Representation and Transfer Learning Using Information-Theoretic Approximations

by

David Qiu

B.S. Electrical Engineering and Computer Science
University of California, Berkeley, 2010

S.M. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2017

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science
at the Massachusetts Institute of Technology
May 2020

Signature of Author: _____

David Qiu
Department of Electrical Engineering and Computer Science
May 15, 2020

Certified by: _____

Lizhong Zheng
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: _____

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Representation and Transfer Learning Using Information-Theoretic Approximations

by David Qiu

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2020, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Learning informative and transferable feature representations is a key aspect of machine learning systems. Mutual information and Kullback-Leibler divergence are principled and very popular metrics to measure feature relevance and perform distribution matching, respectively. However, clean formulations of machine learning algorithms based on these information-theoretic quantities typically require density estimation, which could be difficult for high dimensional problems. A central theme of this thesis is to translate these formulations into simpler forms that are more amenable to limited data. In particular, we modify local approximations and variational approximations of information-theoretic quantities to propose algorithms for unsupervised and transfer learning. Experiments show that the representations learned by our algorithms perform competitively compared to popular methods that require higher complexity.

Thesis Supervisor: Lizhong Zheng
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

This thesis would not have been possible without the impact of so many people.

To start, I want to thank my family for their unwavering love. My road to the PhD is built upon their nurture: they crafted toys and puzzles that cultivated my interest in science and engineering at an early age. It is built upon their sacrifices: they devoted all of their resources to make sure I have the best possible opportunities to succeed. It is built upon their patience: they stood by me through all of my successes and failures over 10 years of higher education. It is through them that I learned to strive toward becoming a better person each day.

I want to thank my advisor, Professor Lizhong Zheng, for his wonderful guidance throughout graduate school. I am one of his first students to focus primarily on machine learning. He gave me considerable freedom to shape the direction of my research in this unfamiliar field. At the same time, he challenged me to never compromise on my development as a researcher and to not merely follow the crowd. This thesis includes many ideas that took shape after discussions with Lizhong and it has been an honor to be his student.

I want to thank my committee members, Professor Gregory Wornell and Professor Guy Bresler, for their valuable time and suggestions. The depth of their experience and the breadth of their knowledge have given many insights that allowed me to formulate new problems and improve this thesis.

I want to thank our lab and collaborators, Anuran, Fabián, Jiejun, Mina, Mohamed, Molly, Joshua, Shao-Lun, Yuguo, and Yuheng, for making every meeting and discussion an enjoyable and stimulating experience.

Finally, I want to thank all of my friends and colleagues from before graduate school and at MIT. I cannot overstate the impact they have had on me. I am truly blessed and grateful to have them in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Learning representations of data is a central theme of machine learning (ML) systems. In addition to accuracy, a major advantage of artificial neural networks is in their ability to learn hierarchical sets of features in their intermediate layers. For example, in convolutional neural networks (CNN) for computer vision, the earlier layers detect for edges while the later layers detect for higher level concepts such as eyes and ears [101]. Parts of the network can then be used as general feature extractors for feature alignment, visualization, transfer learning to other tasks, etc.

However, modern deep neural networks (DNN) have parameter counts on the order of millions [41] to billions [87]. Accordingly, they require a commensurate amount of labeled training data to achieve state of the art performance. The difficulty and expense associated with acquiring manually labeled data have led to increasing interest in unsupervised (e.g., clustering [62], generative adversarial networks [32], infomax [60]), transfer (e.g., parameter transfer [100], domain adaptation [30]), and self-supervised learning (e.g., embedding [71], predictive learning [79]). In these three paradigms, mutual information and the Kullback-Leibler (KL) divergence [18] often serve as principled metrics to guide representation learning, feature selection, and distribution matching.

One of the most prominent drawbacks of using these information-theoretic quantities is

Table 1.1: The rapid increases in the number of parameters in popular deep neural network architectures since year 2012.

Year	Architecture	Parameters
2012	AlexNet	62M
2014	VGGNet	138M
2018	BERT-Large	340M
2019	GPT-2	1.5B
2020	T-NLG	17B

that they are functions of the full distribution. To accurately compute them involve expensive operations such as marginalization and kernel density estimation, which do not scale well in term of either time or sample complexity to be used on today's high dimensional problems. However, ML algorithms do not need to rely on the exact value of these quantities. Approximations would suffice as long as they provide adequate signals to update model parameters. This led to the development of different information-theoretic approximations, two of which are local approximations [44] and variational approximations [78].

Local approximations exploit the fact that when constrained around a local neighborhood of distributions, KL divergence behaves as a squared Euclidean norm. This transforms many difficult information geometric operations into well known linear algebraic operations. Variational approximations write statistical divergence as the result of a maximization problem. The objective functions of these maximizations can be computed directly from empirical estimates without requiring density estimation. Hence, maximizing divergence is still a maximization problem using empirical data while minimizing divergence becomes a min-max problem.

This thesis builds on those efforts by formulating problems such as probabilistic cluster-

ing using matrix norm couplings, task personalization, and the subset Wasserstein coupling. It also derives useful information-theoretic approximation techniques, such as one-shot feature set reduction, non-zero-sum GAN, and the subset Wasserstein GAN. Section 1.3 starts by deriving a local approximation of KL divergence based on Taylor series that we use throughout Chapters 2 and 3.

■ 1.2 Notation

General Notation:

$\mathbb{R}$	set of real numbers
$\mathbb{C}$	set of complex numbers
$\times$	Cartesian product
exp	exponential with base e
log	logarithm with base e
$f(\epsilon) = o(g(\epsilon))$	$\lim_{\epsilon \to 0} \frac{f(\epsilon)}{g(\epsilon)} = 0$

Probability and Random Variables:

pmf	probability mass function
P, Q, R, U	distributions
Bernoulli(p)	Bernoulli distribution with parameter p
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ
$\mathcal{CN}(\mu, \Gamma)$	complex Gaussian distribution with mean μ and covariance Γ
$\mathcal{P}_{\mathcal{X}}$	probability simplex over alphabet $\mathcal{X}$
$\mathcal{P}_{\mathcal{X}}^{\circ}$	relative interior of $\mathcal{P}_{\mathcal{X}}$
Capital X, Y, Z	random variables
Calligraphic $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$	the non-empty sets of possible outcomes (alphabet) for the respective random variables
Lower case x, y, z	particular outcomes of random variables
$\mathbb{E}$	expectation
$X \overset{\mathrm{d}}{=} Y$	X and Y have identical distributions

Matrices:

Boldface $\mathbf{0}, \mathbf{1}$	column vector of all 0's or 1's of appropriate dimensions
$[\,\cdot\,]$	diagonal matrix representation of a vector
$\mathbb{R}^{m\times n}$	set of real $m \times n$ matrices
$\mathcal{P}_{\mathcal{Y}\mid\mathcal{X}}$	set of $\lvert\mathcal{Y}\rvert \times \lvert\mathcal{X}\rvert$ column stochastic matrices
$\mathrm{tr}(\cdot)$	trace of a matrix
$\sigma_i(A)$	ith largest singular values of A
$\lVert A\rVert_p \triangleq \left(\sum_{i=1}^{\min(m,n)} \sigma_i(A)^p\right)^{\frac{1}{p}}$	Schatten ℓ^p-norm of matrix A
$\lVert A\rVert_F = \lVert A\rVert_2$	Frobenius norm of matrix A
$\lVert A\rVert_\star = \lVert A\rVert_1$	nuclear norm of matrix A

Statistics and Machine Learning:

$\mathcal{L}$	loss function in machine learning optimizations
$\hat{\phi}$	empirical version of some probabilistic object ϕ (e.g., random variable, distribution)
$H(X)$	Shannon entropy of random variable X
$I(X,Y)$	Shannon mutual information between random variables X and Y
$D(P \parallel Q)$	Kullback-Leibler (KL) divergence between distributions P and Q
$D_f(P \parallel Q)$	f-divergence between distributions P and Q
$W_1(P,Q)$	Wasserstein-1 distance between distributions P and Q

■ 1.3 Local Approximation of KL Divergence

Definition 1.3.1 (Information Vector)**.** For some fixed reference distribution $R \in \mathcal{P}_{\mathcal{X}}^{\circ}$, we can define any other distribution $P \in \mathcal{P}_{\mathcal{X}}$ as a perturbation around R:

$$P = R + \epsilon\sqrt{R}\,\phi, \tag{1.1}$$

where we call $\phi \in \mathbb{R}^{|\mathcal{X}|}$ the *information vector*.

Remark 1.3.1. To satisfy the first axiom of probability, $\epsilon > 0$ must be a scalar that is small enough such that $P \geq 0$ everywhere. To satisfy the second axiom of probability, $\left\langle \sqrt{R}, \phi \right\rangle = 0$.

In this thesis, we make extensive use of the following approximation to simplify KL divergence for learning algorithms.

Theorem 1.3.1 (Local Approximation of KL Divergence)**.** For distributions P and Q with support on the same alphabet $\mathcal{X}$, and their respective information vectors ϕ and ψ:

$$D(P \parallel Q) = \frac{1}{2}\epsilon^2 \left\|\phi - \psi\right\|_2^2 + o(\epsilon^2), \tag{1.2}$$

where $o(\epsilon^2)$ represents a function satisfying $\lim_{\epsilon\to 0} o(\epsilon^2)/\epsilon^2 = 0$.

Proof. For clarity, we show the proof for the case where $\mathcal{X}$ is a discrete set. It is straightforward to extend to continuous versions. P and Q can be written as

$$\begin{aligned} P &= R + \epsilon\sqrt{R}\,\phi \\ Q &= R + \epsilon\sqrt{R}\,\psi = P + \epsilon\sqrt{R}\,(\psi - \phi) \end{aligned} \tag{1.3}$$

for some reference distribution R on the same support. Then, KL divergence can be written

as

$$
\begin{aligned}
D(P \parallel Q) &= -\sum_{x\in\mathcal{X}} P(x)\log\frac{Q(x)}{P(x)} \\
&= -\sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)+\epsilon\sqrt{R(x)}\,(\psi(x)-\phi(x))}{P(x)} \\
&= -\sum_{x\in\mathcal{X}} P(x)\log\left(1+\frac{\epsilon\sqrt{R(x)}\,(\psi(x)-\phi(x))}{P(x)}\right) \\
&= -\sum_{x\in\mathcal{X}} P(x)\frac{\epsilon\sqrt{R(x)}\,(\psi(x)-\phi(x))}{P(x)} + \frac{1}{2}\sum_{x\in\mathcal{X}} P(x)\left[\frac{\epsilon\sqrt{R(x)}\,(\psi(x)-\phi(x))}{P(x)}\right]^2 + o(\epsilon^2) \\
&= -\sum_{x\in\mathcal{X}} [Q(x)-P(x)] + \frac{1}{2}\sum_{x\in\mathcal{X}} \frac{\epsilon^2 R(x)\,(\psi(x)-\phi(x))^2}{P(x)} + o(\epsilon^2) \\
&= \frac{1}{2}\sum_{x\in\mathcal{X}} \frac{\epsilon^2 R(x)\,(\psi(x)-\phi(x))^2}{P(x)} + o(\epsilon^2) \\
&= \frac{1}{2}\sum_{x\in\mathcal{X}} \frac{\epsilon^2 R(x)\,(\psi(x)-\phi(x))^2}{R(x)+\epsilon\sqrt{R(x)}\cdot\phi} + o(\epsilon^2) \\
&= \frac{1}{2}\sum_{x\in\mathcal{X}} \epsilon^2\,(\psi(x)-\phi(x))^2 - \frac{\epsilon^3\sqrt{R(x)}^3\phi(x)\,(\psi(x)-\phi(x))^2}{R(x)+\epsilon\sqrt{R(x)}\cdot\phi(x)} + o(\epsilon^2) \\
&= \frac{1}{2}\epsilon^2\,\|\phi-\psi\|_2^2 + o(\epsilon^2)
\end{aligned}
$$

where the second, fifth, and seventh equality use (1.3) and the fourth equality uses the Maclaurin series $\log(1+u) = u - \frac{1}{2}u^2 + o(u^2)$. ■

Corollary 1.3.1 (Local Symmetry of KL Divergence)**.** Under the same local conditions as Theorem 1.3.1, KL divergence is symmetric:

$$D(P \parallel Q) = D(Q \parallel P) + o(\epsilon^2). \tag{1.4}$$

1.4 Outline

Chapter 2 uses the local version of mutual information to develop techniques for embedding and clustering on joint probability distributions. Chapter 3 introduces task personalization (a type of transfer learning) and exploits (1.2)'s linear algebraic structure to derive low complexity algorithms for selecting task-specific features. Chapter 4 extends existing training objectives in generative adversarial networks (GAN) and self-supervised learning; it also examines using generative models to capture interference signal constellation to aid wireless communication. Chapter 5 concludes this thesis and discusses future directions.

Chapter 2

Probabilistic Clustering Using Maximal Matrix Norm Couplings

Clustering is one of many important techniques in unsupervised learning that finds structure in unlabeled data. One important class of clustering algorithms is metric based, where each row of the data matrix corresponds to an item's feature vector representation. The most well known example of metric based clustering is k-means clustering (also known as the Lloyd-Max algorithm [62, 66]).

In this chapter, we instead focus on probabilistic clustering, where the data matrix is viewed as the joint co-occurrences (or affinities) between two discrete sets $\mathcal{X}$ and $\mathcal{Y}$ of items and users, respectively. The co-occurrence matrix can be normalized to sum to 1 to represent a joint probability matrix. Much like [22], we want to maximize the "cluster-to-item" mutual information over the set of "user-to-cluster" assignment matrices. Our main contributions include relaxing this mutual information optimization into a Frobenius norm optimization over "DTM" matrices (to be defined later), relating such matrices to graph Laplacians from spectral graph theory, and proposing an alternating maximization algorithm to approximately solve this matrix optimization. Moreover, unlike spectral methods, we directly learn a transition kernel for soft clustering as opposed to following the usual two-step procedure of learning an embedding and then applying k-means clustering.

Note that there are preprocessing techniques to convert a metric based data matrix into

an affinity matrix (e.g., via the heat kernel) that popular manifold learning algorithms such as t-SNE [63] employ. Thus, the techniques presented in this chapter can also be applied to datasets that are more naturally metric based, though we do not study cases with the aforementioned preprocessing.

This chapter is organized as follows: Section 2.1 formulates the mutual information criterion for probabilistic clustering. Section 2.2 reviews the related literature. Section 2.3 defines the divergence transition matrix and derives the relationship between its Frobenius norm and mutual information. Section 2.4 discusses the Frobenius maximization problem for probabilistic clustering and analyzes its convexity and complexity. Section 2.5 relaxes the optimization problem and presents two algorithms based on gradient ascent and alternating maximization, respectively. Section 2.6 presents some experimental results that validate our model.

■ 2.1 Information-Theoretic Motivation

Suppose we are given training data $(x_1, y_1), \ldots, (x_n, y_n)$ that is drawn i.i.d. from a joint pmf $P_{Y,X} \in \mathcal{P}_{\mathcal{Y}\times\mathcal{X}}$ such that $P_Y \in \mathcal{P}_{\mathcal{Y}}^{\circ}$ and $P_X \in \mathcal{P}_{\mathcal{X}}^{\circ}$. Our goal is to perform clustering on $\mathcal{Y}$ by learning the transition probability kernel $P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}$, where $\mathcal{Z}$ is the set of cluster labels with $|\mathcal{Z}| \ll |\mathcal{Y}|$, and $P_{Z|Y=y} \in \mathcal{P}_{\mathcal{Z}}$ represents a soft assignment of $y \in \mathcal{Y}$. Since our training data is "unlabeled", we assume that $X \to Y \to Z$ form a Markov chain to extract information about the clusters from our training data. From hereon, we assume that $P_{Y,X}$ is known as it can be empirically estimated from the data, and $P_Z \in \mathcal{P}_{\mathcal{Z}}^{\circ}$ is known from some prior domain knowledge. For example, when clustering readers of political blogs, $\mathcal{X}$ is the set of blogs, $\mathcal{Y}$ is the set of readers, and P_Z can be set using priors on the distribution of liberals and conservatives in the country.

The following information-theoretic problem can be used to perform probabilistic clus-

tering:

$$\sup_{P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}:\ P_{Z|Y} P_Y = P_Z} I(X;Z), \tag{2.1}$$

where $P_{X,Y}$ and P_Z are fixed, $X \to Y \to Z$ form a Markov chain, and $I(X;Z)$ denotes the mutual information between X and Z (see [18, Section 2.3] for a definition). In the sections that follow, we will refer to $P_{Z|Y} P_Y = P_Z$ as the *constraint on the marginal*. Intuitively, the formulation in (2.1) finds soft clusters by maximizing $I(X;Z)$ and thereby exploiting the information that X contains about Y. Note that $I(X;Z) \leq I(X;Y)$ by the data processing inequality [18, Section 2.8], but $P_{Z|Y} = I_{|\mathcal{Z}|}$ (which denotes the $|\mathcal{Z}| \times |\mathcal{Z}|$ identity matrix) is not a valid solution because $|\mathcal{Z}| \ll |\mathcal{Y}|$.

■ 2.2 Related Work

The paradigm of clustering joint probability distributions is very useful for graphs, networks, and genomics, where items co-occur without explicit labels. Information-theoretic co-clustering [22] alternates between searching for a cluster assignment that minimizes the loss in mutual information and recomputing the induced cluster distributions. [77] exploits properties of the graph Laplacian to search for the closest co-occurence matrix (in term of Frobenius norm) that has the desired number of connected components. [53] uses optimal transport coupling (Wasserstein and Gromov-Wasserstein distance [67, 98]) as the input to a jump detection algorithm to co-cluster. Figure 2.1 shows a visualization the desired output of any co-clustering algorithms: a block structure after rearranging by clusters.

It is also worth mentioning that the general idea of maximizing mutual information down a Markov chain is related to the *information bottleneck method* developed in [96] (which is useful for lossy source compression and clustering), as well as the *linear information coupling problem* introduced in [42] (which provides intuition about network information theory problems).

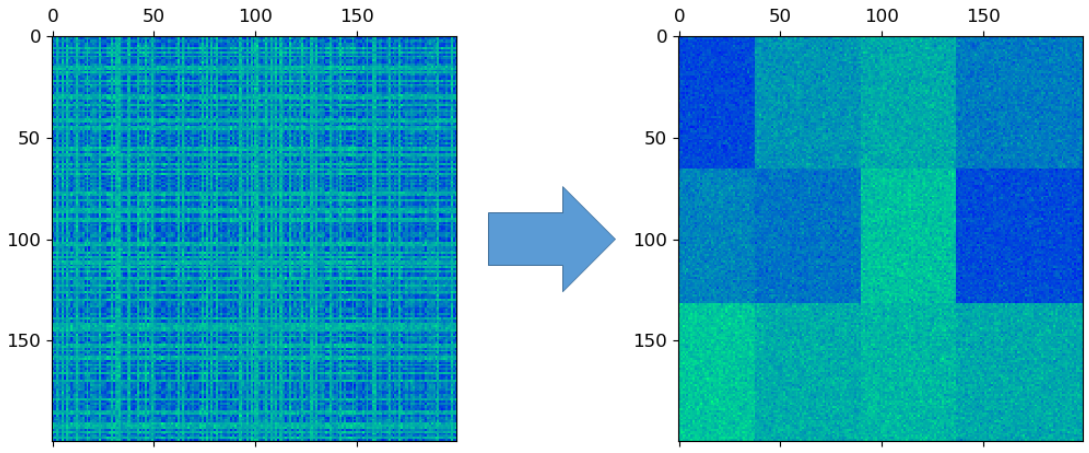

Figure 2.1: Visualization of co-clustering. The algorithm takes a joint probability distribution as input and learns functions that assign items into clusters. Permuting the items according to the assignments yields the underlying block structure.

2.3 Mutual Information Approximation

2.3.1 Local Approximation

We want to use Theorem 1.3.1 to transform (2.1) into a simpler Frobenius norm maximization problem. To do that, we need to write $P_{Z|Y}$ and $P_{Z|X}$ into their information vector forms (1.1):

$$\forall y \in \mathcal{Y}, \;\; P_{Z|Y=y} = P_Z + \epsilon\sqrt{P_Z}\,\psi_y, \tag{2.2}$$

where P_Z serves as the reference distribution.

Due to the Markov relation $X \to Y \to Z$ and after some straightforward computation, the conditions in (2.2) imply that

$$\forall x \in \mathcal{X}, \;\; P_{Z|X=x} = P_Z + \epsilon\sqrt{P_Z}\,\phi_x, \tag{2.3}$$

where the information vectors are given by:

$$\forall x \in \mathcal{X}, \forall z \in \mathcal{Z}, \;\; \phi_x(z) = \sum_{y \in \mathcal{Y}} P_{Y|X}(y|x)\psi_y(z). \tag{2.4}$$

■ 2.3.2 Special Notation

For the remainder of this chapter only, every random variable is discrete and finite, so we exclusively use the following matrix notation. Let $\mathcal{P}_{\mathcal{Y}|\mathcal{X}} \subseteq \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{X}|}$ represents the set of all column stochastic matrices (channels or transition probability kernels) from $\mathcal{X}$ to $\mathcal{Y}$. Analogously, we also write the joint pmf of any two random variables as matrices (i.e., $P_{Y,X} \in \mathcal{P}_{\mathcal{Y}\times\mathcal{X}} \subseteq \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{X}|}$). For any (marginal) pmf P_X, we view it as a column vector and let $[P_X] \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ denote the diagonal matrix with P_X along the principal diagonal.

■ 2.3.3 Divergence Transition Matrix

To succinctly describe the local approximation of the objective function of (2.1) that stems from (2.3), we introduce the *divergence transition matrix*.

Definition 2.3.1 (Divergence Transition Matrix [42])**.** Given a joint pmf $P_{Y,X} \in \mathcal{P}_{\mathcal{Y}\times\mathcal{X}}$, with conditional pmfs $P_{Y|X} \in \mathcal{P}_{\mathcal{Y}|\mathcal{X}}$ and marginal pmfs satisfying $P_X \in \mathcal{P}_{\mathcal{X}}^{\circ}$ and $P_Y \in \mathcal{P}_{\mathcal{Y}}^{\circ}$, the divergence transition matrix (DTM) of $P_{Y,X}$ is defined as:

$$B_{Y,X} = B(P_{Y,X}) \triangleq [P_Y]^{-\frac{1}{2}} P_{Y,X} [P_X]^{-\frac{1}{2}} \tag{2.5}$$

$$= [P_Y]^{-\frac{1}{2}} P_{Y|X} [P_X]^{\frac{1}{2}}. \tag{2.6}$$

It is well-known that the largest singular value of $B_{Y,X}$ is $\sigma_1(B_{Y,X}) = 1$ with corresponding right and left singular vectors $\sqrt{P_X}$ and $\sqrt{P_Y}$, respectively (see e.g. [42, Section

3], [64, Appendix A]):

$$\begin{aligned} B_{Y,X}\sqrt{P_X} &= \sigma_1(B_{Y,X})\sqrt{P_Y} = \sqrt{P_Y}, \\ B_{Y,X}^T\sqrt{P_Y} &= \sigma_1(B_{Y,X})\sqrt{P_X} = \sqrt{P_X}. \end{aligned} \tag{2.7}$$

Moreover, the next proposition decomposes the DTM of random variables in a Markov chain.

Proposition 2.3.1 (Composed DTM)**.** If $X \to Y \to Z$ form a Markov chain, then $B_{Z,X} = B_{Z,Y}B_{Y,X}$.

Proof. Observe using Definition 2.3.1 that:

$$\begin{aligned} B_{Z,X} &= [P_Z]^{-\frac{1}{2}} P_{Z|X} [P_X]^{\frac{1}{2}} \\ &= [P_Z]^{-\frac{1}{2}} P_{Z|Y} P_{Y|X} [P_X]^{\frac{1}{2}} \\ &= \underbrace{[P_Z]^{-\frac{1}{2}} P_{Z|Y} [P_Y]^{\frac{1}{2}}}_{B_{Z,Y}} \underbrace{[P_Y]^{-\frac{1}{2}} P_{Y|X} [P_X]^{\frac{1}{2}}}_{B_{Y,X}} \end{aligned}$$

where the second equality uses the Markov property. ∎

Finally, we locally approximate $I(X;Z)$ using (2.3).

