École Doctorale Paris-Est
Mathématiques & Sciences et Technologies de l'Information et de la Communication

Thèse de doctorat
de l'Université Paris-Est
Domaine : Traitement du Signal et des Images

Présentée par

Shell Xu Hu

pour obtenir le grade de

Docteur de l'Université Paris-Est

Towards Efficient Learning of Graphical Models and Neural Networks with Variational Techniques

---

Soutenue publiquement le 19 décembre 2019
devant le jury composé de :

Guillaume Obozinski	École des Ponts ParisTech	Directeur de thèse
Nikos Komodakis	École des Ponts ParisTech	Directeur de thèse
Simon Lacoste-Julien	Université de Montréal	Rapporteur
Matthew Blaschko	KU Leuven	Rapporteur
Nathalie Peyrard	INRA Toulouse	Examinateur
Jakob Verbeek	Inria Grenoble	Examinateur

École des Ponts ParisTech
LIGM-IMAGINE
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée cedex 2
France

Université Paris-Est Marne-la-Vallée
École Doctorale Paris-Est MSTIC
Département Études Doctorales
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77454 Marne-la-Vallée cedex 2
France

Acknowledgments

TO BE EXTENDED...

As the manuscript will be submitted for review, I would like to take this chance to thank my supervisors Guillaume Obozinski and Nikos Komodakis for organizing the PhD defense. My heartily thanks to Guillaume Obozinski, who has always been very supportive of my work and has given me so many valuable suggestions to my research, life, and this manuscript.

I would like to thank Simon Lacoste-Julien and Matthew Blaschko for taking their time to review this manuscript. I am also grateful to Nathalie Peyrard and Jakob Verbeek for having accepted to be part of the jury.

There are too many people to thank for their encouragement, friendship, support, and help that carried me through the PhD. I will extend this acknowledgement to express my deep thanks to all of them.

Contents

Abstract

Probability theory offers a mathematical framework for quantifying the uncertainty about the model and the data, forms an underpinning of many machine learning systems. Comparing to deterministic models, the prediction of a probabilistic model takes the form of a probability distribution over all possible values. However, it may render the prediction computationally intractable. To overcome the computational issue, variational techniques are often employed, where variational bounds are used to approximate the original objectives, casting the original problem as an optimization problem.

The primary goal of this thesis is to explore better variational formulations for probabilistic graphical models and deep neural networks in terms of computational efficiency or sample efficiency.

For learning discrete graphical models, we propose a new variational inference algorithm for the dual problem of the maximum likelihood estimation, inspired by the recent advances of stochastic variance reduction algorithms. Roughly speaking, we obtain a concave optimization over a convex polytope, where each variable, corresponding to a clique in the graph, is subject to the simplex constraint; besides, the local consistency among cliques induces a set of equality constraints. We thus propose an augmented Lagrangian formulation to solve this optimization problem. Our algorithm, which can be interpreted as an inexact gradient descent for the outer loop, requires only a fixed number of inner block-coordinate updates to obtain a sufficiently good estimate of the gradient for the Lagrange multipliers. We prove that the proposed algorithm enjoys a linear convergence in both the dual and the primal, and compare it with state-of-the-art algorithms on three applications.

Probability theory has a natural application to neural networks if we consider the hidden activations and/or the network parameters as random variables. This type of model is generally known as the Bayesian neural network, and which is typically trained via the minimum description length (MDL) principle, corresponds to formulate supervised learning as lossless compression. We propose an alternative learning formulation based on the rate-distortion theory for lossy compression. In a nutshell, we introduce a mutual information regularizer to control the amount of information stored in the parameters with respect to any minibatch in the train-set, and derive a variational upper bound on the mutual information yielding an EM-like algorithm. Comparing to MDL with stochastic gradient descent, our algorithm is more stable and can potentially achieve a better testing accuracy.

The same formulation can be applied to meta-learning, where the EM-like algorithm can be derived from an empirical Bayes formulation and motivated

from applying a hierarchical Bayes model to the dataset constituting a two-level hierarchy. Recall that the idea behind meta-learning is learning to generalize across tasks. The key difference is that we now restrain the mutual information between the parameters of the model and the dataset associated with a particular task, for which we take a variational upper bound and obtain a sum of local KL divergences between the variational posterior and the true posterior of each task. We derive a novel amortized variational inference that couples all the variational posteriors into a meta-model, which consists of a synthetic gradient network and an initialization network. Moreover, the combination of local KL divergences and synthetic gradient network allows backpropagating information from unlabeled data, thereby attaining better variational posteriors. We further provide a theoretical analysis to justify our formulation in terms of the generalization performance and demonstrate its superior performance on few-shot classification benchmarks.

The second part of this thesis is dedicated to the memory issue of over-parameterized neural networks. To improve memory efficiency, we propose two ideas briefly sketched as follows. The first idea, which is inspired by the information bottleneck principle, makes use of the teacher-student framework, where an over-parameterized teacher network is used as additional supervision for training the thinner student network. The supervision is imposed by maximizing the mutual information between selected pairs of activations, while the intractable mutual information is approximated by a variational lower bound. Another idea comes from an empirical observation that over-parameterized neural networks are robust to symmetry constraints on network parameters, which is embarrassingly simple, has a negligible effect on training and testing accuracy. Both ideas have been extensively tested on standard benchmarks and compared with state-of-the-art methods.

Introduction

Artificial intelligence (AI), suggested by its name, is a type of intelligence distincted from *natural intelligence* (NI) appearing in humans and animals. However, intelligence is itself a chamber of mysteries. We are not close to understanding it any time soon. Over the years, AI researchers attempted to make an AI system indistinguishable from that of its NI sibling. The *Turing test*, which is designed to evaluate proposed AI systems, epitomizes this passion. As AI research advances, the definition of AI is also evolving over time. The term AI was originally used to refer to the computer systems mimicking NI. In its modern usage, AI also refers to as the whole field that studies intelligent machines. As a scientific field, AI was born at the *Dartmouth conference* in 1956 and has become an interdisciplinary field involving computer science, statistics, neuroscience, physics and so on. Jordan (2019) clarified that most of what is being called AI today is what has been called *machine learning* (ML) for the past several decades. ML is a less mysterious term as it is a field that focuses more on the mathematical and engineering aspects of AI systems.

While building an AI system does not necessarily require reverse engineering, drawing inspirations from NI can bring promising ideas. AI research has a long history of emphasis on studying and reproducing partially or entirely the mechanism of NI. The induced methods are sometimes called *brain-inspired methods*. Examples of this kind include a variety of computational models, called *artificial neural networks* (Rumelhart et al. 1986, LeCun et al. 1998, Maass 1997), which loosely imitate biological neurons. This family of methods and models is now largely developed and widely recognized as *deep learning* (LeCun et al. 2015).

As argued by Russell and Norvig (2009), "artificial flights" were made when the Wright brothers and others stopped imitating birds. A different philosophy is to avoid mimicking NI, which stems from mathematical formulations of thinking machines based on concepts such as *logic* and *rationality*. These formulations can date back to *Aristotle's syllogism* and *Boolean algebra*. The most representative examples of this kind are *logical methods*, such as *propositional logic* and *first-order logic*. Unlike artificial neural networks, logical models are more intuitive and transparent from a human perspective. They are able to incorporate domain knowledge in the form of logical formulas and conduct mathematically sound reasoning in a human-understandable way. In the mid-1965s, AI research on this kind of methods was perceived as the dominant paradigm (Haugeland 1989), which has many remarkable applications, such as the general problem solver by Newell and Simon (1961) and the advice taker by McCarthy (1960). These achievements were part of the reasons why AI drew so much attention in the 1960s.

