OPTIMIZATION OF MUTUAL INFORMATION IN LEARNING: EXPLORATIONS IN SCIENCE

DJ STROUSE

A DISSERTATION

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE

BY THE DEPARTMENT OF PHYSICS

ADVISORS: WILLIAM BIALEK & DAVID SCHWAB

SEPTEMBER 2018

Abstract:

This thesis explores three applications of information theory in machine learning, all involving the optimization of information flow in some learning problem. In Chapter 2, we introduce a method for extracting the most informative bits that one signal contains about another. Our method, the deterministic information bottleneck (DIB), is an alternative formulation of the information bottleneck (IB). In Chapter 3, we adapt the DIB to the problem of finding the most informative clusterings of geometric data. We also introduce an approach to model selection that naturally emerges within the (D)IB framework. In Chapter 4 we introduce an approach to encourage / discourage agents in a multi-agent reinforcement learning setting to share information with one another. We conclude in Chapter 5 by discussing ongoing and future work in these directions.

Acknowledgments:

I entered college thinking I would study to be a film director, so it should be abundantly clear that there are many people to thank for their strong influences on me since then. Gratitude is especially directed at the following people...

To my advisors, Bill Bialek & David Scwhab. To Bill for always treating his students as colleagues. He gave me the full freedom to pursue my interests. He surrounded me with excellent scientists, collaborators, and friends. And he set the bar for giving a crystal-clear talk at the chalkboard. To David for being a co-conspirator in all my endeavors. A maximum likelihood model of my publication record during my PhD would suggest that we will be co-authors on every paper I write. I hope that's true.

To my committee members, Jonathan Pillow and Josh Shaevitz, for their time and feedback. Additionally, to Jonathan for taking me in to PNI as a refuge during the construction in Icahn, and for introducing me to the wonderful people in his research group. (And additionally to Josh for introducing me to Żubrówka.)

To my previous research advisors, Paolo Zanardi, Bartlett Mel, Andrew Childs, and Máté Lengyel, for teaching me how to do science. Paolo introduced me to the intellectual adventure that is physics; I wouldn't be a scientist without him. Bartlett introduced me to thinking about computation in the brain and opened many doors for me. Andrew taught me how to write a paper. Máté introduced me to probabilistic inference and machine learning; the seeds of everything I've worked on since were planted during my time working with him.

To the people at Princeton who made my time there enjoyable - Zach Sethna, Farzan Beroz, Bin Xu, Aitor Lewkowycz, Mikio Aoi, Adam Charles, Jasmin Imran-Alsous, Daniel Hellwig, Cheryne Jonay, Bas van Opheusden, Jaan Altosaar, and Dima Krotov. It is no secret to those close to me that I prefer big cities to small towns, but your presence in Princeton made it a place I'll think of nostalgically for years to come.

To my research advisor at DeepMind, Matt Botvinick, for introducing me to reinforcement learning and modern AI. And to the many wonderful colleagues at DeepMind who made my time there fun and educational, including Jane Wang, David Pfau, Neil Rabinowitz, Ben Poole, Kelsey Allen, Nicole Rafidi, Kim Stachenfeld, Vincent Dumoulin, Alex Novikov, Marco Fraccaro, Casper Sønderby, Drew Jaegle, Chongli Qin, Gabriel Pereyra, Kevin McKee, Tina Zhu, Francis Song, and Raphael Köster. I learned more from all of you than I thought possible in four months.

To my colleagues at Spotify, Zack Nichols, Zac Pustejovsky, Ye Zhao, Jeremy Cohen, and the rest of Paradox for giving me my first experience "in industry" and providing me a home in the city before I had one.

To the Hertz Foundation, not for funding me (well, that too I suppose), but for introducing me to many of my favorite people, including Dan Roberts, Max Kleiman-Weiner, Aman Sinha, Ashvin Bashyam, Cooper Rinzler, Vyas Ramanananana, Jim Valcourt, Kelly Moynihan, Jon Russell, and many more.

To the handful of people I pitch every idea, project, and decision - Dan Roberts, Max Kleiman-Weiner, Jeff Seely, and David Schwab. I'm immensely fortunate to count you as friends and colleagues. Dan encourages me to always strive for the upgrade pick. Max provides encouragement when is is needed, and discouragement when it is (more often) needed. Jeff is the mysterious one. David is the bad bo- oh sorry, this is a thesis, not a boy band. In any case, a copy will be provided to you all in the form of a screenshot.

Finally, of course, to my family - to my parents for unconditionally supporting me in every silly thing I pursued, and to my sisters for putting up with me at my most insufferable.

Contents

Chapter 1

Introduction

This is a thesis about measuring and controlling the flow of information in learning problems. That learning should *involve* the flow of information in a colloquial sense should be intuitive - what is learning but the acquisition and processing of information? However, here we mean something more specific, both in terms of *information* and *learning*. By information, we mean it in the sense of Claude Shannon's *information theory (Shannon 1948)*. And by learning, we mean what is commonly referred to today as *machine learning*. We will first provide a brief introduction to both topics, before moving on to outline our program of research at their intersection.

1.1 Information theory

Information theory was originally developed by Claude Shannon to understand the fundamental limits of communication (Shannon 1948). As a member of the research team at Bell Labs, which operated a monopoly on American telecommunications at the time, Shannon and his team had practical reasons to be concerned with such questions. While a full review of information theory is outside of our scope here (Cover and Thomas (2006) is the excellent standard textbook on the topic), we will focus on

two of the most important quantities that Shannon introduced - entropy and mutual information - quantities that will appear again and again in the following chapters.

1.1.1 Entropy

Entropy is a measure of uncertainty. Shannon derived it by stating three axioms he wanted a measure of uncertainty to abide by and showing that entropy was the unique quantity that satisfied them. For a discrete random variable X taking on values in $\mathcal{X}$, the entropy $H(X)$ is defined as:[1]

$$H(X) \equiv -\sum_{x \in \mathcal{X}} p(x) \log p(x)\,, \tag{1.1}$$

where $p(x)$ is the probability mass function of X. Entropy is always positive. The lowest value it takes on is zero. This happens when $p(x)$ has all of its probability mass on one value of x (i.e. $p(x^*) = 1$, $p\big(x^{'}\big) = 0$ for all $x^{'} \neq x^*$), or in other words, when we are *certain* of the value of X. The highest value it takes on is $\log|\mathcal{X}|$. This happens when $p(x)$ is spread evenly across all values of x, or in other words, when we have no knowledge favoring any particular value of X over another and are thus maximally *uncertain* of its value.

Entropy is also defined for continuous variables by replacing the sum with an integral. That is, for a continuous random variable X taking of values in $\mathcal{X}$, the so-called *differential entropy* $h(X)$ is defined as:

$$h(X) = -\int_{\mathcal{X}} f(x) \log f(x)\, dx, \tag{1.2}$$

where $f(x)$ is the probability density function of X. Differential entropy differs from its discrete cousin in that it *can* be negative. To see this, consider the entropy of $f(x) = \frac{1}{a}$ for $x \in [0, a]$. Then $h(X) = -\int_0^a \frac{1}{a} \log \frac{1}{a} dx = \log a$, which for $a < 1$ is

[1] Unless otherwise stated, all logs are base 2, the units of which are typically referred to as *bits*.

negative. In fact, as $a \to 0$, $h(X)$ becomes *infinitely* negative. Thus, if one wished to *minimize* the differential entropy as part of some objective function (see section 1.2), then the objective could potentially diverge, leading to a degenerate solution with $f(X)$ infinitely peaked at some value. This is an issue we run into in our own work in chapter 2.

One reason for the importance of entropy (and differential entropy) is its role in the fundamental limits of data compression. The basic idea behind (lossless) data compression is to use the shortest descriptions for the most frequent messages, and longer descriptions for the less frequent ones. Compression is formalized as as a *source code* C mapping from a random variable X to $\mathcal{B}^*$, the set of finite-length binary strings (i.e. strings of 0s and 1s). Letting $C(x)$ be the codeword assigned to $x \in \mathcal{X}$ and $l(x)$ the length of $C(x)$, then the expected length $L(C)$ of a source code is $L(C) = \sum_{x \in \mathcal{X}} p(x)\, l(x)$. It is natural to wonder: for a given source distribution $p(x)$, what source code minimizes $L(C)$ and what is the minimal value $L^*(C)$? A full formal discussion of this question is unfortunately too long to include here, but (very) roughly speaking, the minimal-length source code achieves $L^*(C) \approx H(X)$. That is, the entropy of a random variable is the fundamental limit on its compression. This fact about entropy will motivate the algorithm we introduce in chapter 2.

Any physicist reading at this point is probably wondering: wait, doesn't the concept of entropy predate Claude Shannon by a long shot? How does the entropy introduced here relate to the one from statistical physics? We answer the second question first. The entropy in physics is typically defined as the logarithm of the number of microstates in a system, which when all microstates are equally likely, corresponds exactly to the definition of entropy above. As for the first question, legend has it that when Claude Shannon derived his measure of uncertainty axiomatically, he approached his colleague John von Neumann and asked what he should call it. von Neumann's response was: "You should call it entropy, for two reasons. In the first place

your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate you will always have the advantage."

1.1.2 Mutual information

Mutual information is a measure of the information that two variables contain about one another. To define and understand it, we first introduce two other quantities: the conditional entropy and the relative entropy.

Conditional entropy is a measure of the uncertainty in one variable after observing another. The conditional entropy of the random variable Y conditioned on the random variable X is defined as:

$$H(Y \mid X) = \sum_{x \in \mathcal{X}} p(x)\, H(Y \mid X = x) \tag{1.3}$$

$$= -\sum_{x \in \mathcal{X}} p(x) \sum_{y} p(y \mid x) \log p(y \mid x)\,. \tag{1.4}$$

Thus, it is simply the average entropy of a variable after conditioning on knowledge of another variable.

Relative entropy, or the Kullback–Leibler (KL) divergence, is a measure of the distance between two probability distributions. The KL divergence between two probability mass functions $p(x)$ and $q(x)$ over the same space $\mathcal{X}$ is defined as:

$$D_{\mathrm{KL}}[p \mid q] \equiv \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{1.5}$$

The KL divergence is bounded below by 0,[2] and saturates this lower bound if and only if $p = q$. The KL divergence is *not* bounded above though: if for any symbol $x \in \mathcal{X}$, $p(x) > 0$ but $q(x) = 0$, then $D_{\mathrm{KL}}[p \mid q] = \infty$. The KL divergence is also not

[2]The nonnegativity of the KL is extensively used in many proofs in information theory and machine learning.

symmetric (i.e. $D_{\mathrm{KL}}[p \mid q] \neq D_{\mathrm{KL}}[q \mid p]$), so the ordering of p and q in the notation matters.

With these two quantities in mind, we provide four equivalent definitions of mutual information. The mutual information $I(X;Y)$ between two random variables X and Y taking values in $\mathcal{X}$ and $\mathcal{Y}$, respectively, is defined as:

$$I(X;Y) \equiv \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)} \tag{1.6}$$

$$= D_{\mathrm{KL}}[p(x,y) \mid p(x)\,p(y)] \tag{1.7}$$

$$= H(X) - H(X \mid Y) \tag{1.8}$$

$$= H(Y) - H(Y \mid X)\,. \tag{1.9}$$

The first definition is written in terms of the elemental probabilities, but it is the latter three that give the intuition behind the mutual information.

In equation 1.7, we see that the mutual information is the KL divergence between the joint distribution $p(x,y)$ and the factorized distribution $p(x)\,p(y)$. Thus, mutual information measures the distance between the true distribution of two variables and their factorized version. When $p(x,y) = p(x)\,p(y)$, we say that $p(x,y)$ "factorizes" and X and Y are "statistically independent." In that case, $I(X;Y) = 0$. The further from statistically independent that X and Y are, or in other words, the more statistically *dependent* they are, the higher the mutual information between them. We use the words "mutual" and "between" because mutual information is symmetric (i.e. $I(X;Y) = I(Y;X)$). Because of this, mutual information can be thought of as a general measure of the association between two variables, one that measures all types of nonlinear dependence, unlike a correlation coefficient (such as a Pearson correlation, which only measures linear dependence) (Kinney and Atwal 2014).

In equations 1.8 and 1.9, we see that the mutual information is the difference between two entropies. We can interpret this difference as the *average reduction*

in uncertainty in one variable when observing the other. Of course, intuitively, a reduction in uncertainty about something means one has gained information about it, and it is this interpretation that gives mutual information its name (the "mutual" part again comes from its symmetry).

Besides its use as a general measure of dependence, mutual information also derives its usefulness from its role in the fundamental limits of communication. Communication is formalized as choosing an encoding function from M possible messages to a sequence of n codewords from a set of input symbols $\mathcal{X}$ which are passed through a fixed noisy channel described by $p(y \mid x)$. The receiver receives n codewords in a set of output symbols $\mathcal{Y}$ and decodes them to guess the original message. The "channel capacity" is then the highest rate in bits per channel use at which information can be communicated with arbitrarily low probability, that is the maximal value of $C = \frac{\log M}{n}$ for which decoding errors tend to zero as $n \to \infty$. Shannon's most famous result is that the channel capacity is equal to the maximum mutual information, $C = \max_{p(x)} I(X;Y)$.[3] We will refer to this result in our discussion of the information bottleneck in chapter 2.

Although less commonly associated with physics than entropy, mutual information has also proved to be a useful concept in physics. In quantum physics, entropies can be generalized to Von Neumman entropies and entanglement entropies. One can form mutual informations using these entropies, which provide a measure of total correlation, which in some special cases can be a measure of pure entanglement. In Hayden et al. (2013), the quantum version of 3-spatial region mutual information was studied. The positivity of the quantity $I_3 = I(A;BC) - I(A;B) - I(A;C)$ is known as monogamy. For holographic quantum field theories, the mutual informations of any three disjoint spatial regions are monogamous. This is interesting because it's a measure of entanglement, and it suggests the the correlations in holographic field theories are quantum and not classical. A pretty famous use of entropy is Page (1993),

[3] Again, this is a highly streamlined discussion; for details, see Cover and Thomas (2006).

which considers the average entropy of a random subsystem. This paved the way for work on applying information theory to study the black hole information problem and quantum chaos. In Shenker and Stanford (2014), mutual information was used in holography to study the butterfly effect. In Hosur et al. (2016), I_3 was studied in operators and related to quantum chaos. In their setup, the I_3 is always less than 0, but they suggest and give evidence that the negativity of I_3 is a measure of information theoretic scrambling.

1.1.3 Summary

To summarize, entropy measures uncertainty, and is the fundamental limit of compression. Mutual information measures dependence, or the average reduction in uncertainty in one variable when observing another, and is the fundamental limit of communication. Both play important roles in physics, probabilistic inference, and across the sciences.

1.2 Machine learning

Machine learning is fundamentally about teaching computers to solve problems that we don't know how to explicitly tell them to solve. In some cases, we humans might be able to perform the task, but we want to teach a computer to do so for reasons of efficiency and scalability. In other cases, we humans might not even know how to perform the task ourselves.

A full introduction to machine learning is again far beyond the scope of this thesis (for an somewhat quirky introduction popular with physicists, seeMacKay (2002); for popular mainstream choices for introductory courses, see Bishop (2006) and Murphy (2012)). Here, we simply introduce a few key concepts necessary to understanding the work in this thesis. First, we outline the general "optimization-based" approach to

machine learning. Then, we describe the three typical categories of machine learning problems.

1.2.1 Optimization-based learning

Imagine you were asked to write a program to calculate a checking account balance, given a list of the deposits and withdrawals. It doesn't take a genius to sum up the deposits and subtract the withdrawals. Now imagine you are asked to write a program to automatically distinguish photos of dogs from cats. You might want to write a rule that checks for pointy versus floppy ears, for example. But how should you tell the computer how to identify ears? Maybe you want to tell the computer to look on the top of the animal's head. But how do you instruct the computer to find a head? What if the ears aren't visible? What if the photo is of an alert Doberman with pointy ears? While distinguishing a dog and cat is extremely easy for you, describing algorithmically how to do so is not.

For this problem, it turns out to be easier to 1) collect many photos of dogs and cats with correct labels and 2) write a *learning algorithm* for the computer that tunes the parameters of a model mapping images to correct labels. By model, we simply mean a parameterized (possibly stochastic) function f_θ mapping from photos to labels (e.g. a convolutional neural network).[4] The learning algorithm to train this model typically involves defining an "objective function" that is then maximized (or a "loss function" that is minimized), such as the probability that your model assigns the correct label to an image in your dataset. In addition to choosing the model parameterization

[4]There appears to be a wide cultural gap in the use of the word "model" in physics and machine learning. In both cases, a model is a mathematical formalization of the relationship between several variables. However, in physics, a model typically has very few parameters, and the parameters are interpretable and set by carrying out a small number of precise experiments. Repetitions of these experiments increase our collective belief over the particular, precise values of the model parameters (the charge of the electron, for example, is measured to about 10 significant digits). In machine learning, on the other hand, a model typically is vastly overparametrized, and the parameters are uninterpretable and set by optimizing a loss over some large dataset. Repeatedly fitting a model could be expected to lead to very different parameter values from experiment to experiment.

and objective, you must also choose a procedure for changing the parameters of your model to optimize your objective (e.g. stochastic gradient descent).

This combination of choosing a dataset, an objective function, a model, and an optimization procedure charaterizes a large fraction of machine learning research today. This approach has enabled computers to play video games (Mnih et al. 2016), synthesize speech (van den Oord et al. 2016), read lips (Shillingford et al. 2018), play board games at super-human level (Silver et al. 2016), and much more. Moreover, the work of the next three chapters will all fall under this category.

1.2.2 Supervised, unsupervised, and reinforcement learning

It is common in machine learning to distinguish between three different types of learning problems.

In *supervised learning*, we are supplied with a data set of N inputs x_i and their correct associated outputs y_i. The goal is to take this data set $\{(x_i, y_i)\}_{i=1}^N$ and fit a model f_θ to the data such that $f_\theta(x) \approx y$ by adjusting the parameters θ. This approach is referred to as "supervised" because we are given the correct outputs y_i. The image classification problem in the last section is an example of supervised learning.

In *unsupervised learning*, no such labels are supplied. Instead, we are simply given a data set $\{x_i\}_{i=1}^N$ and told to find "structure." If this sounds open-ended, that's because it is. Much of the difficulty of unsupervised learning is in defining what "structure" means. Clustering is an example of unsupervised learning.

In *reinforcement learning* (RL), we are supplied with an intermediate level of feedback: a score. Consider learning to play a video game. In a supervised setting, someone would tell us exactly what to do in order to perform well (e.g. "jump on that platform and collect the key"). In an RL setting, we instead only receive points in proportion to how well we did. The goal is to get as many points as possible. In addition to this difference in feedback, RL settings typically also involve *sequential*

decision-making. In (un)supervised learning, the data point we see at time $t+1$ typically does not depend on the data point we receive or the label we apply to it at time t. Instead we receive data points x_i at random. In RL, however, we instead navigate an "agent" through "states" s_t in an "environment", taking "actions" a_t that carry this agent (perhaps stochastically) to new states s_{t+1}. Thus, key features of RL include long-term planning and exploration vs exploitation tradeoffs. In many settings, there are multiple agents present and interacting with one another and the same environment. This setting is known as "multi-agent reinforcement learning" (MARL) and introduces new complexities such as deciding whether to cooperate or compete, and inferring other agent's knowledge and intentions through "theory of mind."

The work in this thesis touches on each of these areas. In chapters 2 and 3, we discuss work at the intersection of supervised and unsupervised learning, while in chapter 4, we discuss work on multi-agent reinforcement learning.

1.2.3 Summary

Machine learning is about teaching machines to learn (duh) so that we can expand the range of tasks they are able to perform beyond those for which we are able to write an explicit algorithm. Tasks are typically categorized as supervised (explicit feedback), unsupervised (no feedback), or reinforcement learning problems (implicit feedback). In all three cases, much of this is done via optimization-based learning.

1.3 Information theory + machine learning

Information theory and machine learning have a long, interwoven history. This should come as no surprise, since uncertainty (entropy) and dependence (mutual information) are key concepts in learning. In fact, MacKay (2002) saw the fields as so inseparable that he wrote a textbook covering both at once.

Our own program of research lies exactly at this intersection. In the following chapters, we employ the information-theoretic concepts of entropy (section 1.1.1) and mutual information (section 1.1.2) in a variety of optimization-based learning problems (section 1.2.1). Chapters 2 and 3 focus on problems that lie somewhere between supervised and unsupervised learning, while chapter 4 focuses on (multi-agent) reinforcement learning (section 1.2.2).

Chapter 2

The deterministic information bottleneck

In any problem of prediction, a primary question is how to select the right "features" that make that prediction problem tractable. Historically, engineers might hand tune such features (e.g. number of blue pixels in an image), and then run a simple predictive model on top (e.g. linear regression). However, a more general and powerful approach is to *learn* the useful features automatically from data. In machine learning, we refer to such prediction problems as *supervised learning* (see section 1.2.2), and the subtask of finding good features as *feature selection* or *representation learning*.

The information bottleneck is a formalization of the feature selection problem in the language of information theory (Tishby et al. 1999) . Imagine we observe one signal, X, and wish to predict another signal, Y, from our measurement; what features of X should we pay attention to? Tishby, Pereria, and Bialek proposed to consider the features as an encoding T of X, and to choose the mapping from X to T which minimizes the objective $L_{\text{IB}} = I(X;T) - \beta I(T;Y)$. The first term in this objective tells us that the features should be *minimal* in the sense that they encode as few bits of the input as possible. The second term tells us that the features should be

predictive in the sense that they contain information about the signal we wish to predict. By measuring minimality as $I(X;T)$, the authors were implicitly "squeezing" the communication channel between X and T (see chapter 1.1.2). The result was a stochastic encoder $q_\beta(T \mid X)$ which balanced the goals of minimality and prediction, with the relative weighting dependent on the tradeoff parameter β.

I first encountered the information bottleneck when I became interested in formalizations of prediction in the brain. There were two things that struck me as odd. First, why measure minimality with mutual information instead of entropy? In the language of compression / source coding (see section 1.1.1), a minimal signal is one with low entropy, because it can be represented with fewer symbols. In terms of neural systems, this would mean being able to use fewer neurons. Second, why was the optimal encoding stochastic? If the goal is to build informative, predictive features, shouldn't noise hurt?

My original thinking on this problem has evolved since then, but that is the setting for the following chapter - how can we alter the information bottleneck picture on representation learning to focus on minimal signals in the language of source coding?

This chapter appeared as Strouse and Schwab (2017).