Theorem 2.3.1 (Local Approximation of Mutual Information)**.** Under the local perturbation conditions in (2.3), we have:

$$I(X;Z) = \frac{1}{2}\left(\|B_{Z,X}\|_F^2 - 1\right) + o(\epsilon^2).$$

Proof. Observe that:

$$\begin{aligned}
I(X;Z) &= \sum_{x\in\mathcal{X}} P_X(x) D(P_{Z|X=x} \parallel P_Z) \\
&= \frac{1}{2}\epsilon^2 \sum_{x\in\mathcal{X}} P_X(x) \|\phi_x\|_2^2 + o(\epsilon^2) \\
&= \frac{1}{2}\epsilon^2 \sum_{x,z} P_X(x) \left(\frac{P_{Z|X}(z|x) - P_Z(z)}{\epsilon\sqrt{P_Z(z)}} \right)^2 + o(\epsilon^2) \\
&= \frac{1}{2} \sum_{x,z} \left(\frac{P_{Z,X}(z,x) - P_Z(z)P_X(x)}{\sqrt{P_Z(z)P_X(x)}} \right)^2 + o(\epsilon^2) \\
&= \frac{1}{2} \left\| B_{Z,X} - \sqrt{P_Z}\sqrt{P_X}^T \right\|_F^2 + o(\epsilon^2) \\
&= \frac{1}{2} \left(\|B_{Z,X}\|_F^2 - 1 \right) + o(\epsilon^2)
\end{aligned}$$

where the first equality follows from a straightforward calculation, the second equality uses Theorem 1.3.1, the third equality uses (2.3), the fifth equality uses Definition 2.3.1, and the final equality holds due to (2.7). ■

■ 2.3.4 Connections to Spectral Graph Theory

In the case of $\mathcal{X} = \mathcal{Y}$, if we view $P_{Y|X}$ as a matrix of Markov transition probabilities, (2.6) is the matrix being factorized in diffusion maps [17]. If we view $P_{Y,X}$ as a weighted adjacency matrix, (2.5) is almost identical to the symmetric normalized graph Laplacian [16, Section 1.2], [6]. Similar to the Laplacian, the DTM carries an important property that we will use later.

Proposition 2.3.2. The multiplicity of the singular value at 1 of $B(P_{Y,X})$ is equivalent to the number of connected components in a bipartite graph that has weighted adjacency matrix $P_{Y,X}$.

For a proof, we refer readers to [84, Theorem 3.1.1], which relates the eigenvalues of the

identity minus the Laplacian to the singular values of the (corresponding) DTM.

■ 2.4 Maximal Frobenius Norm Coupling

Inspired by Theorem 2.3.1, we will learn the $P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}$ that probabilistically clusters each $y \in \mathcal{Y}$ by maximizing $\|B_{Z,X}\|_F^2$ instead of $I(X;Z)$. This Frobenius norm formulation of probabilistic clustering is presented in the next definition.

Definition 2.4.1 (Frobenius Norm Formulation)**.** Given a joint pmf $P_{Y,X} \in \mathcal{P}_{\mathcal{Y}\times\mathcal{X}}$ such that the marginal pmfs satisfy $P_X \in \mathcal{P}_{\mathcal{X}}^{\circ}$ and $P_Y \in \mathcal{P}_{\mathcal{Y}}^{\circ}$, and a target pmf $P_Z \in \mathcal{P}_{\mathcal{Z}}^{\circ}$, we seek to solve the following extremal problem:

$$\max_{P_{Z|Y}\in\mathcal{P}_{\mathcal{Z}|\mathcal{Y}}:\ P_{Z|Y}P_Y=P_Z} \|B_{Z,X}\|_F^2\,, \tag{2.8}$$

where $X \to Y \to Z$ form a Markov chain. We will refer to an optimal argument $P_{Z|Y}^{\star}$ of this problem, which represents a desirable soft clustering assignment, as a *maximal Frobenius norm coupling*.

We make some pertinent remarks about Definition 2.4.1. Firstly, a "coupling" of two marginal pmfs P_Y and P_Z is typically defined as a joint pmf $P_{Z,Y}$ that is consistent with these marginals (and often has additional desirable properties) [56, Section 4.2]. However, since the maximizing conditional pmf $P_{Z|Y}^{\star}$ implicitly defines a joint pmf $P_{Z,Y}^{\star} = P_{Z|Y}^{\star}[P_Y]$, we refer to $P_{Z|Y}^{\star}$ itself as a coupling. Secondly, although the Frobenius norm formulation in (2.8) can be perceived as a local approximation of (2.1) (which nicely connects the two problems), we do not need to require $P_{Z|Y}^{\star}$ to be close to P_Z as in (2.2) (i.e., weak dependence between Z and Y) when using this formulation. Thirdly, the formulation in (2.8) is intuitively well-founded because [43] and [65] illustrate that the singular values of the DTM $B_{Z,X}$ capture how informative or correlated mutually orthogonal embeddings of Z and X are. Hence, maximizing the sum of all squared singular values maximizes the

relevant dependencies between Z and X. Naturally, there are various other reasonable formulations of probabilistic clustering that use singular values of the DTM. We present one such class of formulations in (2.9) in the next subsection.

■ 2.4.1 Theoretical Discussion

Consider the following generalization of (2.8) that also intuitively captures some notion of probabilistic clustering:

$$\max_{P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}:\ P_{Z|Y}P_Y = P_Z} \|B_{Z,X}\|_p^p \tag{2.9}$$

where $P_Z \in \mathcal{P}_{\mathcal{Z}}^\circ$ and $P_{Y,X} \in \mathcal{P}_{\mathcal{Y}\times\mathcal{X}}$ are fixed such that $P_X \in \mathcal{P}_{\mathcal{X}}^\circ$ and $P_Y \in \mathcal{P}_{\mathcal{Y}}^\circ$. Using Proposition 2.3.1, we may rewrite the objective function of (2.9) as

$$\|B_{Z,X}\|_p^p = \left\| [P_Z]^{-\frac{1}{2}} P_{Z|Y} [P_Y]^{\frac{1}{2}} B_{Y,X} \right\|_p^p. \tag{2.10}$$

Since the quantity inside the norm is linear in $P_{Z|Y}$, and the pth power of a Schatten ℓ^p-norm is convex, the objective function is convex. Moreover, the constraints on $P_{Z|Y}$ in (2.9) define a compact and convex set in $\mathbb{R}^{|\mathcal{Z}|\times|\mathcal{Y}|}$. As a result, the maximum in (2.9) can be achieved due to the extreme value theorem. Hence, (2.9) is a maximization of a convex function over a convex set. While convex minimization over convex sets is easy, non-convex problems such as (2.9) are computationally hard [11].

To illustrate this, we consider the notable special case of (2.9) with $p = 2$ which yields the problem in (2.8):

$$\begin{aligned} \max_{P_{Z|Y} \in \mathbb{R}^{|\mathcal{Z}|\times|\mathcal{Y}|}} \quad & \|B_{Z,Y} B_{Y,X}\|_F^2 \\ \text{subject to (s.t.)} \quad & P_{Z|Y} P_Y = P_Z \\ & \mathbf{1}_{|\mathcal{Z}|}^T P_{Z|Y} = \mathbf{1}_{|\mathcal{Y}|}^T \\ & P_{Z|Y} \geq 0 \end{aligned} \tag{2.11}$$

where we use Proposition 2.3.1 to rewrite the objective function, $\mathbf{1}_k \triangleq [1, \ldots, 1]^T \in \mathbb{R}^k$, and the second and third constraints ensure that $P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}$. Letting $A = B_{Z,Y}$ and $B = B_{Y,X}$, we can straightforwardly rewrite this problem as follows:

$$\begin{aligned} \max_{A \in \mathbb{R}^{|\mathcal{Z}| \times |\mathcal{Y}|}} \quad & \|AB\|_F^2 \\ \text{s.t.} \quad & A\sqrt{P_Y} = \sqrt{P_Z} \\ & A^T\sqrt{P_Z} = \sqrt{P_Y} \\ & A \geq 0 \end{aligned} \tag{2.12}$$

This is clearly a non-convex quadratic program (QP). Indeed, letting $a = \text{vec}(A) \in \mathbb{R}^{|\mathcal{Z}||\mathcal{Y}|}$ (which stacks the columns of A to form a vector), $M_1 = (B \otimes I_{|\mathcal{Z}|})(B^T \otimes I_{|\mathcal{Z}|})$, $M_2 = \sqrt{P_Y}^T \otimes I_{|\mathcal{Z}|}$, and $M_3 = I_{|\mathcal{Y}|} \otimes \sqrt{P_Z}^T$, the preceding problem is equivalent to:

$$\begin{aligned} \max_{a \in \mathbb{R}^{|\mathcal{Z}||\mathcal{Y}|}} \quad & a^T M_1 a \\ \text{s.t.} \quad & M_2\, a = \sqrt{P_Z} \quad M_3\, a = \sqrt{P_Y} \\ & a \geq 0 \end{aligned} \tag{2.13}$$

where $\otimes$ denotes the Kronecker product, and we use the fact that $\text{vec}(ABC) = (C^T \otimes A)\,\text{vec}(B)$ for any matrices A, B, and C with valid dimensions. The QP in (2.13) is non-convex because M_1 is positive semidefinite and we are maximizing the associated convex QP. It is proven in [90] that such QPs are NP-hard (also see [31, 80] and the references therein). Therefore, there are no known efficient algorithms to exactly solve (2.8), and we resort to relaxations and other heuristics in the upcoming sections.

Finally, we provide some brief intuition for the NP-hardness of (2.11). The feasible set of (2.11) is the convex polytope $\mathcal{P}_{\mathcal{Z}|\mathcal{Y}} \cap \mathcal{H}$, where $\mathcal{H} \triangleq \{M \in \mathbb{R}^{|\mathcal{Z}| \times |\mathcal{Y}|} : MP_Y = P_Z\}$ is a $|\mathcal{Z}|(|\mathcal{Y}| - 1)$-dimensional affine subspace of $\mathbb{R}^{|\mathcal{Z}| \times |\mathcal{Y}|}$. In general, this convex polytope has super-exponentially many extreme points. To see this, consider the special case where

$m = |\mathcal{Y}| = |\mathcal{Z}|$ and $P_Y = P_Z$ are the uniform pmf. Then, $\mathcal{P}_{\mathcal{Z}|\mathcal{Y}} \cap \mathcal{H}$ is the set of all $m \times m$ doubly stochastic matrices, and its extreme points are the $m!$ different $m \times m$ permutation matrices by the Birkhoff-von Neumann theorem [40, Theorem 8.7.2]. For general $|\mathcal{Y}|$, $|\mathcal{Z}|$, P_Y, and P_Z, the extreme points of $\mathcal{P}_{\mathcal{Z}|\mathcal{Y}} \cap \mathcal{H}$ have more complex structure (see e.g., [45], which studies the uniform P_Y and arbitrary P_Z case). When we maximize a convex function over $\mathcal{P}_{\mathcal{Z}|\mathcal{Y}} \cap \mathcal{H}$ as in (2.11), the optimum is achieved at an extreme point of $\mathcal{P}_{\mathcal{Z}|\mathcal{Y}} \cap \mathcal{H}$. So, we have to search over all super-exponentially many extreme points to find this optimal point. This is computationally very inefficient.

■ 2.4.2 Comparison to Formulations that Directly Modify Co-occurrences

A key feature of our formulation in (2.8) is that it clusters using a transition kernel $P_{Z|Y}$ and keeps the original data distribution $P_{Y,X}$ intact. For comparison, let us consider a different optimization problem that clusters by modifying the non-negative co-occurrence matrix $P \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{X}|}$ directly:

$$\min_{\substack{Q \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{X}|}: \\ Q \geq 0}} \|Q - P\|_F^2 - \lambda \sum_{i=1}^{|\mathcal{Z}|} \sigma_i(B(Q)), \tag{2.14}$$

where $\lambda > 0$ is a hyperparameter that should be set high enough to emphasize the second term in the objective function, $B(Q)$ denotes the DTM corresponding to the joint pmf obtained after normalizing Q, and $|\mathcal{Z}|$ represents the ideal number of clusters we want (note that the set $\mathcal{Z}$ is inconsequential in this formulation). Because (2.14) does not learn a transition kernel, in this subsection we do not normalize the data P to be a valid pmf in order to simplify the presentation.

Intuitively, (2.14) tries to find the closest non-negative matrix Q where $B(Q)$ has the top $|\mathcal{Z}|$ singular values as 1 (i.e., has $|\mathcal{Z}|$ connected components, see Proposition 2.3.2). This is closely related to the model in [77]. One drawback of this type of formulation is

that it has $|\mathcal{Y}||\mathcal{X}|$ parameters to learn. Since the number of clusters is typically much smaller than the number of users (i.e., $|\mathcal{Z}| \ll |\mathcal{Y}|$), our formulation in (2.9) has a much lower number ($|\mathcal{Z}||\mathcal{Y}|$) of parameters to learn.

A more important drawback of (2.14) is that sometimes, the intuitively correct clustering is not the globally optimal solution. We demonstrate this phenomenon via an example. Let $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ for disjoint sets $\mathcal{X}_1$ and $\mathcal{X}_2$, $\mathcal{Y} = \mathcal{Y}_1 \cup \mathcal{Y}_2$ for disjoint sets $\mathcal{Y}_1$ and $\mathcal{Y}_2$, $|\mathcal{X}_1| = |\mathcal{X}_2| = n$, $|\mathcal{Y}_1| = |\mathcal{Y}_2| = m$, and the number of clusters $|\mathcal{Z}| = 2$. Furthermore, let the data matrix P have the following structure:

$$P = \underbrace{\left[\begin{array}{c|c} s\mathbf{1} & \mathbf{1} \\ \hline \mathbf{1} & s\mathbf{1} \end{array}\right]}_{\substack{n \quad n}} \begin{matrix} \} m \\ \} m \end{matrix} \tag{2.15}$$

where $\mathbf{1}$ is a matrix of all 1's of appropriate dimension, and $s > 1$ is some scalar multiplier. Clearly, there are two distinct communities, and the intuitive result with two clusters is:

$$Q_1 = \underbrace{\left[\begin{array}{c|c} s\mathbf{1} & \mathbf{0} \\ \hline \mathbf{0} & s\mathbf{1} \end{array}\right]}_{\substack{n \quad n}} \begin{matrix} \} m \\ \} m \end{matrix} \tag{2.16}$$

where $\mathbf{0}$ is a matrix of all 0's of appropriate dimension. Since Q_1's structure creates two connected components, $\mathcal{X}_1 \cup \mathcal{Y}_1$ and $\mathcal{X}_2 \cup \mathcal{Y}_2$, the largest two singular values of $B(Q_1)$ are both 1. Moreover, the objective function has value $2mn - 2\lambda$.

Now consider a different Q that also creates two connected components by only discon-

necting one item of $\mathcal{X}$ and one item of $\mathcal{Y}$ from the rest of the items:

$$Q_2 = \left[\begin{array}{c|c|c} s\mathbf{1} & \mathbf{1} & \mathbf{0} \\ \hline \mathbf{1} & s\mathbf{1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & s \end{array}\right] \begin{matrix} \} m \\ \} m-1 \\ \} 1 \end{matrix} \qquad \begin{matrix} \\ \\ \\ \underbrace{\qquad}_{n} \quad \underbrace{\qquad}_{n-1} \quad \underbrace{\qquad}_{1} \end{matrix} \tag{2.17}$$

where the disconnected item forms the bottom 1×1 block. The largest two singular values of $B(Q_2)$ are still 1 because of the two connected components. However, the objective function now equals $m + n + s^2(m + n - 2) - 2\lambda$. Thus, when $s < \sqrt{(2mn - m - n)/(m + n - 2)}$, the intuitively correct answer Q_1 is not the global optimum of (2.14).

In contrast, our maximum Frobenius norm formulation in (2.8) (without the constraint on the marginal) easily obtains the two intuitive clusters encoded in P. For example, let $m = n = 50$, $\mathcal{Z} = \{0, 1\}$ denote the cluster labels, and consider the transition kernels $P^1_{Z|Y}$ corresponding to the intuitive clustering shown in (2.16) (defined by $P^1_{Z|Y}(0|y) = 1$ for $y \in \mathcal{Y}_1$ and $P^1_{Z|Y}(1|y) = 1$ for $y \in \mathcal{Y}_2$), and $P^2_{Z|Y}$ corresponding to the clustering shown in (2.17) (defined by $P^2_{Z|Y}(0|y) = 1$ for $y \neq y_0$ and $P^2_{Z|Y}(1|y_0) = 1$ for some $y_0 \in \mathcal{Y}_2$). Then, Figure 2.2 illustrates that the intuitive clustering of $P^1_{Z|Y}$ is greatly preferred by the maximum Frobenius norm formulation. Therefore, our formulation does not exhibit the drawbacks of formulations like (2.14).

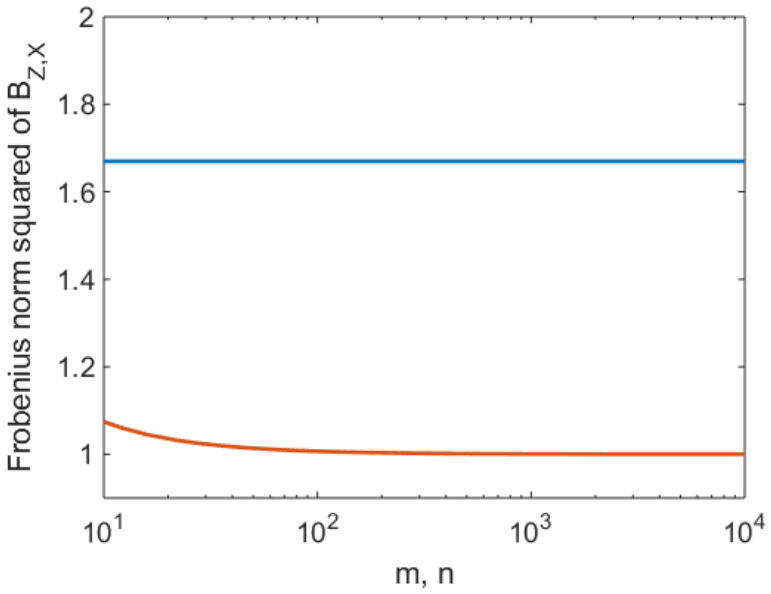

Figure 2.2: Plots of $\|B_{Z,X}\|_F^2$ versus m, n for the different transition kernels $P_{Z|Y}$. In particular, the blue plot corresponds to $B_{Z,X}$ defined by $P^1_{Z|Y}$ (the intuitive clustering), and the red plot corresponds to $B_{Z,X}$ defined by $P^2_{Z|Y}$ (the "one item" clustering).

■ 2.5 Optimization Algorithms

To solve the non-convex QP given by the Frobenius norm formulation of probabilistic clustering in (2.8), we will use a heuristic gradient ascent algorithm (Subsection 2.5.1) as well as a nuclear norm relaxation (Subsection 2.5.2). Although one approach to finding approximate solutions to an NP-hard problem similar to (2.8) is via semidefinite programming (SDP) relaxations, we do not explore SDP based algorithms in this work. Moreover, many of the simpler SDP relaxations for non-convex QPs do not accurately capture our setting because they only appear to be tight when at least one of the constraints is also quadratic [3].

2.5.1 Heuristic Gradient Ascent Algorithm

We now present a gradient-based algorithm for approximating the maximal Frobenius norm coupling defined by the formulation of probabilistic clustering in (2.8), or equivalently, in (2.12). For computational efficiency, we move the first constraint in (2.12) to the objective function to obtain:

$$\begin{aligned} \max_{A \in \mathbb{R}^{|\mathcal{Z}| \times |\mathcal{Y}|}} \quad & \|AB\|_F^2 - \lambda \left\| A\sqrt{P_Y} - \sqrt{P_Z} \right\|_2^2 \\ \text{s.t.} \quad & A^T \sqrt{P_Z} = \sqrt{P_Y} \\ & A \geq 0 \end{aligned} \tag{2.18}$$

where $\lambda > 0$ is a hyperparameter that controls how strictly the $A\sqrt{P_Y} = \sqrt{P_Z}$ constraint is imposed. In other words, the solution no longer has to induce clusters with exactly P_Z as their marginal pmf, but it incurs a penalty proportional to the squared ℓ^2-norm of the difference $A\sqrt{P_Y} - \sqrt{P_Z}$. Note that any other differentiable distance between distributions can be substituted here.

The gradients of the components in the objective function of (2.18) are:

$$\frac{\partial}{\partial A} \|AB\|_F^2 = \frac{\partial}{\partial A} \operatorname{tr}(ABB^T A^T) = 2ABB^T \tag{2.19}$$

$$\frac{\partial}{\partial A} \left\| A\sqrt{P_Y} - \sqrt{P_Z} \right\|_2^2 = 2\left(A\sqrt{P_Y}\sqrt{P_Y}^T - \sqrt{P_Z}\sqrt{P_Y}^T \right) \tag{2.20}$$

where we use denominator layout notation (or Hessian formulation).

Furthermore, since there is an equivalence between (2.11) and (2.12), the constraints in (2.18) correspond exactly to the second and third constraints in (2.11), which are just enforcing $P_{Z|Y}$ to be a valid column stochastic matrix. Thus, we can either use any existing algorithms (e.g. [15, 25]) for projection back onto the simplex and apply them column-wise to $P_{Z|Y}$ or revise them to operate on A directly. Algorithm 1 describes the entire optimization procedure for problem (2.18).

Input: Joint distribution $P_{Y,X}$, target marginal P_Z, marginal penalty multiplier $\lambda > 0$, step size $\alpha > 0$

Output: Soft clusters induced by $P_{Z|Y}$

Initialize $A_0 \in \mathbb{R}^{|\mathcal{Z}|\times|\mathcal{Y}|}$ to be an entry-wise positive matrix
$B \leftarrow [P_Y]^{-\frac{1}{2}} P_{Y,X} [P_X]^{-\frac{1}{2}}$
$M_1 \leftarrow BB^T$
$M_2 \leftarrow \lambda \sqrt{P_Y} \sqrt{P_Y}^T$
$M_3 \leftarrow \lambda \sqrt{P_Z} \sqrt{P_Y}^T$
while A_t not converged **do**
    $A_t \leftarrow A_{t-1}(I_{|\mathcal{Y}|} + \alpha(M_1 - M_2)) + \alpha M_3$
    if A_t violates constraint above tolerance **then**
        $A_t \leftarrow \text{proj}(A_t)$
    end if
end while
return $P_{Z|Y} \leftarrow [P_Z]^{\frac{1}{2}} A_t [P_Y]^{-\frac{1}{2}}$

Algorithm 1: Gradient Ascent Algorithm for the Frobenius Norm Formulation.