However, classical logical models suffer from the so-called *qualification problem* (Russell and Norvig 2009) due to their deductive nature. To handle exceptions in the real world, a method must take into account *uncertainty*, whether due to partial observations, non-deterministic environments, or a combination of both. It is now well known that probability theory offers a mathematically justified framework for quantifying uncertainty. In fact, probability theory is a natural generalization of logic (Jaynes 1996) without making reference to "chances" or "random variables" in contrast to the classical introduction of probability theory by Kolmogorov (1933). This line of research dates back to Laplace (1820), Keynes (1921), Jeffreys (1939). A mathematical justification was given by Cox (1946) based on a few axioms.

The aforementioned lines of research are apparently based on different philosophies and have developed two branches of AI, which, due to their main characteristics, are sometimes labeled as *data-driven* AI and *knowledge-driven* AI respectively. Intuitively, the former relies on general (brain-inspired) function approximators to approximate the underlying input-output mappings of the data, while the latter incorporates human knowledge into the models independent of any particular dataset. Certainly, there is no pure data-driven or knowledge-driven method. Even neural networks cannot generalize without inductive biases.

Probability theory offers an elegant framework to integrate these two kinds of methods in the sense that both high-level knowledge and low-level information are modeled as random variables with either known or implicit distributions. An example in this regard is the *probabilistic graphical model* (PGM) (Pearl 1988, Koller et al. 2009), which inherently supports probabilistic reasoning while possesses a graph knowledge representation inspired by neural networks. Unlike the connections in neural networks, the graph structure in a PGM is often linked to a real-life relationship or its abstraction in the real world, which in turn enables efficient inference and reasoning. Classical examples of PGMs include *hidden Markov models* (Stratonovich 1965), *latent Dirichlet allocation* (Blei et al. 2003), Bayesian networks (Pearl 1985) etc. There are also examples that resemble logical methods or neural networks. For instance, *Markov logic network* (Richardson and Domingos 2006) is a PGM in which random variables are associated with logical propositions; *deep belief networks* (Hinton 2009) and *deep restricted Boltzmann machines* (Salakhutdinov and Hinton 2009) are PGMs with layered graph structures which are similar to that of deep neural networks.

As every sword has two edges, the price to pay for the mathematical rigorousness of uncertainty quantification is computability or scalability. Equipped with the probability framework, we also introduced many intractable quantities, such as the expected value, the denominator of a probability density function, and the marginal likelihood alike, which all involve computing intractable integrals. To approximate such quantities, there are two mainstream ideas: *Markov Chain Monte Carlo* (MCMC) (Metropolis et al. 1953, Hastings 1970, Geman and Geman 1984) and *variational inference* (VI) (Peterson 1987, Jordan et al. 1999, Blei et al. 2017). MCMC is based on Monte Carlo simulation: to draw iid samples from the target probability distribution, it constructs a Markov chain such that the stationary

distribution of the Markov chain is the target probability distribution; these samples are then used to approximate the integrals with tractable sums. Although by construction, MCMC produces unbiased estimates by the strong law of large numbers, it suffers from a slow convergence to the stationary. Variational inference, on the other hand, sacrifices exactness for speed, and casts the approximate inference as an optimization problem, in which a family of simpler distributions (i.e., proposal distributions) is first posited, and then the closest member of that family to the target distribution is chosen with respect to some divergence measure (e.g., the Kullback-Leibler divergence). Compared with MCMC, although being irredeemably biased, VI tends to be faster and more scalable to large data.

About this thesis

In this thesis, I will mainly focus on variational inference and probabilistic models. In particular, I will cover several projects that I have participated in during my PhD about improving the efficiency of AI/ML systems with variational techniques. The thesis consists of two parts. In the first part, the computational efficiency of probabilistic graphical models is studied. In the second part, several problems in deep learning are investigated, which are related to either energy efficiency or sample efficiency. I will briefly show in the next paragraphs the main problems we have encountered and the solutions we have proposed towards more efficient machine learning.

Probabilistic graphical models

The first problem considered in this thesis is the computational efficiency of PGMs. As in many probabilistic models, PGMs suffer from computational intractability issue. Specifically, *probabilistic inference* and *maximum likelihood parameter estimation* of PGMs are in general intractable due to the *partition function*: to compute the function value or its gradient involve an integral over the entire sample space, which is computationally expensive since, even for the discrete case, the size of the sample space grows exponentially as the number of nodes increasing. As such, variational inference has become a workhorse for PGMs. This is also because variational inference is a general framework rather than a particular algorithm. By choosing different proposal distributions, variational inference can take different forms. Along with the insights from stochastic mechanics (Parisi 1988), the *mean-field* variational formulation and the *Bethe* variational formulation are the conventional examples widely used in PGMs (Wainwright 2008). Focusing on the convexified Bethe variational formulation (Wainwright et al. 2005b, Globerson and Jaakkola 2007b, Meshi et al. 2009), we identify that its dual problem is strongly concave but non-smooth, where the non-smoothness comes from the constraints induced by marginalization coupling between the dual variables. We propose a new algorithm inspired by the algorithmic development on convex optimization for empirical risk minimization (Roux et al. 2012, Johnson and Zhang 2013, Defazio et al. 2014a, Shalev-Shwartz and Zhang 2014). In particular, we extend the stochastic dual coordinate ascent (SDCA) algorithm (Shalev-Shwartz and Zhang

2014) and the augmented Lagrangian method of multipliers (ADMM) (Boyd et al. 2011) for the dual problem. Since the classical convergence analysis of ADMM does not hold for convex problems involving more than three blocks of variables (Chen et al. 2016a), we conduct a new convergence analysis for our algorithm based on the analysis of inexact gradients for min-max problems. We also show that both the sequence of primal solutions and the sequence of dual solutions generated by our algorithm enjoy a linear convergence, which is confirmed empirically in three representative experiments of PGMs.

Deep learning

In the second part of the thesis, we focus on deep learning. For deep neural networks, the learning problem is generally a large-scale nonconvex optimization problem, in which neither the objective value nor the gradient can be computed efficiently. Recall that a nonconvex optimization problem is considered extremely difficult since the problem may have many local minima and saddle points. However, when a neural network is over-parameterized, for example, by increasing the depth and/or the width of the network, a simple optimization algorithm like stochastic gradient descent (SGD) (Robbins and Monro 1951) or its variants such as AdaGrad (Duchi et al. 2011), ADAM (Kingma and Ba 2014) can find global optima in the case of supervised image classification (Zhang et al. 2016). Moreover, empirical results by Hinton et al. (2012), He et al. (2016a), Zagoruyko and Komodakis (2016c) show that over-parameterized networks are less prone to overfitting, and often achieve a better generalization performance. This phenomenon has been studied by many theoretical researchers. A few explanations have been published since then, for example, the flat minima conjecture by Keskar et al. (2016), Wu et al. (2017), the PAC-Bayes insights by Dziugaite and Roy (2017), Zhou et al. (2018), Pérez et al. (2018) and the information theory perspective by Tishby and Zaslavsky (2015), Achille and Soatto (2017).