Abstract

Lossy compression and clustering fundamentally involve a decision about what features are relevant and which are not. The information bottleneck method (IB) by Tishby, Pereira, and Bialek formalized this notion as an information-theoretic optimization problem and proposed an optimal tradeoff between throwing away as many bits as possible, and selectively keeping those that are most important. In the IB, compression is measured by mutual information. Here, we introduce an alternative formulation that replaces mutual information with entropy, which we call the deterministic information bottleneck (DIB), that we argue better

captures this notion of compression. As suggested by its name, the solution to the DIB problem turns out to be a deterministic encoder, or hard clustering, as opposed to the stochastic encoder, or soft clustering, that is optimal under the IB. We compare the IB and DIB on synthetic data, showing that the IB and DIB perform similarly in terms of the IB cost function, but that the DIB significantly outperforms the IB in terms of the DIB cost function. We also empirically find that the DIB offers a considerable gain in computational efficiency over the IB, over a range of convergence parameters. Our derivation of the DIB also suggests a method for continuously interpolating between the soft clustering of the IB and the hard clustering of the DIB.

2.1 Introduction

Compression is a ubiquitous task for humans and machines alike (Cover and Thomas 2006, MacKay 2002). For example, machines must turn the large pixel grids of color that form pictures into small files capable of being shared quickly on the web (Wallace 1991), humans must compress the vast stream of ongoing sensory information they receive into small changes in the brain that form memories (Kandel et al. 2013), and data scientists must turn large amounts of high-dimensional and messy data into more manageable and interpretable clusters (MacKay 2002).

Lossy compression involves an implicit decision about what is relevant and what is not (Cover and Thomas 2006, MacKay 2002). In the example of image compression, the algorithms we use deem some features essential to representing the subject matter well, and others are thrown away. In the example of human memory, our brains deem some details important enough to warrant attention, and others are forgotten. And in

the example of data clustering, information about some features is preserved in the mapping from data point to cluster ID, while information about others is discarded.

In many cases, the criterion for "relevance" can be described as information about some other variable(s) of interest. Let's call X the signal we are compressing, T the compressed version, Y the other variable of interest, and $I(T;Y)$ the "information" that T has about Y (we will formally define this later). For human memory, X is past sensory input, T the brain's internal representation (e.g. the activity of a neural population, or the strengths of a set of synapses), and Y the features of the future environment that the brain is interested in predicting, such as extrapolating the position of a moving object. Thus, $I(T;Y)$ represents the predictive power of the memories formed Palmer et al. (2015). For data clustering, X is the original data, T is the cluster ID, and Y is the target for prediction, for example purchasing or ad-clicking behavior in a user segmentation problem. In summary, a good compression algorithm can be described as a tradeoff between the compression of a signal and the selective maintenance of the "relevant" bits that help predict another signal.

This problem was formalized as the "information bottleneck" (IB) by Tishby, Pereira, and Bialek (Tishby et al. 1999). Their formulation involved an information-theoretic optimazation problem, and resulted in an iterative soft clustering algorithm guaranteed to converge to a local (though not necessarily global) optimum. In their cost functional, compression was measured by the mutual information $I(X;T)$. This compression metric has its origins in rate-distortion theory and channel coding, where $I(X;T)$ represents the maximal information transfer rate, or capacity, of the communication channel between X and T (Cover and Thomas 2006). While this approach has its applications, often one is more interested in directly restricting the amount of resources required to represent T, represented by the entropy $H(T)$. This latter notion comes from the source coding literature and implies a restriction on the *representational cost* of T (Cover and Thomas 2006). In the case of human memory, for example, $H(T)$

would roughly correspond to the number of neurons or synapses required to represent or store a sensory signal X. In the case of data clustering, $H(T)$ is related to the number of clusters.

In the following paper, we introduce an alternative formulation of the IB, called the deterministic information bottleneck (DIB), replacing the compression measure $I(X;T)$ with $H(T)$, thus emphasizing contraints on representation, rather than communication. Using a clever generalization of both cost functionals, we derive an iterative solution to the DIB, which turns out to provide a hard clustering, or deterministic mapping from X to T, as opposed to the soft clustering, or probabilitic mapping, that IB provides. Finally, we compare the IB and DIB solutions on synthetic data to help illustrate their differences.

2.2 The original information bottleneck (IB)

Given the joint distribution $p(x, y)$, the encoding distribution $q(t|x)$ is obtained through the following "information bottleneck" (IB) optimization problem:

$$\min_{q(t|x)} L[q(t|x)] = I(X;T) - \beta I(Y;T)\,, \tag{2.1}$$

subject to the Markov constraint $T \leftrightarrow X \leftrightarrow Y$. Here $I(X;T)$ denotes the mutual information between X and T, that is $I(X;T) \equiv H(T) - H(T|X) = \sum_{x,t} p(x,t) \log\left(\frac{p(x,t)}{p(x)p(t)}\right) = D_{KL}[p(x,t) \mid p(x)\,p(t)]$,[1] where D_{KL} denotes the Kullback-Leibler divergence.[2] The

[1] Implicit in the summation here, we have assumed that X, Y, and T are discrete. We will be keeping this assumption throughout for convenience of notation, but note that the IB generalizes naturally to X, Y, and T continuous by simply replacing the sums with integrals (see, for example, ?).

[2] For those unfamiliar with it, mutual information is a very general measure of how related two variables are. Classic correlation measures typically assume a certain form of the relationship between two variables, say linear, whereas mutual information is agnostic as to the details of the relationship. One intuitive picture comes from the entropy decomposition: $I(X;Y) \equiv H(X) - H(X|Y)$. Since entropy measures uncertainty, mutual information measures the *reduction in uncertainty* in one variable when observing the other. Moreover, it is symmetric ($I(X;Y) = I(Y;X)$), so the information is *mutual*. Another intuitive picture comes from the D_{KL} form: $I(X;Y) \equiv D_{KL}[p(x,y) \mid p(x)\,p(y)]$.

first term in the cost function is meant to encourage compression, while the second relevance. β is a non-negative free parameter representing the relative importance of compression and relevance, and our solution will be a function of it. The Markov constraint simply enforces the probabilistic graphical structure of the task; the compressed representation T is a (possibly stochastic) function of X and can only get information about Y through X. Note that we are using p to denote distributions that are given and fixed, and q to denote distributions that we are free to change and that are being updated throughout the optimization process.

Through a standard application of variational calculus (see Section 2.A for a detailed derivation of the solution to a more general problem introduced below), one finds the formal solution:[3]

$$q(t|x) = \frac{q(t)}{Z(x,\beta)} \exp[-\beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]] \tag{2.2}$$

$$q(y|t) = \frac{1}{q(t)} \sum_x q(t|x)\, p(x,y)\,, \tag{2.3}$$

where $Z(x,\beta) \equiv \exp\left[-\frac{\lambda(x)}{p(x)} - \beta \sum_y p(y \mid x) \log \frac{p(y|x)}{p(y)}\right]$ is a normalization factor, and $\lambda(x)$ is a Lagrange multiplier that enters enforcing normalization of $q(t \mid x)$.[4] To get an intuition for this solution, it is useful to take a clustering perspective - since we are compressing X into T, many X will be mapped to the same T and so we can think of the IB as "clustering" xs into their cluster labels t. The solution $q(t|x)$ is then likely to map x to t when $D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]$ is small, or in other words, when the distributions $p(y|x)$ and $q(y|t)$ are similar. These distributions are similar to the

Since D_{KL} measures the distance between two probability distributions, the mutual information quantifies how far the relationship between x and y is from a probabilistically independent one, that is how far the joint $p(x,y)$ is from the factorized $p(x)\,p(y)$. A very nice summary of mutual information as a dependence measure is included in Kinney and Atwal (2014).

[3]For the reader familiar with rate-distortion theory, eqn 2.2 can be viewed as the solution to a rate-distortion problem with distortion measure given by the KL-divergence term in the exponent.

[4]More explicitly, our cost function L also implicitly includes a term $\sum_x \lambda(x)\,[1 - \sum_t q(t|x)]$ and this is where $\lambda(x)$ comes in to the equation. See Section 2.A for details.

extent that x and t provide similar information about y. In summary, inputs x get mapped to clusters t that maintain information about y, as was desired.

This solution is "formal" because the first equation depends on the second and vice versa. However, Tishby et al. (1999) showed that an iterative approach can be built on the the above equations which provably converges to a local optimum of the IB cost function (eqn. 2.1).

Starting with some initial distributions $q^{(0)}(t|x)$, $q^{(0)}(t)$, and $q^{(0)}(y|t)$, the n^{th} update is given by:[5]

$$q^{(n)}(t|x) = \frac{q^{(n-1)}(t)}{Z^{(n)}(x,\beta)} \exp\left[-\beta D_{\mathrm{KL}}\left[p(y|x) \mid q^{(n-1)}(y|t)\right]\right] \tag{2.4}$$

$$q^{(n)}(t) = \sum_x q^{(n)}(t|x)\, p(x) \tag{2.5}$$

$$q^{(n)}(y|t) = \frac{1}{q^{(n)}(t)} \sum_x q^{(n)}(t|x)\, p(x,y)\,. \tag{2.6}$$

Note that the first equation is the only "meaty" one; the other two are just there to enforce consistency with the laws of probability (e.g. that marginals are related to joints as they should be). In principle, with no proof of convergence to a global optimum, it might be possible for the solution obtained to vary with the initialization, but in practice, the cost function is "smooth enough" that this does not seem to happen. This algorithm is summarized in algorithm 2.1. Note that while the general solution is iterative, there is at least one known case in which an analytic solution is possible, namely when X and Y are jointly Gaussian (Chechik et al. 2005).

[5]Note that, if at step m no xs are assigned to a particular $t = t^*$ (i.e. $q(t \mid x) = 0\ \forall x$), then $q^{(m)}(t^*) = q^{(m+1)}(t^*) = 0$. That is, no xs will ever again be assigned to t^* (due to the $q^{(n-1)}(t)$ factor in $q^{(n-1)}(t \mid x)$). In other words, the number of ts "in use" can only decrease during the iterative algorithm (or remain constant). Thus, it seems plausible that the solution will not depend on the cardinality of T, provided it is chosen to be large enough.

Algorithm 2.1 - The information bottleneck (IB) method.

Given $p(x, y)$, $\beta \geq 0$
Initialize $q^{(0)}(t \mid x)$
$q^{(0)}(t) = \sum_x p(x)\, q^{(0)}(t \mid x)$
$q^{(0)}(y \mid t) = \frac{1}{q^{(0)}(t)} \sum_x p(x, y)\, q^{(0)}(t \mid x)$
$n = 0$
while not converged **do**
  $n = n + 1$
  $q^{(n)}(t \mid x) = \frac{q^{(n-1)}(t)}{Z(x,\beta)} \exp\big[-\beta D_{\mathrm{KL}}\big[p(y \mid x) \mid q^{(n-1)}(y \mid t)\big]\big]$
  $q^{(n)}(t) = \sum_x p(x)\, q^{(n)}(t \mid x)$
  $q^{(n)}(y \mid t) = \frac{1}{q^{(n)}(t)} \sum_x q^{(n)}(t \mid x)\, p(x, y)$
end while

In summary, given the joint distribution $p(x, y)$, the IB method extracts a compressive encoder $q(t \mid x)$ that selectively maintains the bits from X that are informative about Y. As the encoder is a function of the free parameter β, we can visualize the entire family of solutions on a curve (figure 2.1), showing the tradeoff between compression (on the x-axis) and relevance (on the y-axis), with β as an implicitly varying parameter. For small β, compression is more important than prediction and we find ourselves at the bottom left of the curve in the high compression, low prediction regime. As β increases, prediction becomes more important relative to compression, and we see that both $I(X;T)$ and $I(T;Y)$ increase. At some point, $I(T;Y)$ saturates, because there is no more information about Y that can be extracted from X (either because $I(T;Y)$ has reached $I(X;Y)$ or because T has too small cardinality). In this regime, the encoder will approach the trivially deterministic solution of mapping each x to its own cluster. At any point on the curve, the slope is equal to β^{-1}, which we can read off directly from the cost functional. Note that the region below the curve is shaded because this area is feasible; for suboptimal $q(t \mid x)$, solutions will lie in this region. Optimal solutions will of course lie on the curve, and no solutions will lie above the curve.

Additional work on the IB has highlighted its relationship with maximum likelihood on a multinomial mixture model (Slonim and Weiss 2002) and canonical correlation

analysis (Creutzig et al. 2009) (and therefore linear Gaussian models (Bach and Jordan 2006) and slow feature analysis (Turner and Sahani 2007)). Applications have included speech recognition (Hecht and Tishby 2005, Hecht et al. 2009), topic modeling (Slonim and Tishby 2000b, 2001, Bekkerman et al. 2001, 2003), and neural coding (Schneidman et al. 2001, Palmer et al. 2015). Most recently, the IB has even been proposed as a method for benchmarking the performance of deep neural networks (Tishby and Zaslavsky 2015).

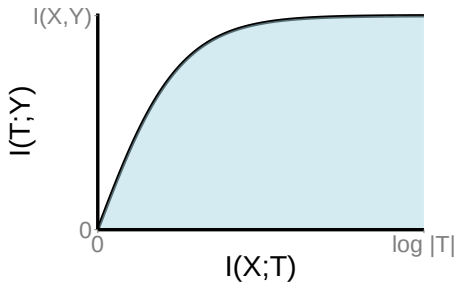

Figure 2.1: **An illustrative IB curve.** $I(T;Y)$ is the relevance term from eqn 2.1; $I(X;T)$ is the compression term. $I(X;Y)$ is an upper bound on $I(T;Y)$ since T only gets its information about Y via X. $\log(|T|)$, where $|T|$ is the cardinality of the compression variable, is a bound on $I(X;T)$ since $I(X;T) = H(T) - H(T \mid X) \leq H(T) \leq \log(|T|)$.

2.3 The deterministic information bottleneck

Our motivation for introducing an alternative formulation of the information bottleneck is rooted in the "compression term" of the IB cost function; there, the minimization of the mutual information $I(X;T)$ represents compression. As discussed above, this measure of compression comes from the channel coding literature and implies a restriction on the *communication cost* between X and T. Here, we are interested in the source coding notion of compression, which implies a restriction on the *representational cost* of T. For example, in neuroscience, there is a long history of work on "redundancy reduction" in the brain in the form of minimizing $H(T)$ (Barlow 1981, 2001a,b).

Let us call the original IB cost function L_{IB}, that is $L_{\mathrm{IB}} \equiv I(X;T) - \beta I(T;Y)$. We now introduce the deterministic information bottleneck (DIB) cost function:

$$L_{\mathrm{DIB}}[q(t|x)] \equiv H(T) - \beta I(T;Y)\,, \tag{2.7}$$

which is to be minimized over $q(t \mid x)$ and subject to the same Markov constraint as the original formulation (eqn 2.1). The motivation for the "deterministic" in its name will become clear in a moment.

To see the distinction between the two cost functions, note that:

$$L_{\mathrm{IB}} - L_{\mathrm{DIB}} = I(X;T) - H(T) \tag{2.8}$$

$$= -H(T \mid X)\,, \tag{2.9}$$

where we have used the decomposition of the mutual information $I(X;T) = H(T) - H(T \mid X)$. $H(T \mid X)$ is sometimes called the "noise entropy" and measures the stochasticity in the mapping from X to T. Since we are minimizing these cost functions, this means that the IB cost function *encourages* stochasticity in the encoding distribution $q(t \mid x)$ relative to the DIB cost function. In fact, we will see that by removing this encouragement of stochasticity, the DIB problem actually produces a deterministic encoding distribution, i.e. an encoding *function*, hence the "deterministic" in its name.

Naively taking the same variational calculus approach as for the IB problem, one cannot solve the DIB problem.[6] To make this problem tractable, we are going to define a family of cost functions of which the IB and DIB cost functions are limiting

[6]When you take the variational derivative of L_{DIB} + Lagrange multiplier term with respect to $q(t \mid x)$ and set it to zero, you get no explicit $q(t \mid x)$ term, and it is therefore not obvious how to solve these equations. We cannot rule that that approach is possible, of course; we have just here taken a different route.

cases. That family, indexed by α, is defined as:[7]

$$L_\alpha \equiv H(T) - \alpha H(T \mid X) - \beta I(T;Y)\,. \tag{2.10}$$

Clearly, $L_{\text{IB}} = L_1$. However, instead of looking at L_{DIB} as the $\alpha = 0$ case, we'll define the DIB solution $q_{\text{DIB}}(t \mid x)$ as the $\alpha \to 0$ limit of the solution to the generalized problem $q_\alpha(t \mid x)$:[8]

$$q_{\text{DIB}}(t \mid x) \equiv \lim_{\alpha \to 0} q_\alpha(t \mid x)\,. \tag{2.11}$$

Taking the variational calculus approach to minimizing L_α (under the Markov constraint), we get the following solution for the encoding distribution (see Section 2.A for the derivation and explicit form of the normalization factor $Z(x, a, \beta)$):

$$q_\alpha(t|x) = \frac{1}{Z(x,\alpha,\beta)} \exp\left[\frac{1}{\alpha}\left(\log q_\alpha(t) - \beta D_{\text{KL}}[p(y|x) \mid q_\alpha(y|t)]\right)\right] \tag{2.12}$$

$$q_\alpha(y|t) = \frac{1}{q_\alpha(t)} \sum_x p(y|x)\, q_\alpha(t|x)\, p(x)\,. \tag{2.13}$$

Note that the last equation is just eqn 2.3, since this just follows from the Markov constraint. With $\alpha = 1$, we can see that the first equation just becomes the IB solution from eqn 2.2, as should be the case.

[7]Note that for $\alpha < 1$, we cannot allow T to be continuous since $H(T)$ can become infinitely negative, and the optimal solution in that case will trivially be a delta function over a single value of T for all X, across all values of β. This is in constrast to the IB, which can handle continuous T. In any case, we continue to assume discrete X, Y, and T for convenience.

[8]Note a subtlety here that we cannot claim that the q_{DIB} is the solution to L_{DIB}, for although $L_{\text{DIB}} = \lim_{\alpha\to 0} L_\alpha$ and $q_{\text{DIB}} = \lim_{\alpha\to 0} q_\alpha$, the solution of the limit need not be equal to the limit of the solution. It would, however, be surprising if that were not the case.

Before we take the $\alpha \to 0$ limit, note that we can now write a generalized IB iterative algorithm (indexed by α) which includes the original as a special case ($\alpha = 1$):

$$q_\alpha^{(n)}(t|x) = \frac{1}{Z(x,\alpha,\beta)} \exp\left[\frac{1}{\alpha}\left(\log q_\alpha^{(n-1)}(t) - \beta D_{\mathrm{KL}}\left[p(y|x) \mid q_\alpha^{(n-1)}(y|t)\right]\right)\right] \tag{2.14}$$

$$q_\alpha^{(n)}(t) = \sum_x p(x)\, q_\alpha^{(n)}(t|x) \tag{2.15}$$

$$q_\alpha^{(n)}(y|t) = \frac{1}{q_\alpha^{(n)}(t)} \sum_x q_\alpha^{(n)}(t|x)\, p(x,y)\,. \tag{2.16}$$

This generalized algorithm can be used in its own right, however we will not discuss that option further here.

For now, we take the limit $\alpha \to 0$ and see that something interesting happens with $q_\alpha(t \mid x)$ - the argument of the exponential begins to blow up. For a fixed x, this means that $q(t \mid x)$ will collapse into a delta function at the value of t which maximizes $\log q(t) - \beta D_{\mathrm{KL}}[p(y \mid x) \mid q(y \mid t)]$. That is:

$$\lim_{\alpha \to 0} q_\alpha(t|x) = f : X \to T, \tag{2.17}$$

where:

$$f(x) = t^* = \operatorname*{argmax}_t \left(\log q(t) - \beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]\right). \tag{2.18}$$

So, as anticipated, the solution to the DIB problem is a deterministic encoding distribution. The $\log q(t)$ above encourages that we use as few values of t as possible, via a "rich-get-richer" scheme that assigns each x preferentially to a t already with many xs assigned to it. The KL divergence term, as in the original IB problem, just makes sure we pick ts which retain as much information from x about y as possible. The parameter β, as in the original problem, controls the tradeoff between how much we value compression and prediction.

Also like in the original problem, the solution above is only a formal solution, since eqn 2.12 depends on eqn 2.13 and vice versa. So we will again need to take an iterative

approach; in analogy to the IB case, we repeat the following updates to convergence (from some initialization):[9]

$$f^{(n)}(x) = \operatorname*{argmax}_t \big(\log q^{(n-1)}(t) - \beta D_{\mathrm{KL}}\big[p(y|x) \mid q^{(n-1)}(y|t)\big]\big) \tag{2.19}$$

$$q^{(n)}(t|x) = \delta\big(t - f^{(n)}(x)\big) \tag{2.20}$$

$$q^{(n)}(t) = \sum_x q^{(n)}(t|x)\, p(x) \tag{2.21}$$

$$= \sum_{x:f^{(n)}(x)=t} p(x) \tag{2.22}$$

$$q^{(n)}(y|t) = \frac{1}{q^{(n)}(t)} \sum_x q^{(n)}(t \mid x)\, p(x,y) \tag{2.23}$$

$$= \frac{\sum_{x:f^{(n)}(x)=t} p(x,y)}{\sum_{x:f^{(n)}(x)=t} p(x)}. \tag{2.24}$$

This process is summarized in Algorithm 2.2.

Note that the DIB algorithm also corresponds to "clamping" IB at every step by assigning each x to its highest probability cluster t. We can see this by taking the argmax of the logarithm of $q(t \mid x)$ in eqn 2.2, noting that the argmax of a positive function is equivalent the argmax of its logarithm, discarding the $\log(Z(x,\beta))$ term since it doesn't depend on t, and seeing that the result corresponds to eqn 2.18. We emphasize, however, that this is not the same as simply running the IB algorithm to convergence and then clamping the resulting encoder; that would, in most cases, produce a suboptimal, "unconverged" deterministic solution.

Like with the IB, the DIB solutions can be plotted as a function of β. However, in this case, it is more natural to plot $I(T;Y)$ as a function of $H(T)$, rather than $I(X;T)$. That said, in order to compare the IB and DIB, they need to be plotted in

[9] As with the IB, the DIB has the property that once a cluster goes unused, it will not be brought back into use in future steps. That is, if $q^{(m)}(t) = 0$, then $\log q^{(m)}(t) = -\infty$ and $q^{(m+1)}(t \mid x) = 0\ \forall x$. So once again, one should conservatively choose the cardinality of T to be "large enough"; for both the IB and DIB, we chose to set it equal to the cardinality of X.

the same plane. When plotting in the $I(X;T)$ plane, the DIB curve will of course lie below the IB curve, since in this plane, the IB curve is optimal; the opposite will be true when plotting in the $H(T)$ plane. Comparisons with experimental data can be performed in either plane.

Algorithm 2.2 - The deterministic information bottleneck (DIB) method.