■ 2.5.2 Nuclear Norm Relaxation

Let us consider a modified problem where we approximate the Frobenius norm in (2.8) using a nuclear norm. This yields the problem in (2.9) specialized to the $p = 1$ case. We further relax this problem by completely disregarding the constraint on the marginal to obtain:

$$\max_{P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}} \|B_{Z,X}\|_* \tag{2.21}$$

which defines a *maximal nuclear norm coupling* representing a desirable clustering assignment. To derive some intuition about this problem, we recall a well known result from the

literature. For any fixed channel $P_{Z|X} \in \mathcal{P}_{\mathcal{Z}|\mathcal{X}}$, the second largest singular value $\sigma_2(B_{Z,X})$ of $B_{Z,X}$ is the *Hirschfeld-Gebelein-Rényi maximal correlation* between Z and X, which is given by:

$$\sigma_2(B_{Z,X}) = \max_{\substack{f:\mathcal{Z}\to\mathbb{R},\, g:\mathcal{X}\to\mathbb{R}:\\ \mathbb{E}[f(Z)]=\mathbb{E}[g(X)]=0\\ \mathbb{E}\left[f(Z)^2\right]=\mathbb{E}\left[g(X)^2\right]=1}} \mathbb{E}[f(Z)g(X)] \tag{2.22}$$

$$= \max_{\substack{f\in\mathbb{R}^{|\mathcal{Z}|},\, g\in\mathbb{R}^{|\mathcal{X}|}:\\ f^T P_Z = g^T P_X = 0\\ f^T[P_Z]f = g^T[P_X]g = 1}} f^T P_{Z,X} g \tag{2.23}$$

where the equality can be easily justified using the *Courant-Fischer variational characterization* of singular values (cf. [88], [64, Definition 2, Proposition 1], and the references therein). In particular, the optimal $f^\star$ and $g^\star$ can be obtained in terms of singular vectors of $B_{Z,X}$ corresponding to the singular value $\sigma_2(B_{Z,X})$, and they serve as useful features that capture the maximal correlation between Z and X [65, 84]. From this perspective, (2.21) maximizes the statistical dependence between Z and X as measured by the sum of maximal correlations (or singular values) subject to the Markov constraint $X \to Y \to Z$ for the purposes of probabilistic clustering.

To derive an algorithm for (2.21) that also uses SVD structure, we consider a generalization of (2.23). Using *Ky Fan's extremum principle*, cf. [39, Theorem 3.4.1], we obtain the relation:

$$\|B_{Z,X}\|_* = \max_{\substack{F\in\mathbb{R}^{|\mathcal{Z}|\times r}\\ G\in\mathbb{R}^{|\mathcal{X}|\times r}}} \operatorname{tr}(F^T P_{Z,X} G) \tag{2.24}$$

$$\text{s.t. } F^T[P_Z]F = G^T[P_X]G = I_r$$

where $r = \min(|\mathcal{X}|, |\mathcal{Z}|)$. The proof of [39, Theorem 3.4.1] also shows that the optimal

solutions of (2.24) are:

$$F^\star = [P_Z]^{-\frac{1}{2}}U \quad \text{and} \quad G^\star = [P_X]^{-\frac{1}{2}}V \tag{2.25}$$

where $U \in \mathbb{R}^{|\mathcal{Z}|\times r}$ and $V \in \mathbb{R}^{|\mathcal{X}|\times r}$ are matrices with orthonormal columns that correspond to the left and right singular vector bases of $B_{Z,X}$, respectively. Thus, since $P_{Z,X} = P_{Z|Y}P_{Y,X}$ by the Markov property, we can rewrite (2.21) as:

$$\begin{aligned} \max_{\substack{P_{Z|Y}\in\mathcal{P}_{\mathcal{Z}|\mathcal{Y}} \\ F\in\mathbb{R}^{|\mathcal{Z}|\times r} \\ G\in\mathbb{R}^{|\mathcal{X}|\times r}}} \quad & \operatorname{tr}(F^T P_{Z|Y} P_{Y,X} G) \\ \text{s.t. } & F^T[P_Z]F = G^T[P_X]G = I_r \end{aligned} \tag{2.26}$$

Inspired by [77], we also use alternating maximization to solve this problem. With $P_{Z|Y}$ fixed, the optimal F and G are given by (2.25). With F and G fixed, the objective function in (2.26) is linear in the entries of $P_{Z|Y}$ and can be solved using any linear programming (LP) packages. Algorithm 2 describes the entire optimization procedure.

We remark that this algorithm does not require any prior knowledge of P_Z. This is one potential advantage of the relaxed nuclear norm formulation in (2.21) over the original Frobenius norm formulation in (2.8). On the other hand, problem (2.26) has the uncommon feature that the constraint on F depends on $P_{Z|Y}$ (or more precisely, on P_Z, which is derived from $P_{Z|Y}$). In typical instances of alternating maximization problems, the feasible sets of the variables (over which we alternate) are "independent" of each other (see e.g. [19]). One way to "decouple" the feasible set of F from $P_{Z|Y}$ is to fix some P_Z (when we have prior knowledge). This imposes an additional linear constraint on $P_{Z|Y}$ which is easily handled by an LP. In our experiments, we do not impose this additional constraint because Algorithm 2 converges to a reasonable solution without the constraint.

Input: Joint distribution $P_{Y,X}$

Output: Clusters induced by $P_{Z|Y}$

Initialize $P_{Z|Y}$ to be a $|\mathcal{Z}| \times |\mathcal{Y}|$ column stochastic matrix
$P_X \leftarrow \mathbf{1}_{|\mathcal{Y}|}^T P_{Y,X}$
while $P_{Z|Y}$ not converged **do**
  $P_{Z,X} \leftarrow P_{Z|Y} P_{Y,X}$
  $P_Z \leftarrow P_{Z,X} \mathbf{1}_{|\mathcal{X}|}$
  $B \leftarrow [P_Z]^{-\frac{1}{2}} P_{Z,X} [P_X]^{-\frac{1}{2}}$
  $U, \Sigma, V \leftarrow \text{SVD}(B)$
  $F \leftarrow [P_Z]^{-\frac{1}{2}} U$
  $G \leftarrow [P_X]^{-\frac{1}{2}} V$
  $P_{Z|Y} \leftarrow \arg\max_{P_{Z|Y} \in \mathcal{P}_{\mathcal{Z}|\mathcal{Y}}} \text{tr}(F^T P_{Z|Y} P_{Y,X} G)$
end while
return $P_{Z|Y}$

Algorithm 2: Alternating Maximization Algorithm for the Nuclear Norm Formulation

■ 2.6 Experiments

■ 2.6.1 Word Embedding

Although this chapter is about clustering, we first want to validate that the DTM is an informative matrix for large scale unsupervised learning. To do this, we use it to learn word embeddings for the Microsoft Research Sentence Completion Challenge [104]. The dataset consists of a training corpus of raw text taken from classic English literature and 1040 Graduate Record Examination (GRE) style sentence completion questions.

Let $P_{Y,X}$ be constructed as the normalized word-word co-occurrence matrix and let $U\Sigma V^T \approx [P_Y]^{-\frac{1}{2}} P_{Y,X} [P_X]^{-\frac{1}{2}}$ be the 640-dimensional truncated SVD of the DTM. We use

Table 2.1: Performance comparison between our model and other single architecture methods as reported in [70].

Model	Accuracy
4-gram	39%
Average LSA similarity	49.6%
Log-bilinear model	54.8%
RNNLMs	55.4%
Skip-gram	48.0%
Our model	**53.94**%

the alternating conditional expectations (ACE) algorithm [13, 65] to approximate $[P_Y]^{-\frac{1}{2}}U$ and use that as the word embedding. We choose ACE because of the difficulty with inserting side information into randomized SVD algorithms [35]. See Section A.1 for a derivation of ACE with side information.

We use various functions of cosine similarity between the candidate word and the surrounding words to select the most probable answer. For details on text preprocessing and the selection metric, refer to Section A.2. Table 2.1 shows that our method is competitive with popular single architecture word embedding techniques. This is not entirely surprising as there are other works [57, 83] that advocate approximately factorizing various versions of the co-occurrence matrix. However, it provides empirical evidence that methods based on the DTM perform well and are worth further investigation (on embedding as well as clustering).

■ 2.6.2 MovieLens 100K

For qualitative validation, we use Algorithm 2 to find 5 clusters in the MovieLens 100K dataset. The data is in the form of a movie-user rating matrix, where each entry can be

Table 2.2: Examples of well known movies divided into the clusters found by Algorithm 2. Note that N/A does not mean cluster 2 is empty, but that it does not contain any movies in the top 100 most rated list.

Cluster 1	Cluster 2	Cluster 3
Raiders of the Lost Ark	N/A	The Terminator
The Godfather	N/A	Terminator 2
Pulp Fiction	N/A	Braveheart
Silence of the Lambs	N/A	The Fugitive
Cluster 4	**Cluster 5**	
Star Wars	Contact	
Return of the Jedi	Liar Liar	
Fargo	The English Patient	
Toy Story	Scream	

blank to denote unrated, or in the range $\{1, \ldots, 5\}$. This is conceptually different from a co-occurrence matrix since a 5-rated movie does not mean a user watched that movie 5 times more frequently compared to a 1-rated movie.

For preprocessing, we replace all blank entries with 0 to denote zero co-occurrence. We assume that each unit increment in rating corresponds to tripling a user's affinity toward a movie. Thus, we map each valid rating using the function $r \mapsto 3^{r-1} - 1$. Then, we row normalize such that each row (corresponding to one movie) sums to 1.

From Table 2.2, we can see an approximate division of genres among clusters 1, 3, 4, and 5. Cluster 2 captures many of the less popular movies and does not contain any one from the set of 100 movies with the most ratings. Since MovieLens 100K does not contain ground truth cluster labels, we do not experiment further beyond this qualitative example.

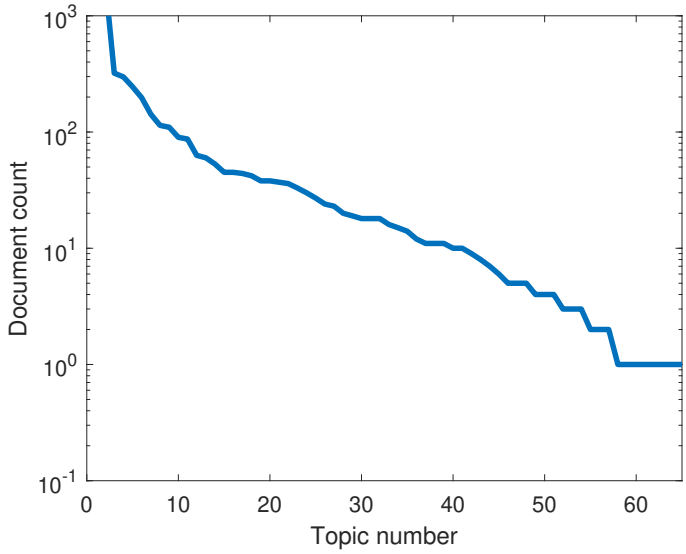

Figure 2.3: Histogram of the number of documents for each topic in the Reuters21578 dataset. The vertical axis is in logarithmic scale. The distribution approximately follows Zipf's law.

■ 2.6.3 Reuters21578

The Reuters21578 dataset contains 8293 documents and their frequencies on 18933 terms. Although the ground truth shows 65 topic clusters, the largest 10 clusters include 87.9% of all documents while the smallest 8 clusters each has 1 document (see Figure 2.3). Thus, we argue that a good algorithm needs to provide a meaningful metric to select a number of clusters that balances between covering many documents and clustering accuracy.

For this experiment, we do not perform any data preprocessing and split all documents into $k \in \{2, 3, 4, 6, 8, 10\}$ clusters. Because we do not have clusters devoted to the $65 - k$ smallest clusters, we report the classification accuracy in Table 2.3 using two different metrics. Overall accuracy counts all documents from those smallest clusters as incorrectly classified, and k-accuracy disregards those documents and only reports accuracy of documents from the top k clusters. In both of these cases, the extra documents from the smallest

Table 2.3: Clustering accuracy on the Reuters21578 dataset using Algorithm 2. The nuclear norm increases more slowly when $k \geq 8$, which implies that $k = 8$ or 10 is the "right" number of clusters.

k	Coverage	Overall acc.	k-acc.	$\|\cdot\|_*$
2	69.55%	65.15%	93.67%	1.71
3	73.42%	65.51%	89.22%	2.33
4	77.01%	62.25%	80.83%	2.85
6	82.35%	57.43%	69.74%	3.72
8	85.43%	54.11%	63.34%	4.49
10	87.85%	48.52%	55.23%	5.14

Table 2.4: Clustering accuracy on the Reuters21578 dataset using Algorithm 1. Knowing the true cluster marginal distribution helps maintain accuracy even as k increases.

k	Coverage	Overall acc.	k-acc.	$\|\cdot\|_F$
2	69.55%	47.86%	68.81%	1.19
3	73.42%	59.60%	81.18%	1.30
4	77.01%	68.64%	89.12%	1.38
6	82.35%	67.70%	82.21%	1.48
8	85.43%	69.12%	80.90%	1.51
10	87.85%	70.73%	80.52%	1.59

clusters are still present in the data, acting as noise.

Similar to spectral clustering [75], we can report the norm given by Algorithm 2 against k to identify the k that strikes a balance between document coverage and clustering accuracy. At the cost of disregarding the smallest clusters, we achieve improved overall accuracy compared to the best algorithm (43.94%) reported in [77, Table 2].

Alternatively, assuming access to the ground truth cluster marginal pmf, we can use Algorithm 1. Table 2.4 shows that this prior information offers significant improvements in accuracy as k increases.

■ 2.7 Summary and Future Work

In this chapter, we reviewed the mutual information formulation for probabilistic clustering (2.1). Then, to convert (2.1) into a matrix optimization (2.8), we locally approximated mutual information as the Frobenius norm of the DTM (Theorem 2.3.1). This allowed us to explicitly learn a maximal matrix norm coupling $P_{Z|Y}$ for clustering as opposed to the standard procedure of embedding then k-means. Learning $P_{Z|Y}$ also let us encode prior information. We saw one example of this with the predefined P_Z in (2.8). To perform semi-supervised learning, we can also add constraints that fix certain columns of $P_{Z|Y}$ for the subset of the data that is labeled.

There are two aspects of our approach that can be improved in future work. Firstly, we can implement more efficient non-convex optimization algorithms that converge to solutions that are closer to the global optimum. Secondly, we can improve our model's robustness to noise. Currently, we treat the observed noisy co-occurrence matrix as a good estimate of the true distribution while matrix factorization (MF) models treat the noise as entry-wise Gaussian perturbations of a low rank model [91]. In our experience, MF tends to perform well on data with high entry-wise noise while our approach performs well on data with complex community structures and lower noise.

Another future direction is to probabilistically cluster X in addition to Y. The opti-

mization problem for this is:

$$\begin{aligned} \max_{\substack{A\in\mathbb{R}^{|\mathcal{Z}|\times|\mathcal{Y}|},\\ C\in\mathbb{R}^{|\mathcal{W}|\times|\mathcal{X}|}}} \quad & \left\|ABC^T\right\|_F^2 \\ \text{s.t. } & A\sqrt{P_Y} = \sqrt{P_Z},\ A^T\sqrt{P_Z} = \sqrt{P_Y}, \\ & C\sqrt{P_X} = \sqrt{P_W},\ C^T\sqrt{P_W} = \sqrt{P_X}, \\ & A \geq 0,\ C \geq 0, \end{aligned} \tag{2.27}$$

where C obtains the clusters of X, cf. (2.12). This parallels the notion of *co-clustering* in the literature [22], and is a topic worthy of further investigation.

Chapter 3

One-Shot Feature Reduction for Transfer Learning and Task Personalization

3.1 Task Personalization

Neural networks have achieved great success in tackling problems in computer vision [41, 51], natural language understanding [21], robotics [28], etc. However, state of the art neural architectures have millions of parameters, and rely on extremely large datasets to reach their advertised performance. For many tasks, it is unrealistic to obtain large amounts of labeled data. For example, a personalized home security system cannot wait until multiple robberies have already occurred before adapting to the homeowner's environment. Multi-task learning [89] enables such rapid adaptation, but it often requires uploading personal labels to a central location, which may be unacceptable due to privacy reasons. Moreover, users will want to design their own tasks, which makes it difficult for centralized training to accommodate millions of unique customers.

In this chapter, we study this problem of *task personalization*. In particular, we model task personalization as dividing some subset of the source labels into more detailed target classes (see Figure 3.1). Building on the previous example, the home security system is

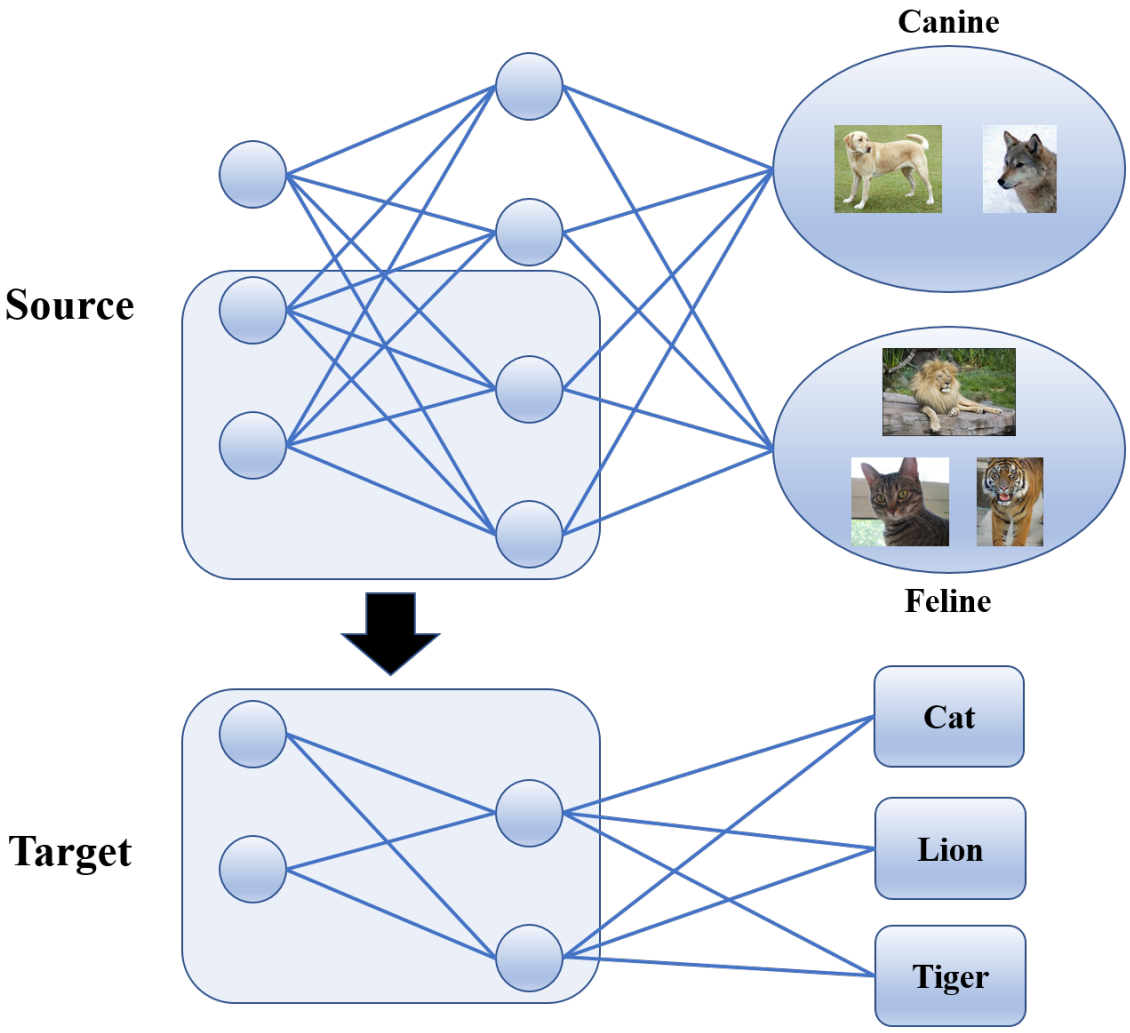

Figure 3.1: Diagram overviewing the method of selecting a part of the source neural network for task personalization.

centrally trained to distinguish between gender, but a customer may want to further train it to distinguish between their multiple daughters. This is different from traditional notions of personalization, where the user features change but the task does not.

Under this setting, overfitting dramatically degrades performance because the size of the task personalization dataset is typically too small to properly train a model with many

parameters. Thus, it is vital that we design a principled framework to process and rank features from a generic source model. Then, as the amount of personalized training data grows, the system can also increase the set of features it uses to achieve a good tradeoff between representation and overfitting.

In particular, we consider the situation where we have a pre-trained deep neural network for some generic purposes, such as CIFAR classification, and a limited number of labeled samples for a specific task of interest. Here, the challenge is to use very few samples to figure out how to use the selected features in the pre-trained model for the target task. For example, we might use a pre-trained Inception model with 2048 features before the final layer for a specific binary classification problem, for which we only have around 200 labeled samples. In such a case, it is necessary to select from these 2048 features a much smaller subset that can optimally contribute to the desired decision. Conceptually, the key question is which features are the most relevant ones to the target task, and how to select them with a single small batch of samples.

The theoretical approach we take in this chapter is based on a local geometric analysis reported in [44]. The concept we borrow is a metric of "relevance", defined as the inner product between feature functions in functional space, and can be approximated relatively easily with a small amount of samples.

Our algorithms make only one pass over the dataset prior to training. Our first algorithm rewrites the classification problem as a least square problem in the information space. Its analytic solution makes it suitable for smaller neural networks with very limited target training data. Our second algorithm performs feature set reduction by returning the source features ranked by their potential contribution. This helps alleviate overfitting and also permits training deeper networks on the target problem. We do not introduce additional architectural elements, and only uses existing neural network building blocks (e.g. softmax, backpropagation, gradient descent, etc.). This is advantageous compared

to popular weight pruning techniques that require training to completion. Weight pruning [36, 73] typically removes connections in a trained network that either have low absolute value or have small impact on the output activation. However, the strategies are mostly designed for during training or post-training. Due to the small amount of training data, the target model typically overfits and causes the weight pruning objectives to be unreliable.

The chapter is organized as follows. Section 3.2 formulates the local geometric framework for KL divergence and derives relevant approximations to classification problems with cross-entropy loss. Section 3.3 reviews the related literature. Section 3.4 reports experimental performance on CIFAR-10 [50] and the NEXET self-driving datasets. Finally, Section 3.5 concludes the chapter and discusses future works.

■ 3.2 Feature Selection Using Local Geometry

■ 3.2.1 Log Likelihood Ratio Function

For clarity, consider a two class problem with input $X \in \mathcal{X}$, $Y \in \{0, 1\}$, $P_Y(0) = P_Y(1) = 0.5$, with condition distributions in information vector form (1.1)

$$P_{X|Y=1} = P_X + \sqrt{P_X}\,\phi \tag{3.1}$$

$$P_{X|Y=0} = P_X - \sqrt{P_X}\,\phi, \tag{3.2}$$

where P_X serves as the reference distribution. Figure 3.2 shows a visualization of the local geometry.

The goal of neural classifiers is to optimally learn the log likelihood functions, or the log likelihood ratio (LLR) in binary classification [10]. Under the geometric framework,

$$\log \frac{P_{X|Y=1}}{P_{X|Y=0}} \approx \frac{\phi}{\sqrt{P_X}}. \tag{3.3}$$

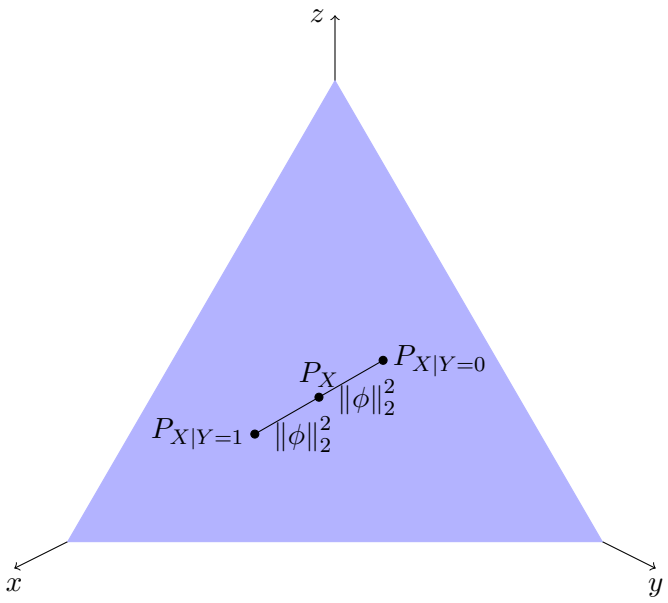

Figure 3.2: Visualization of the conditional distributions and information vector ϕ as perturbations around the marginal distribution P_X. Under the local geometric view, KL divergence is symmetric.