Although it seems over-parameterized networks work well in practice, there are two issues brought by the over-parameterization: the memory/energy inefficiency and the sample inefficiency. The first issue should be attributed to the tremendous number of parameters. Note that due to the highly parallelizable nature of neural networks, over-parameterization may not be a problem in terms of speed. However, the number of parameters to be stored in the memory and the number of operations to be carried out during the forward and backward passes do cause a serious problem when applying deep learning to broader domains with resource constraints. The second issue is more fundamental since neural networks are notoriously known to be data-hungry. Both theoretical and empirical results (Cybenko 1989b, Raghu et al. 2017, Zhang et al. 2016) show that neural networks are capable of expressing a diverse range of functions. According to the classical learning theory (Valiant 1984), this implies that the model complexity can be extremely high, and so does

the sample complexity[1]. In other words, to achieve a small testing error, we will need a large number of training data. Nevertheless, in many cases, the size of the training data does not meet this condition due to either the heavy-tailed data distribution or privacy concerns. These two issues are obviously not isolated. Our ultimate goal is to have a small network trained on a small dataset which still yields comparable performance. To this end, multiple ingredients of the deep learning system should be revised.

The first ingredient we investigate is *regularization*, which is a process of reshaping the learning problem in order to prevent overfitting. In deep learning, traditional regularizers such as *weight decay* (i.e., the L2 penalty) are less effective compared to *dropout* (Srivastava et al. 2014) and *batch normalization* (Ioffe and Szegedy 2015a), both of which have a Bayesian interpretation (Kingma et al. 2015, Teye et al. 2018). In fact, many regularization techniques correspond to imposing prior knowledge from a Bayesian perspective. However, what form of prior knowledge is suitable for deep learning is not clear. In light of the *minimum description length* (MDL) (Rissanen 1978) interpretation of neural networks (Hinton and Van Camp 1993), which formulates the learning as *lossless compression*, we take an alternative path to introduce the prior as a regularization term for the *lossy compression* of the data. Specifically, we take the mutual information between the *bootstrap sample* and the neural-net weights as the regularization term. The proposed algorithm resembles the classical *Blahut-Arimoto algorithm* for lossy compression (Blahut 1972, Arimoto 1972), in which we compute a local *variational posterior* for each bootstrap sample and then use it to update the *aggregated posterior*. Unlike MDL methods (Hinton and Van Camp 1993, Graves 2011, Blundell et al. 2015), we use the aggregated posterior to compute the predictive distribution rather than taking the variational posterior. Empirically, our method exhibits some attractive properties over the conventional MDL methods.

The sample efficiency can be achieved in a special multi-task setting where tasks share some common characteristics. The problem is referred to as *meta-learning* and usually solved by learning-to-learn schemes (Vinyals et al. 2016, Ravi and Larochelle 2016, Finn et al. 2017, Snell et al. 2017, Mishra et al. 2017, Gidaris and Komodakis 2018). Since each task contains its own datasets, the overall data provided in meta-learning is a two-level hierarchy of datasets. We thus consider a *hierarchical Bayes* formulation consisting of two levels of latent variables, where the latent variables in the first level are the parameters of each task, and the latent variable in the second level is the hyperparameter shared across tasks. We opt to approximate the posterior of the hyperparameter by a point estimate, which leads to the so-called *empirical Bayes*. It turns out that the aforementioned mutual information regularizer appears in the *evidence lower bound* of the marginal log-likelihood, but this time we obtain the mutual information between the weights and the data associated with some task. Note that, in empirical Bayes, we treat

[1]This argument may be wrong since, as mentioned by Vapnik et al. (1994), the notion of model complexity should depend on a subset of functions which are reachable from the initialization.

individual weights as iid samples of the same random variable. Finally, we derive an amortized variational inference that couples all local variational posteriors (one for each sample of the parameter) into a meta-model, which consists of a synthetic gradient network and an initialization network. Our empirical results on the mini-ImageNet benchmark for episodic few-shot classification significantly outperform previous state-of-the-art methods.

A seminal work proposed by Ba and Caruana (2014), Hinton et al. (2015) has been shown to be useful for improving both energy efficiency and sample efficiency of deep learning (Yim et al. 2017, Zagoruyko and Komodakis 2016b), which is achieved by forcing the neural network (i.e., the student) to mimic a pretrained teacher neural network. This work is important because we can tailor the student network to be sufficiently small without sacrificing the performance (Furlanello et al. 2018). Note that, without knowledge transfer, the training of small networks tends to underfitting and the optimization could be much more difficult than that of over-parameterized networks. Moreover, provided that the teacher network is trained on a large-scale source dataset (e.g., ImageNet), the teacher-student knowledge transfer offers a natural framework for transfer learning. That being said, existing works require the student network to be sufficiently similar to the teacher network. Yet no mathematical justification is given to explain the mechanism behind the knowledge transfer. Inspired by the information bottleneck interpretation of deep learning (Tishby and Zaslavsky 2015), where the learning is cast as an optimization that minimizes the mutual information between the latent representation and the input while maximizing the mutual information between the latent representation and the label, we propose to maximize the mutual information between the teacher's representation and the student's representation. To incorporate this negative mutual information term as an additional loss, we employ a variational lower bound with the Gaussian assumption. We compare our method with existing knowledge transfer methods on both knowledge distillation and transfer learning tasks and show that our method consistently outperforms existing methods. We further demonstrate the strength of our method on knowledge transfer across heterogeneous network architectures by transferring knowledge from a convolutional neural network (CNN) to a multi-layer perceptron (MLP) on CIFAR-10.

The last work of this thesis was due to an empirical finding – deep neural networks are robust to symmetry constraints imposed on the weights. We propose several symmetrization techniques and show empirically that these techniques are very simple and universally applicable to almost any over-parameterized network without requiring fine-tuning as post-processing, and, depending on network size, symmetry can have little or no negative effect on network accuracy. For instance, our symmetrized ResNet-101 has almost 25% fewer parameters than its original version with only 0.2% accuracy loss on ImageNet. As an application, it can be used to reduce the memory requirement of neural networks with additionally a negligible computational cost at training time.

Organization of the thesis

As we have mentioned, The thesis will be divided into two parts. The first part contains the topics in probabilistic graphical models covered by Chapter 2 and Chapter 3. The second part includes the topics in deep learning covered by Chapter 4, Chapter 5, Chapter 6, Chapter 7 and Chapter 8. The detailed organization of this thesis is sketched as follows.

Chapter 1 covers the prerequisite mathematics for the presentation of the following chapters. It includes a basic introduction to information theory and a basic introduction to the mathematical concepts used by the convergence analysis for convex optimization.

Chapter 2 gives a general introduction to probabilistic graphical models, including common models, variational inference and the formulations for parameter estimation.

Chapter 3 is based on the following publication:

> *Shell X. Hu and Guillaume Obozinski. "SDCA-Powered Inexact Dual Augmented Lagrangian Method for Fast CRF Learning." International Conference on Artificial Intelligence and Statistics (AISTATS), 2018.*

Chapter 4 gives a general introduction to deep neural networks with an emphasis on network compression and the over-parameterization phenomenon of neural networks.

Chapter 5 is based on the following paper:

> *Shell X. Hu, Pablo G. Moreno, Andreas Damianou, Neil D. Lawrence. "β-BNN: A Rate-Distortion Perspective on Bayesian Neural Networks." NeurIPS Workshop on Bayesian Deep Learning, 2018*

Chapter 6 is based on the following paper:

> *Shell X. Hu, Pablo G. Moreno, Xi Shen, Yang Xiao, Guillaume Obozinski, Neil D. Lawrence, Andreas Damianou. "Empirical Bayes Transductive Meta-Learning with Synthetic Gradients." NeurIPS Workshop on Meta-Learning, 2019. (Long version submitted to ICLR 2020)*

Chapter 7 is based on the following publication:

> *Sungsoo Ahn, Shell X. Hu, Andreas Damianou, Neil D. Lawrence, Zhenwen Dai. "Variational Information Distillation for Knowledge Transfer." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.*

Chapter 8 is based on the following publication:

> *Shell X. Hu, Sergey Zagoruyko, Nikos Komodakis. "Exploring Weight Symmetry in Deep Neural Networks." Computer Vision and Image Understanding (CVIU), 187, p.102786, 2019.*

Chapter

1 Background: Convex Optimization and Information Theory

Abstract

In this chapter, we cover several basic definitions and theorems in convex optimization and information theory, which will be used in the subsequent chapters. It is however safe to skip this chapter.