Given $p(x, y)$, $\beta \geq 0$
Initialize $f^{(0)}(x)$
Set $q^{(0)}(t) = \sum_{x:f^{(0)}(x)=t} p(x)$
Set $q^{(0)}(y \mid t) = \frac{\sum_{x:f^{(0)}(x)=t} p(x,y)}{\sum_{x:f^{(0)}(x)=t} p(x)}$
$n = 0$
while not converged **do**
  $n = n + 1$
  $d^{(n-1)}(x, t) \equiv D_{\mathrm{KL}}\big[p(y \mid x) \mid q^{(n-1)}(y \mid t)\big]$
  $\ell_\beta^{(n-1)}(x, t) \equiv \log q(t) - \beta d^{(n-1)}(x, t)$
  $f^{(n)}(x) = \operatorname{argmax}_t \ell_\beta^{(n-1)}(x, t)$
  $q^{(n)}(t) = \sum_{x:f^{(n)}(x)=t} p(x)$
  $q^{(n)}(y \mid t) = \frac{\sum_{x:f^{(n)}(x)=t} p(x,y)}{\sum_{x:f^{(n)}(x)=t} p(x)}$
end while

2.4 Comparison of IB and DIB

To get an idea of how the IB and DIB solutions differ in practice, we generated a series of random joint distributions $p(x, y)$, solved for the IB and DIB solutions for each, and compared them in both the IB and DIB plane. To generate the $p(x, y)$, we first sampled $p(x)$ from a symmetric Dirichlet distribution with concentration parameter α_x (so $p(x) \sim \mathrm{Dir}[\alpha_x]$), and then sampled each row of $p(y \mid x)$ from another symmetric Dirichlet distribution with concentration parameter $\alpha_y^{(i)}$ (so $p(y \mid x_i) \sim \mathrm{Dir}\left[\alpha_y^{(i)}\right]$). In the experiments shown here, we set α_x to 1000, so that each x_i was approximately equally likely, and we set $\alpha_y^{(i)}$ to be equally spaced logarithmically between $10^{-1.3}$ and $10^{1.3}$, in order to provide a range of informativeness in the conditionals. We set the cardinalities of X and Y to $|X| = 256$ and $|Y| = 32$, with $|X| > |Y|$ for two reasons.

First, this encourages overlap between the conditionals $p(y|x)$, which leads to a more interesting clustering problem. Second, in typical applications, this will be the case, such as in document clustering where there are often many more documents than vocabulary words. Since the number of clusters in use for both IB and DIB can only decrease from iteration to iteration (see footnote 9), we always initialized $|T| = |X|$.[10] For the DIB, we initialized the cluster assignments to be as even across the cluster as possible, i.e. each data points belonged to its own cluster. For IB, we initialized using a soft version of the same procedure, with 75% of each conditional's probability mass assigned to a unique cluster, and the remaining 25% a normalized uniform random vector over the remaining $|T| - 1$ clusters.

An illustrative pair of solutions is shown in figure 2.2. The key feature to note is that, while performance of the IB and DIB solutions are very similar in the IB plane, their behavior differs drastically in the DIB plane.

Perhaps most unintuitive is the behavior of the IB solution in the DIB plane, where from an entropy perspective, the IB actually "decompresses" the data (i.e. $H(T) > H(X)$). To understand this behavior, recall that the IB's compression term is the mutual information $I(X,T) = H(T) - H(T \mid X)$. This term is minimized by any $q(t \mid x)$ that maps ts independently of xs. Consider two extremes of such mappings. One is to map all values of X to a single value of T; this leads to $H(T) = H(T \mid X) = I(X,T) = 0$. The other is to map each value of X uniformly across all values of T; this leads to $H(T) = H(T \mid X) = \log |T|$ and $I(X,T) = 0$. In our initial studies, the IB consistently took the latter approach.[11] Since the DIB cost function favors the former approach (and indeed the DIB solution follows this approach), the IB consistently performs poorly by the DIB's standards. This difference

[10] An even more efficient setting might be to set the cardinality of T based on the entropy of X, say $|T| = \text{ceiling}(\exp(H(X)))$, but we didn't experiment with this.

[11] Intuitively, this approach is "more random" and is therefore easier to stumble upon during optimization.

is especially apparent at small β, where the compression term matters most, and as β increases, the DIB and IB solutions converge in the DIB plane.

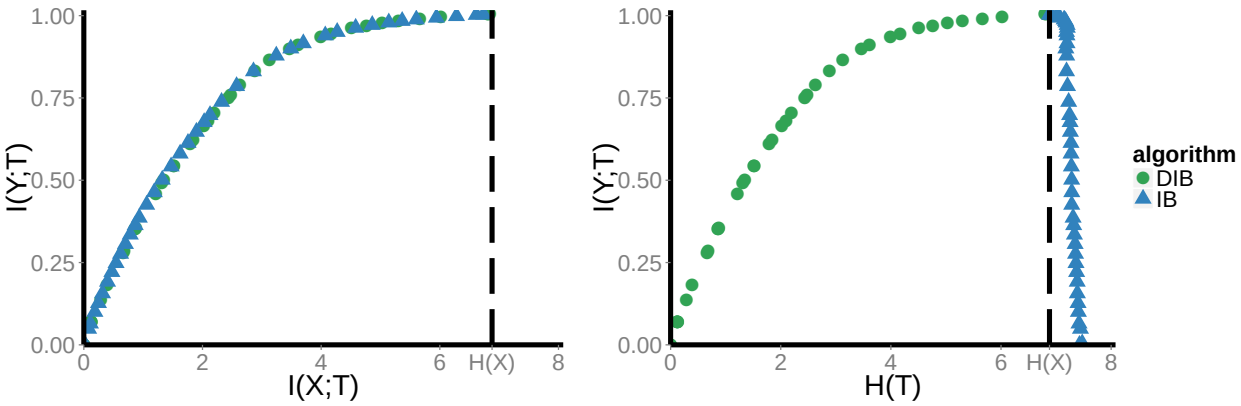

Figure 2.2: **Example IB and DIB solutions.** *Left*: IB plane. *Right*: DIB plane. Upper limit of the y-axes is $I(X,Y)$, since this is the maximal possible value of $I(T;Y)$. Upper limit of the x-axes is $\log(|T|)$, since this is the maximal possible value of $H(T)$ and $I(X,T)$ (the latter being true since $I(X,T)$ is bounded above by both $H(T)$ and $H(X)$, and $|T| < |X|$). The dashed vertical lines mark $H(X)$, which is both an upper bound for $I(X,T)$ and a natural comparison for $H(T)$ (since to place each data point in its own cluster, we need $H(T) = H(X)$).

To encourage the IB to perform closer to DIB optimality at small β, we next altered our initialization scheme of $q(t \mid x)$ to one biased towards single-cluster solutions; instead of each x_i having most of its probability mass on a unique cluster t_i, we placed most of the probability mass for each x_i on the *same* cluster t^*. The intended effect was to start the IB closer to solutions in which all data points were mapped to a single cluster. Results are shown in figure 2.3. Here, p_0 is the amount of probability mass placed on the cluster t^*, that is $q(t^* \mid x) = p_0, \ \forall x$; the probability mass for the remaining $|T| - 1$ clusters was again initialized to a normalized uniform random vector. "random" refers to an initialization which skips placing the p_0 mass and just initializes each $q(t \mid x_i)$ to a normalized uniform random vector.

We note several features. First, although we can see a gradual shift of the IB towards DIB-like behavior in the DIB plane as $p_0 \to 1$, the IB solutions never quite reach the performance of DIB. Moreover, as $p_0 \to 1$, the single-cluster initializations

exhibit a phase transition in which, regardless of β, they "skip" over a sizable fraction of lower-$I(Y;T)$ solutions that are picked up by DIB. Third, even for higher-$I(Y;T)$ solutions, the single-cluster initializations seem to perform suboptimally, not quite extracting all of the information $I(X;Y)$, as DIB and the multi-cluster initialization from the previous section do; this can be seen in both the IB and DIB plane.

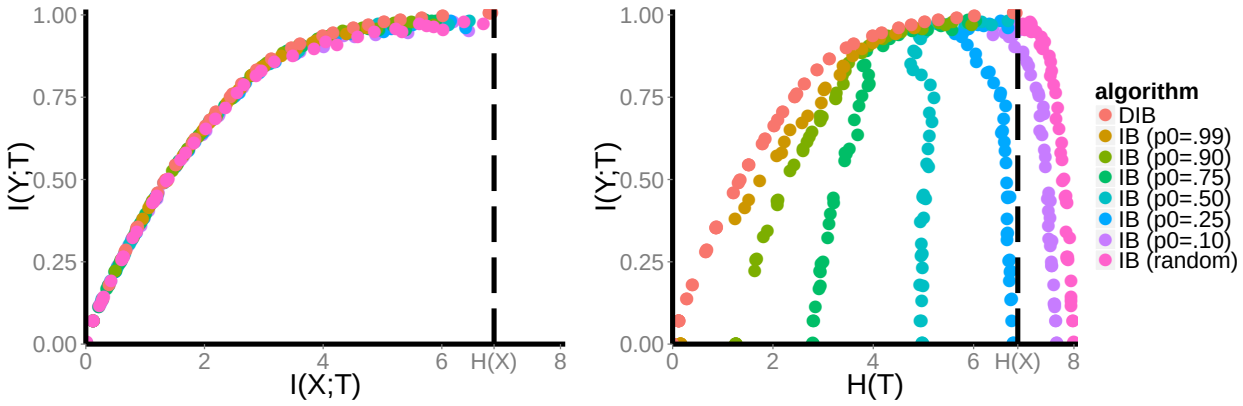

Figure 2.3: **Example IB and DIB solutions across different IB initializations.** Details identical to Figure 2.2, except colors represent different initializations for the IB, as described in the text.

To summarize, the IB and DIB perform similarly by the IB standards, but the DIB tends to outperform the IB dramatically by the DIB's standards. Careful initialization of the IB can make up some of the difference, but not all.

It is also worth noting that, across all the datasets we tested, the DIB also tended to converge faster, as illustrated in figure 2.4. The DIB speedup over IB varied depended on the convergence conditions. In our experiments, we defined convergence as when the relative step-to-step change in the cost functional L was smaller than some threshold ctol, that is when $\frac{L_{n-1}-L_n}{L_{n-1}} <$ ctol at step n. In the results above, we used ctol $= 10^{-3}$. In figure 2.4, we vary ctol, with the IB initialization scheme fixed to the original "multi-cluster" version, to show the effect on the relative speedup of DIB over IB. While DIB remained ≈2-5x faster than IB in all cases tested, that speedup tended to be more pronounced with lower ctol. Since the ideal convergence

conditions would probably vary by dataset size and complexity, it is difficult to make any general conclusions, though our experiments do at least suggest that DIB offers a computational advantage over IB.

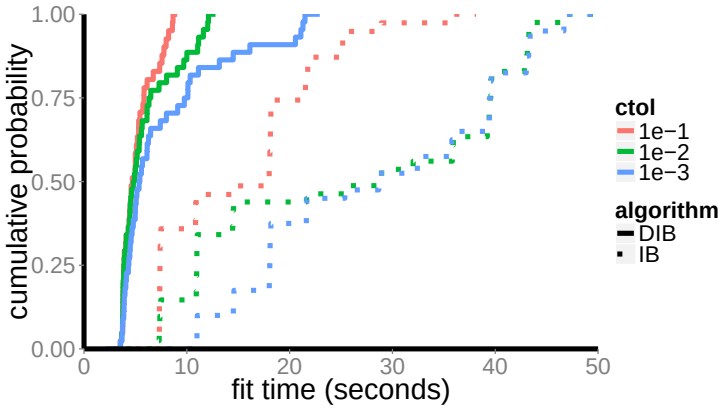

Figure 2.4: **Fit times for IB and DIB.** Cumulative distribution function of fit times across β, for a variety of settings of the convergence tolerance. Note that absolute numbers here depend on hardware, so we emphasize only relative comparisons of IB vs DIB. Note also that across the range of ctol values we tested here, the (D)IB curves vary by less than the width of the data points, and so we omit them.

2.5 Related work

The DIB is not the first hard clustering version of IB.[12] Indeed, the agglomerative information bottleneck (AIB) (Slonim and Tishby 2000a) also produces hard clustering and was introduced soon after the IB. Thus, it is important to distinguish between the two approaches. AIB is a bottom-up, greedy method which starts with all data points belonging to their own clusters and iteratively merges clusters in a way which maximizes the gain in relevant information. It was explicitly designed to produce a hard clustering. DIB is a top-down method derived from a cost function that was not designed to produce a hard clustering. Our starting point was to alter the IB cost function to match the source coding notion of compression. The emergence of hard clustering in DIB is itself a result. Thus, while AIB does provide a hard clustering

[12]In fact, even the IB itself produces a hard clustering in the large β limit. However, it trivially assigns all data points to their own clusters.

version of IB, DIB contributes the following in addition: 1) Our study emphasizes why a stochastic encoder is optimal for IB, namely due to the noise entropy term. 2) Our study provides a principled, top-down derivation of a hard clustering version of IB, based upon an intuitive change to the cost function. 3) Our non-trivial derivation also provides a cost function and solution which interpolates between DIB and IB, by adding back the noise entropy continuously, i.e. with $0 < \alpha < 1$. This interpolation may be viewed as adding a regularization term to DIB, one that may perhaps be useful in dealing with finitely sampled data. Another interpretation of the cost function with intermediate α is as a penalty on *both* the mutual information between X and T *and* the entropy of the compression, $H(T)$.

The original IB also provides a deterministic encoding upon taking the limit $\beta \to \infty$ that corresponds to the causal-state partition of histories (Still et al. 2010). However, this is the limit of no compression, whereas our approach allows for an entire family of deterministic encoders with varying degrees of compression.

2.6 Discussion

Here we have introduced the deterministic information bottleneck (DIB) as an alternative to the information bottleneck (IB) for compression and clustering. We have argued that the DIB cost function better embodies the goal of lossy compression of relevant information, and shown that it leads to a non-trivial deterministic version of the IB. We have compared the DIB and IB solutions on synthetic data and found that, in our experiments, the DIB performs nearly identically to the IB in terms of the IB cost function, but far superior in terms of its own cost function. We also noted that the DIB achieved this performance at a computational efficiency 2-5x better than the IB.

Of course, in addition to the studies with synthetic data here, it is important to compare the DIB and IB on real world datasets as well to see whether the DIB's apparent advantages hold, for example with datasets that have more explicit hierarchical structure for both algorthms to exploit, such as in topic modelling (Blei et al. 2004, Slonim and Weiss 2002).

One particular application of interest is maximally informative clustering, where it would be interesting to know how IB and DIB relate to classic clustering algorithms such as k-means (Strouse and Schwab 2018). Previous work has, for example, offered a principled way of choosing the number of clusters based on the finiteness of the data (Still and Bialek 2004), and similarly interesting results may exist for the DIB. More generally, there are learning theory results showing generalization bounds on IB for which an analog on DIB would be interesting as well (Shamir et al. 2010).

Another potential area of application is modeling the extraction of predictive information in the brain (which is one particular example in a long line of work on the exploitation of environmental statistics by the brain (Barlow 1981, 2001a,b, Atick and Redlich 1992, Olshausen and Field 1996, 1997, 2004, Simoncelli and Olshausen 2001)). There, X would be the stimulus at time t, Y the stimulus a short time in the future $t + \tau$, and T the activity of a population of sensory neurons. One could even consider neurons deeper in the brain by allowing X and Y to correspond not to an external stimulus, but to the activity of upstream neurons. An analysis of this nature using retinal data was recently performed with the IB (Palmer et al. 2015). It would be interesting to see if the same data corresponds better to the behavior of the DIB, particularly in the DIB plane where the IB and DIB differ dramatically.

We close by noting that DIB is an imperfect name for the algorithm introduced here for a couple of reasons. First, there do exist other deterministic limits and approximations to the IB (see, for example, the discussion of the AIB in section 2.5), and so we hesitate to use the phrase "the" deterministic IB. Second, our motivation

here was not to create a deterministic version of IB, but instead to alter the cost function in a way that better encapsulates the goals of certain problems in data analysis. Thus, the deterministic nature of the solution was a result, not a goal. For this reason, "entropic bottleneck" might also be an appropriate name.

Appendix

2.A Derivation of generalized IB solution

Given $p(x, y)$ and subject to the Markov constraint $T \leftrightarrow X \leftrightarrow Y$, the generalized IB problem is:

$$\min_{q(t|x)} L[q(t|x)] = H(T) - \alpha H(T|X) - \beta I(Y;T) - \sum_{x,t} \lambda(x)\, q(t|x)\,, \tag{2.25}$$

where we have now included the Lagrange multiplier term (which enforces normalization of $q(t|x)$) explicitly. The Markov constraint implies the following factorizations:

$$q(t|y) = \sum_x q(t|x)\, p(x|y) \tag{2.26}$$

$$q(t) = \sum_x q(t|x)\, p(x)\,, \tag{2.27}$$

which give us the following useful derivatives:

$$\frac{\delta q(t|y)}{\delta q(t|x)} = p(x|y) \tag{2.28}$$

$$\frac{\delta q(t)}{\delta q(t|x)} = p(x)\,. \tag{2.29}$$

Now taking the derivative of the cost function with respect to the encoding distribution, we get:

$$\frac{\delta L}{\delta q(t|x)} = -\frac{\delta}{\delta q(t|x)} \sum_t q(t) \log q(t) - \frac{\delta}{\delta q(t|x)} \sum_{x,t} \lambda(x)\, q(t|x) \tag{2.30}$$

$$+ \alpha \frac{\delta}{\delta q(t|x)} \sum_{x,t} q(t|x)\, p(x) \log q(t|x) \tag{2.31}$$

$$- \beta \frac{\delta}{\delta q(t|x)} \sum_{y,t} q(t|y)\, p(y) \log \left[\frac{q(t|y)}{q(t)} \right] \tag{2.32}$$

$$= -\log q(t)\, \frac{\delta q(t)}{\delta q(t|x)} - q(t)\, \frac{\delta \log q(t)}{\delta q(t|x)} - \lambda(x)\, \frac{\delta q(t|x)}{\delta q(t|x)} \tag{2.33}$$

$$+ \alpha \left[p(x) \log q(t|x)\, \frac{\delta q(t|x)}{\delta q(t|x)} + q(t|x)\, p(x)\, \frac{\delta \log q(t|x)}{\delta q(t|x)} \right] \tag{2.34}$$

$$- \beta \sum_y \left[p(y) \log \left[\frac{q(t|y)}{q(t)} \right] \frac{\delta q(t|y)}{\delta q(t|x)} \right] \tag{2.35}$$

$$+ \beta \sum_y \left[q(t|y)\, p(y)\, \frac{\delta \log q(t|y)}{\delta q(t|x)} + q(t|y)\, p(y)\, \frac{\delta \log q(t)}{\delta q(t|x)} \right] \tag{2.36}$$

$$= -p(x) \log q(t) - p(x) - \lambda(x) + \alpha \left[p(x) \log q(t|x) + p(x) \right] \tag{2.37}$$

$$- \beta \sum_y \left[p(y) \log \left[\frac{q(t|y)}{q(t)} \right] p(x|y) + p(y)\, p(x|y) - q(t|y)\, p(y)\, \frac{p(x)}{q(t)} \right] \tag{2.38}$$

$$= -p(x) \log q(t) - p(x) - \lambda(x) + \alpha \left[p(x) \log q(t|x) + p(x) \right] \tag{2.39}$$

$$- \beta p(x) \left[\sum_y p(y|x) \log \left[\frac{q(t|y)}{q(t)} \right] + \sum_y p(y|x) - \sum_y q(y|t) \right] \tag{2.40}$$

$$= p(x) \left[-1 - \log q(t) - \frac{\lambda(x)}{p(x)} + \alpha \log q(t|x) + \alpha - \beta \left[\sum_y p(y|x) \log \left[\frac{q(t|y)}{q(t)} \right] \right] \right]. \tag{2.41}$$

Setting this to zero implies that:

$$\alpha \log q(t|x) = 1 - \alpha + \log q(t) + \frac{\lambda(x)}{p(x)} + \beta \left[\sum_y p(y|x) \log \left[\frac{q(t|y)}{q(t)} \right] \right]. \tag{2.42}$$

We want to rewrite the β term as a KL divergence. First, we will need that $\log\left[\frac{q(t|y)}{q(t)}\right] = \log\left[\frac{q(t,y)}{q(t)p(y)}\right] = \log\left[\frac{q(y|t)}{p(y)}\right]$. Second, we will add and subtract $\beta \sum_y p(y|x) \log\left[\frac{p(y|x)}{p(y)}\right]$. This gives us:

$$\alpha \log q(t|x) = 1 - \alpha + \log q(t) + \frac{\lambda(x)}{p(x)} + \beta \sum_y p(y|x) \log\left[\frac{p(y|x)}{p(y)}\right] \tag{2.43}$$

$$- \beta \left[\sum_y p(y|x) \log\left[\frac{p(y|x)}{q(y|t)}\right]\right]. \tag{2.44}$$

The second β term is now just $D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]$. Dividing both sides by α, this leaves us with the equation:

$$\log q(t|x) = z(x, \alpha, \beta) + \frac{1}{\alpha} \log q(t) - \frac{\beta}{\alpha} D_{\mathrm{KL}}[p(y \mid x) \,|q(y|t)]\,, \tag{2.45}$$

where we have absorbed all of the terms that don't depend on t into a single factor:

$$z(x, \alpha, \beta) \equiv \frac{1}{\alpha} - 1 + \frac{\lambda(x)}{\alpha p(x)} + \frac{\beta}{\alpha} \sum_y p(y \mid x) \log\left[\frac{p(y \mid x)}{p(y)}\right]. \tag{2.46}$$

Solving for $q(t|x)$, we get:

$$q(t|x) = \exp[z] \exp\left[\frac{1}{\alpha} \left(\log q(t) - \beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]\right)\right] \tag{2.47}$$

$$= \frac{1}{Z} \exp\left[\frac{1}{\alpha} \left(\log q(t) - \beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]\right)\right], \tag{2.48}$$

where:

$$Z(x, \alpha, \beta) \equiv \exp[-z] \tag{2.49}$$

is just a normalization factor. Now that we're done with the general derivation, let's add a subscript to the solution to distinguish it from the special cases of the IB

and DIB.

$$q_\alpha(t|x) = \frac{1}{Z(x,\alpha,\beta)} \exp\left[\frac{1}{\alpha}\left(\log q(t) - \beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]\right)\right]. \tag{2.50}$$

The IB solution is then:

$$q_{\mathrm{IB}}(t|x) = q_{\alpha=1}(t|x) = \frac{q(t)}{Z} \exp[-\beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]]\,, \tag{2.51}$$

while the DIB solution is:

$$q_{\mathrm{DIB}}(t|x) = \lim_{\alpha\to 0} q_\alpha(t|x) = \delta(t - t^*(x))\,, \tag{2.52}$$

with:

$$t^*(x) = \operatorname*{argmax}_t \left(\log q(t) - \beta D_{\mathrm{KL}}[p(y|x) \mid q(y|t)]\right). \tag{2.53}$$

Chapter 3

The information bottleneck and geometric clustering

In the previous chapter, we introduced the deterministic information bottleneck (DIB). Compared to the original information bottleneck (IB), it 1) is restricted to discrete encodings T, 2) prunes the number of dimensions of T used automatically, and 3) produces hard assignments of inputs to the encodings. These three features make *clustering* a natural application. In clustering problems, we are given some data points $\{\mathbf{x}_i\}_{i=1}^N$ and we wish to find "structure" in the form of n_c groups, or "clusters", that are more similar to one another than to members of other groups. The problems are both to choose the number of groups n_c as well as the assignments of data points to clusters.