Hence, $\frac{\phi}{\sqrt{P_X}}$ is the function of X that we want to learn using the neural network. Notice that converting from function space back to information space just requires multiplying back $\sqrt{P_X}$. We make use of this change of coordinate in Section 3.2.3.

■ 3.2.2 Cross-Entropy Loss

The power of the geometric framework lies in its ability to convert problems without closed form solutions into well known linear algebraic problems [85]. Logistic and softmax regression with cross-entropy loss are two important examples. Their loss function is

$$\mathcal{L}(\widehat{P}, Q) = -\mathbb{E}_{\widehat{P}_X} \mathbb{E}_{\widehat{P}_{Y|X}} \log Q\left(Y|X\right) \tag{3.4}$$

$$= \mathbb{E}_{\widehat{P}_X}\left[H\left(\widehat{P}_{Y|X=x}\right) + D\Big(\widehat{P}_{Y|X=x} \,\Big\|\, Q_{Y|X=x}\Big)\right] \tag{3.5}$$

where $\widehat{P}$ is the empirical distribution and Q is the learned distribution parameterized by a neural network.

The entropy term is a constant, as it is not changed by optimizing over Q. After applying Bayes' rule and Theorem 1.3.1, we can simplify (3.5) as

$$c + \mathbb{E}_{P_Y} D\Big(\widehat{P}_{X|Y} \;\Big\|\; Q_{X|Y}\Big) \approx c + k \left\|\hat{\phi} - \psi\right\|_2^2 \tag{3.6}$$

where c and k are constants, and ψ and $\hat{\phi}$ are the information vectors of Q and $\widehat{P}$, respectively. Thus, minimizing the cross-entropy loss is approximated by minimizing the squared Euclidean distance between ψ and $\hat{\phi}$. Furthermore, if we plot the loss for different networks, the loss should be affine functions of the squared Euclidean norm. This allows us to focus on minimizing $\|\hat{\phi} - \psi\|_2^2$.

Although the local approximation is needed to reduce the cross-entropy loss into squared Euclidean norm, starting with using the χ^2 divergence as the loss function makes the above approximation precise.

■ 3.2.3 Approximate Analytic Solution

For many transfer learning systems with limited target data, parts of the source model are used as a fixed feature extractor while one (or more) linear layers are attached and trained. For a source model with k features, f_i (the i-th feature output) is a function, and can be rescaled into the information space as

$$\psi^{(f_i)} \triangleq \sqrt{P_X} f_i, \; \forall i \in [1, k]. \tag{3.7}$$

Then, minimizing the cross-entropy loss reduces to solving a least square problem in

the information vector space:

$$\arg\min_{w} \ \frac{1}{2}\left\|\hat{\phi} - \sum_{i=1}^{k} w_i \psi^{(f_i)}\right\|_2^2. \tag{3.8}$$

For finite $\mathcal{X}$ and $\Psi \triangleq [\psi^{(f_1)}, \ldots, \psi^{(f_k)}]$, (3.8) can be rewritten into the well known matrix form

$$\arg\min_{w} \ \frac{1}{2}\left\|\hat{\phi} - \Psi w\right\|_2^2, \tag{3.9}$$

with the solution as

$$w^* = (\Psi^T \Psi)^{-1} \Psi^T \hat{\phi}. \tag{3.10}$$

$(\Psi^T \Psi)^{-1}$ is the inverse covariance matrix of the zero-mean source features and is easily estimated from samples. The following calculation shows that under the geometric framework, gradient backpropagation is equivalent to projection in linear algebra:

$$\begin{aligned}
\frac{\partial}{\partial w_j}\mathcal{L} &= \frac{\partial}{\partial w_j}\frac{1}{2}\left\|\hat{\phi} - \sum_{i=1}^{k} w_i \psi^{(f_i)}\right\|_2^2 \\
&= \|\psi^{(f_j)}\|^2 w_i + \sum_{i \neq j} \left\langle \psi^{(f_j)}, \psi^{(f_i)} w_i \right\rangle - \left\langle \psi^{(f_j)}, \hat{\phi} \right\rangle \\
&= -\left\langle \psi^{(f_j)}, \hat{\phi} - \sum_{i=1}^{k} w_i \psi^{(f_i)} \right\rangle,
\end{aligned} \tag{3.11}$$

and the matrix version yields

$$\frac{\partial}{\partial w}\frac{1}{2}\left\|\hat{\phi} - \Psi w\right\|_2^2 = \Psi^T \Psi w - \Psi^T \hat{\phi}. \tag{3.12}$$

When $w = \mathbf{0}$, the gradient reduces to the desired quantity $-\Psi^T \hat{\phi}$.

Input: labeled dataset $[x_i, y_i]_{i=1}^{N_L}$, unlabeled dataset $[x_j]_{j=1}^{N_U}$, trained feature extractor f, target classifier g_w

1: Apply f to $[x_j]_{j=1}^{N_U}$, compute output feature mean and update f's final layer bias to remove mean

2: Stack zero mean version of $[f(x_j)]_{j=1}^{N_U}$ into a data matrix F and compute $F^T F$

3: Initialize $w = \mathbf{0}$, compute $\nabla_w \mathcal{L}$

4: $w \leftarrow -(F^T F)^{-1} \nabla_w \mathcal{L}$

Algorithm 3: Least Square Solution

The least square solution in (3.10) can thus be implemented fairly easily: we can compute $\Psi^T \hat{\phi}$ with (3.12), and $\Psi^T \Psi$ is the covariance matrix of all the features, which can be evaluated with unlabeled data samples. Algorithm 3 describes the full procedure on data samples. One might recognize this method as first whitening the feature functions to eliminate the redundancy of the information they carry, and then making the projection to the target $\hat{\phi}$. This is somewhat in the flavor of classical signal processing.

From an optimization point of view, the least square solution can be understood as first approximating the loss function locally as quadratic, and then directly computing the minimum from the gradients evaluated at a point near the optimum. This approach is quite aggressive. It attempts to form a single linear combination of feature functions $\sum w_i^* f_i$ to approximate $\hat{\phi}$. In practice, we observe that the result is often quite close to ϕ. However, this approach is sensitive to the difference between $\hat{\phi}$ and ϕ, and often do not perform well when only a small batch of samples is used to calculate the w^* coefficients.

■ 3.2.4 One-Shot Feature Set Reduction

Ideally, we want to use the local geometry encapsulated in (3.11) to select for multiple relevant features. In addition to being more robust to empirical variations, this allows for training multilayer target classifiers or additional stages of transfer learning.

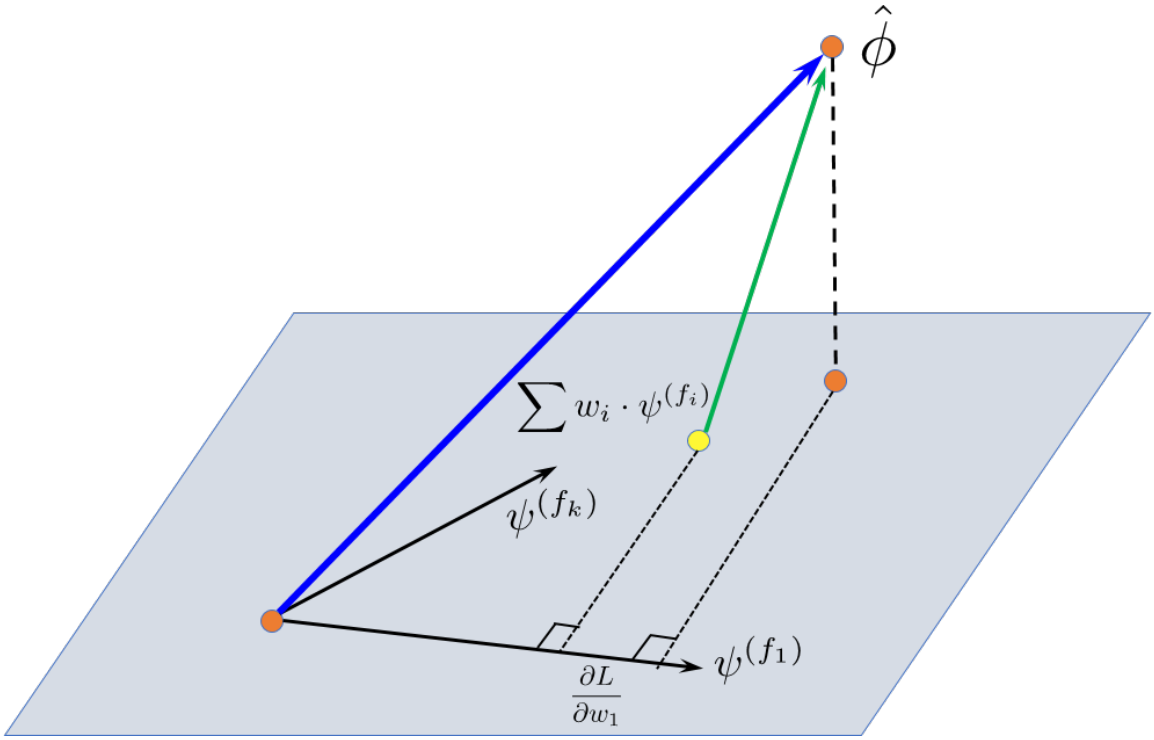

Figure 3.3: Graphical description of the projection view of classification with softmax activation and cross-entropy loss.

Figure 3.3 illustrates the actions described in (3.11). Specifically, for a frozen set of feature functions f_i's, if we initialize the weights $w = \mathbf{0}$ (i.e., the yellow point in the picture is at the origin), the gradient calculation in (3.11) yields

$$\left.\frac{\partial \mathcal{L}}{\partial w_i}\right|_{w_i=0} = -\left\langle \psi^{(f_i)}, \hat{\phi} \right\rangle. \tag{3.13}$$

This inner product is interpreted as a measure of how relevant a feature function f_i is for a target query represented by $\hat{\phi}$. That is, if we have a binary hypothesis testing problem with the LLR function written in the information vector form as $\hat{\phi}$, then the inner product measures the approximate performance loss if we make decisions based on the feature value

Input: labeled dataset $[x_i, y_i]_{i=1}^{N_L}$, unlabeled dataset $[x_j]_{j=1}^{N_U}$, trained feature extractor f, target classifier g_w, number of features to retain r

1: Apply f to $[x_j]_{j=1}^{N_U}$, compute output feature mean and variance.

2: Apply f and standardization to the labeled dataset input to form $[\tilde{f}(x_i), y_i]_{i=1}^{N_L}$

3: Initialize $w = \mathbf{0}$, compute $\nabla_w \mathcal{L}$, and rank individual w_i in descending order of $|\nabla_{w_i} \mathcal{L}|$

4: Retain the top r weights and make the remaining non-trainable

5: Update w with mini-batch gradient descent over $[\tilde{f}(x_i), y_i]_{i=1}^{N_L}$

Algorithm 4: One-Shot Feature Set Reduction

of f_i instead of the LLR. An advantage of this simple measure of relevance is that it can be evaluated fairly easily, since the inner product of the information vectors can be written as the correlation between the feature function f_i and the target LLR function. In practice, the gradient computed during backpropagation is the empirical average version of this correlation, averaged with a single batch of samples.

As a compromise, we select not one, but a small collection of features. We do not require these features to be uncorrelated, and use some labeled samples on a fully connected softmax layer to learn the right way to combine these features. To that end, we only require that each feature function is individually relevant to the target problem. This leads to a natural solution that picks a subset of features with the largest absolute values of their gradients.[1] Since the picking phase of the algorithm does not involve training for multiple iterations, we call this *one-shot feature set reduction* (FSR). Algorithm 4 describes the full procedure on data samples.

A key feature of Algorithm 4 is that the feature selection is performed under a linear approximation while the classifier is a potentially nonlinear or generalized linear model (GLM). Section 3.4 empirically shows that this approach outperforms both purely linear

[1] For multi-class problems, we use the sum of square of the weight gradients associated with a particular feature.

approximations to the cross-entropy loss and GLM with regularization.

3.3 Related Work

3.3.1 General Transfer Learning

Many methods fall under the umbrella of transfer learning. [71] uses self-supervision over unannotated corpora to learn vector representations of words, which are useful for many downstream tasks. [21, 87] take it a step further by learning deep transformer-based [97] language models and fine-tuning on natural language understanding tasks. Today, these models are viewed as the building blocks for many state of the art systems in natural language understanding.

The source task can also be completely unsupervised. For example, [12] exploits the structure of convolutional neural networks (CNN) to learn unsupervised representations via mapping images to randomly drawn noise samples on the hypersphere. The authors then show that the representations are informative for ImageNet [20] and Pascal VOC [27]. However, all the aforementioned works assume a large set of labeled target data to bypass the need for feature set reduction.

Meta-learning, especially in the form of learning a transferable initialization [28, 76], is another related research area. These methods formulate the meta-objective as the loss after the current network parameters have had k steps of adaptation via gradient descent. This encourages the learned network parameters to operate as good initialization points for adapting to a distribution of tasks. They have also been shown to achieve higher accuracy than methods based on matching [49, 99] on few-shot learning problems such as Omniglot [54]. However, few-shot learning in this form assumes the presence of a distribution of related tasks during training time. It would be challenging to obtain these while preserving privacy.

Domain adaptation assumes that only the input distribution changes between the source

and target problems [7]. The most popular methods (e.g., [30, 52]) use an adversarial loss [32] during source training to align the source and target latent space. We do not focus on source training in this work, but our method also allows for unsupervised domain adaptation techniques during that stage.

■ 3.3.2 Specific Related Methods

The area of subcategory-aware classification [14, 23] in the computer vision community is conceptually similar to the goal of task personalization. The hypothesis is that there is too much intraclass diversity such that automatically learning subcategories would improve classification accuracy on the (larger) classes. Usually, the subcategories are learned using unsupervised learning methods such as clustering. However, this is different from our goal. Subcategory-aware classification typically only considers the source problem of class accuracy, while we tackle the problem of transferring to subcategory classification with few samples. Combining both methods would potentially yield interesting results.

Within the area of transfer learning and feature selection, [29] also uses a fully supervised image classification source task and proposes methods to transfer to unseen target classes with few examples. While their goal is most similar to ours, they assume access to natural language descriptions of the unseen classes and leverage pre-trained semantic embeddings to represent them. We do not assume access to such side information. [82] uses information-theoretic quantities to measure feature relevance. They propose estimating mutual information of continuous random variables using discretization and Parzen windows, which is more computationally challenging than our methods. Additionally, they do not explore the effect of their algorithm on transfer learning or task personalization.

3.4 Experimental Results

3.4.1 Introduction

Recall that we are primarily interested in task personalization, defined as dividing some subset of the source labels into more detailed target classes. We focus on comparing one-shot FSR (Algorithm 4) against standard regularization and automatic feature selection strategies such as ridge and lasso regression [95]. For all three methods, we extract the output features from the penultimate layer of some source-trained deep neural network. In this work, we then freeze the source network and train an additional linear layer for classification on a small task personalization dataset. Because we use different source models for CIFAR-10 and NEXET, their source training procedures are described in their respective subsections.

3.4.2 Task Personalization Training Setup

After extracting the features using the source model, we standardize every feature by making them zero mean and unit variance. This could be done using labeled or unlabeled data. For Algorithm 4, we sweep the number of feature in $[1, k]$ (approximately logarithmic spaced), where k is the number of output features in the penultimate layer of the source model. For ridge and lasso regression, we sweep $\lambda \in [10^{-5}, 10^{0}]$. We select the best performing hyperparameter using the validation set and then report the performance on the test set. We repeat each experiment 10 times to obtain the mean accuracy and 95% confidence interval.

For training, we use the Adam optimizer [47] with a batch size of 50 and an initial learning rate of 0.001. We decay the learning rate by a factor of 10 after five consecutive epochs without validation accuracy improvement.

All experiments were performed using PyTorch 1.3.0 [81], on an Ubuntu 16.04 desktop equipped with one Intel i7-8700k CPU and one Nvidia RTX 2080 Ti GPU.

Figure 3.4: Example images from the CIFAR-10 dataset.

■ 3.4.3 CIFAR-10

CIFAR-10 [50] is a popular benchmark dataset for computer vision[2]. It contains 50000 training and 10000 validation images of size 32×32. There are 10 different categories of different types of animal and vehicles. We split the official validation set into two halves to form our validation and test sets. During source and target training, we use random crop and horizontal flip as data augmentation.

We use binary classification between classes $\{1, \ldots, 5\}$ and $\{6, \ldots, 10\}$ as the source task and train a ResNet-18 [37] from random initialization as our source model. For this part, we train for 100 epochs using stochastic gradient descent (SGD) with momentum. The learning rate is 0.1 for epochs $[1, 50]$, 0.001 for epochs $[51, 75]$, and 0.0001 for epochs $[76, 100]$. The batch size is 128 and L_2 regularization is $5 \cdot 10^{-4}$.

We define the two target tasks as 5-way classification within classes $\{1, \ldots, 5\}$, corresponding to {airplane, automobile, bird, cat, deer}, and $\{6, \ldots, 10\}$, corresponding to

[2] https://www.cs.toronto.edu/~kriz/cifar.html

Table 3.1: Mean accuracy and 95% confidence interval for 5-way classification within classes {airplane, automobile, bird, cat, deer} in the CIFAR-10 dataset.

	$N = 100$	$N = 300$	$N = 1000$
Ridge	70.34 ± 0.06	74.35 ± 0.15	77.69 ± 0.21
Lasso	70.33 ± 0.23	75.75 ± 0.25	77.74 ± 0.17
One-shot FSR	$\mathbf{73.32 \pm 0.21}$	$\mathbf{75.88 \pm 0.22}$	$\mathbf{78.82 \pm 0.18}$
Features selected	96	160	192

Table 3.2: Mean accuracy and 95% confidence interval for 5-way classification within {dog, frog, horse, ship, truck} in the CIFAR-10 dataset.

	$N = 100$	$N = 300$	$N = 1000$
Ridge	84.12 ± 0.05	85.65 ± 0.32	89.58 ± 0.21
Lasso	87.96 ± 0.30	87.36 ± 0.23	89.99 ± 0.06
One-shot FSR	$\mathbf{88.76 \pm 0.14}$	$\mathbf{88.98 \pm 0.28}$	$\mathbf{90.57 \pm 0.17}$
Features selected	96	128	224

{dog, frog, horse, ship, truck}. The empirical results are found in Table 3.1 and Table 3.2, respectively. As expected, both tables show that when data is scarce, ridge regression performs poorly because it does not reduce the feature set. One-shot FSR's accuracy is consistently highest across all scenarios, but the gap closes as the number of target samples increase. Interestingly, the accuracy for classes $\{6, \ldots, 10\}$ is significantly higher than for classes $\{1, \ldots, 5\}$. This suggests that it is easier to distinguish between {dog, frog, horse, ship, truck} than {airplane, automobile, bird, cat, deer}.

Figure 3.5: Example images from the NEXET dataset.

■ 3.4.4 NEXET

NEXET is a dataset of street images for training autonomous vehicles[3]. It contains 50000 training images with multiple bounding boxes. The images come from cities such as San Francisco and Tel Aviv, and have lighting conditions such as daylight, nightime, and twilight. There are five categories: car, van, truck, pickup truck, and bus. We extract, resize to 299×299, and save each unique bounding box and its associated label. We do not use any data augmentation for NEXET.

Because the car category contains the most samples, we define the source task as binary classification between car and non-car. We train an Inception-v3 neural network [94] on the source task starting from the PyTorch Inception-v3 ImageNet checkpoint. For this part, we train for 100 epochs using SGD with momentum. The initial learning rate is 0.02, and

[3] https://blog.getnexar.com/https-medium-com-itayklein-intro-nexet-50e9b596d0e5

Table 3.3: Mean accuracy and 95% confidence interval for classification between {van, bus}, and {truck, pickup truck} in NEXET. The source model is trained on car vs. non-car.

	$N = 30$	$N = 100$	$N = 300$	$N = 1000$
RIDGE	77.59 ± 0.28	80.09 ± 0.05	86.52 ± 0.36	88.90 ± 0.14
LASSO	80.68 ± 0.13	84.03 ± 0.27	86.81 ± 0.17	89.37 ± 0.12
ONE-SHOT FSR	$\mathbf{83.85 \pm 0.05}$	$\mathbf{84.59 \pm 0.35}$	$\mathbf{88.61 \pm 0.05}$	$\mathbf{90.02 \pm 0.06}$
FEATURES SELECTED	64	64	256	320

it decays by a factor of 10 every 25 epochs. The batch size is 64 and L_2 regularization is 10^{-5}.

We define the target task as binary classification between {van, bus}, and {truck, pickup truck}. Because the number of target classes is smaller than that of CIFAR-10, we sweep the number of target training samples over $N \in \{30, 100, 300, 1000\}$. Table 3.3 reports the empirical results. Similar to the CIFAR-10 results, one-shot FSR's accuracy is consistently highest across all scenarios, but the gap closes as the number of target samples increase.

To further show that one-shot FSR is picking features in a statistically meaningful way, we plot the mean accuracy as we sweep the number of features chosen. Here, the baseline which we compare to is a simple strategy that just randomly picks a subset of features. Figure 3.6 shows the plot for $N = 1000$. From that, it is clear that for as low as 20 features, one-shot FSR already captures enough information to approach maximum accuracy. However, because the target set is fairly large, one-shot FSR does not gain much compared to pure regularization strategies (i.e., not picking, right endpoint of the blue line).

The more interesting outcome occurs when the target set is smaller. Figure 3.7 shows the plot for $N = 100$. Here, if we use regularization alone, the performance is lower than one-shot FSR or the least square solution. Since the least square solution does not have

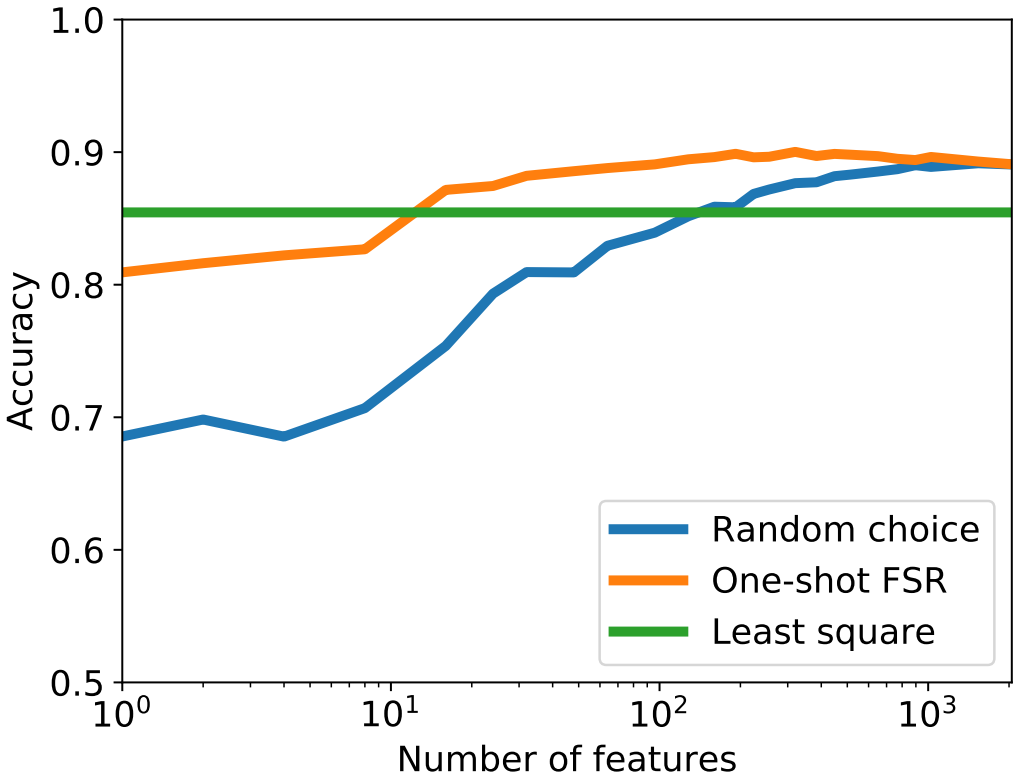

Figure 3.6: Plot of accuracy vs. number of feature chosen for randomly selecting features, one-shot FSR, and least square solution. $N = 1000$.

hyperparameters, it is a useful method to achieve good (but not maximum) performance on smaller datasets without needing hyperparameter sweep or validation.