1.1 Convex Optimization

Definition 1.1.1 (Convexity)**.** *A function $f(x)$ is convex if $dom(f)$ is a convex set and*

$$\alpha f(x) + (1-\alpha)f(y) \geq f\Big(\alpha x + (1-\alpha)y\Big),\ \forall x, y \in dom(f), \alpha \in [0,1].$$

If the above inequality holds only for "$>$", we say $f(x)$ is strictly convex.

Lemma 1.1.1 (First order condition)**.** *A differentiable function $f(x)$ is convex iff*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle, \quad \forall x, y \in dom(f).$$

Proof. By the definition of convexity,

$$f(y) \geq f(x) + \frac{f\Big(\alpha x + (1-\alpha)y\Big) - f(x)}{\alpha}. \tag{1.1}$$

Let $g(\alpha) = f\Big(\alpha x + (1-\alpha)y\Big)$, then $f(x) = g(0)$ and $f(y) = g(1)$. By taking limits on both sides, we can rewrite the above inequality as

$$\begin{aligned} f(y) &\geq f(x) + \lim_{\alpha \to 0} \frac{g(\alpha) - g(0)}{\alpha} && (1.2)\\ &= f(x) + \nabla g(0) && (1.3)\\ &= f(x) + \langle \nabla f(x), y - x \rangle. && (1.4) \end{aligned}$$

Note that $\nabla g(\alpha) = \langle \nabla f(x + \alpha(y-x)), y - x \rangle$. □

Corollary 1.1.1. *A differentiable function $f(x)$ is convex iff*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0, \quad \forall x, y \in dom(f).$$

Proof. By the Lemma 1.1.1,

$$\begin{aligned} f(y) &\geq f(x) + \langle \nabla f(x), y - x \rangle && (1.5)\\ f(x) &\geq f(y) + \langle \nabla f(y), x - y \rangle. && (1.6) \end{aligned}$$

The result follows by combining the above inequalities, which cancels out $f(x)$ and $f(y)$ on both sides. □

Definition 1.1.2 (Lipschitz continuous gradient)**.** *A differentiable function is said to have Lipschitz continuous gradient if there exists a constant $L > 0$ such that*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in dom(f), \tag{1.7}$$

where L is called the Lipschitz constant of f.

Theorem 1.1.2. *The following statements are equivalent:*

1. *A differentiable convex function $f(x)$ has Lipschitz continuous gradient.*

2. $f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|^2$, $\forall x, y \in dom(f)$.
3. $\alpha f(x) + (1-\alpha) f(y) \leq f\Big(\alpha x + (1-\alpha) y\Big) + \frac{\alpha(1-\alpha)L}{2}\|x - y\|^2$, $\forall x, y \in dom(f)$.
4. $\frac{L}{2}\|x\|^2 - f(x)$ *is convex.*
5. *If* f *is twice differentiable,* $\nabla^2 f(x) \preceq LI$, $\forall x \in dom(f)$.
6. $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2$, $\forall x, y \in dom(f)$.

Proof. We first prove $1 \Leftrightarrow 4$. Let $g(x) := \frac{L}{2}\|x\|^2 - f(x)$, then $\nabla g(x) = Lx - \nabla f(x)$. Since by Corollary 1.1.1 and Cauchy-Shwartz inequality, we have

$$\begin{aligned} \langle \nabla f(x) - \nabla f(y), x - y \rangle &\leq \|\nabla f(x) - \nabla f(y)\| \|x - y\| && (1.8) \\ &\leq L\|x - y\|^2, && (1.9) \end{aligned}$$

where the last inequality follows from the fact that $\nabla f(x)$ is Lipschitz continuous. By rearranging terms, we obtain

$$\langle (Lx - \nabla f(x)) - (Ly - \nabla f(y)), x - y \rangle \geq 0. \tag{1.10}$$

Again, applying Corollary 1.1.1 on $g(x)$, we conclude that $g(x)$ is convex, which can be used to show the equivalence $1 \Leftrightarrow 3$ by expanding

$$g(\alpha x + (1-\alpha) y) \leq \alpha g(x) + (1-\alpha) g(y). \tag{1.11}$$

Next, it is straightforward to show the equivalences $4 \Leftrightarrow 2$ and $4 \Leftrightarrow 5$, since

$$\begin{aligned} g(x) \text{ is convex} &\Leftrightarrow \nabla^2 g(x) = LI - \nabla^2 f(x) \succeq 0 \text{ if } f \text{ is twice differentiable} && (1.12) \\ &\Leftrightarrow g(y) - g(x) \geq \langle \nabla g(x), y - x \rangle && (1.13) \\ &\Leftrightarrow f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|^2. && (1.14) \end{aligned}$$

To show the equivalence $1 \Leftrightarrow 6$, we first introduce

$$\begin{aligned} f_x(z) &= f(z) - \langle \nabla f(x), z \rangle && (1.15) \\ f_y(z) &= f(z) - \langle \nabla f(y), z \rangle. && (1.16) \end{aligned}$$

Note that $\nabla f_x(z) = \nabla f(z) - \nabla f(x)$. Therefore,

$$\begin{aligned} \|\nabla f_x(z) - \nabla f_x(z')\| &= \|\nabla f(z) - \nabla f(z')\| && (1.17) \\ &\leq L\|z - z'\|, && (1.18) \end{aligned}$$

which indicates $\nabla f_x(z)$ is Lipschitz continuous. Similarly, $\nabla f_y(z)$ can also be shown to be Lipschitz. Then, by the statement 2, we have

$$f_x(y) \leq f_x(z) + \langle \nabla f_x(z), y - z \rangle + \frac{L}{2}\|z - y\|^2. \tag{1.19}$$

Minimizing both sides with respect to y yields

$$\min_y f_x(y) \leq \min_y \left[f_x(z) + \langle \nabla f_x(z), y - z \rangle + \frac{L}{2}\|z - y\|^2 \right] \tag{1.20}$$

$$= f_x(z) - \frac{1}{2L}\|\nabla f_x(z)\|^2. \tag{1.21}$$

Since $f_x(z)$ is convex in z, it follows that $z^* = \arg\min_z f_x(z) = x \Leftrightarrow \nabla f_x(z) = 0$. The above inequality can be rewritten as

$$\frac{1}{2L}\|\nabla f_x(z)\|^2 = \frac{1}{2L}\|\nabla f(z) - \nabla f(x)\|^2 \leq f_x(z) - f_x(x) \tag{1.22}$$

$$= f(z) - f(x) + \langle \nabla f(x), x - z \rangle. \tag{1.23}$$

Let $z = y$, we have

$$\frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2 \leq f(y) - f(x) + \langle \nabla f(x), x - y \rangle. \tag{1.24}$$

Similarly, we obtain a symmetric inequality from $f_y(z)$:

$$\frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|^2 \leq f(x) - f(y) + \langle \nabla f(y), y - x \rangle. \tag{1.25}$$

Merging these two inequalities, we show that

$$\frac{1}{L}\|\nabla f(z) - \nabla f(y)\|^2 \leq \langle \nabla f(x) - \nabla f(y), x - y \rangle, \tag{1.26}$$

which concludes the proof for $1 \Leftrightarrow 6$. □

In addition to the statements in Theorem 1.1.2, we have also derived two intermediate results, i.e., eq.(1.9) and eq.(1.24), from the proof, which may also be of interests.