However, phrasing clustering in the language of (D)IB is non-trivial. First, we introduced DIB and IB as approaches to *supervised learning*, and yet clustering is a problem of *unsupervised learning*. If we are only given the data points $\{\mathbf{x}_i\}_{i=1}^N$, what should, in the language of (D)IB, X and Y be? Second, the data points $\{\mathbf{x}_i\}_{i=1}^N$ often live in a space endowed with some sort of geometry in which there is naturally a notion of distance. How should that be incorporated into the picture?

We became interested in this question after writing the previous chapter, and quickly discovered a paper claiming that IB was equivalent (in certain limits) to a classic clustering algorithm called k-means (Still et al. 2003). This introduced several questions to us. First, if IB is equivalent to k-means, what is DIB equivalent to? Second, how could a stochastic algorithm like IB be equivalent to a deterministic one like k-means? Investigating these two questions led us to find several errors in Still et al. (2003) which ultimately nullified the claim of an IB / k-means equivalence. However, in understanding these errors, we were able to understand how to fix them and build on much of what was correct about their setup to properly phrase geometric clustering in the language of information theory.

This chapter has been accepted for publication as Strouse and Schwab (2018).

Abstract

The information bottleneck (IB) approach to clustering takes a joint distribution $P(X, Y)$ and maps the data X to cluster labels T which retain maximal information about Y (Tishby et al. 1999). This objective results in an algorithm that clusters data points based upon the similarity of their conditional distributions $P(Y \mid X)$. This is in contrast to classic "geometric clustering" algorithms such as k-means and gaussian mixture models (GMMs) which take a set of observed data points $\{\mathbf{x}_i\}_{i=1:N}$ and cluster them based upon their geometric (typically Euclidean) distance from one another. Here, we show how to use the deterministic information bottleneck (DIB) (Strouse and Schwab 2017), a variant of IB, to perform geometric clustering, by choosing cluster labels that preserve information about data point location on a smoothed dataset. We also introduce a novel intuitive method to choose the number of clusters, via kinks in the information curve. We apply this approach to a variety of simple clustering problems, showing that DIB with our model selection procedure recovers the generative cluster labels. We also show that, for one simple case, DIB interpolates between the cluster boundaries of GMMs and

k-means in the large data limit. Thus, our IB approach to clustering also provides an information-theoretic perspective on these classic algorithms.

3.1 Introduction

Unsupervised learning is a crucial component of building intelligent systems (LeCun 2016), since such systems need to be able to leverage experience to improve performance even in the absence of feedback. One aspect of doing so is discovering discrete structure in data, a problem known as clustering (MacKay 2002). In the typical setup, one is handed a set of data points $\{\mathbf{x}_i\}_{i=1}^N$, and asked to return a mapping from data point label i to a finite set of cluster labels c. The most basic approaches include k-means and gaussian mixture models (GMMs). GMMs cluster data based on maximum likelihood fitting of a probabilistic generative model. k-means can either be thought of as directly clustering data based on geometric (often Euclidean) distances between data points, or as a special case of GMMs with the assumptions of evenly sampled, symmetric, equal variance components.

The information bottleneck (IB) is an information-theoretic approach to clustering data X that optimizes cluster labels T to preserve information about a third "target variable" of interest Y. The resulting (soft) clustering groups data points based on the similarity in their conditional distributions over the target variable through the KL divergence, $\mathrm{KL}[p(y \mid x_i) \mid p(y \mid x_j)]$. An IB clustering problem is fully specified by the joint distribution $P(X, Y)$ and the tradeoff parameter β quantifying the relative preference for fewer clusters and more informative ones.

At first glance, it is not obvious how to use this approach to cluster geometric data, where the input is a set of data point locations $\{\mathbf{x}_i\}_{i=1}^N$. For example, what is the target variable Y that our clusters should retain information about? What should $P(X, Y)$ be? And how should one choose the tradeoff parameter β?

Still et al. (2003) were the first to attempt to do geometric clustering with IB, and claimed an equivalence (in the large data limit) between IB and k-means. Unfortunately, while much of their approach is correct, it contained some fundamental errors that nullify the main results. In the next section, we describe those errors and how to correct them. Essentially, their approach did not properly translate geometric information into a form that could be used correctly by an information-theoretic algorithm.

In addition to fixing this issue, we also choose to use a recently introduced variant of the information bottleneck called the deterministic information bottleneck (DIB) (Strouse and Schwab 2017). We make this choice due to the different way in which IB and DIB use the number of clusters provided to them. IB is known to use all of the clusters it has access to, and thus clustering with IB requires a search both over the number of clusters N_c as well as the the parsimony-informativeness tradeoff parameter β (Slonim et al. 2005). DIB on the other hand has a built-in preference for using as few clusters as it can, and thus only requires a parameter search over β. Moreover, DIB's ability to select the number of clusters to use for a given β leads to a intuitive model selection heuristic based on the robustness of a clustering result across β that we show can recover the generative number of clusters in many cases.

In the next section, we more formally define the geometric clustering problem, the IB approach of Still et al. (2003), and our own DIB approach. In section 3.3, we show that our DIB approach to geometric clustering behaves intuitively and is able to recover the generative number of clusters with only a single free parameter (the data smoothing scale s). In section 3.4, we discuss the relationship between our approach and GMMs and k-means, proving that at least in one simple case, DIB interpolates between GMM and k-means cluster boundaries by varying the data smoothing scale s. Our approach thus provides a novel information-theoretic approach to geometric clustering, as well as an information-theoretic perspective on these classic clustering methods.

3.2 Geometric clustering with the (deterministic) information bottleneck

In a geometric clustering problem, we are given a set of N observed data points $\{\mathbf{x}_i\}_{i=1:N}$ and asked to provide a weighting $q(c \mid i)$ that categorizes data points into (possibly multiple) clusters such that data points "near" one another are in the same cluster. The definition of "near" varies by algorithm: for k-means, for example, points in a cluster are closer to their own cluster mean than to any other cluster mean.

In an information bottleneck (IB) problem, we are given a joint distribution $P(X, Y)$ and asked to provide a mapping $q(t \mid x)$ such that T contains the "relevant" information in X for predicting Y. This goal is embodied by the information-theoretic optimization problem:

$$q^*_{\mathrm{IB}}(t \mid x) = \underset{q(t|x)}{\operatorname{argmin}}\ I(X, T) - \beta I(T, Y)\,, \tag{3.1}$$

subject to the Markov constraint $T \leftrightarrow X \leftrightarrow Y$. β is a free parameter that allows for setting the desired balance between the compression encouraged by the first term and the relevance encouraged by the second; at small values, we throw away most of X in favor of a succint representation for T, while for large values of β, we retain nearly all the information that X has about Y.

This approach of squeezing information through a latent variable bottleneck might remind some readers of a variational autoencoder (VAE) (Kingma and Welling 2013), and indeed IB has a close relationship with VAEs. As pointed out in (Alemi et al. 2016), a variational version of IB can essentially be seen as the supervised generalization of a VAE, which is typically an unsupervised algorithm.

We are interested in performing geometric clustering with the information bottleneck. For the purposes of this paper, we will focus on a recent alternative formulation of the IB, called the deterministic information bottleneck (DIB) (Strouse and Schwab 2017). We do this because the DIB's cost function more directly encourages the use of as few clusters as possible, so initialized with $n_c^{\max}$ clusters, it will typically converge to a solution with far fewer. Thus, it has a form of model selection built in that will prove useful for geometric clustering (Strouse and Schwab 2017). IB, on the other hand, will tend to use all $n_c^{\max}$ clusters, and thus requires an additional search over this parameter (Slonim et al. 2005). DIB also differs from IB in that it leads to a hard clustering instead of a soft clustering.

Formally, the DIB setup is identical to that of IB except that the mutual information term $I(X;T)$ in the cost functional is replaced with the entropy $H(T)$:

$$q^*_{\mathrm{DIB}}(t \mid x) = \underset{q(t|x)}{\operatorname{argmin}} \; H(T) - \beta I(T,Y)\,. \tag{3.2}$$

This change to the cost functional leads to a hard clustering with the form (Strouse and Schwab 2017):

$$q^*_{\mathrm{DIB}}(t \mid x) = \delta(t - t^*(x)) \tag{3.3}$$

$$t^* = \underset{t}{\operatorname{argmax}} \; \log q(t) - \beta d(x,t) \tag{3.4}$$

$$d(x,t) \equiv \mathrm{KL}[p(y \mid x) \mid q(y \mid t)] \tag{3.5}$$

$$q(t) = \sum_x q(t \mid x)\, p(x) \tag{3.6}$$

$$q(y \mid t) = \frac{1}{q(t)} \sum_x q(t \mid x)\, p(x)\, p(y \mid x)\,, \tag{3.7}$$

where the above equations are to be iterated to convergence from some initialization. The IB solution (Tishby et al. 1999) simply replaces the first two equations with:

$$q^*_{\mathrm{IB}}(t \mid x) = \frac{q(t)}{Z(x,\beta)} e^{-\beta d(x,t)}, \tag{3.8}$$

which can be seen as replacing the argmax in DIB with an exponential and a soft max.

The (D)IB is referred to as a "distributional clustering" algorithm (Slonim and Tishby 2001) due to the KL divergence term $d\,(x,t) = \mathrm{KL}[p(y \mid x) \mid q(y \mid t)]$, which can be seen as measuring how similar the data point conditional distribution $p(y \mid x)$ is to the cluster conditional, or mixture of data point conditionals, $q(y \mid t) = \sum_x q(x \mid t)\, p(y \mid x)$. That is, a candidate point $x^{'}$ will be assigned to a cluster based upon how similar its conditional $p\big(y \mid x^{'}\big)$ is to the conditionals $p(y \mid x)$ for the data points x that make up that cluster. Thus, both DIB and IB cluster data points based upon the conditionals $p(y \mid x)$.

To apply (D)IB to a geometric clustering problem, we must choose how to map the geometric clustering dataset $\{\mathbf{x}_i\}_{i=1:N}$ to an appropriate IB dataset $P(X,Y)$. First, what should X and Y be? Since X is the data being clustered by IB, we'll choose that to be the *data point index* i. As for the target variable Y that we wish to maintain information about, it seems reasonable to choose the *data point location* $\mathbf{x}$ (though we will discuss alternative choices later). Thus, we want to cluster data indices i into cluster indices c in a way that maintains as much possible info about the location $\mathbf{x}$ as possible (Still et al. 2003).

Now, how should we choose the joint distribution $p(i,\mathbf{x}) = p(\mathbf{x} \mid i)\, p(i)$? At first glance, one might choose $p(\mathbf{x} \mid i) = \delta_{\mathbf{x}\mathbf{x}_i}$, since data point i was observed at location $\mathbf{x}_i$. The reason *not* to do this lies with the fact that (D)IB is a distributional clustering

algorithm, as discussed two paragraphs above. Data points are compared to one another through their conditionals $p(\mathbf{x} \mid i)$, and with the choice of a delta function, there will be no overlap unless two data points are on top of one another. That is, choosing $p(\mathbf{x} \mid i) = \delta_{\mathbf{x}\mathbf{x}_i}$ leads to a KL divergence that is either infinite for data points at different locations, or zero for data points that lie exactly on top of one another, i.e. $\mathrm{KL}[p(\mathbf{x} \mid i) \mid p(\mathbf{x} \mid j)] = \delta_{\mathbf{x}_i\mathbf{x}_j}$. Trivially, the resulting clustering would assign each data point to its own cluster, grouping only data points that are identical. Put another way, all relational information in an IB problem lies in the joint distribution $P(X, Y)$. If one wants to perform geometric clustering with an IB approach, then geometric information must somehow be injected into that joint distribution, and a series of delta functions does not do that. A previous attempt at linking IB and k-means made this mistake (Still et al. 2003). Subsequent algebraic errors were tantamount to incorrectly introducting geometric information into IB, precisely in the way that such geometric information appears in k-means, and resulting in an algorithm that is not IB. We describe these errors in more detail in an appendix (section 3.A).

Based on the problems identified with using delta functions, a better choice for the conditionals is something spatially extended, such as:

$$p(\mathbf{x} \mid i) \propto \exp\left[-\frac{1}{2s^2} d(\mathbf{x}, \mathbf{x}_i)\right], \tag{3.9}$$

where s sets the geometric scale or units of distance, and d is a distance metric, such as the Euclidean distance $d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|^2$. If we indeed use the Euclidean distance, then $p(\mathbf{x} \mid i)$ will be (symmetric) gaussian (with variance s^2), and this corresponds to gaussian smoothing our data. In any case, the obvious choice for the marginal is $p(i) = \frac{1}{N}$, where N is the number of data points, unless one has a reason a priori to favor certain data points over others. These choices for $p(i)$ and $p(\mathbf{x} \mid i)$ determine completely our dataset $p(i, \mathbf{x}) = p(\mathbf{x} \mid i)\, p(i)$. Figure 3.1 contains

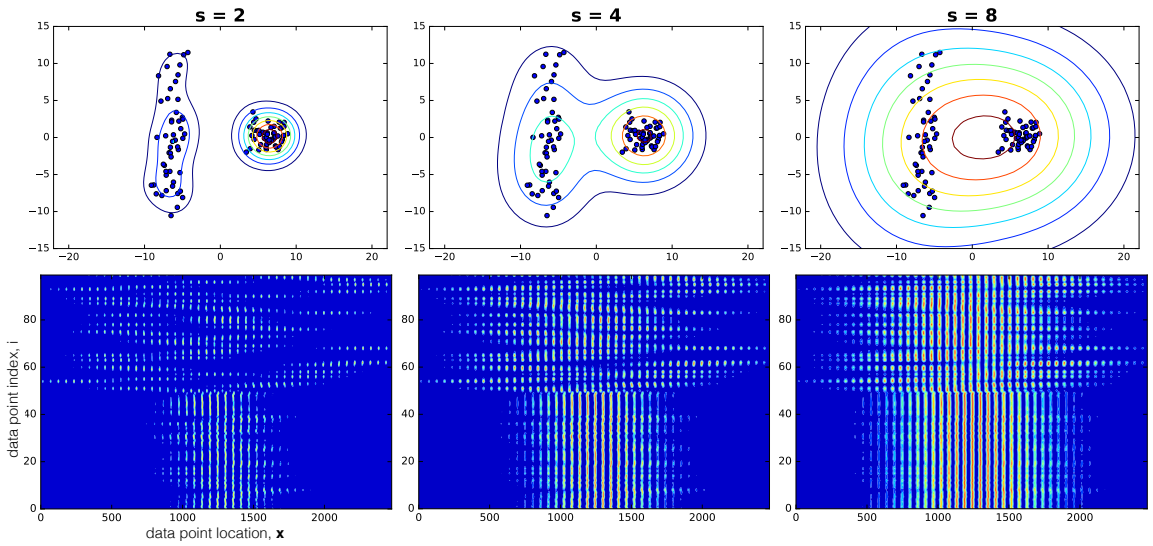

Figure 3.1: **Illustration of data smoothing procedure.** Example dataset with one symmetric and one skew cluster. *Top row*: scatterplot of data points with smoothed probability distribution overlaid. *Bottom row*: heat map of the joint distribution $P(i, \mathbf{x})$ that is fed into DIB. The two spatial dimensions in the top row are binned and concatenated into a single dimension (on the horizontal axis) in the bottom row, which is the source of the "striations."

an illustration of this data smoothing procedure. We will explore the effect of the choice of smoothing scale s throughout this paper.

With the above choices, we have a fully specified DIB formulation of a geometric clustering problem. Using our above notational choices, the equations for the n^{th} step in the iterative DIB solution is (Strouse and Schwab 2017):

$$c^{*(n)}(i) = \underset{c}{\text{argmax}} \ \log q^{(n-1)}(c) - \beta d_n(i, c) \tag{3.10}$$

$$d_n(i, c) \equiv \text{KL}\big[p(\mathbf{x} \mid i) \mid q^{(n)}(\mathbf{x} \mid c)\big] \tag{3.11}$$

$$q^{(n)}(c \mid i) = \delta\big(c - c^{*(n)}(i)\big) \tag{3.12}$$

$$q^{(n)}(c) = \frac{n_c^{(n)}}{N} \tag{3.13}$$

$$q^{(n)}(\mathbf{x} \mid c) = \sum_i q^{(n)}(i \mid c)\, p(\mathbf{x} \mid i) = \frac{1}{n_c^{(n)}} \sum_{i : c^{*(n)}(i) = c} p(\mathbf{x} \mid i)\,, \tag{3.14}$$

where $n_c^{(n)}$ the number of data points assigned to cluster c at step n, $n_c^{(n)} \equiv Nq^{(n)}(c) = \sum_i q^{(n)}(c \mid i) = \left|i : c^{*(n)}(i) = c\right|$.

Note that this solution contains β as a free parameter. As discussed above, it allows us to set our preference between solutions with fewer clusters and those that retain more spatial information. It is common in the IB literature to run the algorithm for multiple values of β and to plot the collection of solutions in the "information plane" with the relevance term $I(Y;T)$ on the y-axis and the compression term $I(X;T)$ on the x-axis (Palmer et al. 2015, Creutzig et al. 2009, Chechik et al. 2005, Slonim et al. 2005, Still and Bialek 2004, Tishby and Zaslavsky 2015, Rubin et al. 2016, Strouse and Schwab 2017, Ravid and Tishby 2017). The natural such plane for the DIB is with the relevance term $I(Y;T)$ on the y-axis and *its* compression term $H(T)$ on the x-axis (Strouse and Schwab 2017). The curve drawn out by (D)IB solutions in the information plane can be viewed as a Pareto-optimal boundary of how much relevant information can be extracted about Y given a fixed amount of information about X (IB) or representational capacity by T (DIB) (Strouse and Schwab 2017). Solutions lying below this curve are of course suboptimal, but a priori, the (D)IB formalism doesn't tell us how to select a single solution from the family of solutions lying on the (D)IB boundary. Intuitively however, when faced with a boundary of Pareto-optimality, if we must pick one solution, its best to choose one at the "knee" of the curve. Quantitatively, the "knee" of the curve is the point where the curve has its maximum magnitude second derivative. In the most extreme case, the second derivative is infinite when there is a "kink" in the curve, and thus the largest kinks might correspond to solutions of particular interest. In our case, since the slope of the (D)IB curve at any given solution is β^{-1} (which can be read off from the cost functionals), kinks indicate solutions that are valid over a wide range of β. So large kinks also correspond to robust solutions, in the sense that they optimize a wide range of (D)IB tradeoffs. Quantitatively, we can measure the size of a kink by the angle θ of

the discontinuity it causes in the slope of the curve; see figure 3.2 for details. We will show in the next section that searches for solutions with large θ result in recovering the generative cluster labels for geometric data, including the correct number of clusters.

Note that this model selection procedure would not be possible if we had chosen to use IB instead of DIB. IB uses all the clusters available to it, regardless of the choice of β. Thus, all solutions on the curve would have the same number of clusters anyway, so any knees or kinks cannot be used to select the number of clusters.

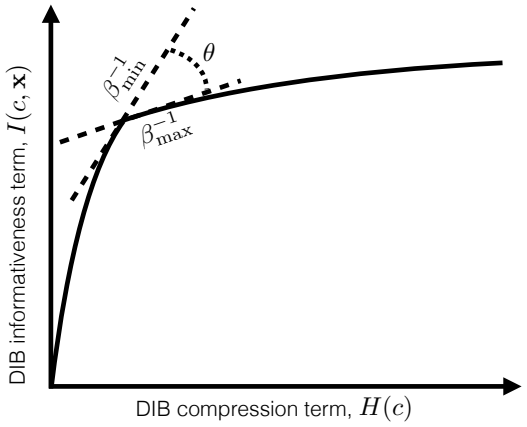

Figure 3.2: **"Kinks" in DIB information curve as model selection**. $\beta_{\min}$ and $\beta_{\max}$ are the smallest and largest β at which the solution at the kink is valid. Thus, $\beta_{\min}^{-1}$ and $\beta_{\max}^{-1}$ are the slopes of upper and lower dotted lines. The "kink angle" is then $\theta = \frac{\pi}{2} - \arctan(\beta_{\min}) - \arctan(\beta_{\max}^{-1})$. It is a measure of how robust a solution is to the choice of β; thus high values of θ indicate solutions of particular interest.

3.3 Results: geometric clustering with DIB

We ran the DIB as described above on four geometric clustering datasets, varying the smoothing width s (see eqn 3.9) and tradeoff parameter β, and measured for each solution the fraction of spatial information extracted $\tilde{I}(c; \mathbf{x}) = \frac{I(c;\mathbf{x})}{I(i;\mathbf{x})}$[1] and the number of clusters used n_c , as well as the kink angle θ. We iterated the DIB equations above just as in Strouse and Schwab (2017) with one difference. Iterating greedily

[1]Note that $I(i; \mathbf{x})$ is an upper bound on $I(c; \mathbf{x})$ due to the data processing inequality,(**?**) so $\tilde{I}(c; \mathbf{x})$ is indeed the fraction of potential geometric information extracted from the smoothed $P(i, \mathbf{x})$.

from some initialization can lead to local minima (the DIB optimization problem is non-convex). To help overcome suboptimal solutions, upon convergence, we checked whether merging any two clusters would improve the value L of the cost functional in eqn 3.2. If so, we chose the merging with the highest such reduction, and began the iterative equations again. We repeated this procedure until the algorithm converged and no merging reduced the value of L. We found that these "non-local" steps worked well in combination with the greedy "local" improvements of the DIB iterative equations. While not essential to the function of DIB, this improvement in performance produced cleaner information curves with less "noise" caused by convergence to local minima. Similar to (Strouse and Schwab 2017), the automated search over β began with an initial set of values, and then iteratively inserted more values where there were large jumps in $H(c)$, $I(c;\mathbf{x})$, or the number of clusters used, or where the largest value of β did not lead to a clustering solution capturing nearly all of the available geometric information (that is, with $I(c;\mathbf{x}) \approx I(i;\mathbf{x})$. For more details, see our code repository at https://github.com/djstrouse/information-bottleneck.

Results are shown in figure 3.3. Each large row represents a different dataset. The left column shows fractional spatial information $\tilde{I}(c;\mathbf{x})$ versus number of clusters used n_c, stacked by smoothing width s.[2] The center column shows the kink angle θ for each cluster number n_c, again stacked by smoothing width s. Finally, the right column shows example solutions.

In general, note that as we increase β, we move right along the plots in the left column, that is towards higher number of clusters n_c and more spatial information $\tilde{I}(c;\mathbf{x})$. Not all values of n_c are present because while varying the implicit parameter β, DIB will not necessarily "choose" to use all possible cluster numbers. For example, for small smoothing width s, most points won't have enough overlap in $p(\mathbf{x} \mid i)$ with their neighbors to support solutions with few clusters, and for large smoothing width

[2]Note that this is *not* the same as the information plane curve from figure 3.2. While the y-axes are the same (up to the normalization), the x-axes are different.