■ 3.4.5 NEXET Without Finetuning

For a complete empirical evaluation, we shift gears from task personalization to evaluating our method on a problem that one-shot FSR is not necessarily designed for: general transfer learning. In the most general case, transfer learning assumes that both the input distribution and the task set / distribution will change drastically [103]. Depending on the

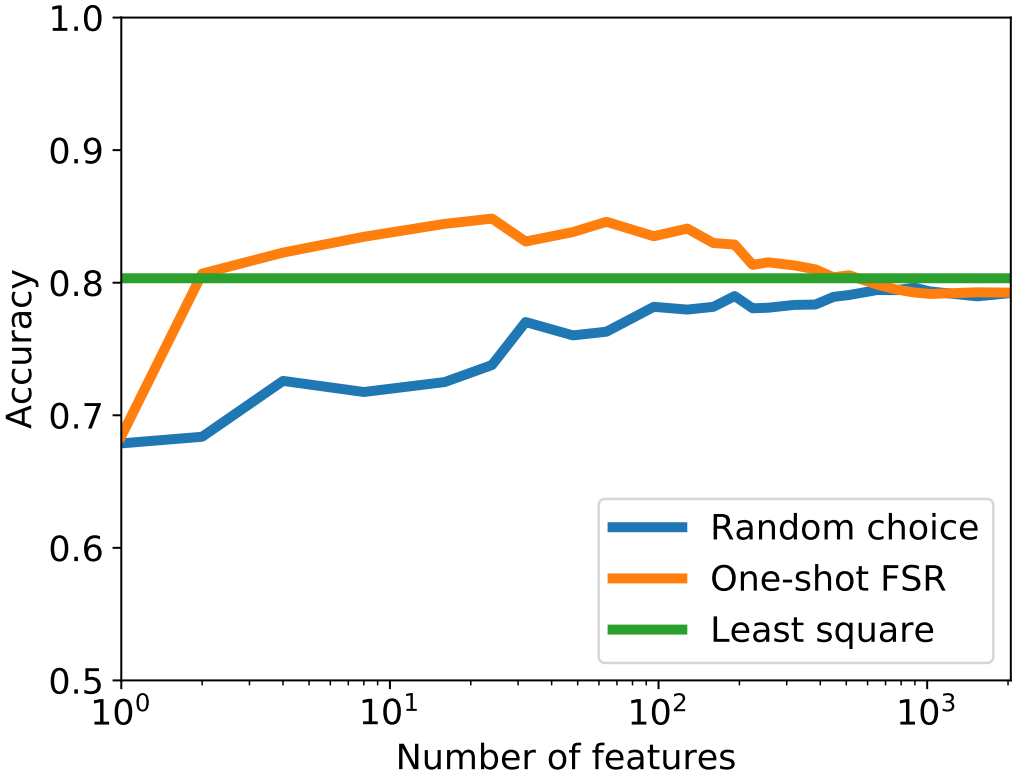

Figure 3.7: Plot of accuracy vs. number of feature chosen for randomly selecting features, one-shot FSR, and least square solution. $N = 100$.

architecture of the source model, which layer we choose to extract, and the similarity between source and target task, either a larger proportion of the source features are irrelevant to the target task or the learned representations are generic enough that every feature can contribute to target accuracy improvement.

We model this scenario by transferring from a model pre-trained on ImageNet to NEXET. We still use Inception-v3, but do not train on car vs. non-car. Instead, we use the PyTorch ImageNet checkpoint as our source model. All other experimental settings are identical to the previous subsection.

Table 3.4: Mean accuracy and 95% confidence interval for classification between {van, bus}, and {truck, pickup truck} in NEXET. The source model is trained on ImageNet.

	$N = 30$	$N = 100$	$N = 300$	$N = 1000$
Ridge	67.91 ± 0.13	79.61 ± 0.06	83.47 ± 0.08	86.01 ± 0.11
Lasso	$\mathbf{71.45 \pm 0.56}$	$\mathbf{79.69 \pm 0.13}$	$\mathbf{83.83 \pm 0.10}$	$\mathbf{86.28 \pm 0.11}$
One-shot FSR	69.57 ± 0.73	78.95 ± 0.07	82.77 ± 0.13	85.97 ± 0.15
Features selected	64	Full	Full	Full

Table 3.4 reports the empirical results. Since one-shot FSR tends to select all features, it is a sign that the representations learned by Inception-v3 on ImageNet tend to be generic and informative to the target task. Although lasso performs the best in this case, our method does not significantly degrade performance.

■ 3.5 Summary and Future Directions

As machine learning becomes more prevalent in our devices, there is more opportunity and demand for task personalization. There have been many works tackling personalization problems where the user feature changes but the task does not, or transfer learning problems where the input does not change but the tasks are completely different. We bridge the gap by proposing and studying the problem where the target divides some subset of the source labels into more detailed classes.

We derive a local information geometric approximation to classification with softmax activation and cross-entropy loss. Under the framework, classification problems can be rewritten as a least square problem in the information space. Under the projection view of least square, the least square weights represent the features' relevance to the log likelihood function that we are trying to learn.

Then, we empirically show that our one-shot feature set reduction algorithm beats regularization and automatic feature selection strategies for task personalization. The classification accuracy on CIFAR-10 and NEXET are especially strong when the target dataset is small, such that it is difficult to train a large target model without overfitting.

There are many possible future directions for this work. For example, designing decorrelation strategies to account for redundancy could further reduce the number of features needed. Extending the geometric framework to analyze multi-layer or recurrent target classifiers could enable algorithms that can capture more complex task personalization classes. Testing on multi-modal datasets beyond just images could improve performance and discover latent interactions between the different modalities.

Chapter 4

Topics in Generative Models

Deep generative models, especially generative adversarial networks (GAN) are intimately related to approximations of statistical divergences. In particular, divergences have variational forms and can be written as maximization problems. Then, minimizing the divergence between a generated distribution and a data distribution yields GAN's min-max structure [78, Section 2]. This chapter presents some analyses and experiments in the generative modeling space. Section 4.1 proves that vanilla GAN does not have be framed as a zero-sum game to perform implicit f-divergence minimization. Section 4.2 derives a variant of the Wasserstein critic that enables learning to generate a subset of a distribution. Section 4.3 proposes a generative approximation of conditional entropy and experiments with its self-normalization property in classification problems. Section 4.4 examines using generative models to learn interference signal constellations to aid multi-user wireless communication.

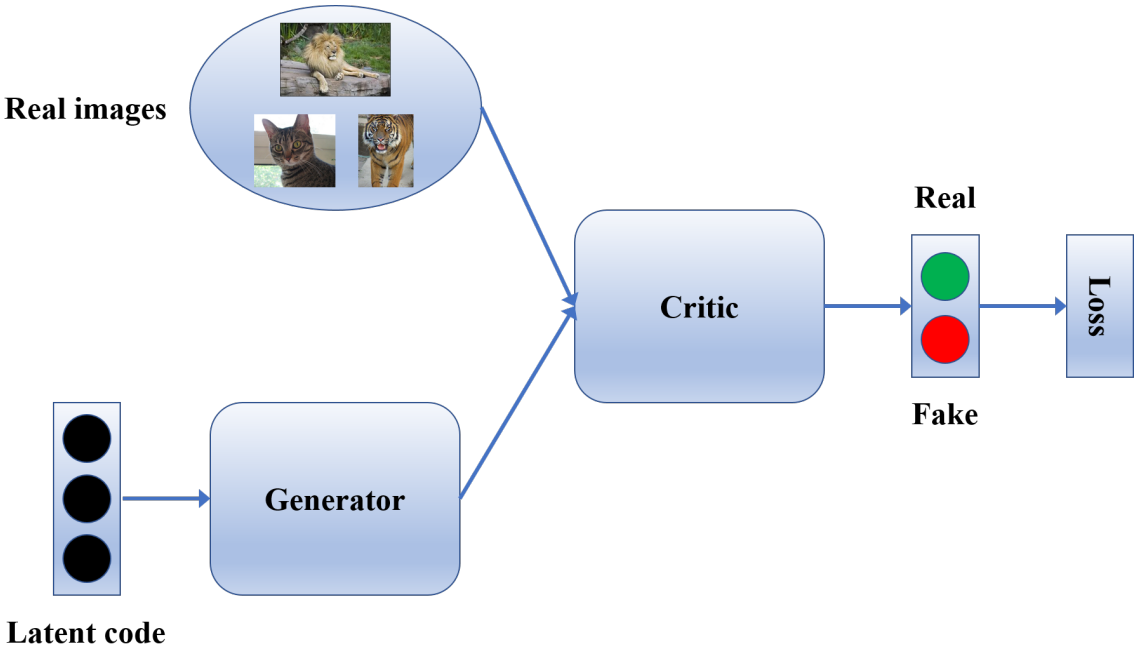

Figure 4.1: Architectural diagram for generative adversarial networks.

■ 4.1 Alternative GAN Objective

GANs consist of a generator (G) and a critic[1] (C), both parameterized by neural networks (see Figure 4.1). They are trained with the following optimization [32, Section 3]:

$$\min_G \max_C \; \mathbb{E}_{x\sim P}\left[\log(C(x))\right] + \mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2 I)}\left[\log(1-C(G(z)))\right], \tag{4.1}$$

where P is the data distribution. In essence, C tries to discriminate between inputs from the data distribution (real) and inputs from the generated distribution (fake) while G tries to generate increasingly realistic samples to fool C.

GANs achieve global optimality when $G^*(Z) \overset{\mathrm{d}}{=} X$ and $C^*(x) = 0.5, \forall x \in \mathcal{X}$, with a min-

[1] This is usually called a discriminator when the objective is binary classification. We use the more general term *critic* to reserve the letter D for f-divergence.

max loss of $-\log(4)$ [32, Theorem 1]. However, empirical results show that GANs rarely replicate the entire data distribution, especially as the dimensionality of data increases. One such phenomenon is called *mode collapse* by the generative models community, where only a subset of modes are outputted by G. We now provide a brief qualitative description of the cause of mode collapse.

Imagine we are training a GAN to generate MNIST handwritten digits [55] and at some point during training, the majority of G's outputs are 0's and none of G's outputs are 9's. When C sees this through data, it learns to predict 0's as fake and 9's as real with high confidence. Now, because G is trained to maximally fool C into thinking its generated samples are real, it can do that by outputting mostly 9's. Then, C adapts to the updated distributions and the cat and mouse game continues indefinitely.

One reason for this instability is due to the first order optimization methods used to train neural networks. Neither G or C can anticipate how the other network will react to its parameter updates. A method called unrolled GAN [69] attempts to alleviate this problem by training G with a surrogate "lookahead" loss. However, the full algorithm involves unrolling the computation graph and computing kth order derivatives.

■ 4.1.1 Non-Zero-Sum GAN

We take a different approach and show that GAN does not have to be trained as a zero-sum game. Consider the following loss functions:

$$\mathcal{L}_G = \mathbb{E}_{x\sim P}[D_f(\text{Bern}(0.5) \parallel \text{Bern}(C(x)))] + \mathbb{E}_{x\sim Q}[D_f(\text{Bern}(0.5) \parallel \text{Bern}(C(x)))] \tag{4.2}$$

$$\mathcal{L}_C = -\mathbb{E}_{x\sim P}\left[\log(C(x))\right] - \mathbb{E}_{x\sim Q}\left[\log(1-C(x))\right], \tag{4.3}$$

where D_f denotes any f-divergence, Q is a shorthand notation for the distribution of $G(Z)$. Instead of having $\mathcal{L}_G = -\mathcal{L}_C$ (i.e., training G to maximally fool C), we build the inductive bias of $C^*(x) = 0.5$ into G's loss function. The following theorem establishes minimizing

(4.2) as implicit divergence minimization.

Lemma 4.1.1. For a fixed G, the optimal C is $C^*(x) = \frac{P(x)}{P(x)+Q(x)}$.

Proof. See [32, Proposition 1].

Theorem 4.1.1. For C that is optimally trained with respect to G, $\mathcal{L}_G = D_f(\frac{P+Q}{2} \parallel P) + D_f(\frac{P+Q}{2} \parallel Q)$. The global minimum generator loss $\mathcal{L}_G = 0$ when $P = Q$.

Proof. From Lemma 4.1.1, the optimal critic outputs $C^*(x) = \frac{P(x)}{P(x)+Q(x)}$. Then, substitute $C^*(x)$ into (4.2):

$$\begin{aligned}
\mathcal{L}_G &= \mathbb{E}_{x\sim P}[D_f(U \parallel \text{Bernoulli}(C^*(x)))] + \mathbb{E}_{x\sim Q}[D_f(U \parallel \text{Bernoulli}(C^*(x)))] \\
&= \mathbb{E}_{x\sim P}\left[\frac{P(x)}{P(x)+Q(x)} f\left(\frac{\frac{1}{2}(P(x)+Q(x))}{P(x)}\right) + \frac{Q(x)}{P(x)+Q(x)} f\left(\frac{\frac{1}{2}(P(x)+Q(x))}{Q(x)}\right)\right] \\
&+ \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{P(x)+Q(x)} f\left(\frac{\frac{1}{2}(P(x)+Q(x))}{P(x)}\right) + \frac{Q(x)}{P(x)+Q(x)} f\left(\frac{\frac{1}{2}(P(x)+Q(x))}{Q(x)}\right)\right] \\
&= \mathbb{E}_{x\sim P}\left[f\left(\frac{\frac{1}{2}(P(x)+Q(x))}{P(x)}\right)\right] + \mathbb{E}_{x\sim Q}\left[f\left(\frac{\frac{1}{2}(P(x)+Q(x))}{Q(x)}\right)\right] \\
&= D_f\left(\frac{P+Q}{2} \,\middle\|\, P\right) + D_f\left(\frac{P+Q}{2} \,\middle\|\, Q\right),
\end{aligned} \tag{4.4}$$

where the second and fourth equality use the definition of f-divergence and the third equality uses a straightforward rearrangement of the expectations and probabilities. Due to the positive definiteness of divergence, $P = Q$ implies $\mathcal{L}_G = 0$. ■

We can reverse the arguments of the f-divergences in (4.2) and show a similar result.

$$\mathcal{L}_G^R = \mathbb{E}_{x\sim P}\left[D_f(\text{Bern}(C(x)) \parallel \text{Bern}(0.5))\right] + \mathbb{E}_{x\sim Q}[D_f(\text{Bern}(C(x)) \parallel \text{Bern}(0.5))] \tag{4.5}$$

$$\mathcal{L}_C^R = -\mathbb{E}_{x\sim P}\left[\log(C(x))\right] - \mathbb{E}_{x\sim Q}\left[\log(1-C(x))\right], \tag{4.6}$$

Theorem 4.1.2. For C that is optimally trained with respect to G, $\mathcal{L}_G^R = D_f(P \parallel \frac{P+Q}{2}) + D_f(Q \parallel \frac{P+Q}{2})$. The global minimum generator loss $\mathcal{L}_G = 0$ when $P = Q$.

Proof. From Lemma 4.1.1, the optimal critic outputs $C^*(x) = \frac{P(x)}{P(x)+Q(x)}$. Then, substitute $C^*(x)$ into (4.5):

$$\begin{aligned}
\mathcal{L}_G^R &= \mathbb{E}_{x\sim P}\left[D_f(\text{Bernoulli}(C^*(x)) \parallel U)\right] + \mathbb{E}_{x\sim Q}[D_f(\text{Bernoulli}(C^*(x)) \parallel U)] \\
&= \mathbb{E}_{x\sim P}\left[\frac{1}{2}f\left(\frac{P(x)}{\frac{1}{2}(P(x)+Q(x))}\right) + \frac{1}{2}f\left(\frac{Q(x)}{\frac{1}{2}(P(x)+Q(x))}\right)\right] \\
&+ \mathbb{E}_{x\sim Q}\left[\frac{1}{2}f\left(\frac{P(x)}{\frac{1}{2}(P(x)+Q(x))}\right) + \frac{1}{2}f\left(\frac{Q(x)}{\frac{1}{2}(P(x)+Q(x))}\right)\right] \\
&= \mathbb{E}_{x\sim \frac{P+Q}{2}}\left[f\left(\frac{P(x)}{\frac{1}{2}(P(x)+Q(x))}\right) + f\left(\frac{Q(x)}{\frac{1}{2}(P(x)+Q(x))}\right)\right] \\
&= D_f\left(P \,\middle\|\, \frac{P+Q}{2}\right) + D_f\left(Q \,\middle\|\, \frac{P+Q}{2}\right)
\end{aligned} \tag{4.7}$$

where the second and fourth equality use the definition of f-divergence and the third equality combines the expectations. Due to the positive definiteness of divergence, $P = Q$ implies $\mathcal{L}_G^R = 0$. ■

We conclude the proofs with a remark that when KL divergence is chosen in (4.5), (4.7) becomes proportional to the Jensen-Shannon divergence [58].

■ 4.1.2 Algorithm

Algorithmically, (4.2) with KL divergence is straightforward to implement in modern deep learning frameworks with automatic differentiation. Since the binary uniform distribution is the first argument of the KL divergence, it is sufficient to modify the ground truth probabilites of all generated samples to be uniform during generator training. Loss functions such as tf.nn.sigmoid_cross_entropy_with_logits in TensorFlow [1] and torch.nn.BCEWithLogitsLoss in PyTorch [81] fully support this operation. Algorithm 5

Input: unlabeled data distribution P, noise distribution $\mathcal{N}$, mini-batch size m

Output: trained generator G and critic C

1: **for** number of training iterations **do**

2: Draw m samples $[x_i]_{i=1}^m$ from P

3: Draw m noise samples $[z_i]_{i=1}^m$ from $\mathcal{N}$

4: Update C by descending on the gradient of

$$-\frac{1}{2m}\sum_{i=1}^{m}\log(C(x_i)) - \frac{1}{2m}\sum_{i=1}^{m}\log(1 - C(G(z_i))$$

5: Draw m noise samples $[z_i]_{i=1}^m$ from $\mathcal{N}$

6: Update G by descending on the gradient of

$$-\frac{1}{2m}\sum_{i=1}^{m}\log(C(G(z_i))) - \frac{1}{2m}\sum_{i=1}^{m}\log(1 - C(G(z_i)))$$

7: **end for**

Algorithm 5: Modified GAN training procedure [32] with the alternative generator loss in (4.2).

describes the training procedure in detail. For (4.5), the output from the critic is the first argument of the KL divergence. Supporting this is also possible through automatically differentiation of $u\log(u)$.

■ 4.1.3 Stacked MNIST Experiment

For the experiment, we use the Stacked MNIST code and hyperparameter settings from [59] and modify the DCGAN [86] generator loss to use (4.2) with KL divergence. Stacked MNIST is an image dataset where three random images from MNIST [55] are stacked on top of each other in the three color channels (see Figure 4.2). Thus, there are 1000 modes

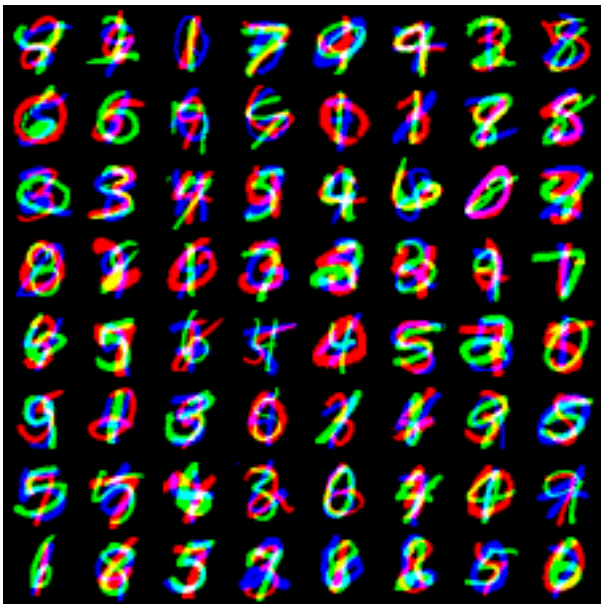

Figure 4.2: Example images from the Stacked MNIST dataset.

to cover.

To compute the number of modes covered, [59] uses a pre-trained MNIST classifier to classify individual color channels of all generated samples. Then, mode coverage is defined as the number of modes with at least one generated sample. Additionally, KL divergence can be computed between the generated PMF and the uniform PMF over the 1000 modes. Table 4.1 reports the Stacked MNIST performance of our method and other single sample (unpacked) GAN architectures. Our method is competitive with popular GAN architecture without introducing nearly as much complexity as VEEGAN [93] does. The simplicity of our method also makes it straightforward to extend to multiple samples (packing).

Table 4.1: Number of modes covered (higher is better) by the generator and the KL divergence (lower is better) between the real and generated distributions for different GAN architectures.

Architecture	Modes	KL
Adversarially Learned Inference (ALI) [26]	16.0	5.40
Unrolled GAN [69]	48.7	4.32
VEEGAN [93]	150.0	2.95
DCGAN [86]	78.9 ± 6.46	4.50 ± 0.127
Ours	111.8 ± 12.43	3.76 ± 0.20

4.2 Subset Wasserstein GAN

Wasserstein GAN (WGAN) [2] is another class of generative model based on the same idea of competing neural networks. Instead of the convex conjugate[2] of f-divergence [78], it uses the Wasserstein distance (see Definition 4.2.1) and Kantorovich-Rubinstein duality [98] to provide the training signal to the generator (G). As the full proof of the Kantorovich-Rubinstein duality is beyond the scope of this work, we use discrete Wasserstein distance for the derivations in this section.

Definition 4.2.1 (Discrete Wasserstein-1 Distance)**.** For pmfs P and Q with identical finite alphabet $\mathcal{U}$ and some distance function $d : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, the Wasserstein-1 distance

[2] Also known as the Fenchel conjugate.

W_1 is defined as the solution to

$$W_1(P,Q) = \min_{\gamma \in \mathcal{P}_{\mathcal{U},\mathcal{U}}} \sum_{\substack{x \in \mathcal{U} \\ y \in \mathcal{U}}} d(x,y)\gamma(x,y) \tag{4.8}$$
$$\text{s.t.} \sum_{y \in \mathcal{U}} \gamma(x,y) = P(x), \forall x \in \mathcal{U},$$
$$\sum_{x \in \mathcal{U}} \gamma(x,y) = Q(y), \forall y \in \mathcal{U}.$$

Theorem 4.2.1 (Kantorovich-Rubinstein Duality)**.** The dual problem of (4.8) is

$$\max_f \sum_{y \in \mathcal{U}} f(y)Q(y) - \sum_{x \in \mathcal{U}} f(x)P(x) \tag{4.9}$$
$$\text{s.t. } |f(x) - f(y)| \leq d(x,y), \forall x, y \in \mathcal{U}.$$

In addition, strong duality holds.

WGAN [2] and WGAN-GP [33] use G to generate Q, use the dataset to sample from P, and parameterize f with a critic network. They choose d as the L_1 distance and enforce the resulting Lipschitz constraint with weight clipping and a gradient penalty term [33, Section 4] , respectively. However, $W_1 = 0$ if and only if $P = Q$, which is not conducive for use cases where only a subset of the distribution needs to be captured.