For many convex optimization problems, the Lipschitz continuity is an essential assumption in order to prove the convergence of a proposed algorithm. In addition to that, if we further assume the *strong convexity* condition holds, there exists algorithms that guarantee a linear convergence.

Definition 1.1.3 (Strong convexity)**.** *A differential function $f(x)$ is strongly convex if*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|x - y\|^2 \quad \forall x, y \in dom(f), \tag{1.27}$$

for some strong-convexity constant $\mu > 0$.

Without loss of generality, we used the differentiability in the definition of strong convexity. However, this is not required since one could replace the gradients with sub-gradients.

1.2 Information Theory

We will encounter the *Jensen's inequality* many times in this thesis, which forms the underpinning of variational inference.

Lemma 1.2.1 (Jensen's inequality)**.** *If f is a concave function and $X \in dom(f)$ is a random variable, then*

$$\mathbb{E}[f(X)] \leq f\Big(\mathbb{E}[X]\Big). \tag{1.28}$$

Moreover, if f is strictly concave, then

$$\mathbb{E}[f(X)] = f\Big(\mathbb{E}[X]\Big) \Leftrightarrow X = \textit{ constant.} \tag{1.29}$$

Proof. Since f is concave, for any $x, y \in \mathrm{dom}(f)$,

$$f(x) \leq f(y) + \nabla f(y)^\top (x - y). \tag{1.30}$$

Choosing $y = \mathbb{E}[X]$ and taking expectations over both sides, we have

$$\begin{aligned} \mathbb{E}[f(X)] &\leq f(y) + \nabla f(y)^\top (\mathbb{E}[X] - x) && (1.31)\\ &= f(\mathbb{E}[X]) + \nabla f(\mathbb{E}[X])^\top (\mathbb{E}[X] - \mathbb{E}[X]) && (1.32)\\ &= f(\mathbb{E}[X]). && (1.33) \end{aligned}$$

If f is strictly concave and X is a random variable, then eq.(1.30) holds only for "$<$", and $\mathbb{E}[f(X)] < f(\mathbb{E}[X])$. Hence, $\mathbb{E}[f(X)] = f(\mathbb{E}[X])$ holds only when X is a constant. In this case, $X \equiv \mathbb{E}[X]$. □

The mathematical concept of entropy or Shannon entropy was first introduced by Shannon (1948). Intuitively, if one attempts to encode a random variable X and assigns a code length to it inversely proportional to $p(X)$, then the code of $X = x$ will take $\log_2 \frac{1}{p(x)}$ bits. Thus, the entropy can be interpreted as the average code length in the binary format. The formal definition reads as follows:

Definition 1.2.1 (Entropy)**.** *Let $X \in \mathcal{X}$ be a discrete random variable with distribution $p(X)$. The quantity*

$$H(X) = \mathbb{E}_{X \sim p(X)}\left[\log_2 \frac{1}{p(x)}\right] = \sum_{x \in \mathcal{X}} p(x)\left[\log_2 \frac{1}{p(x)}\right] \tag{1.34}$$

is called the (Shannon) entropy of X. $H(X)$ is also written as $H(p(X))$ to emphasize that it is a functional of the distribution p.

Note that it is valid to use logarithm with respect to other bases, since the definition of the entropy differs only by a multiplicative constant. The definition of entropy also holds when $X = (X_1, \ldots, X_n)$ as a vector of random variables, for which $p(X_1, \ldots, X_n)$ is a joint distribution accordingly, and $H(X_1, \ldots, X_n)$ is called the *joint entropy*. When considering interactions between two random variables, we need an extension of the entropy induced by the conditional distribution.

Definition 1.2.2 (Conditional entropy)**.** *Let X, Y be two discrete random variables with the joint distribution $p(X, Y)$. The quantity*

$$H(X|Y) = \mathbb{E}_{y\sim p(Y)}\Big[H\Big(p(X|Y=y)\Big)\Big] \tag{1.35}$$

$$= \mathbb{E}_{x,y\sim p(X,Y)}\Big[\frac{1}{\log p(x|y)}\Big] \tag{1.36}$$

is called the conditional entropy.

We have only defined the entropy for discrete random variables. The continuous counterpart of entropy is called the *differential entropy*.

Definition 1.2.3 (Differential entropy)**.** *Let X be a continuous random variable with probability density function $f(x)$. The quantity*

$$h(X) = \int_{\mathcal{X}} f(x) \log \frac{1}{f(x)} dx \tag{1.37}$$

is called the differential entropy of X.

It turns out that the definition of entropy for both discrete and continuous cases can be unified using *Lebesgue integral*:

$$h(X) = -\int_{\mathcal{X}} f \log f d\mu, \tag{1.38}$$

where μ is the Lebesgue measure. In particular, for the discrete case, μ is the *counting measure* on $\mathcal{X}$. In both cases, f is the *Radon-Nikodym* derivative of the probability measure/distribution P induced by X, namely, $f = \frac{dP}{d\mu}$, then

$$h(X) = -\int_{\mathcal{X}} \frac{dP}{d\mu} \log \frac{dP}{d\mu} d\mu = -\int_{\mathcal{X}} \log \frac{dP}{d\mu} dP. \tag{1.39}$$

A related quantity is the *Kullback-Leibler (KL) divergence*, which is also known as the *relative entropy*.

Definition 1.2.4 (Kullback-Leibler divergence)**.** *Let P, Q be two probability measures over $\mathcal{X}$. If P is absolutely continuous with respect to Q, that is, $Q(X) = 0$ implies $P(X) = 0$, then the Radon-Nikodym derivative $\frac{dP}{dQ}$ exists and the quantity*

$$D_{\mathrm{KL}}(P\|Q) = \int_{\mathcal{X}} \frac{dP}{dQ} \log \frac{dP}{dQ} dQ = \int_{\mathcal{X}} \log \frac{dP}{dQ} dP \tag{1.40}$$

is called the Kullback-Leibler divergence.

Note that the difference between the entropy and the KL divergence is that the underlying measure μ is replaced by another probability measure Q and the negative sign is dropped in the definition of KL divergence. For continuous random variables, the Radon-Nikodym derivative $\frac{dP}{dQ}$ is computed as the density ratio $\frac{dP/d\mu}{dQ/d\mu}$.

In practice, the entropy and the KL divergence have very distinct usages, although their definitions are similar. The entropy is usually used as an uncertainty "measure", while the KL divergence is considered as a "distance" between two probability distributions. A highly related quantity to both quantities is the *mutual information*.

Definition 1.2.5 (Mutual information)**.** *Let X, Y be two random variables with joint distribution P_{XY} and marginal distributions P_X and P_Y. The mutual information of X and Y is defined by*

$$I(X;Y) = D_{\mathrm{KL}}(P_{XY} \| P_X P_Y). \tag{1.41}$$

By the definition of mutual information, $I(X;Y) \geq 0$ and $I(X;Y) = 0$ if and only if $X \perp\!\!\!\perp Y$. It is also symmetric: $I(X;Y) = I(Y;X)$. Thus, $I(X;Y)$ is widely used to measure the statistical dependence between X and Y. In other words, it measures the information about one random variable provided by another random variable. There are a few other useful properties that connects the mutual information to the entropy.