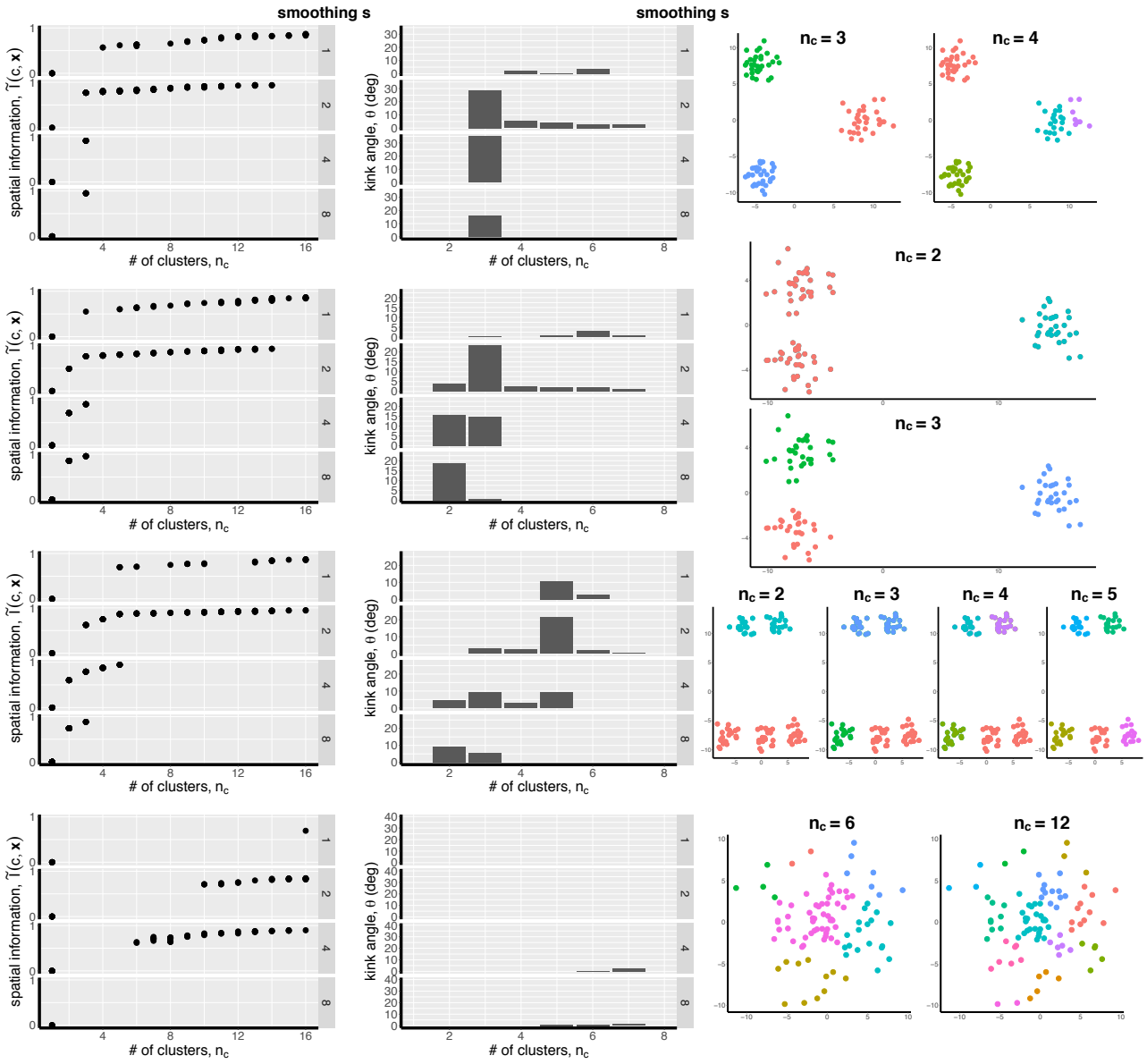

Figure 3.3: **Results: model selection and clustering with DIB.** Results for four datasets. Each row represents a different dataset. *Left column*: fraction of spatial information extracted, $\tilde{I}(c;\mathbf{x}) = \frac{I(c;\mathbf{x})}{I(i;\mathbf{x})}$, versus number of clusters used, n_c, across a variety of smoothing scales, s. *Center column*: kink angle θ (of the $I(c;\mathbf{x})$ vs $H(c)$ curve) versus number of clusters used, n_c, across a variety of smoothing scales, s. *Right column*: example resulting clusters.

s, local spatial information is thrown out and *only* solutions with few clusters are possible. More interestingly, DIB may retain or drop solutions based on how well they match the structure of the data, as we will discuss for each dataset below. Additionally, solutions that match well the structure in the data (for example, ones with n_c matched to the generative parameters) tend to be especially robust to β, that is they have a large kink angle θ. Thus, θ can be used to perform model selection. For datasets with structure at multiple scales, the kink angle θ will select different solutions for different values of the smoothing width s. This allows us to investigate structure in a dataset at a particular scale of our choosing. We now turn to the individual datasets.

The first dataset (top large row) consists of 3 equally spaced, equally sampled symmetric gaussian clusters (see solutions in right column). We see that the 3-cluster solution stands out in several ways. First, it is robust to spatial scale s. Second, the 3-cluster solution extract nearly all of the available spatial information; solutions with $n_c \geq 4$ extract little extra $\tilde{I}(c; \mathbf{x})$. Third and perhaps most salient, the 3-cluster solution has by far the largest value of kink angle θ across a wide range of smoothing scales. In the right column, we show examples of 3 and 4-cluster solutions. Note that while all 3-cluster solutions look exactly like this one, the 4-cluster solutions vary in how they chop one true cluster into two.

The second dataset (second row) consists of 3 more equally sampled symmetric gaussian clusters, but this time not equally spaced; two are much closer to one another than the third. This is a dataset with multiple scales present, thus we should expect that the number of clusters picked out by any model selection procedure, e.g. kink angle, should depend on the spatial scale of interest. Indeed, we see that to be true. The 3-cluster solution is present for all smoothing widths shown, but is only selected out as the best solution by kink angle for intermediate smoothing widths ($s = 2$). For large smoothing widths ($s = 8$), we see that the 2-cluster solution is chosen as best. For smoothing widths in between ($s = 4$), the 2 and 3-cluster solutions are roughly

equally valid. In terms of spatial information, the 2 and 3-cluster solutions are also prominent, with both transitions from $n_c = 1 \to 2$ and $n_c = 2 \to 3$ providing significant improvement in $\tilde{I}(c; \mathbf{x})$ (but little improvement for more fine-grained clusterings).

The third dataset (third row) features even more multi-scale structure, with 5 symmetric, equally sampled gaussians, again with unequal spacing. Sensible solutions exist for $n_c = 2 - 5$, and this can be seen by the more gradual rise of the fractional spatial information $\tilde{I}(c; \mathbf{x})$ with n_c in that regime. We also again see a transition in the model selection by θ from the 5-cluster solution at small smoothing widths ($s = 1, 2$) and the 2-cluster solution at larger smoothing widths ($s = 8$), with intermediate n_c favoring those and intermediate solutions. Example clusters for $n_c = 2 - 5$ are shown at right.

Finally, we wanted to ensure that DIB and our model selection procedure would not halluscinate structure where there is none, so we applied it to a single gaussian blob, with the hope that no solution with $n_c > 1$ would stand out and prove robust to β. As can be seen in the fourth row of figure 3.3, that is indeed true. No solution at any smoothing width had particuarly high kink angle θ, and no solution remained at the "knee" of the $\tilde{I}(c; \mathbf{x})$ versus n_c curve across a wide range of smoothing widths.

Overall, these results suggest that DIB on smoothed data is able to recover generative geometric structure at multiple scales, using built-in model selection procedures based on identifying robust, spatially informative solutions.

3.4 Results: DIB vs GMMs & k-means

Here we show that in the limit of infinite data and small smoothing scale s, the behavior of (D)IB is intimately related to the hard cluster boundaries implied by GMMs. We assume we have one gaussian cluster centered at $\mu_1 = (0, 0)$ with covariance $\Sigma_1 = \mathrm{diag}(\sigma_1, \sigma_2)$, and a second gaussian cluster centered at $\mu_2 = (L, 0)$

with covariance $\Sigma_2 = \mathrm{diag}(\sigma, \sigma)$. If we have a mixture model with weights w_1 and w_2, then the hard maximum likelihood boundary between these two clusters in the (x_1, x_2) plane is given by:

$$\begin{aligned} T_1 &\equiv \frac{x_1^2}{2\sigma_1^2} + \frac{x_2^2}{2\sigma_2^2} + \log \sigma_1 \sigma_2 \\ T_2 &\equiv \frac{1}{2\sigma_2}\left(x_1^2 + x_2^2 + L^2 - 2Lx_1\right) \\ \log w_1 - T_1 &= \log w_2 - T_2. \end{aligned}$$

On the other hand, the (D)IB algorithm would classify a new test point at location (x_1, x_2) gaussian smoothed by s based on the KL divergence between its smoothed distribution and the two cluster gaussians:

$$\mathrm{KL}_1 = s^2\left(\frac{\sigma_1^2 + \sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) + \frac{x_1^2}{2\sigma_1^2} + \frac{x_2^2}{2\sigma_2^2} - \frac{k}{2} + \log\frac{\sigma_1\sigma_2}{s^2} \tag{3.15}$$

$$\mathrm{KL}_2 = \frac{s^2}{\sigma^2} + \frac{1}{2\sigma^2}\left((x_1 - L)^2 + x_2^2\right) - \frac{k}{2} + \log\frac{\sigma^2}{s^2}, \tag{3.16}$$

where $k = 2$ is the number of dimensions. The boundary implied by DIB is found by setting:

$$\log w_1 - \beta \mathrm{KL}_1 = \log w_2 - \beta \mathrm{KL}_2. \tag{3.17}$$

Notice that for small smoothing width $s \to 0$ and either $\beta = 1$ or evenly sampled clusters $w_1 = w_2$, this is identical to the hard boundary implied by the GMM. For $\beta > 1$, $w_1 \neq w_2$, and small smoothing width, we see that, compared with a GMM, DIB encourages capturing more information about spatial location at the expense of using clusters more equally. Put another way, the effect of the cluster prior term $\log \frac{w_1}{w_2}$ is reduced by pulling it closer to zero, i.e. replacing it with $\log\left(\frac{w_1}{w_2}\right)^{1/\beta}$. This

provides an interesting information theoretic interpretation of GMMs and also shows the manner in which clustering with DIB is a generalization.

To see the effect of larger smoothing widths, we compared the numerically calculated DIB, GMM, and k-means cluster boundaries for the "true" assignments with $n_c = 2$ over a range of smoothing widths (see figure 3.4). The data consisted of 1000 points sampled equally ($w_1 = w_2$) from one isotropic and one skew gaussian as shown. We can see that for small smoothing widths, the DIB boundary indeed approaches that of the GMM. For larger smoothing widths, the effect of the "shape" of the clusters is muted and the DIB boundary approaches k-means. Note that this is just one particular example however and DIB need not approach k-means in the large s limit in general.

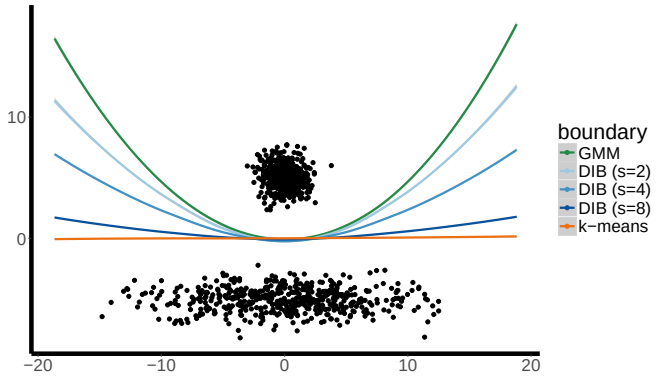

Figure 3.4: **Cluster boundaries for different algorithms**. Colored lines show boundaries separating two clusters for 3 different algorithms: k-means, GMMs, and DIB with 3 different levels of smoothing. Dataset was 1000 points drawn equally from a single symmetric gaussian and a single skew gaussian. Black points show data.

3.5 Discussion

Here, we have shown how to use the formalism of the information bottleneck to perform geometric clustering. A previous paper (Still et al. 2003) claimed to contribute similarly, however for the reasons discussed in sections 3.2 and 3.A, their approach contained fundamental flaws. We amend and improve upon that paper in four ways.

First, we show to fix the errors they made in their problem setup (with the data preparation). Second, we argue for using DIB over IB in this setting for its preference for using as few clusters as it can. Third, we introduce a novel form of model selection for the number of clusters based on discontinuities (or "kinks") in the slope of the DIB curve, which indicate solutions that are robust across the DIB tradeoff parameter β. We show that this information-based model selection criterion allows us to correctly recover generative structure in the data at multiple spatial scales. Finally, we compare the resulting clustering algorithm to k-means and gaussian mixture models (GMMs). We found that for large smoothing width, s, the performance of the method seems to behave similarly to k-means. More interestingly, we found that for small smoothing width, the method behaves as a generalization of a GMM, with a tunable tradeoff between compression and fidelity of the representation.

We have introduced one way of doing geometric clustering with the information bottleneck, but we think it opens avenues for other ways as well. First, the uniform smoothing we perform above could be generalized in a number of ways to better exploit local geometry and better estimate the "true" generative distribution of the data. For example, one could do gaussian smoothing with mean centered on each data point but the covariance estimated by the sample covariance of neighboring data points around that mean. Indeed, our early experiments with this alternative suggest it may be useful for certain datasets. Second, while choosing spatial location as the relevant variable for DIB to preserve information about seems to be the obvious first choice to investigate, other options might prove interesting. For example, preserving information about the identity of neighbors, if carefully formulated, might make fewer implicit assumptions about the shape of the generative distribution, and enable the extension of our approach to a wider range of datasets.

Scaling the approach introduced here to higher-dimensional datasets is non-trivial because the tabular representation used in the original IB (**?**)and DIB (**?**) algorithms

leads to an exponential scaling with the number of dimensions. Recently, however, three groups simultaneously introduced a variational version of IB (Alemi et al. 2016, Chalk et al. 2016, Achille and Soatto 2016), in which one parameterizes the encoder $q(t \mid x)$ (and "decoder" $q(y \mid t)$) with a function approximator, e.g. a deep neural network. This has the advantage of allowing scaling to much larger datasets. Moreover, the choice of parameterization often implies a smoothness constraint on the data, relieving the problem encountered above of needing to smooth the data. It would be interesting to develop a variational version of DIB, which could then be used to perform information-theoretic clustering as we have done here, but on larger problems and perhaps with no need for data smoothing.

Appendix

3.A Errors in (Still et al. 2003)

A previous attempt was made to draw a connection between IB and k-means (Still et al. 2003). Even before reviewing the algebraic errors that lead their result to break down, there are two intuitive reasons why such a claim is unlikely to be true. First, IB is a soft clustering algorithm, and k-means is a hard clustering algorithm. Second, the authors made the choice not to smooth the data and to set $p(\mathbf{x} \mid i) = \delta_{\mathbf{x}\mathbf{x}_i}$. As discussed in section 3.2, (D)IB clusters data points based on these conditionals, and delta functions trivially only overlap when they are identical.

The primary algebraic mistake appears just after eqn 14, in the claim that $p_n(\mathbf{x} \mid c) \propto p_{n-1}(\mathbf{x} \mid c)^{1/\lambda}$. Combining the previous two claims in that proof, we obtain:

$$p_n(\mathbf{x} \mid c) = \frac{1}{N} \sum_i \frac{\delta_{\mathbf{x}\mathbf{x}_i}}{Z_n(i, \lambda)} p_{n-1}(\mathbf{x}_i \mid c)^{1/\lambda} \,. \tag{3.18}$$

Certainly, this does not imply that $p_n(\mathbf{x} \mid c) \propto p_{n-1}(\mathbf{x} \mid c)^{1/\lambda}$ everywhere, because of the $\delta_{\mathbf{x}\mathbf{x}_i}$ factor which picks out only a finite number of points.

One might wonder why with these mistakes, the authors still obtain an algorithm that looks and performs like k-means. The reason is because their sequence of mistakes leads to the result in eqn 15 that effectively *assumes* that IB has access to geometric

information it should not, namely the cluster centers at step n. Since these are exactly what k-means uses to assign points to clusters, it is not surprising that the behavior then resembles k-means.

Chapter 4

Learning to share and hide intentions using information regularization

In the last two chapters, we discussed the application of information-theoretic tools to feature selection and clustering. Even though these were not the original intended applications of information theory, they nonetheless led to interesting solutions. In the present chapter, we turn to applying information theory to problems more similar to its original intended application - communication.

Here, we consider a setting in which multiple agents with asymmetric information seek to accomplish some goal. In acting to carry out their goals, they may convey some of the information they hold privately to other agents. If the other agent is a friend, this is a good thing, and should be encouraged. However, if the other agent is an enemy, this is a bad thing, and should be discouraged. In either case, one might want to *explicitly* include the information conveyed to other agents in the training objective. In what follows, we develop the technical tools to do so, and then show experimentally that they lead to effective cooperative / competitive strategies.

This chapter is under review as Strouse et al. (2018).

Abstract

Learning to cooperate with friends and compete with foes is a key component of multi-agent reinforcement learning. Typically to do so, one requires access to either a model of or interaction with the other agent(s). Here we show how to learn effective strategies for cooperation and competition in an asymmetric information game with no such model or interaction. Our approach is to encourage an agent to reveal or hide their intentions using an information-theoretic regularizer. We consider both the mutual information between goal and action given state, as well as the mutual information between goal and state. We show how to stochastically optimize these regularizers in a way that is easy to integrate with policy gradient reinforcement learning. Finally, we demonstrate that cooperative (competitive) policies learned with our approach lead to more (less) reward for a second agent in two simple asymmetric information games.

4.1 Introduction

In order to effectively interact with others, an intelligent agent must understand the intentions of others. In order to successfully cooperate, collaborative agents that share their intentions will do a better job of coordinating their plans together (Tomasello et al. 2005). This is especially salient when information pertinent to a goal is known asymmetrically between agents. When competing with others, a sophisticated agent might aim to hide this information from its adversary in order to deceive or surprise them. This type of sophisticated planning is thought to be a distinctive aspect of human intelligence compared to other animal species (Tomasello et al. 2005).

Furthermore, agents that share their intentions might have behavior that is more interpretable and understandable by people. Many reinforcement learning (RL) systems often plan in ways that can seem opaque to an observer. In particular, when an agent's reward function is not aligned with the designer's goal the intended behavior

often deviates from what is expected (Hadfield-Menell et al. 2016). If these agents are also trained to share high-level and often abstract information about its behavior (i.e. intentions) it is more likely a human operator or collaborator can understand, predict, and explain that agents decision. This is key requirement for building machines that people can trust.

Previous approaches have tackled aspects of this problem but all share a similar structure (Dragan et al. 2013, Ho et al. 2016, Hadfield-Menell et al. 2016, Shafto et al. 2014). They optimize their behavior against a known model of an observer which has a theory-of-mind (Baker et al. 2009, Ullman et al. 2009, Rabinowitz et al. 2018) or is doing some form of inverse-RL (Ng et al. 2000, Abbeel and Ng 2004). In this work we take an alternative approach based on an information theoretic formulation of the problem of sharing and hiding intentions. This approach does not require an explicit model of or interaction with the other agent, which could be especially useful in settings whether interactive training is expensive or dangerous. Our approach also naturally composes with scalable policy-gradient methods commonly used in deep reinforcement learning.

4.2 Hiding and revealing intentions via information-theoretic regularization

We consider multi-goal environments in the form of a discrete-time finite-horizon discounted Markov decision process (MDP) defined by the tuple $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, \mathcal{G}, P, \rho_G, r, \rho_S, \gamma_R, T)$, where $\mathcal{S}$ is a state set, $\mathcal{A}$ an action set, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ a (goal-independent) probability distribution over transitions, $\mathcal{G}$ a goal set, $\rho_G : \mathcal{G} \to \mathbb{R}_+$ a distribution over goals, $r : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$ a (goal-dependent) reward function, $\rho_S : \mathcal{S} \to \mathbb{R}_+$ a probability distribution over initial states, $\gamma_R \in [0, 1]$ a (reward) discount factor, and T the horizon.

In each episode, a goal is sampled and determines the reward structure for that episode. One agent, Alice, will have access to this goal and thus knowledge of the environment's reward structure, while a second agent, Bob, will not and instead must infer it from observing Alice. We assume that Alice knows in advance whether Bob is a friend or foe and wants to make his task easier or harder, respectively, but that she has no model of him and must train without any interaction with him.

Of course, Alice also wishes to maximize her own expected reward $\eta[\pi] = \mathbb{E}_\tau\left[\sum_{t=0}^{T} \gamma_R^t r(s_t, g)\right]$, where $\tau = (g, s_0, a_0, s_1, a_1, \ldots, s_T)$ denotes the episode trajectory, $g \sim \rho_G$, $s_0 \sim \rho_S$, $a_t \sim \pi_g(a_t \mid s_t)$, and $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$, and $\pi_g(a \mid s; \theta) : \mathcal{G} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ is Alice's goal-dependent probability distribution over actions (policy) parameterized by θ.

It is common in RL to consider loss functions of the form $J[\pi] = \eta[\pi] + \beta\ell[\pi]$, where ℓ is a regularizer meant to help guide the agent toward desirable solutions. For example, the policy entropy is a common choice to encourage exploration (Mnih et al. 2016), while pixel prediction and control have been proposed to encourage exploration in visually rich environments with sparse rewards (Jaderberg et al. 2016).

Below, we will consider two different information regularizers meant to encourage/discourage Alice from sharing goal information with Bob: the (conditional) mutual information between goal and action given state, $I_{\text{action}}[\pi] \equiv I(A; G \mid S)$, which we will call the "action information", and the mutual information between state and goal, $I_{\text{state}}[\pi] \equiv I(S; G)$, which we will call the "state information." Since the mutual information is a general measure of dependence (linear and non-linear) between two variables, I_{action} and I_{state} measure the ease in inferring the goal from the actions and states, respectively, generated by the policy π. Thus, if Alice wants Bob to do well, she should choose a policy with high information, and vice versa if not.

We consider both action and state informations because they have different advantages and disadvantages. Using action information assumes that Bob (the observer) can see both Alice's states *and* actions, which may be unrealistic in some

environments, such as one in which the actions are the torques a robot applies to its joint angles (Eysenbach et al. 2018). Using state information relaxes the assumption to that Bob only can observe Alice's states, however it does so at the cost of requiring Alice to count goal-dependent state frequencies under the current policy. Optimizing action information, on the other hand, does not require state counting. So, in summary, action information is simpler to optimize, but state information may be more appropriate to use in a setting where an observer can't observe (or infer) the observee's actions.