For example, we may want to use WGAN as the base architecture for a style transfer system [102] between two unlabeled datasets drawn from P and R. Although they are unlabeled, we know that R contains only zebras and P contains various classes of animals including different species of horses. If we only want the system to learn how to translate between zebras and horses, it needs to automatically learn to treat horses and other irrelevant species from P differently without manual labeling. To alleviate that problem,

consider a modification of W_1 that we name the *subset Wasserstein distance*:

$$W_1^{(S)}(P,Q) = \min_{\gamma \in \mathcal{P}_{\mathcal{U},\mathcal{U}}} \sum_{\substack{x \in \mathcal{U} \\ y \in \mathcal{U}}} d(x,y)\gamma(x,y) \tag{4.10}$$
$$\text{s.t.} \sum_{y \in \mathcal{U}} \gamma(x,y) \leq \frac{P(x)}{\alpha}, \forall x \in \mathcal{U},$$
$$\sum_{x \in \mathcal{U}} \gamma(x,y) = Q(y), \forall y \in \mathcal{U},$$

where α is the prior belief on the probability of horses in Q. The dual of (4.10) is

$$\max_{f \geq 0} \sum_{x \in \mathcal{U}} f(y)Q(y) - \frac{1}{\alpha} \sum_{x \in \mathcal{U}} f(x)P(x) \tag{4.11}$$
$$\text{s.t.} \; |f(x) - f(y)| \leq d(x,y), \forall x, y \in \mathcal{U}.$$

For more details on the derivation of (4.11), refer to Section C.1.

To use WGAN to model (4.10) for image distributions, it is necessary to extend (4.11) to continuous distributions and to design G with inductive biases to penalize large changes between the input and output (in the example, this would bias G to map from zebras to animals with similar geometry, such as horses). Then, the Subset WGAN can be trained with the following optimization:

$$\min_G \max_{C \geq 0} \; \mathbb{E}_{r \sim R}\left[C(G(r))\right] - \frac{1}{\alpha}\mathbb{E}_{x \sim P}\left[C(x)\right], \tag{4.12}$$

where $G(r)$ induces the distribution Q and C is a Wasserstein critic with a non-negative output activation such as the softplus function.

■ 4.3 Batch Softmax Normalization

Classification problems aim to predict label Y from input X. In this section, we rewrite classification with softmax activation and cross-entropy objective as estimating the conditional entropy $-H(Y|X)$. Since conditional entropy is a function of KL divergence, we can choose to approximate its discriminative or generative forms [74]. The tool we use is the Donsker-Varadhan (DV) characterization of KL divergence (Theorem 4.3.1) and we apply it to either $D(P_{Y|X=x} \parallel \cdot)$ or $D(P_{X,Y} \parallel \cdot)$.

Theorem 4.3.1 (Donsker-Varadhan [24])**.** KL divergence can be written in its variational form as

$$D(P \parallel Q) = \sup_{g} \ \mathbb{E}_P[g(X)] - \log \mathbb{E}_Q[\exp(g(X))] \tag{4.13}$$

where the function $g : \text{range}(X) \to \mathbb{R}$ is constrained such that $\log \mathbb{E}_Q[\exp(g(X))] < \infty$.

Lemma 4.3.1. The optimal $g(x) = c + \log\left(\frac{P(x)}{Q(x)}\right)$ for some $c \in \mathbb{R}$.

Proof. See Appendix C.2.

For the remainder of this section, we assume X and Y to be discrete, though the derivations also hold for continuous X.

■ 4.3.1 Softmax as Discriminative DV

Define U to be a uniform random variable on the alphabet of Y. From definition of conditional entropy,

$$-H(Y|X) = -\sum_x P_X(x) H(Y|X=x) = \sum_x P_X(x)[D(P_{Y|X=x} \parallel P_U) - H(U)]. \tag{4.14}$$

Now, apply (4.13). Because there is a different divergence term for each realization of X, the supremum is over functions g parameterized by X.

$$\begin{aligned} &\mathbb{E}_{P_X}[\sup_{g_X} \mathbb{E}_{P_{Y|X}}[g_X(Y)] - \log \mathbb{E}_{P_U}[\exp(g_X(U))]] - H(U) \\ =&\mathbb{E}_{P_X}\left[\sup_{g_X} \mathbb{E}_{P_{Y|X}}[g_X(Y)] - \log \frac{\sum_u P_U(u)\exp(g_X(u))]}{P_U(u)}\right] \end{aligned} \tag{4.15}$$

However, all of the supremizations are independent because they operate on g parameterized by different outcomes of X. Thus, we can write g as a function of two variables and move the sup to the outside of the expectation:

$$\sup_g \mathbb{E}_{P_X}[\mathbb{E}_{P_{Y|X}}[g(X,Y)] - \log \sum_u \exp(g(X,u))] \tag{4.16}$$

$$= \sup_g \mathbb{E}_{P_{X,Y}}[g(X,Y)] - \mathbb{E}_{P_X}[\log \sum_u \exp(g(X,u))] \tag{4.17}$$

$$\approx \sup_g \frac{1}{N}\sum_{x_i,y_i} \log\left(\frac{\exp(g(x_i,y_i))}{\sum_u \exp(g(x_i,u))}\right), \tag{4.18}$$

where (4.18) is the empirical version of (4.17) and N is the batch size. If g is parameterized by a neural network, (4.18) becomes the optimization criterion for softmax regression with cross-entropy objective. Thus, softmax regression is implicitly performing estimation of the conditional entropy $-H(Y|X)$ using (4.13). Lemma 4.3.1 implies that the optimal $\exp(g^*(x,y)) \propto P_{Y|X}(y|x)$.

■ 4.3.2 Batch Softmax Normalization as Generative DV

Applying (4.13) to $D(P_{X,Y} \parallel P_X U_Y)$ instead of $D(P_{Y|X} \parallel U_Y)$ yields a different optimization problem.

Proposition 4.3.1. $-H(Y|X) = \sup_g \mathbb{E}_{P_{X,Y}}[g(X,Y)] - \log \mathbb{E}_{P_X} \sum_u \exp(g(X,u))$.

Proof. Define U to be a uniform random variable on the support of Y. Then,

$$\begin{aligned} -H(Y|X) &= -\sum_x P_X(x) H(Y|X=x) = \sum_x P_X(x)[D(P_{Y|X=x} \parallel P_U) - H(U)] \\ &= D(P_{X,Y} \parallel P_X P_U) - H(U). \end{aligned}$$

Now, apply DV to $D(P_{X,Y} \parallel P_X P_U)$ while treating the arguments as bivariate distributions:

$$\begin{aligned} &\sup_g \ \mathbb{E}_{P_{X,Y}}[g(X,Y)] - \log \mathbb{E}_{P_X P_U}[\exp(g(X,U))]] - H(U) \\ =&\sup_g \ \mathbb{E}_{P_{X,Y}}[g(X,Y)] - \log \frac{\mathbb{E}_{P_X} \sum_u P_U(u) \exp(g(X,U))}{P_U(u)} \\ =&\sup_g \ \mathbb{E}_{P_{X,Y}}[g(X,Y)] - \log \mathbb{E}_{P_X} \sum_u \exp(g(X,u)) \end{aligned} \tag{4.19}$$

■

In this form, the log is outside of $\mathbb{E}_{P_X}$, which means the softmax normalization term is computed across the entire batch. Although this makes the empirical gradient estimates biased, Lemma 4.3.1 still implies that the optimal $\exp(g^*(x,y)) \propto P_{Y|X}(y|x)$. Interestingly, the normalization term coupled with randomization between batches encourages $\sum_u \exp(g(x,u)) \approx c$, for all x and some $c \in \mathbb{R}$. Section 4.3.6 examines this effect in detail.

■ 4.3.3 Relation to Mutual Information

The entropy of discrete random variables is non-negative. Hence, $I(X;Y) = H(Y) - H(Y|X) \geq -H(Y|X)$, which shows that estimating $-H(Y|X)$ also yields a lower bound on the mutual information $I(X;Y)$. In fact, using $I(X;Y)$ as the starting point results in almost identical derivations. Section C.3 draws some connections to mutual information based self-supervised learning algorithms.

4.3.4 Experimental Setup

For all the self-supervised learning algorithms mentioned in Section C.3, the original papers have extensive experiments on their representation learning performances. One shared observation is that MINE and Deep InfoMax (generative DV) tend to be less stable than InfoNCE and Jensen-Shannon Deep InfoMax (discriminatve DV), especially on easier datasets. Since we have drawn the connection between softmax and DV, we investigate the difference between discriminative DV (softmax) and generative DV on supervised learning. In this subsection, we report performance for classification accuracy, self-normalization, and generalization on the CIFAR-10 dataset [50].

For the experiments, we build on top of the PyTorch [81] code from [61] for ResNet-18 [37] and DenseNet-121 [41]. We train the neural networks for 100 epochs using SGD with momentum of 0.9 and mini-batch size of 128. The initial learning rate is 0.1, and decays to 0.01 and 0.001 at epoch 50 and 75, respectively. We use a weight decay of 0.0005 and random crop and horizontal flip for data augmentation.

4.3.5 Classification Accuracy

To establish generative DV as a competitive objective, we first compare the classification accuracy on CIFAR-10. Table 4.2 reports that for two popular architectures, using generative DV does not degrade performance significantly. Figure 4.3 and Figure 4.4 show that the training progress of the two methods closely track each other.

4.3.6 Self-Normalization

For tasks such as sampling or language modeling, it may be useful to induce self-normalized outputs without needing to apply the softmax normalization. After training is complete, we compute the mean and standard deviation of $\sum_u \exp(g(X, u))$, where $g(X, u)$ is the u-th final layer activation for input random variable X before any output nonlinearity

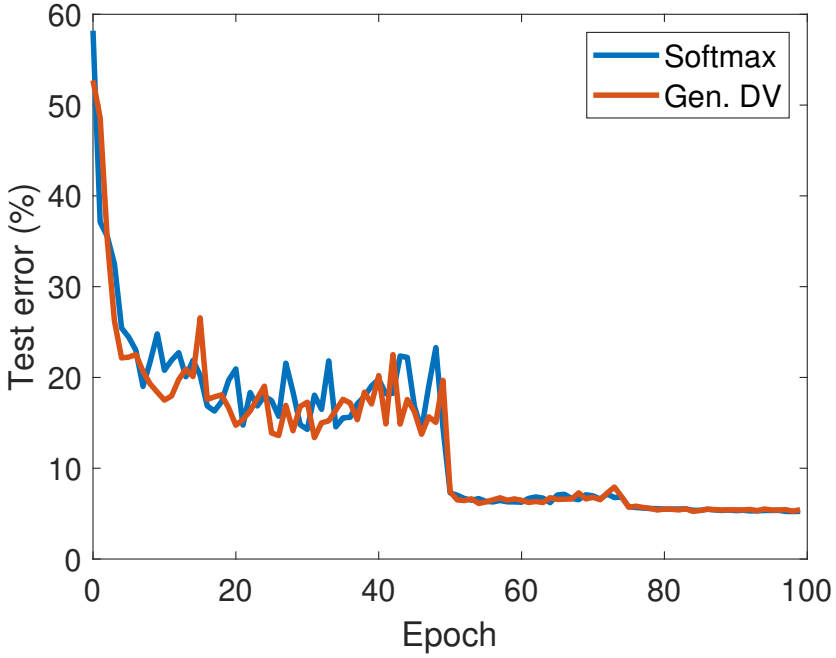

Figure 4.3: ResNet-18 test error rate (%) as a function of training epoch.

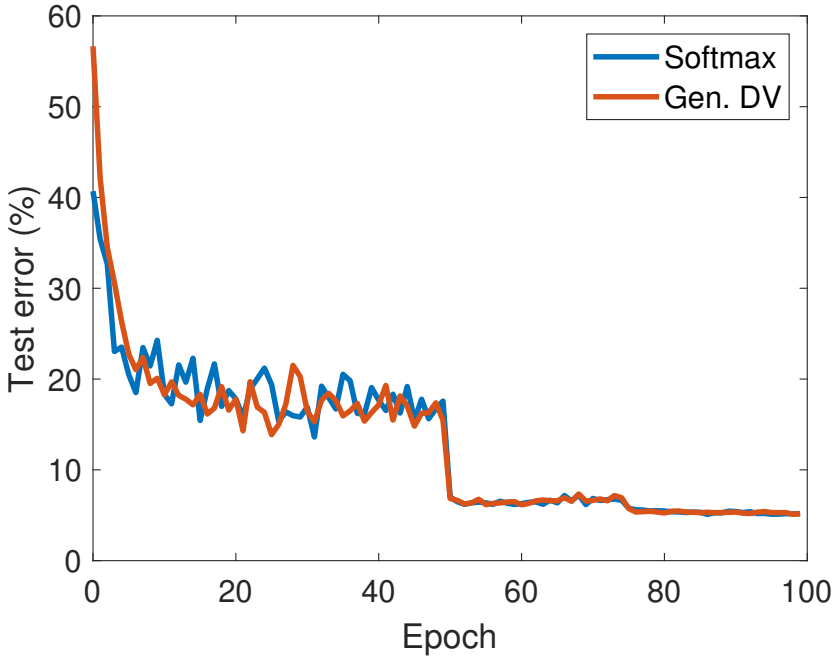

Figure 4.4: DenseNet-121 test error rate (%) as a function of training epoch.

Table 4.2: Classification error rates (%) for softmax and generative DV objectives on the CIFAR-10 dataset.

Network	Softmax	Gen. DV
ResNet-18	5.23	5.27
DenseNet-121	5.1	5.11

Table 4.3: Mean±standard deviation for $\sum_u \exp(g(X,u))$ on the CIFAR-10 training and test set.

Network	Data	Softmax	Gen. DV
ResNet-18	Train	$3.8e4 \pm 8.9e4$	1.49 ± 0.03
ResNet-18	Test	$5.2e4 \pm 1.5e5$	1.46 ± 0.15
DenseNet-121	Train	$1.5e5 \pm 1.0e6$	15.76 ± 0.43
DenseNet-121	Test	$3.1e5 \pm 1.8e6$	15.75 ± 0.89

(e.g., softmax). Table 4.3 shows that the softmax activation does not perform any self-normalization. In fact, the standard deviation of $\sum_u \exp(g(X,u))$ is consistently larger than its mean. On the other hand, generative DV controls that quantity to be tight around the mean across all input, even for unseen data (i.e., the test set).

■ 4.3.7 Generalization

We sweep the amount of training data from 6.125% to 100% of the CIFAR-10 training set and compare the sensitivity of softmax and generative DV to the availability of data. Figure 4.5 shows generative DV noticeably underperforming softmax when fewer than 25% of the training set is used, which appears to corroborate the trends observed in [79] and [38].

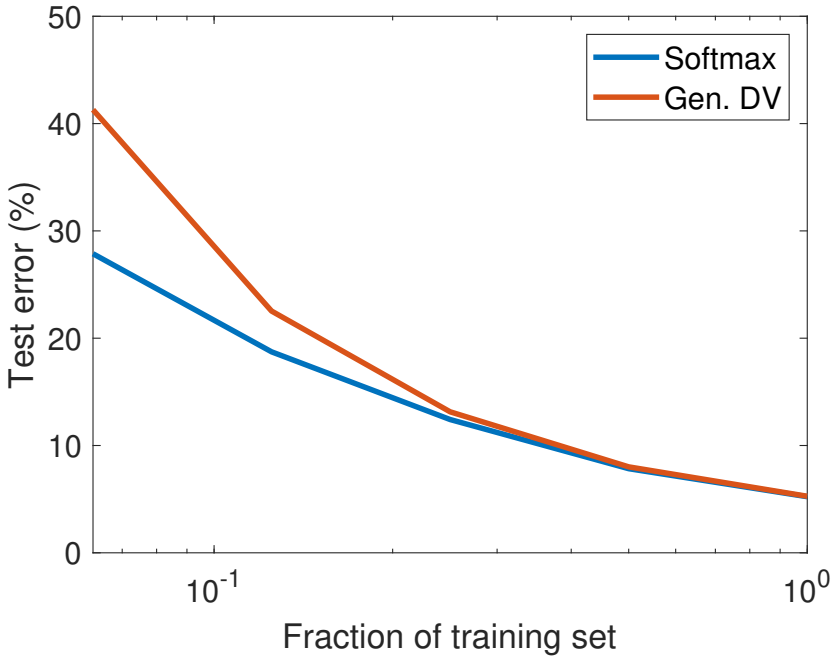

Figure 4.5: Generalization performance for ResNet-18 as a function of the fraction of data used for training. Softmax (blue) outperforms generative DV (red) as the availability of data decreases.

■ 4.4 Wireless Interference Signal Constellation Modeling

■ 4.4.1 Communication Background

As the number of wireless devices increase, they inevitably interfere with each other and degrade overall performance. Since the focus of this work is on learning, we choose a simple, match filtered and sampled baseband model for $M = 2$ transmitters (TX) interfering with each other at N receivers (RX):

$$Y(n) = H_1 X_1(n) + H_2 X_2(n) + Z(n), \tag{4.20}$$

where $Y(n), H_1, H_2 \in \mathbb{C}^N$, $X_1(n), X_2(n) \in \mathbb{C}$, and $Z(n) \sim \mathcal{CN}(0,1)$. $X_1(n)$ and $X_2(n)$ are structured. For the examples in this section, we choose $X_1(n)$ to be binary phase-shift keying (BPSK) and $X_2(n)$ to be quadrature amplitude modulation (16-QAM) with the following constellations:

$$X_1(n) \in \{1, -1\} \tag{4.21}$$

$$X_2(n) \in \{a + bi \mid a, b \in \{\frac{3}{\sqrt{10}}, \frac{1}{\sqrt{10}}, -\frac{1}{\sqrt{10}}, -\frac{3}{\sqrt{10}}\}\}. \tag{4.22}$$

The goal of X_1's receiver is to detect whether 1 or -1 is sent by computing the log likelihood ratio (LLR):

$$\log\left(\frac{P(X_1(n) = 1|Y)}{P(X_1(n) = -1|Y)}\right), \tag{4.23}$$

and there are algorithms with varying degrees of complexity (see Table 4.4) to approximate (4.23) in the presence of interference:

1. Ignore the structure in $H_2X_2(n)$, treat it as additional Gaussian noise, and perform binary LLR calculation on $X_1(n)$

2. Solve for $X_1(n)$ and $X_2(n)$ using regularized least squares (minimum mean square error) and round to the nearest neighbor in the constellation

3. Perform maximum a posteriori (MAP) detection by marginalizing out $X_2(n)$

Table 4.4: Advantages and disadvantages of three different receiver designs in increasing complexity.

Algorithm	Advantages	Disadvantages
1	Very simple to implement. No need to modify RX. Only needs to estimate channel for primary user (H_1).	Poor performance. Accuracy goes to 50% as interference power increases.
2	Polynomial time complexity in M and N.	Needs to estimate channel for all users. Does not perform well if $N < M$.
3	Optimal in term of error probability.	Needs to estimate channel for all users. Time complexity scales with the product of every user's constellation size (exponential in M).

■ 4.4.2 Detection Using Neural Networks

There has been recent progress on using neural networks to perform channel decoding [46], modulation classification [68], and multiple-input and multiple-output (MIMO) detection [92]. Knowing the full model (4.20), we can also generate data to train a bank of classifiers indexed by the channel state information (H_1, H_2) to detect for $X_1(n)$. Figure 4.6 shows one example channel state's generated dataset. When deployed, the receiver estimates (H_1, H_2) and computes (4.23) using the pre-trained classifier with the closest channel.

■ 4.4.3 Adapting Between Different Channels

In an environment where the interference is slowly varying, the receiver can exploit the structure in the signal to adapt from a source model trained under (H_1, H_2) to any target

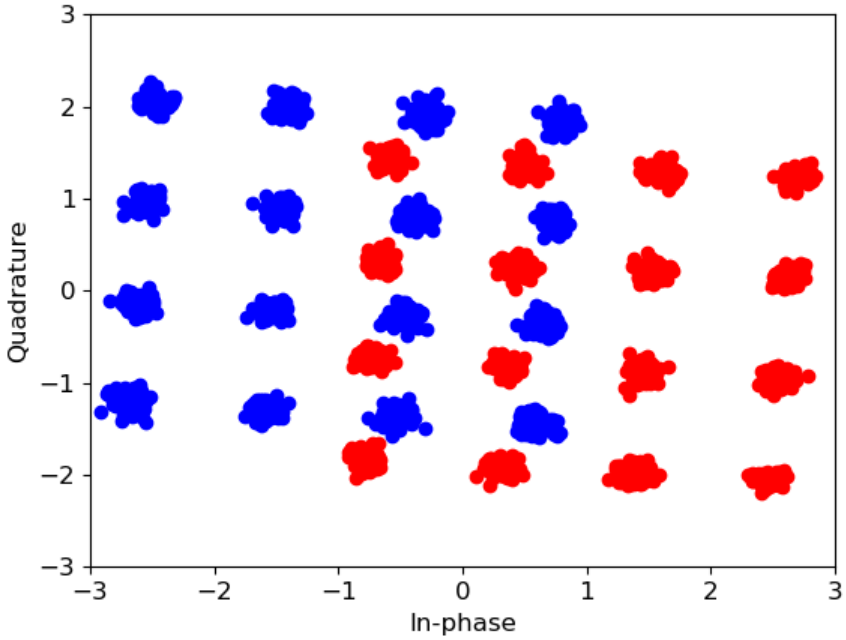

Figure 4.6: Scatterplot of observed data for one BPSK user and one 3x (4.8dB) stronger 16-QAM interferer. Red and blue correspond to the symbols 1 and −1, respectively.

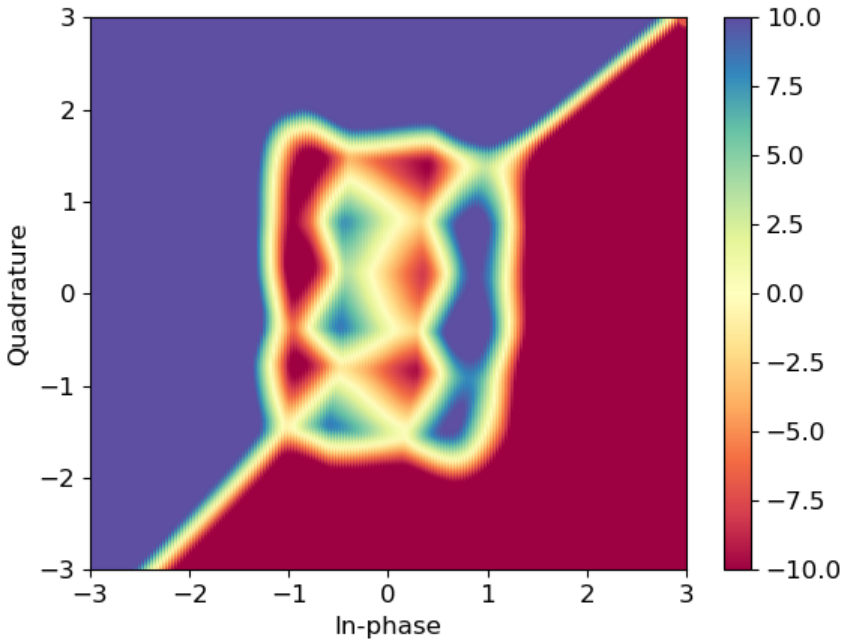

Figure 4.7: Neural classifier's output LLR for the example dataset in Figure 4.6.

channel realization (H_1', H_2). For simplicity, we focus on memoryless receivers and drop the dependence on n. Start by rewriting (4.23) as

$$\begin{aligned}\log\left(\frac{P(X_1=1|Y)}{P(X_1=-1|Y)}\right) &= \log\left(\frac{P(X_1=1)P(Y|X_1=1)}{P(X_1=-1)P(Y|X_1=-1)}\right)\\ &= \log\left(\frac{P(X_1=1)}{P(X_1=-1)}\right)+\log\left(\frac{P(Y|X_1=1)}{P(Y|X_1=-1)}\right).\end{aligned}$$

Typically, $P(X_1=1)=P(X_1=-1)=0.5$ in order to transmit the maximum amount of information, which makes the 1st term 0. By the linear relationship in (4.20), the 2nd term can be written as

$$\log\left(\frac{P(H_2X_2+Z=Y-H_1')}{P(H_2X_2+Z=Y+H_1')}\right). \tag{4.24}$$

In other words, having a trained density estimator for the random variable $H_2X_2(n)+Z(n)$ allows us to compute the LLR for any target channel realization H_1'. Although generative models such as variational autoencoders [48] can effectively estimate lower bounds of general densities, using them involves training a separate model purely for adaptation. We want to show that the source model can be used to approximate (4.24).

