dec. 2016, pp. 5076–5084.

[92] J.-H. Jacobsen, A. W. Smeulders, and E. Oyallon, "i-RevNet: Deep invertible networks," in *Proc. International Conference on Learning Representations (ICLR)*, Vancouver, May 2018.

[93] H. Xu and S. Mannor, "Robustness and generalization," *Machine Learning*, vol. 86, pp. 391–423, Mar. 2012.

[94] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: an empirical study," in *Proc. International Conference on Learning Representations (ICLR)*, Vancouver, May 2018.

[95] T. Zahavy, B. Kang, A. Sivak, J. Feng, H. Xu, and S. Mannor, "Ensemble robustness and generalization of stochastic deep learning algorithms," in *Proc. International Conference on Learning Representations (ICLR)*, Vancouver, May 2018.

[96] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu, "Reshaping deep neural network for fast decoding by node-pruning," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, May 2014, pp. 245–249.

[97] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. International Conference on Learning Representations (ICLR)*, Toulon, Apr. 2017.

[98] K. Liu, R. A. Amjad, and B. C. Geiger, "Understanding individual neuron importance using information theory," arXiv:1804.06679v1 [cs.LG], Apr. 2018.

[99] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, New Orleans, May 2019.

[100] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, Aug. 2013.

[101] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *Journal of Machine Learning Research*, vol. 19, pp. 1–34, 2018.

[102] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *Proc. International Conference on Learning Representations (ICLR)*, Vancouver, May 2018.

[103] A. Kolchinsky, B. D. Tracey, and S. Van Kuyk, "Caveats for information bottleneck in deterministic scenarios," in *Proc. International Conference on Learning Representations (ICLR)*, New Orleans, May 2019.

[104] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[105] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," in *Proc. International Conference on Learning Representations (ICLR)*, Toulon, Apr. 2017.

[106] A. Achille and S. Soatto, "Information dropout: Learning optimal representations through noisy computation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 2897–2905, Dec. 2018.

[107] D. J. Strouse and D. J. Schwab, "The deterministic information bottleneck," in *Proc. of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, Jun. 2016, pp. 696–705.

[108] Z. Goldfeld, E. van den Berg, K. H. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, "Estimating information flow in neural networks," arXiv:1810.05728v3 [cs.LG], Nov. 2018.

[109] M. Vera, P. Piantanida, and L. R. Vega, "The role of the information bottleneck in representation learning," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Vail, CO, Jun. 2018, pp. 1580–1584.

[110] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv:1312.6114v10 [cs.LG], Dec. 2013.

[111] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[112] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, pp. 463–482, Nov. 2002.

[113] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *Proc. 34th International Conference on Machine Learning (ICML)*, Sydney, Aug. 2017, pp. 233–242.

[114] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," in *Proc. International Conference on Learning Representations (ICLR)*, Toulon, Apr. 2017.

[115] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, "Reversible architectures for arbitrarily deep residual neural networks," in *Proc. AAAI Conference on Artificial Intelligence*, New Orleans, Feb. 2018.

[116] B. C. Geiger and G. Kubin, *Information Loss in Deterministic Signal Processing Systems*, ser. Understanding Complex Systems. Springer, 2018.

[117] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[118] J. Pearl, "Probabilistic reasoning in intelligent systems: Networks of plausible reasoning," 1988.

[119] I. Csiszár, "On the dimension and entropy of order α of the mixture of probability distributions," *Acta Mathematica Hungarica*, vol. 13, pp. 245–255, 1962.

[120] A. Rényi, "On the dimension and entropy of probability distributions," *Acta Mathematica Hungarica*, vol. 10, pp. 193–215, Mar. 1959.

[121] B. R. Hunt and V. Y. Kaloshin, "How projections affect the dimension spectrum of fractal measures," *Nonlinearity*, vol. 10, p. 1031, 1997.

[122] P. Mattila, M. Morán, and J.-M. Rey, "Dimension of a measure," *Studia Math*, vol. 142, pp. 219–233, 2000.

[123] Y. Wu, S. Shamai (Shitz), and S. Verdú, "Information dimension and the degrees of freedom of the interference channel," *IEEE Trans. Inf. Theory*, vol. 61, pp. 256–279, Jan. 2015.

[124] Y. Wu and S. Verdú, "Rényi information dimension: Fundamental limits of almost lossless analog compression," *IEEE Trans. Inf. Theory*, vol. 56, pp. 3721–3748, Aug. 2010.