The generality with which mutual information measures dependence is at once its biggest strength and weakness. On the one hand, using information allows Alice to prepare for interaction with Bob with neither a model of nor interaction with him. On the other hand, Bob might have limited computational resources (for example, perhaps his policy is linear with respect to his observations of Alice) and so he may not be able to "decode" all of the goal information that Alice makes available to him. Nevertheless, I_{action} and I_{state} can at least be considered upper bounds on Bob's inference performance; if $I_{\text{action}} = 0$ or $I_{\text{state}} = 0$, it would be impossible for Bob to guess the goal (above chance) from Alice's actions or actions, respectively, alone.

4.2.1 Optimizing action information: $I_{\text{action}} \equiv I(A; G \mid S)$

First, we discuss regularization via optimizing the mutual information between goal and action (conditioned on state), $I_{\text{action}} \equiv I(A; G \mid S)$, where G is the goal for the episode, A is the chosen action, and S is the state of the agent. That is, we will train an agent to maximize the objective $J_{\text{action}}[\pi] \equiv \mathbb{E}[r] + \beta I_{\text{action}}$, where β is a tradeoff parameters whose sign determines whether we want the agent to signal (positive) or hide (negative) their intentions, and whose magnitude determines the relative preference for rewards and intention signaling/hiding.

I_{action} is a functional of the multi-goal policy $\pi_g(a \mid s) \equiv p(a \mid s, g)$, that is the probability distribution over actions given the current goal and state, and is given by:

$$I_{\text{action}} \equiv I(A; G \mid S) = \sum_s p(s)\, I(A; G \mid S = s) \tag{4.1}$$

$$= \sum_g \rho_G(g) \sum_s p(s \mid g) \sum_a \pi_g(a \mid s) \log \frac{\pi_g(a \mid s)}{p(a \mid s)}. \tag{4.2}$$

The quantity involving the sum over actions is a KL divergence between two distributions: the goal-dependent policy $\pi_g(a \mid s)$ and a goal-independent policy $p(a \mid s)$. This goal-independent policy comes from marginalizing out the goal, that is $p(a \mid s) = \sum_g \rho_G(g)\, \pi_g(a \mid s)$, and can be thought of as a fictitious policy that represents the agent's "habit" in the absence of knowing the goal. We will denote $\pi_0(a \mid s) \equiv p(a \mid s)$ and refer to it as the "base policy," whereas we will refer to $\pi_g(a \mid s)$ as simply the "policy." Thus, we can rewrite the information above as:

$$I_{\text{action}} = \sum_g \rho_G(g) \sum_s p(s \mid g)\, \mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)] \tag{4.3}$$

$$= \mathbb{E}_\tau[\mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)]]\,. \tag{4.4}$$

Writing the information this way suggests a method for stochastically estimating it. First, we sample a goal g from $p(g)$, that is we initialize an episode of some task. Next, we sample states s from $p(s \mid g)$, that is we generate state trajectories using our policy $\pi_g(a \mid s)$. At each step, we measure the KL between the policy and the base policy. Averaging this quantity over episodes and steps give us our estimate of I_{action}.

Optimizing I_{action} with respect to the policy parameters θ is a bit trickier, however, because the expectation above is with respect to a distribution that depends on θ.

Thus, the gradient of I_{action} with respect to θ has two terms:

$$\nabla_\theta I_{\text{action}} = \sum_g \rho_G(g) \sum_s \left(\nabla_\theta p(s \mid g)\right) \mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)] \tag{4.5}$$

$$+ \sum_g \rho_G(g) \sum_s p(s \mid g)\, \nabla_\theta \mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)]\,. \tag{4.6}$$

The second term involves the same sum over goals and states as in equation 4.3, so it can be written as an expectation over trajectories, $\mathbb{E}_\tau[\nabla_\theta \mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)]]$, and therefore is straightforward to estimate from samples. The first term is more cumbersome, however, since it requires us to model (the policy dependence of) the goal-dependent state probabilities, which in principle involves knowing the dynamics of the environment. Perhaps surprisingly, however, the gradient can still be estimated purely from sampled trajectories, by employing the so-called "log derivative" trick to rewrite the term as an expectation over trajectories. The calculation is similar to the proof of the policy gradient theorem (Sutton et al. 1999) and details can be found in the supplemental information.

The resulting Monte Carlo policy gradient (MCPG) update is:

$$\nabla_\theta J_{\text{action}}(t) = A_{\text{action}}(t)\, \nabla_\theta \log \pi_g(a_t \mid s_t) + \beta \nabla_\theta \mathrm{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)]\,, \tag{4.7}$$

where $A_{\text{action}}(t) \equiv R_t + \beta R_{\text{action}}(t) - V_g(s_t)$ is the advantage, $R_t = \sum_{\tau=t}^{T} \gamma_R^{\tau-t} r_t$ is the discounted reward return, $V_g(s_t)$ is a goal-state value function regressed toward $R_t + \beta R_{\text{action}}(t)$, and we have introduced an "action information return":

$$R_{\text{action}}(t) \equiv \sum_{u=t+1}^{\min(t+n,T)} \gamma_I^{u-t} \mathrm{KL}[\pi_g(a \mid s_u) \mid \pi_0(a \mid s_u)]\,, \tag{4.8}$$

where γ_I is an information discount factor (leading to exponential discounting) and n is the maximum number of terms to include in the info return (leading to Heaviside

Algorithm 4.1 Action information regularized REINFORCE with value baseline.

Input: β, n, ρ_G, γ_I, γ_R, and ability to sample MDP $\mathcal{M}$
Initialize π, parameterized by θ
Initialize V, parameterized by ϕ
for $i = 1$ **to** N_{episodes} **do**
  Generate trajectory $\tau = (g, s_0, a_0, s_1, a_1, \ldots, s_T)$
  for $t = 0$ **to** $T - 1$ **do**
    Marginalize to obtain base policy: $\pi_0(a \mid s_t) = \sum_g \rho_G(g)\, \pi_g(a \mid s_t)$
    Calculate reward return: $R_t = \sum_{\tau=t}^{T} \gamma_R^{\tau-t} r_t$
    Calculate action information return $R_{\text{action}}(t)$ using equation 4.8
    Update policy in direction of $\nabla_\theta J_{\text{action}}(t)$ using equation 4.7
    Update value function in direction of $-\nabla_\phi \left\| V_g(s_t) - (R_t + \beta R_{\text{action}}(t)) \right\|^2$
  end for
end for

discounting).[1] Thus, the agent optimizes a pseudo-reward, which is a weighted sum of the usual (reward) return and the info return. The gradient update above seeks to maximize this pseudo-reward, along with the KL divergence between the policy and base policy in the present state. This algorithm is summarized in algorithm 4.1.

4.2.2 Optimizing state information: $I_{\text{state}} \equiv I(S; G)$

We now consider how to regularize an agent by the information one's *states* give away about the goal, using the mutual information between state goal, $I_{\text{state}} \equiv I(S; G)$. This can be written:

$$I_{\text{state}} = \sum_g \rho_G(g) \sum_s p(s \mid g) \log \frac{p(s \mid g)}{p(s)} = \mathbb{E}_\tau \left[\log \frac{p(s \mid g)}{p(s)} \right]. \tag{4.9}$$

In order to *estimate* this quantity, we could track and plug into the above equation the empirical state frequencies $p_{\text{emp}}(s \mid g) \equiv \frac{N_g(s)}{N_g}$ and $p_{\text{emp}}(s) \equiv \frac{N(s)}{N}$, where $N_g(s)$ is the number of times state s was visited during episodes with goal g, $N_g \equiv \sum_s N_g(s)$ is the total number of steps taken under goal g, $N(s) \equiv \sum_g N_g(s)$ is the number of

[1] n enters into the calculation as the approximation depth for $p(s \mid g)$. The introduction of γ_I is typically done by defining $p(s)$ and $p(s \mid g)$ to be themselves discounted. (**?**)

times state s was visited across all goals, and $N \equiv \sum_{g,s} N_g(s) = \sum_g N_g = \sum_s N(s)$ is the total number of state visits across all goals and states. Thus, keeping a moving average of $\log \frac{p_{\text{emp}}(s_t|g)}{p_{\text{emp}}(s_t)}$ across episodes and steps yields an estimate of I_{state}.

However, we are of course interested in *optimizing* I_{state} and so, as in the last section, we need to employ a slightly more sophisticated estimate procedure. Taking the gradient of I_{state} with respect to the policy parameters θ, we get:

$$\nabla_\theta I_{\text{state}} = \sum_g \rho_G(g) \sum_s \left(\nabla_\theta p(s \mid g)\right) \log \frac{p(s \mid g)}{p(s)} \tag{4.10}$$

$$+ \sum_g \rho_G(g) \sum_s p(s \mid g) \left(\frac{\nabla_\theta p(s \mid g)}{p(s \mid g)} - \frac{\nabla_\theta p(s)}{p(s)} \right). \tag{4.11}$$

The calculation is similar to that for evaluating $\nabla_\theta I_{\text{state}}$ and details can again be found in the supplemental information. The resulting MCPG update is:

$$\nabla_\theta J_{\text{state}}(t) = A_{\text{state}}(t)\, \nabla_\theta \log \pi_g(a_t \mid s_t) \tag{4.12}$$

$$- \beta \sum_{g' \neq g} \rho_G\left(g'\right) R_{\text{cf}}\left(t, g'\right) \nabla_\theta \log \pi_{g'}(a_t \mid s_t)\,, \tag{4.13}$$

where $A_{\text{state}}(t) \equiv R_t + \beta R_{\text{state}}(t) - V_g(s_t)$ is the advantage, $V_g(s_t)$ is a goal-state value function regressed toward $R_t + \beta R_{\text{state}}(t)$, and we have introduced the "state information return" and "counterfactual goal return":

$$R_{\text{state}}(t) \equiv \sum_{u=t+1}^{\min(t+n,T)} \gamma_I^{u-t} \left(1 - p_{\text{emp}}(g \mid s_t) + \log \frac{p_{\text{emp}}(s_u \mid g)}{p_{\text{emp}}(s_u)} \right) \tag{4.14}$$

$$R_{\text{cf}}\left(t, g'\right) \equiv \sum_{v=t+1}^{\min(t+n,T)} \gamma_I^{u-t} \frac{p_{\text{emp}}\left(s_{v-n} \mid g'\right)}{p_{\text{emp}}(s_{v-n} \mid g)} \prod_{u=1}^{n} \frac{\pi_{g'}(a_{v-u} \mid s_{v-u})}{\pi_g(a_{v-u} \mid s_{v-u})} \frac{p_{\text{emp}}(s_v \mid g)}{p_{\text{emp}}(s_v)}, \tag{4.15}$$

as well as $p_{\text{emp}}(g \mid s_t) \equiv \rho_G(g)\, \frac{p_{\text{emp}}(s_t|g)}{p_{\text{emp}}(s_t)}$. The first term in equation 4.12 encourages the present action to be more likely to the extent that it leads to a future trajectory over states unique to the present goal. The second term discourages the present action

Algorithm 4.2 State information regularized REINFORCE with value baseline.

Input: β, n, ρ_G, γ_I, γ_R, and ability to sample MDP $\mathcal{M}$
Initialize π, parameterized by θ
Initialize V, parameterized by ϕ
Initialize the state counts $N_g(s)$
for $i = 1$ **to** N_{episodes} **do**
  Generate trajectory $\tau = (g, s_0, a_0, s_1, a_1, \ldots, s_T)$
  Update $N_g(s)$ (and therefore $p_{\text{emp}}(s \mid g)$) according to τ
  for $t = 0$ **to** $T - 1$ **do**
    Calculate reward return: $R_t = \sum_{\tau=t}^{T} \gamma_R^{\tau-t} r_t$
    Calculate state information return $R_{\text{state}}(t)$ using equation 4.14
    Calculate counterfactual goal return $R_{\text{cf}}\left(t, g^{'}\right)$ using equation 4.15
    Update policy in direction of $\nabla_\theta J_{\text{state}}(t)$ using equation 4.12
    Update value function in direction of: $-\nabla_\phi \left\| V_g(s_t) - (R_t + \beta R_{\text{state}}(t)) \right\|^2$
  end for
end for

under *other goals*, again to the extent that the present action leads to states unique to the *present* goal. The second term also includes an importance sample weight over trajectories, since we are considering the probability of a trajectory generated under the present goal, but evaluating it relative to another goal.

4.3 Related work

Whye Teh et al. (2017) recently proposed an algorithm similar to our action information regularized approach (algorithm 4.1), but with very different motivations. They argued that constraining goal-specific policies to be close to a distilled base policy promotes transfer by sharing knowledge across goals. Due to this difference in motivation, they only explored the $\beta < 0$ regime (i.e. our "competitive" regime). They also did not derive their update from an information-theoretic cost function, but instead proposed the update directly. Because of this, their approach differs in that it did not include the $\beta \nabla_\theta \text{KL}[\pi_g \mid \pi_0]$ term, and instead only included the modified return. Moreover, they did not calculate the full KLs in the modified return, but instead estimated them from single samples (e.g. $\text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \approx \log \frac{\pi_g(a_t|s_t)}{\pi_0(a_t|s_t)}$).

Nevertheless, the similarity in our approaches suggest a link between transfer and competitive strategies, although we do not explore this here.

Eysenbach et al. (2018) also recently proposed an algorithm similar to ours, which used both I_{state} and I_{action} but with the "goal" replaced by a randomly sampled "skill" label in an unsupervised setting (i.e. no reward). Their motivation was to learn a diversity of skills that would later would be useful for a supervised (i.e. reward-yielding) task. Their approach to optimizing I_{state} differs from ours in that it uses a discriminator, a powerful approach but one that, in our setting, would imply a more specific model of the observer which we wanted to avoid.

Dragan et al. (2013) considered training agents to reveal their goals (in the setting of a robot grasping task), but did so by building an explicit model of the observer. Ho et al. (2016) uses a similar model to capture human generated actions that "show" a goal also using an explicit model of the observer. There is also a long history of work on training RL agents to cooperate and compete through interactive training and a joint reward (e.g. (Littman 1994, 2001, Kleiman-Weiner et al. 2016, Leibo et al. 2017)). Our approach differs in that it enables us to train an agent to cooperate or compete *without* needing to interact with or model the other agent(s).

4.4 Experiments

We demonstrate the effectiveness of our approach in two stages. First, we show that training Alice (who has access to the goal of the episode) with information regularization effectively encourages both goal signaling and hiding, depending on the sign of the coefficient β. Second, we show that Alice's goal signaling and hiding translate to higher and lower rates of reward acquisition for Bob (who does not have access to the goal and must infer it from observing Alice), respectively. We demonstrate these results in two different simple settings.

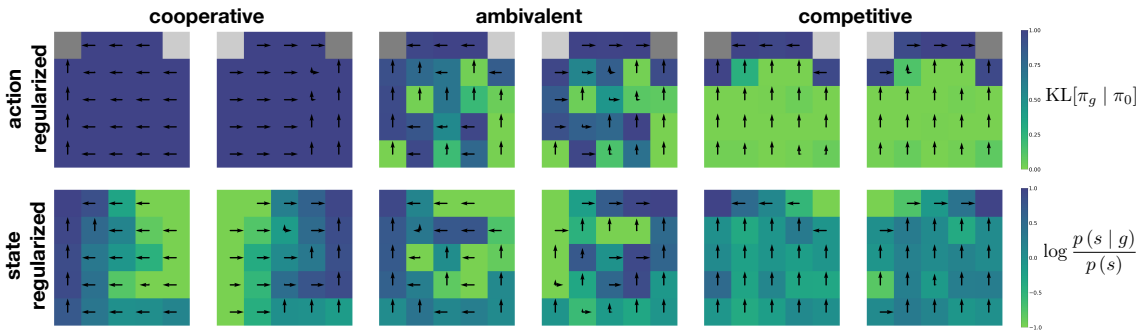

Figure 4.1: **Information-regularized policies**. Top row: regularization with I_{action}. Bottom row: regularization with I_{state}. Left column: $\beta = .025$. Center column: $\beta = 0$. Right column: $\beta = .025$. See main text for additional details.

4.4.1 Spatial navigation

The first setting we consider is a simple grid world spatial navigation task, where we can fully visualize and understand Alice's regularized policies. The 5×5 environment contains two possible goals: the top left state or the top right. On any given episode, one goal is chosen randomly (so ρ_G is uniform) and that goal state is worth $+1$ reward. The other goal state is then worth -1. Both are terminal. Each of Alice and Bob spawn in a random (non-terminal) state and take actions in $\mathcal{A} = \{\text{left}, \text{right}, \text{up}, \text{down}, \text{stay}\}$. A step into a wall is equivalent to the stay action but results in a penalty of $-.1$ reward. We first train Alice alone, and then freeze her parameters and introduce Bob.

Alice was trained using implementations of algorithms 4.1 and 4.2 in TensorFlow (**?**). Given the small, discrete environment, we used tabular representations for both π and V. See supplemental information for training parameters.

Examples of Alice's resulting policies are shown in figure 4.1. The top row contains policies regularized with I_{action}, the bottom with I_{state}. The left column contains "cooperative" policies encouraged to share goal information ($\beta = .025$), the middle "ambivalent" policies that are unregularized ($\beta = 0$), and the right "competitive" policies encouraged to hide goal information ($\beta = -.025$). In each case, two policies are shown - at left, the policy when the goal is in the top left, and at right, when it is in the top right. The arrows are proportional to Alice's action probabilities. The color

scale indicates $\mathrm{KL}[\pi_g \mid \pi_0]$ and $\log \frac{p_{\mathrm{emp}}(s|g)}{p_{\mathrm{emp}}(s)}$ in bits for action and state information regularization, respectively. Thus, bluer indicates more goal information shared in that state.

In the cooperative case with action information regularization, Alice wants to maximize $\mathrm{KL}[\pi_g \mid \pi_0]$ and thus she wants her goal-dependent policies to differ as much as possible. This primarily results in her going left in most states when the goal is in the top left, and right when its in the top right. She can also choose to have some probability of going up in those states, as long as she only does so under one goal or the other and not both. Ultimately this means that no matter what state Alice spawns in, she immediately signals the goal with her very first action, and continues to do so with each subsequent action.

In the competitive case with action information regularization, Alice instead wants to *minimize* $\mathrm{KL}[\pi_g \mid \pi_0]$ and thus she wants her goal-dependent policies to *match* as much as possible. In the present case, this means that she chooses to go up as far as she can until she must pivot in order to still reach the goal in the minimal number of steps. This leads to her hiding the goal for as long as possible without sacrificing reward.

Without information regularization, Alice's policy is a random mixture of the competitive and cooperative strategies, the details of which are determined by initialization and the randomness of training trajectories. Regularization with state information yields similar policies to those generated with action information in all three cases, since in this deterministic setting, different action choices are tightly coupled with different state trajectories.

To demonstrate that Alice's goal revealing and hiding behaviors are useful for cooperation and competition, respectively, we then trained a second agent, Bob, who does not have access to the goal and instead must infer it from observing Alice. Thus, while Alice's inputs at time t were the present goal g and her state s_t^{alice}, Bob's are

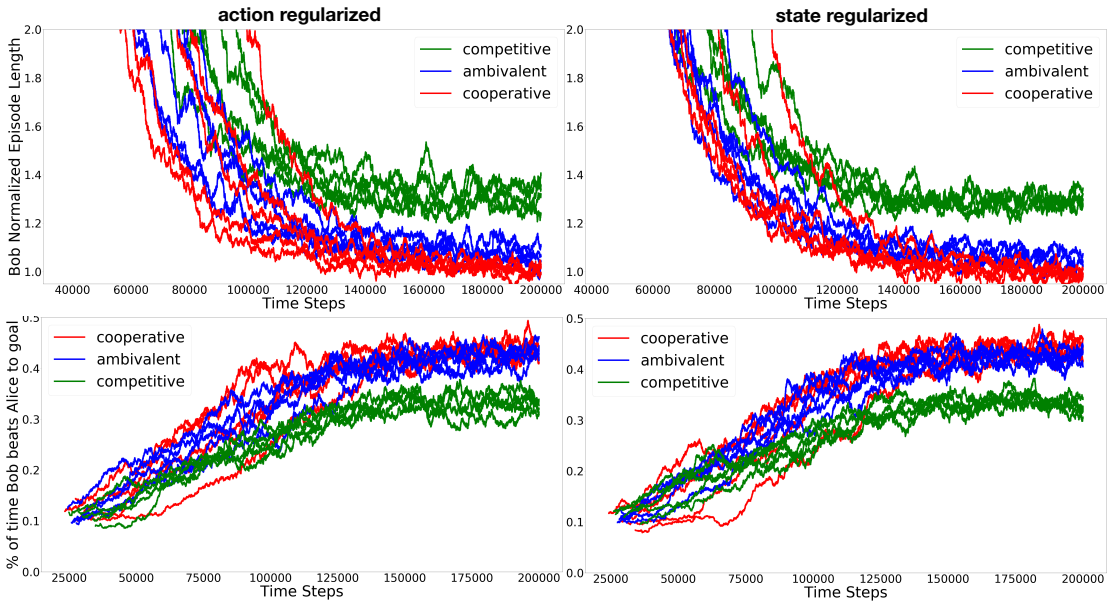

Figure 4.2: **The effect of Alice hiding/sharing goal information on Bob's performance.** Left column: regularization with I_{action}. Right column: regularization with I_{state}. Top row: Bob's episode length relative to Alice's (moving average over 500 episodes). Bottom row: the percentage of time Alice beats Bob to the goal (moving average over 1000 episodes).

Alice's present state and action s_t^{alice} and a_t^{alice}, as well as his own state s_t^{bob}. Details are available in the supplemental information, but in brief, Bob processes Alice's state-action trajectories with an RNN to form a belief about the goal, which then feeds into his policy, all of which is trained end-to-end via REINFORCE.

We trained 5 of each of the 3 versions of Alice above, and 10 Bobs per Alice. We plot the results for the best performing Bob for each Alice (so $5 \times 3 = 15$ curves) in figure 4.2. We use all 5 Alices to estimate the variance in our approach, but the best-of-10 Bob to provide a reasonable estimate of the *best* performance of a friend/foe.

We measure Bob's performance in terms of his episode length, relative to Alice's, as well as the percentage of time he beats Alice to the goal. For both action and state information regularization, encouraging Alice to hide goal information leads to Bob taking about 30% longer to reach the goal relative to when Alice is encouraged to share goal information. Information-hiding Alice receives a boost of similar magnitude

in the frequency with which she beats Bob to the goal. Training without information regularization leads to results in between the competitive and cooperative strategies, although closer to the cooperative strategy in this case. We also note that the variance in Bob's performance was higher for the unregularized case, and much higher for the competitive case, with nearly zero variance in performance for the cooperative case, indicating that information hiding and sharing make training harder and easier, respectively.

4.4.2 Key-and-door game

In the above spatial navigation task, information regularization of Alice breaks symmetries between equally-preferred (in terms of reward) navigation strategies. However, in many scenarios, it might be worthwhile for an agent to give up some reward if it means large gains in the ability to hide or share information. To demonstrate that our approach could also discover such "lossy" strategies, we designed a simple key-and-door game with this feature (figure 4.3, left).