Based on the same derivation as (4.24), the source model f_S computes

$$f_S(Y)=\log\left(\frac{P(H_2X_2+Z=Y-H_1)}{P(H_2X_2+Z=Y+H_1)}\right) \tag{4.25}$$

and outputs

$$f_S(Y-H_1'+H_1)=\log\left(\frac{P(H_2X_2+Z=Y-H_1')}{P(H_2X_2+Z=Y-H_1'+2H_1)}\right) \tag{4.26}$$

$$f_S(Y+H_1'+H_1)=\log\left(\frac{P(H_2X_2+Z=Y+H_1')}{P(H_2X_2+Z=Y+H_1'+2H_1)}\right). \tag{4.27}$$

for the chosen shifted inputs. Although (4.26) and (4.27) cannot be combined to compute

(4.25) exactly, an approximation is possible. Intuitively, (4.26) and (4.27) guess that 1 and -1 are the transmitted target symbol, respectively, and subtract the guessed symbols' contributions ($\pm H_1'$) from Y. One of the two shifts is correct and successfully cancels the target symbol's contribution while the other introduces an erroneous bias of $\pm 2H_1'$. After adding an arbitrary source symbol H_1 to align with the source domain, the correct shift lands in a region of the input space with high likelihood. Conversely, it is highly unlikely to be able to bias that point by $\pm 2H_1'$ and remain in a region with equally high likelihood.

The problem remains that classifiers are trained to estimate the conditional probability $P(X_1|Y)$ as opposed to the data likelihood $P(Y)$. For example, in the gap regions with no datapoints in Figure 4.6, $P(Y)$ should be small, but Figure 4.7 shows that f_S outputs LLRs with large absolute values. To remedy that, we can augment the source training set with noise data drawn from

$$Y \sim \text{unif}(\min(H_1X_1 + H_2X_2), \max(H_1X_1 + H_2X_2))$$
$$X_1 \sim \text{Rademacher}$$

Figure 4.8 shows the classifier's output LLR when trained using the same (augmented) dataset as Figure 4.7. This trick allows us to approximate the target LLR as

$$\log\left(\frac{P(Y|X_1=1)}{P(Y|X_1=-1)}\right) = \log\left(\frac{P(H_2X_2+Z=Y-H_1')}{P(H_2X_2+Z=Y+H_1')}\right) \tag{4.28}$$
$$\approx f_S(Y-H_1'+H_1) - f_S(Y+H_1'+H_1) \tag{4.29}$$

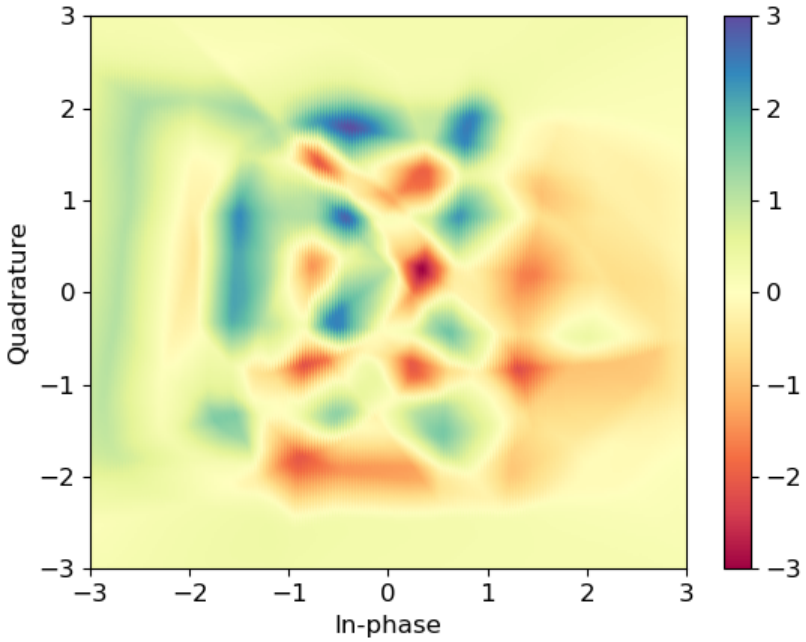

Figure 4.8: Neural network classifier's output LLR for the example dataset in Figure 4.6 augmented with noise samples.

Chapter 5

Conclusion

In this thesis, we studied information-theoretic approximations for machine learning algorithms. In particular, we formulated and designed techniques based on local information geometry to 1) cluster joint probability distributions in an end-to-end fashion as opposed to embedding then k-means, and 2) compute principled feature relevance scores before target task training to improve generalization for task personalization. We also modified and improved variational approximations of statistical divergences to 1) ameliorate mode collapse in GAN, 2) automatically learn to capture subsets of a data distribution using WGAN, and 3) self-normalize in the final layer of neural classifiers. Finally, we proposed using generative models for wireless communication in the presence of interference and presented a heuristic on how to approximate the generative models using a classifier.

There are many future directions related to the topics presented in this thesis, including:

- Extend the local information geometric framework beyond single layer classifiers to show how information flows in multilayer perceptions (MLP) and recurrent neural networks (RNN)
- Rigorously analyze the convergence properties of the proposed GAN / WGAN structures and conduct experiments on large scale tasks
- Use the self-normalization property from Section 4.3 to decrease the need for noise sampling for the classifier from Section 4.4

Appendix A

Supplementary Results from Chapter 2

A.1 Alternating Conditional Expectation With Side Information

A.1.1 Information Vector Derivation

To show the relation between DTM factorization with side information and the alternating conditional expectation (ACE) algorithm, consider the following rank-k matrix factorization problem:

$$\min_{\substack{\psi\in\mathbb{R}^{|\mathcal{Y}|\times k}\\ \phi\in\mathbb{R}^{|\mathcal{X}|\times k}}} \left\|B-\psi\phi^T\right\|_F^2+\lambda_\psi \operatorname{tr}(\psi^T A\psi)+\lambda_\phi \operatorname{tr}(\phi^T C\phi), \tag{A.1}$$

where B is the DTM and A and C are additional loss matrices. With the appropriate choice of B vs. B^T, (A.1) is symmetric between ψ and ϕ and all derivations apply to both variables.

When ϕ is fixed, the problem is convex and the solution occurs when the gradient is

zero.

$$\begin{aligned}
&\frac{\partial}{\partial \psi}\left[\left\|B-\psi\phi^T\right\|_F^2+\lambda_\psi \operatorname{tr}(\psi^T A\psi)+\lambda_\phi \operatorname{tr}(\phi^T C\phi)\right] \\
=&\frac{\partial}{\partial \psi}\left[\operatorname{tr}\left(\left(B-\psi\phi^T\right)\left(B^T-\phi\psi^T\right)\right)\right]+\lambda_\psi\left(A\psi+A^T\psi\right) \\
=&\frac{\partial}{\partial \psi}\left[\operatorname{tr}\left(BB^T-2\psi\phi^T B^T+\psi\phi^T\phi\psi^T\right)\right]+\lambda_\psi\left(A\psi+A^T\psi\right) \\
=&-2B\phi+2\psi\phi^T\phi+\lambda_\psi\left(A\psi+A^T\psi\right)=0. \qquad \text{(A.2)}
\end{aligned}$$

For general matrix A, (A.2) is an instance of a Sylvester equation, which is often studied in numerical linear algebra and control theory. Solving it using the Bartels–Stewart algorithm [4] has cubic time complexity.

For diagonal A, the objective $\lambda_\psi \operatorname{tr}(\psi^T A\psi)$ reduces to

$$\lambda_\psi \sum_{i=1}^{|\mathcal{Y}|} A_{i,i}\left\|\psi(i,:)\right\|_2^2, \qquad \text{(A.3)}$$

where $\psi(i,:)$ denotes the i-th row of ψ. It is clear to see that (A.3) has the form of L_2 regularization with an additional row-specific multiplier $A_{i,i}$. To decouple the $A_{i,i}$'s, we can combine (A.2) and (A.3) row-by-row

$$-B(i,:)\phi+\psi(i,:)\phi^T\phi+\lambda_\psi A_{i,i}\psi(i,:)=-B(i,:)\phi+\psi(i,:)\left(\phi^T\phi+\lambda_\psi A_{i,i}I_k\right)=0 \qquad \text{(A.4)}$$

to obtain the row-wise stationary points:

$$\psi(i,:)=B(i,:)\phi\left(\phi^T\phi+\lambda_\psi A_{i,i}I_k\right)^{-1}. \qquad \text{(A.5)}$$

A.1.2 Conditional Expectation Operator of Embedding Functions

To show that ACE can solve (A.5), first observe that $\left(\phi^T\phi + \lambda_\psi A_{i,i} I_k\right)^{-1}$ in (A.5) only performs a row-wise normalization. Hence, all we need to show are that $\psi = B\phi$ and $\phi = B^T\psi$ are equivalent to conditional expectation operations. Start by defining $f \triangleq [P_Y]^{-\frac{1}{2}}\psi$ and $g \triangleq [P_X]^{-\frac{1}{2}}\phi$ as the desired embedding functions for Y and X, respectively. Then,

$$[P_Y]^{\frac{1}{2}} f = \psi = B\phi = \left([P_Y]^{-\frac{1}{2}} P_{Y,X} [P_X]^{-\frac{1}{2}}\right)\left([P_X]^{\frac{1}{2}} g\right) \tag{A.6}$$

$$[P_X]^{\frac{1}{2}} g = \phi = B^T\psi = \left([P_X]^{-\frac{1}{2}} P_{Y,X}^T [P_Y]^{-\frac{1}{2}}\right)\left([P_Y]^{\frac{1}{2}} f\right) \tag{A.7}$$

simplify to

$$f = [P_Y]^{-1} P_{Y,X} \cdot g = P_{Y|X} \cdot g \tag{A.8}$$

$$g = [P_X]^{-1} P_{Y,X}^T \cdot f = P_{X|Y} \cdot f. \tag{A.9}$$

A.2 Word Embedding

To test the performance of DTM factorization on a real world dataset, we focus on the Microsoft Research Sentence Completion Challenge [104]. The training corpus contains 522 classic literature texts. The test set consists of 1040 sentences from Sherlock Holmes novels, where one word is replaced with a blank. Similar to the Graduate Record Examination (GRE), there are five candidate words for the algorithm to pick from (see Figure A.1). However, the candidates are selected to have roughly equal unigram probability to prevent models from exploiting raw word count information.

A.2.1 Preprocessing

1. Remove hyphens and quotation marks.

2. Start every sentence on a new line.

10) The sun had set and _____ was settling over the moor.
a) dusk
b) mischief
c) success
d) disappointment
e) laughter

Figure A.1: Example of a sentence completion question from the Microsoft Research Sentence Completion Challenge dataset.

3. Remove numbers and proper nouns and replace them with special indicators words.

4. Remove punctuations and lowercase the remaining words.

5. Remove determiners and prepositions such as the, after, where, etc.

6. Use Natural Language Toolkit [9] WordNetLemmatizer to aggregate different grammatical representations of the same root word (e.g., quick, quicker, quickly).

■ A.2.2 Constructing $P_{Y,X}$

We iterate over each word in a sentence. At word w_i, we add a co-occurrence count to $(w_i, w_{i+1}), \ldots, (w_i, w_{i+l})$ and $(w_{i+1}, w_i), \ldots, (w_{i+l}, w_i)$, where l is the window length. The window can also be weighted to deemphasize words that are far from each other, but we do not report the performance of that scheme.

■ A.2.3 Choosing the Most Likely Word

Define $f : \mathcal{W} \to \mathbb{R}^k$, where $\mathcal{W}$ is the set of words in the vocabulary and k is the number of word embedding dimensions. Recall that f is fully determined by $[P_Y]^{-\frac{1}{2}}U$, which has unit variance in all dimensions. Using f as the word embedding assumes the same semantic importance across all k dimensions. We can also use $[P_Y]^{-\frac{1}{2}}U\sqrt{\Sigma}$ to deemphasize the later dimensions with lower correlation.

Table A.1: Accuracy of DTM factorization with different weighting functions on the Microsoft Research Sentence Completion Challenge.

SCHEME	WEIGHTING	ACCURACY
(A.10)	$[P_Y]^{-\frac{1}{2}}U$	53.94%
(A.10)	$[P_Y]^{-\frac{1}{2}}U\sqrt{\Sigma}$	51.06%
(A.11)	$[P_Y]^{-\frac{1}{2}}U$	51.82%
(A.11)	$[P_Y]^{-\frac{1}{2}}U\sqrt{\Sigma}$	49.81%
(A.12)	$[P_Y]^{-\frac{1}{2}}U$	52.40%
(A.12)	$[P_Y]^{-\frac{1}{2}}U\sqrt{\Sigma}$	50.67%

To choose the answer from the candidate set, we focus on three schemes based on the cosine distance and report the accuracy in Table A.1.

1. Sum the individual cosine distances of every word in the sentence with the candidate:

$$\text{score}(\text{candidate}) = \sum_{w \in \text{sentence}} \cos\left(\frac{f(\text{candidate}) \cdot f(w)}{\|f(\text{candidate})\| \cdot \|f(w)\|}\right). \tag{A.10}$$

2. Sum the individual cosine distances of every word in a length $2l$ window with the candidate word:

$$\text{score}(\text{candidate}) = \sum_{w \in \text{window}} \cos\left(\frac{f(\text{candidate}) \cdot f(w)}{\|f(\text{candidate})\| \cdot \|f(w)\|}\right). \tag{A.11}$$

3. Sum the embedding of every word in the sentence. Then use the cosine distance of that sum with the candidate word:

$$\text{score}(\text{candidate}) = \cos\left(\frac{f(\text{candidate}) \cdot \sum_{\text{word} \in \text{sentence}} f(w)}{\|f(\text{candidate})\| \cdot \left\|\sum_{\text{word} \in \text{sentence}} f(w)\right\|}\right). \tag{A.12}$$

Appendix B

Supplementary Results from Chapter 3

B.1 Synthetic Data Experiment

We also test on a discrete synthetic dataset, where $X \in \{0, \ldots, 59\}$ and $Y \in \{0, 1\}$. P_Y is uniform and $P_{X|Y}$ is generated by entry-wise sampling from unif$(0, 1)$ and normalizing to sum to 1. The numpy seed used is 10. The input is first converted to one-hot vector representation and then fed into a two layer neural network with softmax output activation and cross-entropy loss. The network is trained using Adam (learning rate 0.01) on 8000 samples and tested on 2000 samples. Figure B.1 and Figure B.2 shows the accuracy and loss, which has similar trends as Figure 3.6 and 3.7. Because it is a discrete problem with known distribution, we also plot the ideal test set performance of a classifier trained on the training set and test set, respectively.

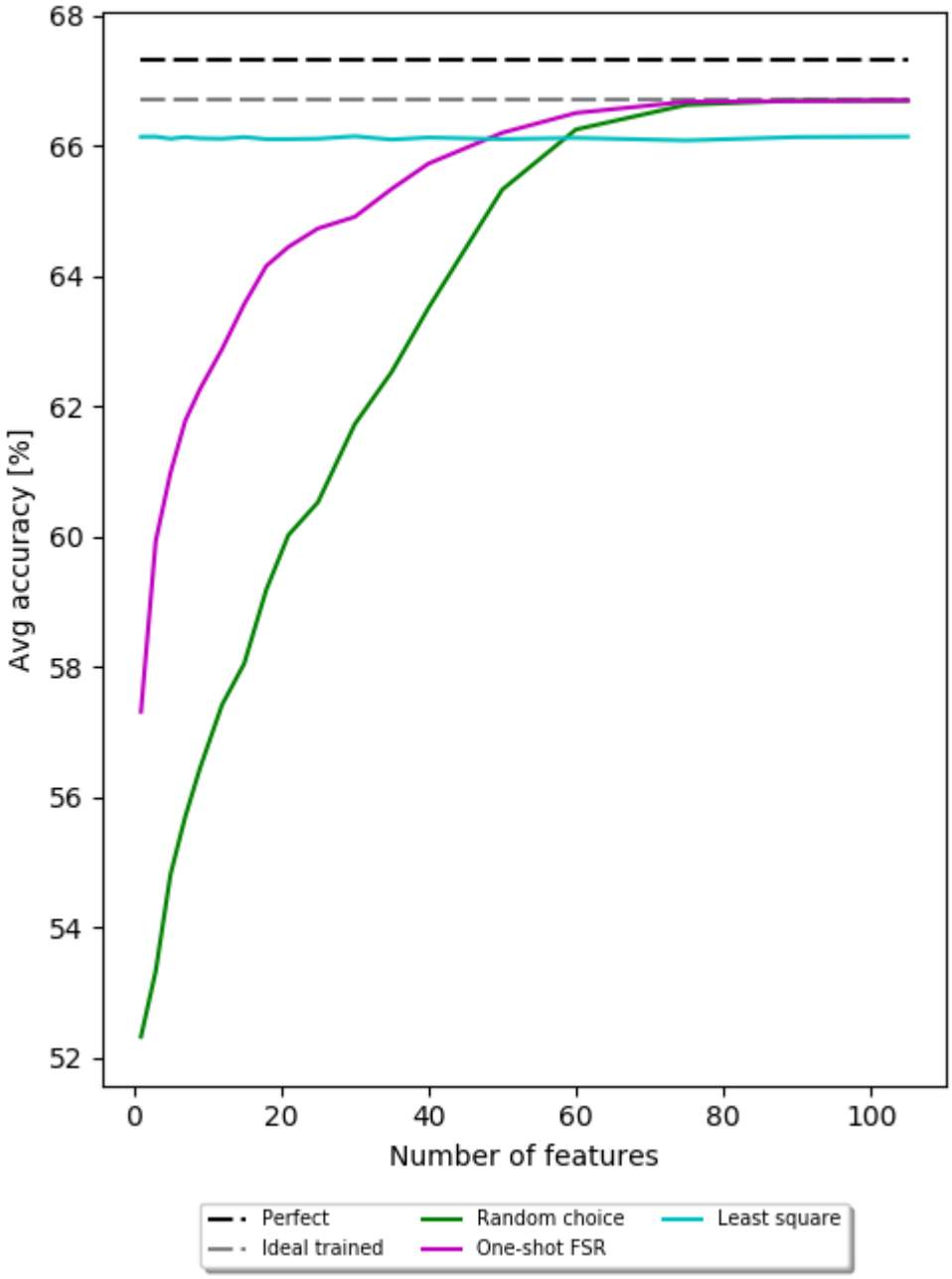

Figure B.1: Plot of accuracy vs. number of feature chosen for randomly selecting features, one-shot FSR, and the least square solution. Results are averaged over 50 runs.

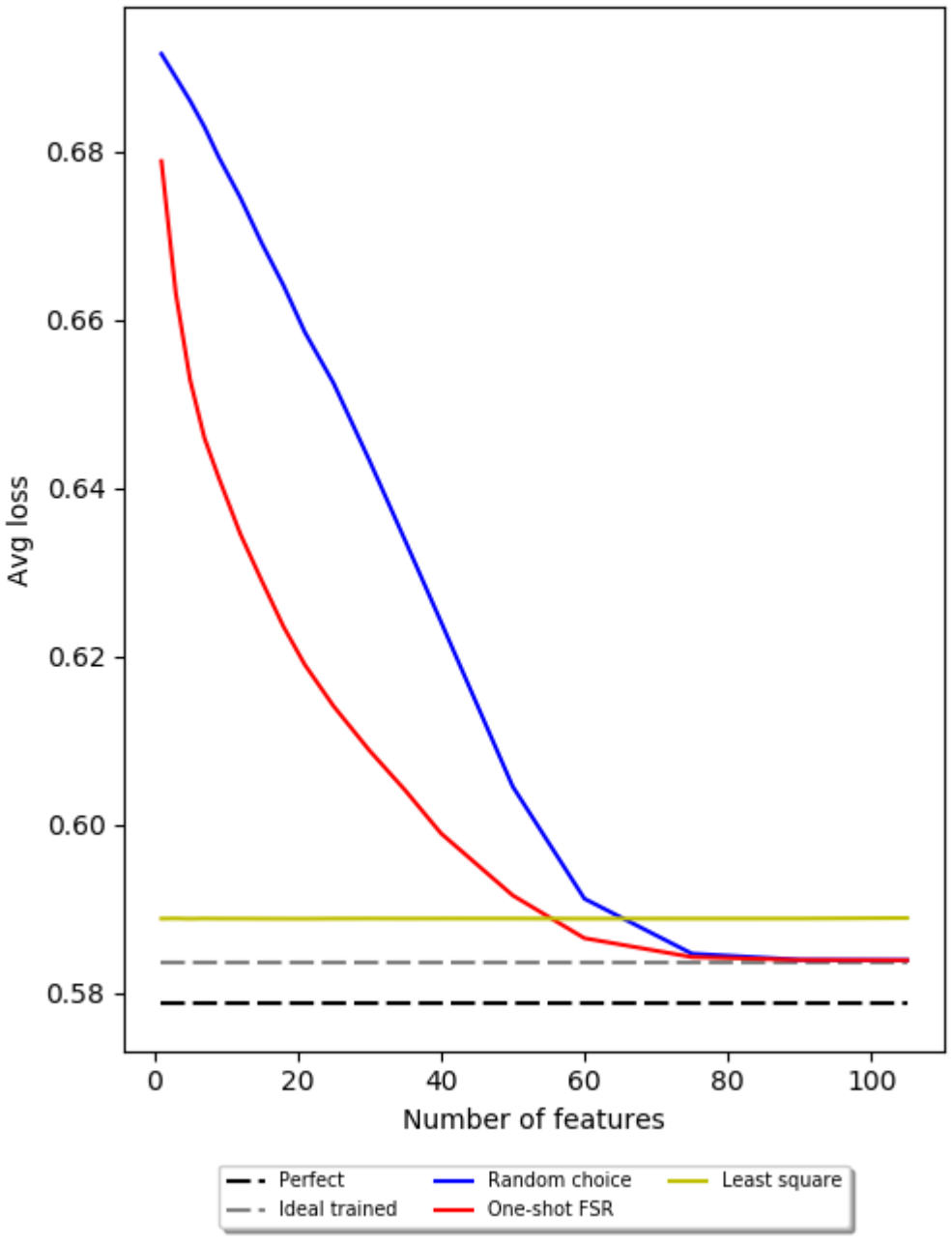

Figure B.2: Plot of loss vs. number of feature chosen for randomly selecting features, one-shot FSR, and the least square solution. Results are averaged over 50 runs.

Appendix C

Supplementary Results from Chapter 4

C.1 Derivation of Subset Kantorovich-Rubinstein Duality for PMFs

Recall the formulation of the subset Wasserstein distance:

$$
\begin{aligned}
W_1^{(S)}(P,Q) = \min_{\gamma \in \mathcal{P}_{\mathcal{U},\mathcal{U}}} & \sum_{\substack{x \in \mathcal{U} \\ y \in \mathcal{U}}} d(x,y)\gamma(x,y) \\
\text{s.t.} & \sum_{y \in \mathcal{U}} \gamma(x,y) \leq \frac{P(x)}{\alpha}, \forall x \in \mathcal{U}, \\
& \sum_{x \in \mathcal{U}} \gamma(x,y) = Q(y), \forall y \in \mathcal{U}.
\end{aligned} \tag{C.1}
$$

This is a linear program (LP). It is bounded because both the alphabet $\mathcal{U}$ and distance function d are defined to be finite and bounded (see Definition 4.2.1). It is feasible because a coupling that satisfies the regular Wasserstein constraints is also a solution to (C.1). Thus, the dual LP is bounded, feasible, and strong duality holds [8].