Lemma 1.2.2. *Let X, Y be two random variables with joint distribution P_{XY} and marginal distributions P_X and P_Y. The mutual information of X and Y can be decomposed as*

$$\begin{aligned} I(X;Y) &= h(X) + h(Y) - h(X,Y) && (1.42)\\ &= h(X) - h(X|Y) && (1.43)\\ &= h(Y) - h(Y|X). && (1.44) \end{aligned}$$

There is an analogy between the aforementioned information quantities and the true measures over sets. Namely, the joint entropy, conditional entropy and mutual information can be considered as the measure of a set union, set difference and set intersection respectively.

1.3 Rate-distortion theory

The rate-distortion theory was invoked to define the "goodness" of a representation of an information source (such as the weights of a neural network). The rate is referred to the length of the representation and the distortion is introduced to measure the quality of the representation. In other words, rate-distortion theory is concerned with the average amount of information about the information source that must be preserved by any lossy data compression scheme such that the reproduction can be subsequently generated from the compressed data with average distortion less than or equal to some specified error.

The readers are invited to follow a detailed description of the rate-distortion theory at Chapter 10 of Cover and Thomas (2012). This section will only cover the minimal material.

Let us first introduce some notations. Denote by $X \in \mathcal{X}$ the input random variable, $\mathrm{enc}(X)$ the code of X produced by an encoder $\mathrm{enc}(\cdot)$ and $\hat{X} \in \hat{\mathcal{X}}$ the estimate of X produced by a decoder $\mathrm{dec}(\cdot)$.

Definition 1.3.1. *A distortion function is a mapping $d\colon \mathcal{X} \times \hat{\mathcal{X}} \to \mathcal{R}^+$.*

Definition 1.3.2. *The distortion associated with an encoder $enc\colon \mathcal{X} \to \{1, \ldots, K\}$ and a decoder $dec\colon \{1, \ldots, K\} \to \hat{\mathcal{X}}$ is defined as*

$$D := \mathbb{E}_{x \sim p(x)} d(x, dec(enc(x))).$$

Definition 1.3.3. *The rate associated with an encoder enc*: $\mathcal{X} \to \{1, \dots, K\}$ *is defined as* $\log K$.

Definition 1.3.4. *A* (R, D) *pair is achievable for a distribution* $p(X)$ *and distortion function* $d(\cdot, \cdot)$*, if there exists an encoder enc*: $\mathcal{X} \to \{1, \dots, \lfloor e^R \rfloor\}$ *and a decoder dec*: $\{1, \dots, \lfloor e^R \rfloor\} \to \hat{\mathcal{X}}$ *such that* $\mathbb{E}_{x \sim p(x)} d(x, dec(enc(x))) \leq D$*. The closure of the set of achievable pairs is called the rate-distortion region.*

Definition 1.3.5. *The rate-distortion function* $R(D)$ *is the infimum of rates* R *such that* (R, D) *is achievable.*

There is a monotonic trade-off between the rate of the code and the expected distortion. Essentially, the larger the rate, the smaller is the achievable distortion. This trade-off is characterized by the celebrated rate-distortion theorem of Shannon and Kolmogorov.

Theorem 1.3.1 (Rate-distortion theorem, Cover and Thomas (2012) (Theorem 10.2.1))**.** *The rate-distortion function for an iid random variable* X *with distribution* $p(x)$ *and a bounded distortion function* $d(\cdot, \cdot)$ *is given by*

$$R(D) = \min_{p(\hat{x}|x):\ \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X}) \quad \textit{with} \tag{1.45}$$

$$I(X; \hat{X}) := \sum_{x,\hat{x}} p(x)p(\hat{x}|x) \log \frac{p(\hat{x}|x)}{p(\hat{x})} \quad \textit{and} \quad p(\hat{x}) = \sum_x p(x)p(\hat{x}|x). \tag{1.46}$$

In general, $R(D)$ is difficult to calculate directly since the mutual information is intractable. However, since the mutual information has a nice variational characterization, we are able to compute $R(D)$ approximately by an iterative procedure. We will first introduce the related properties of $I(X; \hat{X})$ and $R(D)$, and then derive the variational characterization of $R(D)$.

Lemma 1.3.1. *The mutual information* $I(X; \hat{X})$ *is functional of* $p(x)$ *and* $p(\hat{x}|x)$*, thus it can be equivalently rewriten as* $I(X; \hat{X}) = I(p(x), p(\hat{x}|x))$*. Moreover, the mutual information is convex in* $p(\hat{x}|x)$ *and concave in* $p(x)$*.*

Proof. First, we decompose

$$\begin{aligned} I(X; \hat{X}) &= H(\hat{X}) - H(\hat{X}|X), \\ &= H(p(\hat{x})) - H(p(\hat{x}|x)). \end{aligned}$$

Since $p(\hat{x})$ is a linear function of $p(x)$, and $H(p(\hat{x}))$ is a concave function in $p(\hat{x})$, it immediately follows that $I(X; \hat{X}) = I(p(x), p(\hat{x}|x))$ and it is concave in $p(x)$.

To prove the convexity wrt $p(\hat{x}|x)$, we rewrite

$$I(X; \hat{X}) = \mathbb{E}_{p(x)} D_{\mathrm{KL}}(p(\hat{x}|x) \| p(\hat{x})).$$

Since the KL divergence is convex in both arguments, fixing $p(x)$, we have $I(X; \hat{X})$ is convex in $p(\hat{x}|x)$. □

Proposition 1.3.2. *For a rate-distortion function $R(D)$ wrt a distribution $p(X)$, we have the following properties*

1. *$R(D)$ is nonincreasing in D for all $D \geq 0$.*
2. *$R(D)$ is convex in D for all $D \geq 0$.*
3. *$R(D)$ is continuous in D for all $D \geq 0$.*
4. *$R(D) \leq H(p), \forall D \geq 0$.*

Proof. Having defined $R(D)$ as the solution of a constrained minimization, we have

1. The constraint set will be enlarged if D increases. Thus, the minimum can only remain unchanged or smaller, which implies that $R(D)$ is nonincreasing.
2. Consider two rate-distortion pairs (R_0, D_0) and (R_1, D_1), which are achieved at $q_0(\hat{x}|x)$ and $q_1(\hat{x}|x)$ respectively. Now define $q_\alpha(\hat{x}|x) = \alpha q_0(\hat{x}|x) + (1-\alpha)q_1(\hat{x}|x)$. Then, $D_\alpha = \alpha D_0 + (1-\alpha)D_1$. By Lemma 1.3.1, we have
$$\begin{aligned} R(D_\alpha) \leq I(p, q_\alpha) &\leq \alpha I(p, q_0) + (1-\alpha) I(p, q_1), \\ &= \alpha R(D_0) + (1-\alpha) R(D_1), \\ &= \alpha R_0 + (1-\alpha) R_1, \end{aligned}$$
which implies the convexity.
3. Assume by contradiction that $R(D)$ is discontinous at D_α such that $R(D_\alpha) = \lim_{D \to D_\alpha^-} R(D) > \lim_{D \to D_\alpha^+} R(D)$. Then, there exists two end points $D_0 \in (0, D_\alpha)$ and $D_1 \in (D_\alpha, +\infty)$ such that $\alpha R(D_0) + (1-\alpha)R(D_1) < R(D_\alpha)$ for some $\alpha \in (0, 1)$, which is a contradiction to the convexity of $R(D)$.
4. This is due to $I(X; \hat{X}) = H(X) - H(X|\hat{X}) \leq H(X)$. Then,
$$R(D) \leq \min_{p(\hat{x}|x):\ \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} H(X) = H(X),$$
since $H(X)$ is independent of $p(\hat{x}|x)$.