It is again a two-goal (door) game with the same action space and reward structure as the spatial navigation setting. Alice again alone receives the goal, and Bob must infer it from observing her. The difference is that, in order to enter the terminal states, Alice and Bob must first pick up an appropriate key. Each agent has goal-specific keys that only they can pick up (top/bottom rows, color-coded to door, labeled with A/B for Alice/Bob). Alice also has access to a master key that can open both doors (center right). Agents can only pick up one key per episode - the first they encounter. Bob spawns in the same location every time (the "B"), while Alice spawns in any of the 3 spaces between her two goal-specific keys (the "A" and spaces above/below). This means that Bob has a shorter path to the goals, and thus if Alice telegraphs the goal right away, Bob will beat her to it. While Alice's master key is strictly on a

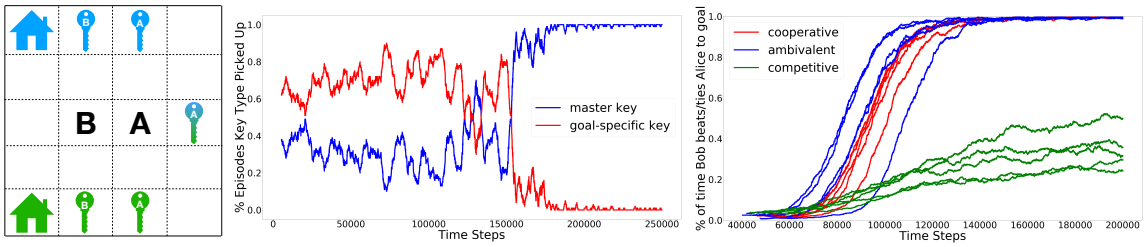

Figure 4.3: **Key-and-door game results**. Left: depiction of game. Center: percentage episodes in which Alice picks up goal-specific vs master key during training in an example run (moving average over 100 episodes). Right: percentage episodes in which Bob beats/tie Alice to the goal (moving average over 1000 episodes).

longer path to the goal, picking it up allows her to delay informing Bob of the goal such that she can beat him to it.

We trained Alice with action information regularization as in the previous section (see supplemental information for training parameters). When unregularized or encouraged to share goal information ($\beta = .25$), Alice simply took the shortest path to the goal, never picking up the master key. When Bob was trained on these Alices, he beat/tied her to the goal on approximately 100% of episodes (figure 4.3, right). When encouraged to hide information ($\beta = -.25$), however, we found that Alice learned to take the longer path via the master key on about half of initializations (example in figure 4.3, center). When Bob was trained on these Alices, he beat/tied her to the goal much less than half the time (figure 4.3, right). Thus, our approach successfully encourages Alice us to forgo rewards during solo training in order to later compete more effectively in an interactive setting.

4.5 Discussion

In this work, we developed a new framework for building agents that balance reward-seeking with information-hiding/sharing behavior. We demonstrate that our approach allows agents to learn effective cooperative and competitive strategies in asymmetric information games without an explicit model or interaction with the other

agent(s). Such an approach could be particularly useful in settings where interactive training with other agents could be dangerous or costly, such as the training of expensive robots or the deployment of financial trading strategies.

We have here focused on simple environments with discrete and finite states, goals, and actions, and so we briefly describe how to generalize our approach to more complex environments. When optimizing I_{action} with many or continuous actions, one could stochastically approximate the action sum in $\text{KL}[\pi_g \mid \pi_0]$ and its gradient (as in Whye Teh et al. (2017)). Alternatively, one could choose a form for the policy π_g and base policy π_0 such that the KL is analytic. For example, it is common for π_g to be Gaussian when actions are continuous. If one also chooses to use a Gaussian approximation for π_0 (forming a variational bound on I_{action}), then $\text{KL}[\pi_g \mid \pi_0]$ is closed form. For optimizing I_{state} with continuous states, one can no longer count states exactly, so these counts could be replaced with, for example, a pseudo-count based on an approximate density model (Ostrovski et al. 2017). Of course, for both types of information regularization, continuous states or actions also necessitates using function approximation for the policy representation. Finally, although we have assumed access to the goal distribution ρ_G, one could also approximate it from experience.

Appendix

4.A Calculating $\nabla_\theta I_{\text{action}}(t)$

We seek to evaluate the first term (see main text for discussion of the second term) in:

$$\nabla_\theta I_{\text{action}} = \sum_g \rho_G(g) \sum_s \left(\nabla_\theta p(s \mid g)\right) \mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)] \tag{4.16}$$

$$+ \sum_g \rho_G(g) \sum_s p(s \mid g)\, \nabla_\theta \mathrm{KL}[\pi_g(a \mid s) \mid \pi_0(a \mid s)]\,. \tag{4.17}$$

Doing so requires us to model (the policy dependence of) the goal-dependent state probabilities, which in principle involves knowing the dynamics of the environment. The simplest such "model" we could use are the empirical goal-dependent state frequencies $p_{\text{emp}}(s \mid g) \equiv \frac{N_g(s)}{N_g}$, where $N_g(s)$ is the number of times state s was visited during episodes with goal g and $N_g = \sum_s N_g(s)$ is the total number of steps taken under goal g. However, if we simply plugged in $\hat{p}_0(s_t \mid g) \equiv p_{\text{emp}}(s \mid g)$, there is no policy dependence and so the first term would vanish (since $\nabla_\theta p_{\text{emp}}(s \mid g) = 0$). This yields what we will call the "zeroth-order" approximation to the gradient in equation 4.16:

$$\nabla_\theta I^0_{\text{action}}(t) \equiv \nabla_\theta \mathrm{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)]\,, \tag{4.18}$$

where the superscript 0 refers to the order of the approximation of $p(s \mid g)$ (i.e. $\hat{p}_0(s \mid g)$). A "first-order" approximation, going one step back in time, would be:[2]

$$\hat{p}_1(s_t \mid g) \equiv \sum_{s_{t-1}, a_{t-1}} p_{\text{emp}}(s_{t-1} \mid g)\, \pi_g(a_{t-1} \mid s_{t-1})\, P(s_t \mid s_{t-1}, a_{t-1}) \tag{4.19}$$

The idea here is to model the probability of reaching state s_t as the probability of reaching another state s_{t-1}, choosing an action a_{t-1} in that state, and then transitioning into state s_t, summed over s_{t-1} and a_{t-1}. The n-step generalization of this approximation is then:

$$\hat{p}_n(s_t \mid g) \equiv \sum_{\tau(t,n)} p_{\text{emp}}(s_{t-u} \mid g) \prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u})\,, \tag{4.20}$$

where $\tau(t, n) = \{s_{t-u}, a_{t-u}\}_{u=1}^{n}$ is the set of trajectories of length n going backwards from time t. That is, we approximate $p(s_t \mid g)$ by summing over all paths of length n ending in state s_t, considering the probabilities of the originating state, and the action choices and transitions along the way.

Since the policy-dependence of $\hat{p}_n$ is now explicit in equation 4.20, we can take the derivative with respect to the policy parameters θ:

$$\nabla_\theta \hat{p}_n(s_t \mid g) = \sum_{\tau(t,n)} p_{\text{emp}}(s_{t-n} \mid g) \times \sum_{v=1}^{u} P(s_{t-v+1} \mid s_{t-v}, a_{t-v})\, \nabla_\theta \pi_g(a_{t-v} \mid s_{t-v}) \tag{4.21}$$

$$\times \prod_{u=1, u \neq v}^{n} P(s_{t-u+1} \mid s_{t-u}, a_{t-u})\, \pi_g(a_{t-u} \mid s_{t-u})\,. \tag{4.22}$$

To massage this into an expectation over trajectories so that we can estimate from samples, we will employ the so-called log-derivative trick. That is, we multiply and

[2]We have added time indices for clarity, but note that s_{t-1} and a_{t-1} are dummy variables that are summed over, and are *not* the *actual* previous states and actions in the present trajectory, but rather represent *all possible* previous states and actions.

divide by the $\pi_g(a_{t-v} \mid s_{t-v})$ to get:

$$\nabla_\theta \hat{p}_n(s_t \mid g) = \sum_{\tau(t,n)} p_{\text{emp}}(s_{t-n} \mid g) \tag{4.23}$$

$$\times \sum_{v=1}^{u} P(s_{t-v+1} \mid s_{t-v}, a_{t-v})\, \pi_g(a_{t-v} \mid s_{t-v})\, \frac{\nabla_\theta \pi_g(a_{t-v} \mid s_{t-v})}{\pi_g(a_{t-v} \mid s_{t-v})} \tag{4.24}$$

$$\times \prod_{u=1, u\neq v}^{n} P(s_{t-u+1} \mid s_{t-u}, a_{t-u})\, \pi_g(a_{t-u} \mid s_{t-u}) \tag{4.25}$$

$$= \sum_{\tau(t,n)} p_{\text{emp}}(s_{t-n} \mid g) \prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u}) \tag{4.26}$$

$$\times \sum_{v=1}^{n} \frac{\nabla_\theta \pi_g(a_{t-v} \mid s_{t-v})}{\pi_g(a_{t-v} \mid s_{t-v})} \tag{4.27}$$

$$= \sum_{\tau(t,n)} p_{\text{emp}}(s_{t-n} \mid g) \prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u}) \tag{4.28}$$

$$\times \sum_{v=1}^{n} \nabla_\theta \log \pi_g(a_{t-v} \mid s_{t-v})\,. \tag{4.29}$$

Plugging this back into the first term of equation 4.16, which we denote T_1^n (adding a superscript n to indicate that we have plugged in the n-step approximation):

$$T_1^n = \sum_{g,\tau(t,n),s_t} \rho_G(g)\, p_{\text{emp}}(s_{t-n} \mid g) \prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u}) \tag{4.30}$$

$$\times \mathrm{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \sum_{v=1}^{n} \nabla_\theta \log \pi_g(a_{t-v} \mid s_{t-v}) \tag{4.31}$$

$$= \mathbb{E}_\tau \left[\mathrm{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \sum_{v=1}^{n} \nabla_\theta \log \pi_g(a_{t-v} \mid s_{t-v}) \right], \tag{4.32}$$

where we have now written this term as an expectation over trajectories and so can therefore estimate the gradient from samples. For trajectories of length less than n, the approximation in equation 4.20 would "ground out" at the spawning state. Therefore, the sum in the expectation should instead be from $v = 1$ to $\min(n, t)$.

This yields the n^{th}-order gradient estimator:

$$\nabla_\theta I^n_{\text{action}}(t) \equiv \nabla_\theta \text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \tag{4.33}$$

$$+ \text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \sum_{u=1}^{\min(n,t)} \nabla_\theta \log \pi_g(a_{t-u} \mid s_{t-u}) \,. \tag{4.34}$$

While there are many approaches to maximizing expected reward, because the information is a functional of the policy and we wish to simultaneously optimize rewards and information, it will be most natural to take a policy gradient approach. That is, the gradient of our objective (with respect to policy parameters θ) at time step t in an episode with goal g will given by:

$$\nabla_\theta J^n_{\text{action}}(t) = R_t \nabla_\theta \log \pi_g(a_t \mid s_t) + \beta \nabla_\theta \text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \tag{4.35}$$

$$+ \beta \text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \sum_{u=1}^{\min(n,t)} \nabla_\theta \log \pi_g(a_{t-u} \mid s_{t-u}) \,. \tag{4.36}$$

where $R_t = \sum_{\tau=t}^{T} \gamma_R^{\tau-t} r_t$ is the discounted return, T is the length of the episode, γ_R the reward discount factor, and r_t the reward at time step t. By using the actual episode return R_t, we are assuming finite episodes and taking an approach known as the REINFORCE algorithm (**?**). However, one could replace R_t with an estimated return such as a learned Q-value $Q(s_t, a_t)$.

We can understand this update better if we take the sum over terms going backward in time and group them with the gradient updates at those times. That yields:

$$\nabla_\theta J^n_{\text{action}}(t) = (R_t + \beta R_{\text{action}}) \nabla_\theta \log \pi_g(a_t \mid s_t) + \beta \nabla_\theta \text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \,, \tag{4.37}$$

where $R_t = \sum_{\tau=t}^{T} \gamma_R^{\tau-t} r_t$ is the discounted reward return, and we have introduced an "action information return":

$$R^n_{\text{action}}(t) \equiv \sum_{u=t+1}^{\min(t+n,T)} \text{KL}[\pi_g(a \mid s_u) \mid \pi_0(a \mid s_u)] \,. \tag{4.38}$$

Thus, the agent optimizes a pseudo-reward, which is a weighted sum of the usual (reward) return and the info return. The gradient update above seeks to maximize this pseudo-reward, along with the KL divergence between the policy and base policy in the present state.

Of course, the info return could grow unbounded with large n and T, so just as one includes a discount factor for the reward, one should also include it for the info return.[3] That is, we revise our above definition to:

$$R^n_{\text{action}}(t) \equiv \sum_{u=t+1}^{\min(t+n,T)} \gamma_I^{u-t} \text{KL}[\pi_g(a \mid s_u) \mid \pi_0(a \mid s_u)] \,, \tag{4.39}$$

where γ_I is the discount for the information return.

In the main text, we drop the n superscripts and include a value function baseline, yielding:

$$\nabla_\theta J_{\text{action}}(t) = A_{\text{action}}(t) \, \nabla_\theta \log \pi_g(a_t \mid s_t) + \beta \nabla_\theta \text{KL}[\pi_g(a \mid s_t) \mid \pi_0(a \mid s_t)] \,, \tag{4.40}$$

where $A_{\text{action}}(t) \equiv R_t + \beta R_{\text{action}}(t) - V_g(s_t)$ is the advantage and $V_g(s_t)$ is a goal-state value baseline regressed toward $R_t + \beta R_{\text{action}}(t)$.

[3] This is typically motivated by *defining* the state distribution $p(s)$ to include the discount factor, **?** but since its introduction is clumsy no matter where you add it, we do so at the end.

4.B Calculating $\nabla_\theta I_{\text{state}}(t)$

We want to evaluate:

$$\nabla_\theta I_{\text{state}} = \sum_g \rho_G(g) \sum_s (\nabla_\theta p(s \mid g)) \log \frac{p(s \mid g)}{p(s)} \tag{4.41}$$

$$+ \sum_g \rho_G(g) \sum_s p(s \mid g) \frac{\nabla_\theta p(s \mid g)}{p(s \mid g)} \tag{4.42}$$

$$- \sum_g \rho_G(g) \sum_s p(s \mid g) \frac{\nabla_\theta p(s)}{p(s)} \tag{4.43}$$

$$\equiv T_1 + T_2 - T_3, \tag{4.44}$$

where we denote the three terms by T_1 , T_2, and T_3 (emphasizing that we are now overriding the definitions T_1 and T_2 in the last section). The calculation of T_1 procedes just as for the same term in the previous section, and again results in supplementing the usual return with an info return, this time over the log state odds:

$$\sum_{u=t+1}^{t+n} \log \frac{p_{\text{emp}}(s_u \mid g)}{p_{\text{emp}}(s_u)}. \tag{4.45}$$

By the same argument, $T_2 = \sum_g p(g) \sum_s \nabla_\theta p(s \mid g)$ simply results in the addition of 1 to the info return at each time step.

Finally, we have the third term. For $\nabla_\theta p(s)$, we can use the same n-step approximation as in eqn 4.20 and just marginalize out the goal, that is:

$$\nabla_\theta \hat{p}_n(s_t) \equiv \sum_{g,\tau(t,n)} \rho_G(g)\, p_{\text{emp}}(s_{t-n} \mid g) \prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u}) \tag{4.46}$$

$$\times \sum_{v=1}^{n} \nabla_\theta \log \pi_g(a_{t-v} \mid s_{t-v})\,. \tag{4.47}$$

Plugging this into T_3 yields:

$$T_3^n = \sum_{g,g^{'},\tau(t,n),s_t} \rho_G(g)\,\rho_G\Big(g^{'}\Big)\,\frac{p_{\text{emp}}(s_t \mid g)}{p_{\text{emp}}(s_t)} p_{\text{emp}}\Big(s_{t-n} \mid g^{'}\Big) \tag{4.48}$$

$$\times \prod_{u=1}^{n} \pi_{g^{'}}(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u}) \sum_{v=1}^{n} \nabla_\theta \log \pi_{g^{'}}(a_{t-v} \mid s_{t-v})\,. \tag{4.49}$$

To turn this into an expectation over trajectories, we multiply and divide by $p_{\text{emp}}(s_{t-n} \mid g)$ and $\prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})$:

$$T_3^n = \sum_{g,\tau(t,n),s_t} \rho_G(g)\, p_{\text{emp}}(s_{t-n} \mid g) \prod_{u=1}^{n} \pi_g(a_{t-u} \mid s_{t-u})\, P(s_{t-u+1} \mid s_{t-u}, a_{t-u}) \tag{4.50}$$

$$\times \sum_{g^{'}} \rho_G\Big(g^{'}\Big)\,\frac{p_{\text{emp}}\big(s_{t-n} \mid g^{'}\big)}{p_{\text{emp}}(s_{t-n} \mid g)} \prod_{u=1}^{n} \frac{\pi_{g^{'}}(a_{t-u} \mid s_{t-u})}{\pi_g(a_{t-u} \mid s_{t-u})} \frac{p_{\text{emp}}(s_t \mid g)}{p_{\text{emp}}(s_t)} \tag{4.51}$$

$$\times \sum_{v=1}^{n} \nabla_\theta \log \pi_{g^{'}}(a_{t-v} \mid s_{t-v}) \tag{4.52}$$

$$= \mathbb{E}_\tau \left[\sum_{g^{'}} \rho_G\Big(g^{'}\Big)\,\frac{p_{\text{emp}}\big(s_{t-n} \mid g^{'}\big)}{p_{\text{emp}}(s_{t-n} \mid g)} \prod_{u=1}^{n} \frac{\pi_{g^{'}}(a_{t-u} \mid s_{t-u})}{\pi_g(a_{t-u} \mid s_{t-u})} \frac{p_{\text{emp}}(s_t \mid g)}{p_{\text{emp}}(s_t)} \sum_{v=1}^{n} \nabla_\theta \log \pi_{g^{'}}(a_{t-v} \mid s_{t-v}) \right]. \tag{4.53}$$

Breaking off the $g^{'} = g$ term, this gives:

$$T_3^n = \mathbb{E}_\tau \left[\rho_G(g)\,\frac{p_{\text{emp}}(s_t \mid g)}{p_{\text{emp}}(s_t)} \sum_{v=1}^{n} \nabla_\theta \log \pi_g(a_{t-v} \mid s_{t-v}) \right] \tag{4.54}$$

$$+ \mathbb{E}_\tau \left[\sum_{g^{'} \neq g} \rho_G\Big(g^{'}\Big)\,\frac{p_{\text{emp}}\big(s_{t-n} \mid g^{'}\big)}{p_{\text{emp}}(s_{t-n} \mid g)} \prod_{u=1}^{n} \frac{\pi_{g^{'}}(a_{t-u} \mid s_{t-u})}{\pi_g(a_{t-u} \mid s_{t-u})} \frac{p_{\text{emp}}(s_t \mid g)}{p_{\text{emp}}(s_t)} \sum_{v=1}^{n} \nabla_\theta \log \pi_{g^{'}}(a_{t-v} \mid s_{t-v}) \right]. \tag{4.55}$$

The $g^{'} = g$ term suggests we should further amend our info return by subtracting the term $p_{\text{emp}}(g \mid s_t) \equiv \rho_G(g) \frac{p_{\text{emp}}(s_t|g)}{p_{\text{emp}}(s_t)}$, that is:

$$R^n_{\text{state}}(t) \equiv \sum_{u=t+1}^{t+n} 1 - p_{\text{emp}}(g \mid s_t) + \log \frac{p_{\text{emp}}(s_u \mid g)}{p_{\text{emp}}(s_u)}. \tag{4.56}$$

Putting it altogether, we have:

$$\nabla_\theta J^n_{\text{state}}(t) = (R_t + \beta R^n_{\text{state}}(t)) \nabla_\theta \log \pi_g(a_t \mid s_t) \tag{4.57}$$

$$- \beta \sum_{g' \neq g} \rho_G\Big(g^{'}\Big) R^n_{\text{cf}}\Big(t, g^{'}\Big) \nabla_\theta \log \pi_{g'}(a_t \mid s_t)\,, \tag{4.58}$$

where we have the included the "counterfactual goal" return:

$$\tilde{R}^n_{\text{cf}}\Big(t, g^{'}\Big) \equiv \sum_{v=t+1}^{t+n} \frac{p_{\text{emp}}\big(s_{v-n} \mid g^{'}\big)}{p_{\text{emp}}(s_{v-n} \mid g)} \prod_{u=1}^{n} \frac{\pi_{g'}(a_{v-u} \mid s_{v-u})}{\pi_g(a_{v-u} \mid s_{v-u})} \frac{p_{\text{emp}}(s_v \mid g)}{p_{\text{emp}}(s_v)}. \tag{4.59}$$

Including the effects of trajectories less than length n and adding a discount factor as in the previous section, we finally amend the state and counterfactual info returns to:

$$R^n_{\text{state}}(t) \equiv \sum_{u=t+1}^{\min(t+n,T)} \gamma_I^{u-t} \left(1 - p_{\text{emp}}(g \mid s_t) + \log \frac{p_{\text{emp}}(s_u \mid g)}{p_{\text{emp}}(s_u)}\right) \tag{4.60}$$

$$R^n_{\text{cf}}\Big(t, g^{'}\Big) \equiv \sum_{v=t+1}^{\min(t+n,T)} \gamma_I^{u-t} \frac{p_{\text{emp}}\big(s_{v-n} \mid g^{'}\big)}{p_{\text{emp}}(s_{v-n} \mid g)} \prod_{u=1}^{n} \frac{\pi_{g'}(a_{v-u} \mid s_{v-u})}{\pi_g(a_{v-u} \mid s_{v-u})} \frac{p_{\text{emp}}(s_v \mid g)}{p_{\text{emp}}(s_v)}. \tag{4.61}$$

In the main text, we drop the n superscripts and include a value function baseline, yielding:

$$\nabla_\theta J_{\text{state}}(t) = A_{\text{state}}(t)\, \nabla_\theta \log \pi_g(a_t \mid s_t) - \beta \sum_{g' \neq g} \rho_G\Big(g'\Big)\, R_{\text{cf}}\Big(t, g'\Big)\, \nabla_\theta \log \pi_{g'}(a_t \mid s_t)\,, \tag{4.62}$$

where $A_{\text{state}}(t) \equiv R_t + \beta R_{\text{state}}(t) - V_g(s_t)$ is the advantage and $V_g(s_t)$ is a goal-state value function regressed toward $R_t + \beta R_{\text{state}}(t)$.