After direct application of duality, the dual of (C.1) is

$$\begin{aligned} \max_{f,g} \ & \sum_{x \in \mathcal{U}} f(y)Q(y) + \frac{1}{\alpha} \sum_{x \in \mathcal{U}} g(x)P(x) \\ \text{s.t. } & |f(y) - g(x)| \leq d(x,y), \forall x, y \in \mathcal{U}, \\ & g(x) \leq 0, \forall x \in \mathcal{U}. \end{aligned} \tag{C.2}$$

All that is left is to show $g = -f$ via proof by contradiction. Observe that for $x, y = c$, $d(c,c) = 0$. Then, $g(c) \neq -f(c)$ (i.e., the inequality not being tight) combined with complementary slackness imply that $\gamma(c,c) = 0$ in the primal problem. We can focus on the case where $P(c), Q(c) > 0$ because otherwise, $\gamma(c,c)$ can be eliminated from the set of primal variables.

Qualitatively, $\gamma(c,c) = 0$ and $P(c), Q(c) > 0$ imply that even though at least $|P(c) - Q(c)|$ amount of mass exists in both P and Q at c, the transportation plan still chooses to move that amount of mass elsewhere. Then, for some $a, b \neq c$ such that $\gamma(a,c), \gamma(c,b) > 0$, $d(a,b) + d(c,c) \leq d(a,c) + d(c,b)$ by the triangle inequality. Reassigning the transportation plan to move an additional ϵ amount of mass from a to b instead of a to c and c to b would achieve a net saving. This means the primal cannot be optimal when $g \neq f$ in the dual.

C.2 Proof of Lemma 4.3.1

Let $g(X) \triangleq c + \log\left(\frac{P(X)}{Q(X)}\right)$. We will show that it achieves the maximum.

$$\begin{aligned}
\mathbb{E}_P[g(x)] - \log \mathbb{E}_Q[\exp(g(x))] &= \mathbb{E}_P\left[c + \log\left(\frac{P(X)}{Q(X)}\right)\right] - \log \mathbb{E}_Q\left[\exp\left(c + \log\left(\frac{P(X)}{Q(X)}\right)\right)\right] \\
&= c + \mathbb{E}_P\left[\log\left(\frac{P(X)}{Q(X)}\right)\right] - c - \log \mathbb{E}_Q\left[\exp\left(\log\left(\frac{P(X)}{Q(X)}\right)\right)\right] \\
&= \mathbb{E}_P\left[\log\left(\frac{P(X)}{Q(X)}\right)\right] - \log \mathbb{E}_Q\left[\frac{P(X)}{Q(X)}\right] \\
&= \mathbb{E}_P\left[\log\left(\frac{P(X)}{Q(X)}\right)\right] = D(P \parallel Q).
\end{aligned}$$

■

C.3 Connections to Self-Supervised Learning

For tasks where it is expensive to obtain labeled examples, the self-supervised learning paradigm defines proxy tasks that enable neural networks to capture the structure of the underlying data distribution. These proxy tasks are still prediction problems, which means they can be related to DV and converted into the generative DV form. This section shows three such connections.

Often, these algorithms learn by contrasting data drawn from the true joint distribution against data drawn from a noise distribution. Notation-wise, we use X and Y as the random variables for the true data and $Z \sim \text{Bernoulli}(\frac{1}{k+1})$ as the indicator of whether to draw from $P_{X,Y}$ or a problem-specific noise distribution $W_{X,Y}$. We define $\tilde{X}$ and $\tilde{Y}$ as the random variables with mixture distribution $\frac{1}{k+1}P_{X,Y} + \frac{k}{k+1}W_{X,Y}$.

C.3.1 Noise Contrastive Estimation and Negative Sampling for Word Embedding

One of the most successful application of self-supervised learning is word embedding [71, 83]. By learning a low dimensional model to approximate the word-context conditional

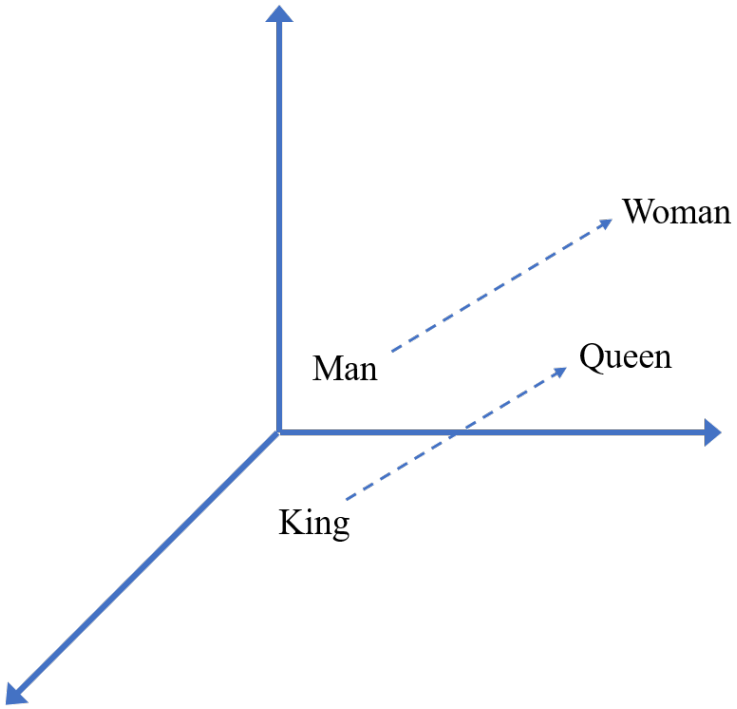

Figure C.1: The canonical example of vector arithmetic and analogy in the word embedding space.

distributions, words that have similar conditional distributions are mapped to be close[1]. Figure C.1 shows the famous example, where the semantics of male vs. female is captured by one direction in the embedding space. Two popular methods for learning word embedding are noise contrastive estimation (NCE) [34, 72] and negative sampling (NS).

Proposition C.3.1. Let $\tilde{X}$ and $\tilde{Y}$ be the input and output word random variable with k noise (negative) samples per true sample. Both the NCE and NS objectives implicitly estimate $-H(Z|\tilde{X},\tilde{Y})$, which is a lower bound on $I(\tilde{X},\tilde{Y};Z)$.

Proof. Recall that Z is an indicator variable that is 1 when we draw from the joint word-

[1] This is known as the distributional hypothesis.

context distribution and 0 when we draw from the noise distribution. For word embedding, the noise is typically chosen to be $P_X W_Y$ for some specified W_Y. If we draw k noise samples per true sample, then

$$\begin{aligned} Q_{\tilde{X},\tilde{Y},Z=1} &= \frac{1}{k+1} P_{X,Y} \\ Q_{\tilde{X},\tilde{Y},Z=0} &= \frac{k}{k+1} P_X W_Y. \end{aligned} \tag{C.3}$$

To start, expand $I(\tilde{X},\tilde{Y};Z)$ and apply Theorem 4.3.1 to $D(Q_{Z|\tilde{X},\tilde{Y}} \parallel Q_Z)$:

$$\begin{aligned} I(\tilde{X},\tilde{Y};Z) &= D(Q_{\tilde{X},\tilde{Y},Z} \parallel Q_{\tilde{X},\tilde{Y}} Q_Z)) = \mathbb{E}_{Q_{\tilde{X},\tilde{Y}}} D(Q_{Z|\tilde{X},\tilde{Y}} \parallel Q_Z) \\ &= \sup_g \ \mathbb{E}_{Q_{\tilde{X},\tilde{Y}}} \left[\mathbb{E}_{Q_{Z|\tilde{X},\tilde{Y}}} g(\tilde{X},\tilde{Y},Z) - \log \left(\mathbb{E}_{Q_Z} \exp(g(\tilde{X},\tilde{Y},Z)) \right) \right] \\ &= \sup_g \ \mathbb{E}_{Q_{\tilde{X},\tilde{Y},Z}} g(\tilde{X},\tilde{Y},Z) - \mathbb{E}_{Q_{\tilde{X},\tilde{Y}}} \log \left(\mathbb{E}_{Q_Z} \exp(g(\tilde{X},\tilde{Y},Z)) \right). \end{aligned} \tag{C.4}$$

Now, use (C.3) to rewrite the expectations in (C.4) as

$$\begin{aligned} &\sup_g \ \frac{1}{k+1} \mathbb{E}_{P_X} \mathbb{E}_{P_{Y|X}} g(X,Y,1) + \frac{k}{k+1} \mathbb{E}_{P_X} \mathbb{E}_{W_Y} g(X,Y,0) \\ &- \frac{1}{k+1} \mathbb{E}_{P_X} \mathbb{E}_{P_{Y|X}} \log \left(\mathbb{E}_{Q_Z} \exp(g(X,Y,Z)) \right) - \frac{k}{k+1} \mathbb{E}_{P_X} \mathbb{E}_{W_Y} \log \left(\mathbb{E}_{Q_Z} \exp(g(X,Y,Z)) \right), \end{aligned}$$

and combine terms:

$$\begin{aligned} \sup_g \ \mathbb{E}_{P_X} \Bigg[&\frac{1}{k+1} \mathbb{E}_{P_{Y|X}} \log \left(\frac{\exp[g(X,Y,1)]}{\mathbb{E}_{Q_Z} \exp(g(X,Y,Z))} \right) \\ &+ \frac{k}{k+1} \mathbb{E}_{W_Y} \log \left(\frac{\exp[g(X,Y,0)]}{\mathbb{E}_{Q_Z} \exp(g(X,Y,Z))} \right) \Bigg]. \end{aligned}$$

Finally, write out the terms of $\mathbb{E}_{Q_Z}$, and rearrange some constants:

$$\begin{aligned}
&\sup_g \ \mathbb{E}_{P_X}\left[\frac{1}{k+1}\mathbb{E}_{P_{Y|X}}\log\left(\frac{\exp[g(X,Y,1)]}{\frac{1}{k+1}\exp(g(X,Y,1))+\frac{k}{k+1}\exp(g(X,Y,0))}\right)\right.\\
&\left.+\frac{k}{k+1}\mathbb{E}_{W_Y}\log\left(\frac{\exp[g(X,Y,0)]}{\frac{1}{k+1}\exp(g(X,Y,1))+\frac{k}{k+1}\exp(g(X,Y,0))}\right)\right]\\
=&\sup_g \ \mathbb{E}_{P_X}\left[\frac{1}{k+1}\mathbb{E}_{P_{Y|X}}\log\left(\frac{\frac{1}{k+1}\exp[g(X,Y,1)]}{\frac{1}{k+1}\exp(g(X,Y,1))+\frac{k}{k+1}\exp(g(X,Y,0))}\right)\right.\\
&\left.+\frac{k}{k+1}\mathbb{E}_{W_Y}\log\left(\frac{\frac{k}{k+1}\exp[g(X,Y,0)]}{\frac{1}{k+1}\exp(g(X,Y,1))+\frac{k}{k+1}\exp(g(X,Y,0))}\right)\right]\\
&-\frac{1}{k+1}\log\left(\frac{1}{k+1}\right)-\frac{k}{k+1}\log\left(\frac{k}{k+1}\right)\\
=&\sup_g \ \mathbb{E}_{P_X}\left[\frac{1}{k+1}\mathbb{E}_{P_{Y|X}}\log\left(\frac{\exp[g(X,Y,1)]}{\exp(g(X,Y,1))+k\exp(g(X,Y,0))}\right)\right.\\
&\left.+\frac{k}{k+1}\mathbb{E}_{W_Y}\log\left(\frac{k\exp[g(X,Y,0)]}{\exp(g(X,Y,1))+k\exp(g(X,Y,0))}\right)\right]+H(Z). \qquad \text{(C.5)}
\end{aligned}$$

If we move $H(Z)$ to the other side, the remaining optimization estimates $I(\tilde{X},\tilde{Y};Z) - H(Z) = H(Z|\tilde{X},\tilde{Y})$.

Using Lemma 4.3.1, we can compute what the optimal g should be:

$$\exp(g(X,Y,1)) = C_{X,Y}\frac{Q(Z=1|X,Y)}{Q(Z=1)} = C_{X,Y}\frac{P(Y|X)}{\frac{1}{k+1}P(Y|X)+\frac{k}{k+1}W(Y)} \qquad \text{(C.6)}$$

$$\exp(g(X,Y,0)) = C_{X,Y}\frac{Q(Z=0|X,Y)}{Q(Z=0)} = C_{X,Y}\frac{W(Y)}{\frac{1}{k+1}P(Y|X)+\frac{k}{k+1}W(Y)}, \qquad \text{(C.7)}$$

where $C_{X,Y}$ is a multiplicative factor. Because of the free multiplicative factor, only the ratio between (C.6) and (C.7) matter:

$$\frac{\exp(g(X,Y,1))}{\exp(g(X,Y,0))} = \frac{P(Y|X)}{W(Y)}. \qquad \text{(C.8)}$$

Thus, if we set $\exp(g(X,Y,0)) = \frac{1}{k}$, we recover the NS logistic regression objective [71,

Section 2.2]

$$\sup_g \mathbb{E}_{P_X}\left[\frac{1}{k+1}\mathbb{E}_{P_{Y|X}}\log\left(\frac{\exp[g(X,Y,1)]}{\exp(g(X,Y,1))+1}\right)+\frac{k}{k+1}\mathbb{E}_{W_Y}\log\left(\frac{1}{\exp(g(X,Y,1))+1}\right)\right]. \tag{C.9}$$

(C.8) implies $\exp(g^*(X,Y,1)) = \frac{P(Y|X)}{kW(Y)}$. Interestingly, if $W(Y) = P(Y)$ (i.e., the noise distribution is chosen to be the same as the unigram distribution),

$$g^*(X,Y,1) = \log\frac{P(Y|X)}{kP(Y)} = \log\frac{P(X,Y)}{P(X)P(Y)} - \log(k). \tag{C.10}$$

Word2vec restricts g to be a dot product, which means (C.10) is implicitly factorizing the shifted pointwise mutual information matrix. Thus, Theorem 4.3.1 leads to an alternate proof for the result of [57, Section 3] without using any gradient calculations.

Alternatively, if we set $\exp(g(X,Y,0)) = W(Y)$, we recover the noise contrastive estimation (NCE) objective [34, 72]:

$$\begin{aligned}\sup_g \mathbb{E}_{P_X}\Bigg[&\frac{1}{k+1}\mathbb{E}_{P_{Y|X}}\log\left(\frac{\exp[g(X,Y,1)]}{\exp(g(X,Y,1))+kW(Y)}\right)\\ &+\frac{k}{k+1}\mathbb{E}_{W_Y}\log\left(\frac{kW(Y)}{\exp(g(X,Y,1))+kW(Y)}\right)\Bigg].\end{aligned} \tag{C.11}$$

(C.8) implies $\exp(g^*(X,Y,1)) = P(Y|X)$. ■

■ C.3.2 InfoNCE

[79, Appendix A.1] already draws connections between InfoNCE and DV-based Mutual Information Neural Estimation (MINE) [5]. We only make the further observation that the derivation uses the discriminative DV form for MINE. Due to its softmax objective, InfoNCE also can be interpreted as discriminative DV where one of the random variables is a sequence.

■ C.3.3 Deep InfoMax

Deep InfoMax [38] learns a feature encoder E_ψ such that $I(X; E_\psi(X))$ is maximized:

$$\arg\max_{\omega,\psi} \hat{I}_\omega(X; E_\psi(X)) \tag{C.12}$$

Define $\mathbb{J}$ as the joint distribution of $(X, E_\psi(X))$ and $\mathbb{M}$ as the product of their marginals. To estimate mutual information, the authors propose to substitute it with its DV form:

$$\hat{I}_\omega(X; E_\psi(X)) = \mathbb{E}_{\mathbb{J}}[T_\omega(X, E_\psi(X))] - \log \mathbb{E}_{\mathbb{M}}[\exp(T_\omega(X, E_\psi(X)))] \tag{C.13}$$

The authors note that a key difference in their formulation is the log on the outside of all expectations. Section 4.3 shows it results from applying DV to the generative form.

An additional insight from [38, Section 3.1] is replacing (C.12) with maximizing a non-KL divergence based mutual information. In particular, they propose using the mutual information defined via the Jensen-Shannon divergence [58]. We derive the following relation:

Proposition C.3.2. Let $Z \sim \text{Bernoulli}(\frac{1}{2})$, and the noise distribution be $P_X P_{E_\psi(X)}$. The Jensen-Shannon Deep InfoMax objective is equivalent to estimating $I(\tilde{X}, \tilde{E}_\psi(X); Z)$ using DV. More generally, $I^{(JSD)}(X; Y) = I(\tilde{X}, \tilde{Y}; Z)$.

Proof. Written in the notation from [38, Equation 4], the objective of Jensen-Shannon Deep InfoMax is

$$\hat{I}^{(JSD)}_{\omega,\psi}(X; E_\psi(X)) = \mathbb{E}_{\mathbb{P}}[-\text{sp}(-T_\omega(X, E_\psi(X)))] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[\text{sp}(T_\omega(X', E_\psi(X)))] \tag{C.14}$$

where X is the input, X' is the input drawn from $\tilde{\mathbb{P}} = \mathbb{P}$, and $\text{sp}(\cdot) = \log(1 + \exp(\cdot))$.

If we write out the sp function, the objective becomes

$$\begin{aligned}&\mathbb{E}_{\mathbb{P}}[-\log(1+\exp(-T_\omega(X,E_\psi(X))))] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[\log(1+\exp(T_\omega(X',E_\psi(X))))] \\ &= \mathbb{E}_{\mathbb{P}}\left[\log\left(\frac{1}{1+\exp(-T_\omega(X,E_\psi(X)))}\right)\right] + \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\log\left(\frac{1}{1+\exp(T_\omega(X',E_\psi(X)))}\right)\right].\end{aligned} \tag{C.15}$$

$E_\psi(X)$ is a deterministic function of X and the expectations are unweighted[2]. If we define $Y \triangleq E_\psi(X)$ and $k=1$, (C.15) is equivalent to the objective in (C.9) by a change of variable[3]. Thus, (C.15) estimates the same quantity as (C.9): $H(Z|\tilde{X},\tilde{Y})$, where $Z \sim \text{Bernoulli}(\frac{1}{2})$.

For the purpose of optimization, [38] discards a constant $\log(2)$ term from the convex conjugate formulation in [78]. When $k=1$, $H(Z)=\log(2)$. Adding that back, we see that Jensen-Shannon Deep InfoMax estimates $I(\tilde{X},\tilde{E}_\psi(X);Z)$. For general Y that is not a deterministic function of X, the first expectation in (C.14) is over the joint distribution, and the same proof works to show that $I^{(JSD)}(X;Y) = I(\tilde{X},\tilde{Y};Z)$ ■

[2]This does not preclude having more than one negative sample per positive sample to approximate the 2nd expectations. k is only a parameter that controls the mixture distribution of $(\tilde{X},\tilde{Y})$.

[3]This is sometimes called the law of the unconscious statistician.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[3] Xiaowei Bao, Nikolaos V. Sahinidis, and Mohit Tawarmalani. Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Mathematical Programming, Series B*, 129(1):129–157, September 2011.

[4] Richard H. Bartels and George W Stewart. Solution of the matrix equation ax+ xb= c [f4]. *Communications of the ACM*, 15(9):820–826, 1972.

[5] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540, 2018.

[6] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[7] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[8] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[9] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[10] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[11] Hans L. Bodlaender, Peter Gritzmann, Victor Klee, and Jan Van Leeuwen. Computational complexity of norm-maximization. *Combinatorica*, 10(2):203–225, June 1990.

[12] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 517–526. JMLR. org, 2017.

[13] Leo Breiman and Jerome H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80 (391):580–598, September 1985.

[14] Tao Chen, Shijian Lu, and Jiayuan Fan. S-cnn: Subcategory-aware convolutional networks for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2522–2528, 2017.

[15] Yunmei Chen and Xiaojing Ye. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

[16] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[17] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[18] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[19] Imre Csiszár and Gábor Tusnády. Information geometry and alternating minimization procedures. *Statistics and Decisions*, Supplement Issue 1:205–237, 1984.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[22] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003.

[23] Jian Dong, Wei Xia, Qiang Chen, Jianshi Feng, Zhongyang Huang, and Shuicheng Yan. Subcategory-aware object classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 827–834, 2013.

[24] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983.

[25] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279. ACM, 2008.

[26] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[27] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[28] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[29] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[30] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[31] David Yang Gao. Canonical duality theory and solutions to constrained nonconvex quadratic programming. *Journal of Global Optimization*, 29(4):377–399, August 2004.

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[33] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[34] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[35] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[36] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[39] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, New York, 1991.

[40] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, second edition, 2013.

[41] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[42] Shao-Lun Huang and Lizhong Zheng. Linear information coupling problems. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, pages 1029–1033, Cambridge, MA, USA, July 1-6 2012.

[43] Shao-Lun Huang, Anuran Makur, Lizhong Zheng, and Gregory W Wornell. An information-theoretic approach to universal feature selection in high-dimensional inference. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1336–1340. IEEE, 2017.

[44] Shao-Lun Huang, Anuran Makur, Gregory W Wornell, and Lizhong Zheng. On universal features for high-dimensional learning and inference. *arXiv preprint arXiv:1911.09105*, 2019.

[45] W. B. Jurkat and H. J. Ryser. Term ranks and permanents of nonnegative matrices. *Journal of Algebra*, 5(3):342–357, March 1967.

[46] Hyeji Kim, Yihan Jiang, Ranvir Rana, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Communication algorithms via deep learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[47] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[48] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[49] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[50] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[52] Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, Bill Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 9345–9356, 2018.

[53] Charlotte Laclau, Ievgen Redko, Basarab Matei, Younes Bennani, and Vincent Brault. Co-clustering through optimal transport. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1955–1964, 2017.

[54] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[55] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

[56] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, Providence, first edition, 2009.

[57] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[58] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[59] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in neural information processing systems*, pages 1498–1507, 2018.

[60] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[61] Kuang Liu. https://github.com/kuangliu/pytorch-cifar, 2018.

[62] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, IT-28(2):129–137, March 1982.

[63] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[64] Anuran Makur and Lizhong Zheng. Linear bounds between contraction coefficients for f-divergences. arXiv:1510.01844v3 [cs.IT], August 2017.

[65] Anuran Makur, Fabián Kozynski, Shao-Lun Huang, and Lizhong Zheng. An efficient algorithm for information decomposition and extraction. In *Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing*, pages 972–979, Allerton House, UIUC, Illinois, USA, September 29-October 2 2015.

[66] Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1):7–12, March 1960.

[67] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

[68] Gihan J Mendis, Jin Wei, and Arjuna Madanayake. Deep learning-based automated modulation classification for cognitive radio. In *2016 IEEE International Conference on Communication Systems (ICCS)*, pages 1–6. IEEE, 2016.

[69] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[70] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[71] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[72] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[73] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[74] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[75] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[76] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[77] Feiping Nie, Xiaoqian Wang, Cheng Deng, and Heng Huang. Learning a structured optimal bipartite graph for co-clustering. In *Advances in Neural Information Processing Systems*, pages 4132–4141, 2017.

[78] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[79] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[80] Panos M. Pardalos and Stephen A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22, March 1991.

[81] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[82] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[83] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[84] David Qiu. Embedding and latent variable models using maximal correlation. Masters thesis in electrical engineering and computer science, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 2017.

[85] David Qiu, Anuran Makur, and Lizhong Zheng. Probabilistic clustering using maximal matrix norm couplings. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1020–1027. IEEE, 2018.

[86] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[87] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1 (8):9, 2019.

[88] Alfréd Rényi. On measures of dependence. *Acta Mathematica Academiae Scientiarum Hungarica*, 10(3-4):441–451, 1959.

[89] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[90] Sartaj Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4): 262–279, December 1974.

[91] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[92] Neev Samuel, Tzvi Diskin, and Ami Wiesel. Deep mimo detection. In *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2017.

[93] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

[94] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[95] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[96] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pages 368–377, Allerton House, UIUC, Illinois, USA, September 22-24 1999.

[97] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[98] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[99] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[100] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[101] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[102] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[103] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *arXiv preprint arXiv:1911.02685*, 2019.

[104] Geoffrey Zweig and Christopher JC Burges. The microsoft research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft, 2011.