□

Since $R(D)$ is convex, for any $\beta \leq 0$, there exists a point on the curve of $R(D)$ for $0 \leq D \leq D_{\max}$, such that the slope of a tangent to the curve of $R(D)$ at that point is equal to $-\beta$. Denote such a point by $(D_s, R(D_s))$, which is not necessarily unique. Then the tangent line intersects with the R-axis at $(0, R(D_s) - s \cdot D_s)$, where $s = -\beta$. See Figure 1.1 for an example.

On the other hand, given an autoencoder induced by $p(\hat{x}|x) \in \mathcal{P}_{\hat{x}}$, where $\mathcal{P}_{\hat{x}} := \{p(\hat{x}|x) \geq 0 \mid \sum_{\hat{x}} p(\hat{x}|x) = 1\}$, we rewrite the rate and the distortion respectively as

$$I(p(\hat{x}|x)) := I(X; \hat{X}), \tag{1.47}$$
$$D(p(\hat{x}|x)) := \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x, \hat{x}). \tag{1.48}$$

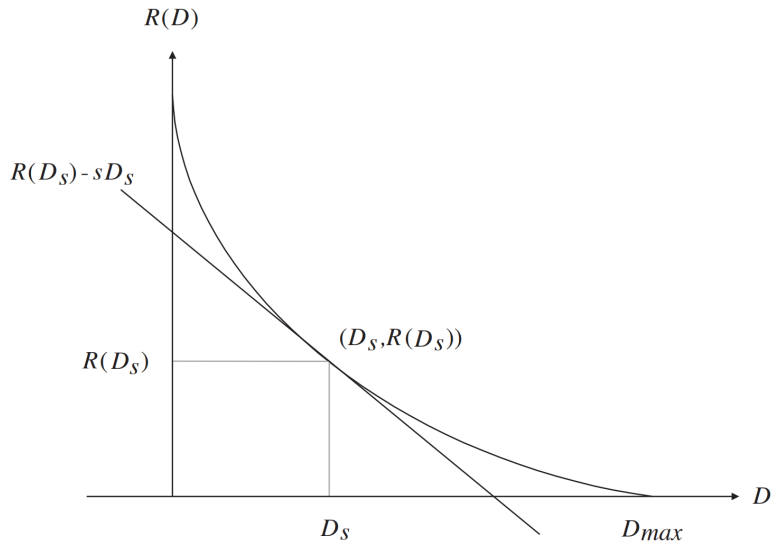


Figure 1.1: The rate-distortion plane.

Note that we leave the dependency on $p(x)$ implicit as $p(x)$ is not tunable by the autoencoder. For any $p(\hat{x}|x) \in \mathcal{P}_{\hat{x}}$, we obtain an achievable pair $(I(p(\hat{x}|x)), D(p(\hat{x}|x)))$ by definition. For each achievable pair and a given slope $s = -\beta$, it also specifies a line with R-intercept $I(p(\hat{x}|x)) - s \cdot D(p(\hat{x}|x))$. Among all the parallel lines defined by an achievable pair and the slope s, it is clear that $R(D_s) - s \cdot D_s$ yields the smallest R-intercept, namely,

$$R(D_s) + \beta \cdot D_s = \min_{p(\hat{x}|x) \in \mathcal{P}_{\hat{x}}} I(p(\hat{x}|x)) + \beta \cdot D(p(\hat{x}|x)). \tag{1.49}$$

Thus, by computing the RHS of (1.49), we can compute the value of the rate-distortion function $R(D)$ at D_s. This can also be understood as the geometrical interpretation of the Lagrangian of (1.45)

$$L(p(\hat{x}|x), \beta) := I(p(\hat{x}|x)) + \beta\Big[D(p(\hat{x}|x)) - D_s\Big]. \tag{1.50}$$

To further simplify the minimization of the RHS of (1.49), we resort to the following variational characterization of the mutual information.

Lemma 1.3.2 (Cover and Thomas (2012) (Theorem 10.8.1))**.** *The mutual information has a variational form:*

$$I(X;Y) = \min_{m(y)} D_{\mathrm{KL}}(p(x,y) \| p(x)m(y)),$$

where the argmin is attained at $m^*(y) = p(y) = \sum_x p(x,y)$.

By applying Lemma 1.3.2, the optimization in (1.49) becomes

$$\min_{p(\hat{x}|x) \in \mathcal{P}_{\hat{x}}} \min_{m(\hat{x}) \in \mathcal{M}} I(p(\hat{x}|x), m(\hat{x})) + \beta \cdot D(p(\hat{x}|x), m(\hat{x})), \tag{1.51}$$

where $\mathcal{M}$ is the set of valid instances of $m(\hat{x})$.

For the above characterization, Arimoto (1972), Blahut (1972) provide a convergent iterative algorithm for computing $p^*(\hat{x}|x)$ and $m^*(\hat{x})$ given a fixed β.

Proposition 1.3.3 (Blahut-Arimoto Algorithm)**.** *Let* $\mathcal{M} := \{m \mid m(\hat{x}) > 0, \forall \hat{x}\}$. *Provided an initial* $p_t(\hat{x})$ *at* $t = 0$. *At iteration* $t > 0$, *the Blahut-Arimoto algorithm takes the following steps:*

$$p_t(\hat{x}|x) = \frac{m_t(\hat{x})e^{-\beta d(x,\hat{x})}}{\sum_{\hat{x}'} m_t(\hat{x}')e^{-\beta d(x,\hat{x})}}, \tag{1.52}$$

$$m_{t+1}(\hat{x}) = \sum_x p(x)p_t(\hat{x}|x). \tag{1.53}$$

Moreover, the algorithm converges to the global minimum.

Proof. The proof can be found in Blahut (1972) or Cover and Thomas (2012). □

Chapter

2 Introduction to Probabilistic Graphical Models

Abstract

There exists many introductory materials for probabilistic graphical models, such as the textbook by Koller et al. (2009), Murphy (2012) and the monograph by Wainwright (2008). However, different materials have relatively different emphases. To give a modern review, it is important to revisit existing materials from an optimization perspective. Following the classical presentation, I will cover three main topics: model, inference and learning. I first explain the most fundamental idea of probabilistic graphical models, that is, the conditional independence and its graph representation. Next, I introduce the inference problem with an emphasis on the variational techniques. In particular, for the exponential family, different variational formulations correspond to different approximations to the log-parition function. Finally, I present two formulations for the learning problem with a focus on structured-output support vector machine. For both learning and inference, I also review the classical algorithms.

2.1 Introduction

Most of problems in machine learning can be seen as *inverse problems*, that is, given an iid sample of input-output pairs of the form (x, y), the goal is to estimate the *mapping* between x and y. In the case when $\mathcal{Y}$ is high-dimensional and represents a space of structured objects, the problem is known as *structured prediction*. It includes applications such as semantic image segmentation where $\mathcal{Y} = \mathbb{N}^k$ for an image with k pixels, human pose estimation where $\mathcal{Y} = \mathbb{R}^k$ for k body joints, dependency parsing where $\mathcal{Y}$ is the set of spanning trees, and so on. Accordingly, the input space $\mathcal{X}$ may also be structured, but the structure among the input is usually implicitly captured by the feature representation.

Structured prediction is computationally expensive even it has already saved a large amount of computation compared with its unstructured counterpart. For example, a discrete output space, e.g, $\mathcal{Y} = \mathbb{N}^k$, contains an exponential number of elements and grows very quickly when k increases. Therefore, the operations involving the whole output space is generally intractable, e.g., computing the mean $\mathbb{E}[Y]$ is NP-hard. A probabilistic graphical model (PGM) is a special family of distributions, where *conditional independence* (CI) assumptions among the domain are made based on human knowledge. The CI assumptions enable a *factorization* in the distribution, which reduces a global computation to local computations within the *factors*. Another key idea of PGMs is that the CI assumption can be linked to *separation* in a graph by associating each node a random variable. This is where the name of PGM comes from. Adding conditional independences will basically reduce the *treewidth*[1] of the graph, and thus make the statistical inference (approximately) solvable in polynomial time.