4.C Experimental parameters and details

4.C.1 Simple spatial navigation

In order to allow Bob to integrate information about the goal over time and remember it to guide future actions, we endow Bob with a recurrent neural network (RNN) to process Alice's state-action pairs. We used a gated recurrent unit (GRU) (Cho et al. 2014) to which Alice's state-action pairs are fed as a one-hot vector. We chose to use a scalar core state for the GRU since it was simply tasked with tracking Bob's belief about one of two goals, and could thus assign each goal to a sign of the GRU core state/output, which is what Bob chose to do in practice. The GRU output $z_t = \text{RNN}\big(s_t^{\text{alice}}, a_t^{\text{alice}}\big)$ was then concatenated with a one-hot representation of Bob's own state s_t^{bob} and fed into a fully-connected, feed-forward layer of 128 units with two readout heads: a policy head (a linear layer with $|\mathcal{A}|$ units followed by a softmax, yielding $a_t^{\text{bob}} \sim \pi^{\text{bob}}\big(s_t^{\text{bob}}, z_t\big)$) and a value head (a single linear readout node, yielding $v_t = V^{\text{bob}}\big(s_t^{\text{bob}}, z_t\big)$).

	Alice	**Bob**
training time, in steps	100k	200k
max episode length, in steps	100	100
entropy bonus (logarithmically annealed from/to)	.5, .005	.5, .01
learning rate (Adam)	2.5×10^{-2}	5×10^{-5}
weight on value function regression term	.5	.5
reward discount γ_R	.8	.8
$p(s \mid g)$ approximation depth n (action, state)	0, 1	N/A
information discount γ_I	N/A, 1	N/A

Table 4.1: **Training parameters**.

4.C.2 Key game

The only difference from the previous set of training parameters is that Alice now trains longer (250k instead of 100k steps).

	Alice	**Bob**
training time, in steps	250k	200k
max episode length, in steps	100	100
entropy bonus (logarithmically annealed from/to)	.5, .005	.5, .01
learning rate (Adam)	2.5×10^{-2}	5×10^{-5}
weight on value function regression term	.5	.5
reward discount γ_R	.8	.8
$p(s \mid g)$ approximation depth n (action, state)	0, 1	N/A
information discount γ_I	N/A, 1	N/A

Table 4.2: **Training parameters**.

Chapter 5

Conclusion

5.1 Recap

In this thesis, we have applied the tools of information theory to developing useful algorithms for supervised learning (chapter 2), unsupervised learning (chapter 3), and reinforcement learning (chapter 4). We believe information theory is a natural set of tools for machine learning problems for two primary reasons. First, its core concepts of entropy and mutual information quantify uncertainty and correlation, respectively - two crucial concepts in learning. Second, its chief accomplishments in measuring the limits of compression and communication are directly useful in learning problems. Compression is intimately related to the notion of choosing a minimal model, whereas communication is a key concept in getting artificially intelligent agents to coordinate with one another.

One might wander if the focus on DIB in this thesis is meant as an endorsement of it over IB. In truth, when we originally developed the DIB, we found it difficult to come up with cases in which one would be better off using IB. However, since then, it has become abundantly clear that both are useful in different settings. One influential paper on our point of view is Alemi et al. (2016). In that work, the authors used a

scalable version of IB to train a neural network to categorize images, but to do so with minimal encodings (in an IB sense). This both promoted better generalization to unseen data, as well as resistance to so-called "adversarial attacks", in which one tries to fool a model into miscategorizing data with as small a modification to the input as possible. In the case of these two goals, generalization (i.e. the avoidance of overfitting) and adversarial robustness (i.e. reduced model sensitivity), *information* seems like the natural constraint to add, and therefore IB is an appropriate choice. As argued in chapters 2 and 3, DIB remains an appropriate choice when the *size* of some representation of data is the constraint (such as in clustering), or when a deterministic encoding is preferred.

In the next and final section, we discuss ongoing and future work applying and scaling the tools of information theory to machine learning.

5.2 Future work

A PhD is long, but never long enough to do all of the things. As such, there are many threads of the work discussed above that we are actively pursuing or interested in pursuing soon.

5.2.1 IB and deep learning

Historically, one of the impediments to the practical usage of IB on realistic problems is that one must have full access to the joint distribution $P(X, Y)$. This has restricted IB to be applied only to small datasets where $P(X, Y)$ can be expected to be estimated accurately, or in the jointly gaussian case where an analytic solution is known (Chechik et al. 2005, Palmer et al. 2015). These restrictions also apply to DIB; again, one must have full access to $P(X, Y)$ to proceed.

Recently, however, multiple research groups (Achille and Soatto 2016, Alemi et al. 2016, Chalk et al. 2016) have simultaneously pointed out that one can scale IB using techniques from variational inference to optimize the IB objective directly from *samples from* $P(X, Y)$, rather than the distribution itself, in an approach dubbed the "variational information bottleneck" (VIB) (Alemi et al. 2016). One does so by parameterizing the encoder $q(t \mid x)$ with a neural network and optimizing a bound on the original IB objective. From a neural networks standpoint, this can be viewed as a form of "information dropout" (Achille and Soatto 2016), encouraging the hidden units to encode only as much information as necessary (and no more).

While these techniques were originally applied to IB, the application to DIB is relatively straightforward. By parameterizing the DIB encoder with a neural network and optimizing a variational bound on the DIB objective, one could also scale DIB. Doing so would, for example, allow the application of our approach to clustering (Chapter 3) to much larger datasets. More generally, such a variational DIB (or VDIB) would allow applying the information dropout idea of Achille and Soatto (2016) to neural networks with discrete latents.

5.2.2 Interpolating between IB and DIB

In Chapter 2, we derived the DIB solution by introducing a general cost function that interpolates between the DIB and IB cost functions, solving to optimize that general cost function, and then taking the limit of the solution towards the DIB objective. The introduction of this generalized IB cost function was thus simply a tool to obtain the DIB solution. However, the generalized IB solution is also a well-defined algorithm in its own right, one that allows for interpolating between the soft clustering of IB and the hard clustering of DIB. We have yet to thoroughly explore applications of this interpolated method, but we speculate that it might, for example, be useful

when one actually wants to use the DIB, but due to uncertainty caused by limited data, one needs to "soften" the clustering assignments.

5.2.3 Improvements to information sharing in MARL

In Chapter 4, we introduced a method for encouraging agents to share/hide information with other agents in order to cooperate/compete in a multi-agent reinforcement learning (MARL) setting. This is an ongoing project and we are actively working on generalizing the method to make it applicable in a wider variety of settings.

In the derivation of our Monte Carlo mutual information gradient estimators (sections 4.A and 4.B), we employed an n-step approximation to $p(s \mid g)$ in order to calculate the gradient with respect to the policy parameters. After the submission of this work, we noticed that a simpler derivation was possible that removed the need for this approximation. The resulting algorithms are the same as the ones introduced in Chapter 4, but with n set to the length of the episode, thus removing one free parameter.

Our approach as presented requires Alice to know in advance whether she wishes to cooperate or compete with Bob. However, in many realistic settings, one might not know whether a new agent is a friend or a foe before interacting with them. One way to generalize our approach to this setting is have Alice learn a parameterized family of policies that include both cooperative and competitive ones. To do this, our approach could be modified as follows: on each episode, a value of β (the information regularizer weight) is chosen randomly a prior distribution $p(\beta)$. On that episode, Alice is trained to optimize her objective with that particular value of β. In addition, she receives as input to her policy this value of β in a new channel alongside her state and the goal. Over time, she learns to associate different values of β with different objectives. Then, when Bob is introduced, her policy network is frozen, her objective is replaced with a joint one describing her cooperative / competitive interactions with Bob (e.g. the

sum / difference of their individual rewards), and she now receives control of β as a parameter to optimize. By optimizing the joint reward over β, Alice chooses whether to cooperate or compete online. Moreover, since she only has a single parameter to optimize, she would learn far faster than if her entire policy was learned from scratch through interaction with Bob.

We framed our approach as useful when we'd like to finish training Alice *before* interacting with Bob. However, there is a continuum of situations between this, and training with Bob in the loop in which our approach might also be useful. For example, it might be useful to pre-train with information regularization, and then fine tune Alice's policy through interaction with Bob. In many situations, this would greatly speed up convergence to a good policy. Moreover, even without pre-training, it might be useful to include information regularization in a joint reward with Bob in order to help "shape" the reward and promote cooperation / competition more quickly.

Finally, we tested our approach in simple grid world settings, where states, actions, and goals are discrete. In many realistic settings, however, any of these might be continuous, thus generalizing to these situations is a priority. There are 3 modifications needed to deal with these scenarios: 1) the replacement of Alice's tabular policy with a parameterized one (e.g. with a neural network), 2) forming a variational approximation to the "base policy" $\tilde{\pi}_0 \approx \pi_0 \equiv \sum_g p(g)\, \pi_g$, an d 3) choosing a parameterization of π_g and $\tilde{\pi}_0$ such that $\mathrm{KL}[\pi_g \mid \pi_0]$ is analytic (or at least tractable to approximate). Working with continuous states requires modification 1, continuous goals 1 and 2, and continuous actions all 3.

5.2.4 Synergy and MARL

Imagine you are a preschool teacher designing a class activity to promote your students to communicate and work together. If you give them an activity in which the task can be best completed alone, they have no incentive to work together. An

ideal activity is one that *requires* them to share information with one another and coordinate their actions in order to perform most effectively. If one could quantify the extent to which a task requires this type of information-sharing, then one could directly optimize the task for promoting teamwork.

Information theory provides just such a measure - synergy . The synergy between N variables $\mathbf{x} = \{x_i\}_{i=1}^{N}$ and y is defined as: $\text{Syn}(\mathbf{x}; y) = I(\mathbf{x}, y) - \sum_i I(x_i, y)$. In other words, synergy measures how much more information the collective group of variables $\mathbf{x}$ contain about y than the sum of the individual informations that each x_i has about y. Note that synergy is not necessarily symmetric: $\text{Syn}(x; y) \neq \text{Syn}(y; x)$. Synergy also can be positive (the x_is combine synergistically), negative (the x_is are redundant), or zero (the x_is are additive).

In our above example, imagine that there are N children each given a message x_i, and they must collectively infer the identity y of an object in the classroom. For narrowing down the number of potential objects to n, the students receive reward equal to $\log(n)$. In other words, they are rewarded for the reduction in entropy over the identity of the object (assuming uniform uncertainty over the original and remaining objects). Points are divided evenly among the students who together put forth an answer, and multiple groups can put forth an answer (points are divided within a team, but not between them). Thus, for group i with m_i students and a reduction to n_i objects, each student in group i receives $\frac{\log n_i}{m_i}$ points. If the all of the messages are the same (i.e. $x_i = x_j$ for all i, j), then there is no incentive to cooperate - the students do better guessing on their own. If the messages are different but combine additively ($\text{Syn}(\mathbf{x}; y) = 0$), then the students should be agnostic between cooperating or not. But if the messages combine synergistically ($\text{Syn}(\mathbf{x}; y) > 0$), then the students will do better to cooperate. Thus, as the teacher, one can quantitatively optimize the messages given to the students to promote cooperation.

We are currently working out the details of the task structure to test this idea, but outline it here for its connection to the content of Chapter 4.

Bibliography

Abbeel, P. and Ng, A. Y.: 2004, Apprenticeship learning via inverse reinforcement learning, *Proceedings of the Twenty-First International Conference on Machine learning*, ACM.

Achille, A. and Soatto, S.: 2016, Information Dropout: Learning Optimal Representations Through Noisy Computation, *ArXiv e-prints* .

Alemi, A. A., Fischer, I., Dillon, J. V. and Murphy, K.: 2016, Deep variational information bottleneck, *CoRR* **abs/1612.00410**.
URL: *http://arxiv.org/abs/1612.00410*

Atick, J. J. and Redlich, A. N.: 1992, What Does the Retina Know about Natural Scenes?, *Neural Computation* **4**(2), 196–210.

Bach, F. R. and Jordan, M. I.: 2006, A Probabilistic Interpretation of Canonical Correlation Analysis, *Technical report*.

Baker, C. L., Saxe, R. and Tenenbaum, J. B.: 2009, Action understanding as inverse planning, *Cognition* **113**(3), 329–349.

Barlow, H. B.: 1981, Critical Limiting Factors in the Design of the Eye and Visual Cortex, *Proceedings of the Royal Society of London B: Biological Sciences* **212**(1186), 1–34.

Barlow, H. B.: 2001a, Redundancy reduction revisited., *Network Computation in Neural Systems* **12**(3), 241–253.

Barlow, H. B.: 2001b, The exploitation of regularities in the environment by the brain., *Behavioral and Brain Sciences* **24**(4), 602–607.

Bekkerman, R., El-Yaniv, R., Tishby, N. and Winter, Y.: 2001, On feature distributional clustering for text categorization, *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 146–153.

Bekkerman, R., El-Yaniv, R., Tishby, N. and Winter, Y.: 2003, Distributional word clusters vs. words for text categorization, *Journal of Machine Learning Research* **3**, 1183–1208.

Bishop, C. M.: 2006, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg.

Blei, D. M., Griffiths, T. L., Jordan, M. I. and Tenenbaum, J. B.: 2004, Hierarchical Topic Models and the Nested Chinese Restaurant Process, *Advances in Neural Information Processing Systems* pp. 17–24.

Chalk, M., Marre, O. and Tkacik, G.: 2016, Relevant sparse codes with variational information bottleneck, *ArXiv e-prints* .

Chechik, G., Globerson, A., Tishby, N. and Weiss, Y.: 2005, Information Bottleneck for Gaussian Variables, *Journal of Machine Learning Research* **6**, 165–188.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y.: 2014, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, *ArXiv e-prints* .

Cover, T. M. and Thomas, J. A.: 2006, *Elements of Information Theory*, Wiley-Interscience.

Creutzig, F., Globerson, A. and Tishby, N.: 2009, Past-future information bottleneck in dynamical systems, *Physical Review E* **79**(4), 041925–5.

Dragan, A. D., Lee, K. C. and Srinivasa, S. S.: 2013, Legibility and predictability of robot motion, *International Conference on Human-Robot Interaction (HRI)* pp. 301–308.
URL: *http://dl.acm.org/citation.cfm?id=2447556.2447672*

Eysenbach, B., Gupta, A., Ibarz, J. and Levine, S.: 2018, Diversity is All You Need: Learning Skills without a Reward Function, *ArXiv e-prints* .

Hadfield-Menell, D., Russell, S. J., Abbeel, P. and Dragan, A.: 2016, Cooperative inverse reinforcement learning, *Advances in neural information processing systems*, pp. 3909–3917.

Hayden, P., Headrick, M. and Maloney, A.: 2013, Holographic Mutual Information is Monogamous, *Phys. Rev.* **D87**(4), 046003.

Hecht, R. M., Noor, E. and Tishby, N.: 2009, Speaker recognition by Gaussian information bottleneck., *Proceedings of the Annual Conference of the International Speech Communication Association.*

Hecht, R. M. and Tishby, N.: 2005, Extraction of relevant speech features using the information bottleneck method, *Proceedings of the Annual Conference of the International Speech Communication Association.*

Ho, M. K., Littman, M., MacGlashan, J., Cushman, F. and Austerweil, J. L.: 2016, Showing versus doing: Teaching by demonstration, *Advances In Neural Information Processing Systems*, pp. 3027–3035.

Hosur, P., Qi, X.-L., Roberts, D. A. and Yoshida, B.: 2016, Chaos in quantum channels, *JHEP* **02**, 004.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D. and Kavukcuoglu, K.: 2016, Reinforcement Learning with Unsupervised Auxiliary Tasks.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A. and Hudspeth, A. J.: 2013, *Principles of neural science*, 5th edn, McGraw-Hill, Health Professions Division, New York.

Kingma, D. P. and Welling, M.: 2013, Auto-Encoding Variational Bayes, *ArXiv e-prints* .

Kinney, J. B. and Atwal, G. S.: 2014, Equitability, mutual information, and the maximal information coefficient, *Proceedings of the National Academy of Sciences* **111**(9), 3354–3359.

Kleiman-Weiner, M., Ho, M. K., Austerweil, J. L., Littman, M. L. and Tenenbaum, J. B.: 2016, Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction, *Proceedings of the 38th Annual Conference of the Cognitive Science Society*.

LeCun, Y.: 2016, Predictive learning. NIPS.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J. and Graepel, T.: 2017, Multi-agent reinforcement learning in sequential social dilemmas, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, pp. 464–473.

Littman, M. L.: 1994, Markov games as a framework for multi-agent reinforcement learning, *Machine Learning Proceedings 1994*, Elsevier, pp. 157–163.

Littman, M. L.: 2001, Friend-or-foe q-learning in general-sum games, *ICML*, Vol. 1, pp. 322–328.

MacKay, D. J. C.: 2002, Information theory, inference, and learning algorithms.

Mnih, V., Puigdomènech Badia, A., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D. and Kavukcuoglu, K.: 2016, Asynchronous Methods for Deep Reinforcement Learning, *ArXiv e-prints* .

Murphy, K. P.: 2012, *Machine Learning: A Probabilistic Perspective*, The MIT Press.

Ng, A. Y., Russell, S. J. et al.: 2000, Algorithms for inverse reinforcement learning., *Icml*, pp. 663–670.

Olshausen, B. A. and Field, D. J.: 1996, Emergence of simple-cell receptive field properties by learning a sparse code for natural images., *Nature* **381**(6583), 607–609.

Olshausen, B. A. and Field, D. J.: 1997, Sparse coding with an overcomplete basis set: A strategy employed by V1?, *Vision research* **37**(23), 3311–3325.

Olshausen, B. A. and Field, D. J.: 2004, Sparse coding of sensory inputs, *Current Opinion in Neurobiology* **14**(4), 481–487.

Ostrovski, G., Bellemare, M. G., van den Oord, A. and Munos, R.: 2017, Count-Based Exploration with Neural Density Models, *ArXiv e-prints* .

Page, D. N.: 1993, Average entropy of a subsystem, *Phys. Rev. Lett.* **71**, 1291–1294.

Palmer, S. E., Marre, O., Berry II, M. J. and Bialek, W.: 2015, Predictive information in a sensory population, *Proceedings of the National Academy of Sciences* **112**(22), 6908–6913.

Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S. M. A. and Botvinick, M.: 2018, Machine Theory of Mind, *ArXiv e-prints* .

Ravid, S. and Tishby, N.: 2017, Opening the black box of deep neural networks via information, *arXiv.org* **cs.LG**.

Rubin, J., Ulanovsky, N., Nelken, I. and Tishby, N.: 2016, The representation of prediction error in auditory cortex, *PLoS computational biology* **12**(8), e1005058–28.

Schneidman, E., Slonim, N., Tishby, N., de Ruyter van Steveninck, R. R. and Bialek, W.: 2001, Analyzing neural codes using the information bottleneck method, *Advances in Neural Information Processing Systems* .

Shafto, P., Goodman, N. D. and Griffiths, T. L.: 2014, A rational account of pedagogical reasoning: Teaching by, and learning from, examples, *Cognitive psychology* **71**, 55–89.

Shamir, O., Sabato, S. and Tishby, N.: 2010, Learning and Generalization with the Information Bottleneck, *Theoretical Computer Science* **411**(29-30), 2696–2711.

Shannon, C. E.: 1948, A mathematical theory of communication, *The Bell System Technical Journal* **27**(3), 379–423.
URL: *https://ieeexplore.ieee.org/document/6773024/*

Shenker, S. H. and Stanford, D.: 2014, Black holes and the butterfly effect, *JHEP* **03**, 067.

Shillingford, B., Assael, Y., Hoffman, M. W., Paine, T., Hughes, C., Prabhu, U., Liao, H., Sak, H., Rao, K., Bennett, L., Mulville, M., Coppin, B., Laurie, B., Senior, A. and de Freitas, N.: 2018, Large-Scale Visual Speech Recognition, *ArXiv e-prints* .

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M.,

Kavukcuoglu, K., Graepel, T. and Hassabis, D.: 2016, Mastering the game of Go with deep neural networks and tree search, *Nature* **529**(7587), 484–489.

Simoncelli, E. P. and Olshausen, B. A.: 2001, Natural image statistics and neural representation., *Annual Review of Neuroscience* **24**(1), 1193–1216.

Slonim, N., Atwal, G., Tkacik, G. and Bialek, W.: 2005, Information-based clustering., *Proceedings of the National Academy of Sciences* **102**(51), 18297–18302.

Slonim, N. and Tishby, N.: 2000a, Agglomerative Information Bottleneck, *Advances in Neural Information Processing Systems* pp. 617–623.

Slonim, N. and Tishby, N.: 2000b, Document clustering using word clusters via the information bottleneck method, *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 208–215.

Slonim, N. and Tishby, N.: 2001, The Power of Word Clusters for Text Classification, *European Colloquium on Information Retrieval Research*.

Slonim, N. and Weiss, Y.: 2002, Maximum Likelihood and the Information Bottleneck, *Advances in Neural Information Processing Systems* .

Still, S. and Bialek, W.: 2004, How many clusters an information-theoretic perspective, *Neural Computation* **16**(12), 2483–2506.

Still, S., Bialek, W. and Bottou, L.: 2003, Geometric clustering using the information bottleneck method, *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, MIT Press, Cambridge, MA, USA, pp. 1165–1172.

URL: *http://dl.acm.org/citation.cfm?id=2981345.2981489*

Still, S., Crutchfield, J. P. and Ellison, C. J.: 2010, Optimal causal inference: Estimating stored information and approximating causal architecture, *Chaos* **20**.

Strouse, D. and Schwab, D. J.: 2017, The Deterministic Information Bottleneck, *Neural Computation* .

Strouse, D. and Schwab, D. J.: 2018, The Information Bottleneck And Geometric Clustering, *Neural Computation* .

Strouse, D., Schwab, D. J., Botvinick, M., Tenenbaum, J. T. and Kleiman-Weiner, M.: 2018, Learning to Share and Hide Intentions using Information Regularization, *NIPS* .

Sutton, R. S., McAllester, D., Singh, S. and Mansour, Y.: 1999, Policy gradient methods for reinforcement learning with function approximation, *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, MIT Press, Cambridge, MA, USA, pp. 1057–1063.
URL: *http://dl.acm.org/citation.cfm?id=3009657.3009806*

Tishby, N., Pereira, F. C. and Bialek, W.: 1999, The information bottleneck method, *Proceedings of The 37th Allerton Conference on Communication, Control, and Computing* pp. 368–377.

Tishby, N. and Zaslavsky, N.: 2015, Deep Learning and the Information Bottleneck Principle, *arXiv.org* .

Tomasello, M., Carpenter, M., Call, J., Behne, T. and Moll, H.: 2005, Understanding and sharing intentions: The origins of cultural cognition, *Behavioral and brain sciences* **28**(05), 675–691.

Turner, R. E. and Sahani, M.: 2007, A maximum-likelihood interpretation for slow feature analysis., *Neural Computation* **19**(4), 1022–1038.

Ullman, T., Baker, C., Macindoe, O., Evans, O., Goodman, N. and Tenenbaum, J. B.: 2009, Help or hinder: Bayesian models of social goal inference, *Advances in neural information processing systems*, pp. 1874–1882.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K.: 2016, WaveNet: A Generative Model for Raw Audio, *ArXiv e-prints* .

Wallace, G. K.: 1991, The JPEG still picture compression standard, *Commun. ACM* **34**(4), 30–44.

Whye Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N. and Pascanu, R.: 2017, Distral: Robust Multitask Reinforcement Learning, *ArXiv e-prints* .