Although the CI assumptions yield computational benefits, one would ask whether these assumptions are realistic? A probabilistic graphical model is sometimes referred to as a knowledge-driven method in constrast with data-driven methods such as deep learning. It is knowledge-driven since we derive a model from high-level human knowledge to hypothesize the observed data. Thus, the model itself is, at least partially, *interpretable*. For example, a time series is naturally modeled as a *chain graph* in the context of PGMs. This amounts to introducing the *first-order Markovian* assumption. For real problems, such as weather forecasting, this assumption leads to a probabilistic model based only on the observation of yesterday. This is certainly not the best choice, but it is intuitive from a human perspective. In general, PGM offers a practical framework to embed a wide range of independence assumptions, which is not easy for data-driven methods.

In the following, I will present the key ingredients of PGMs. For a sake of clarity the presentation will concentrate on variational inference.

[1] The treewidth of a graph is the size of the largest vertex set in a *tree decomposition* of the graph minus one.

2.2 Models

Let $X := (X_1, \ldots, X_n)$ denote a random vector, where each element X_i is a random variable taking value in the space $\mathcal{X}_i$. A PGM with respect X includes a probability distribution $p(x)$ over X and a graph $G = (V, E)$, such that each node in V is associated with a random variable and the edges in E encode the relationships/dependencies between X_i's. Following Wainwright (2008), I treat $p(x)$ as the Radon-Nikodym derivative of the probability measure with respect to the base measure (which is counting measure in the discrete case and Lebesgue measure in the continuous case).

Factorization

Given G, we are able to incorporate the CI assumptions into the definition of $p(x)$ through the *factorization* according to G. Specifically, $p(x)$ is expressed as a product of nonnegative local functions (called *potential functions*), each of which literally involves only a small set of variables forming a *factor*. In the directed case, a factor denotes the parent-child neighborhood. In the undirected case, a factor is a *clique*, which is a fully connected subgraph, such as a node or an edge. Note that the set of factors is not necessarily the set of *maximal cliques*, i.e., the set of cliques that are not properly contained within any other clique. Therefore, the factorization according to G may not be unique. From a modeling perspective, a model of $p(x)$ is more powerful by including *higher-order* cliques, since it requires fewer CI assumptions, but at the same time the computational overload is increased.

Denote by $\mathcal{A}$ the set of factors. For $a \in \mathcal{A}$, we have a potential function $\psi_a : \mathcal{X}_a \to \mathbb{R}$, where $\mathcal{X}_a := \bigtimes_{i \in a} \mathcal{X}_i$. We say that $p(x)$ factorizes according to G if

$$p(x) = \frac{1}{Z} \prod_{a \in \mathcal{A}} \psi_a(x_a), \text{ with } Z = \sum_{x \in \mathcal{X}} \prod_{a \in \mathcal{A}} \psi_a(x_a), \tag{2.1}$$

where Z is called the *partition function* with respect to ψ, serving as a normalizing constant. For undirected graphs, the corresponding PGMs are called *Markov random fields* (MRFs) and $p(x)$ is called the *Gibbs distribution*. For DAGs, the corresponding PGMs are called *Bayesian networks* (BNs) and $p(x)$ takes a special form:

$$p(x) = \prod_{i \in V} \psi_i(x_i, \text{pa}(x_i)), \tag{2.2}$$

where $\psi_i \geq 0$ and $\forall i, \forall \text{pa}(x_i)$, $\sum_{x_i} \psi_i(x_i, \text{pa}(x_i)) = 1$. In this case, we always have $Z = 1$, since the potential function $\psi_i(x_i, \text{pa}(x_i))$ is exactly the conditional probability distribution $p(x_i|\text{pa}(x_i))$.

Note that, without the factorization, the time complexity for computing Z is $O(|\mathcal{X}|)$, where $|\mathcal{X}|$ returns the cardinality of the space $\mathcal{X}$. For PGMs, the time complexity reduces to $O(n \cdot \prod_{i \in a^*} |\mathcal{X}_i|)$, where a^* is the factor with the maximum number of nodes.

A slightly more general graphical representation is called the *factor graph* (Kschischang et al. 2001), which is a bipartite graph $G = (V, E, \mathcal{A})$ including the set of variable nodes V, the set of factor nodes $\mathcal{A}$, and the set of edges between a factor and a variable. An example of the factor graph is shown in Figure 2.1. The factor graph is particularly useful in understanding the *belief propagation* algorithms for probabilistic inference (Kschischang et al. 2001).

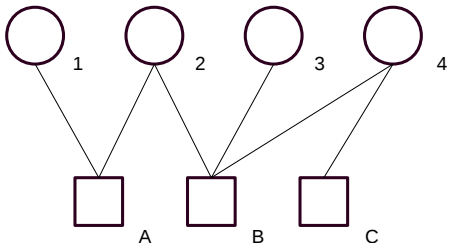


Figure 2.1: A factor graph example. Circles represent variable nodes. Boxes represent factor nodes. Note that there are only edges between circles and boxes.

Equivalence between conditional independence and factorization

The key feature of PGMs is the connection between the graph and the factorization of the probability distribution. The connection is built upon the CI assumptions. In fact, the graph itself does not specify the conditional independences. The *connectivity* is a proxy to reason the dependencies between random variables since each random variable is associated with a node. In analogy, verifying a conditional independence, $S \perp\!\!\!\perp T \mid E$, is equivalent to checking if S is *separated* from T by E, where S, T, E are three disjoint sets of nodes. The concept of separation is not uniquely defined. For a undirected graph, the definition of *separation* is straightforward:

S is separated from T given E, if E intersect with every path between S and T.

For a directed acyclic graph (DAG), the *d-separation* (Koller et al. 2009) is used, for which I adopt an informal definition from (Murphy 2012): S and T are d-separated given E, if any undirected path between S and T satisfies one of the following conditions:

1. The path contains an *indirect causal-effect structure*, $A \to M \to B$ or an *indirect evidential-effect structure* $A \leftarrow M \leftarrow B$, where $M \in E$.

2. The path contains a *common-causal structure*, $A \leftarrow M \to B$, where $M \in E$.

3. The path contains a *V-structure*, $A \to M \leftarrow B$, where neither M nor its descendants is in E.

Now, we are ready to define the two types of CI in PGMs induced by the factorization and the graph respectively:

- *Independences in distribution*:

$$I(p) := \{(A \perp\!\!\!\perp B \mid E) \,\big|\, p(A, B|E) = p(A|E)p(B|E)\}.$$

- *Independences in graph*:

$$I(G) := \{(A \perp\!\!\!\perp B \mid E) \,\big|\, A \text{ is separate/d-separated from } B \text{ given } E\}.$$

We say G is an *I-Map* of $p(x)$ if $I(G) \subseteq I(p)$, which means that G does not contain any local independence that does not hold in $p(x)$. Conversely, $p(x)$ may have additional independences that are not reflected in G. As a special case, if G is fully connected, then it is an I-Map for any distribution since it does not assert any conditional independence assumption.

There is a fundamental connection between CI assumptions and the factorization. Koller et al. (2009, Theorem 3.1, 3.2, 4.1, 4.2) have shown this connection for the directed case and the undirected case respectively. The results can be summarized as follows.

1. If $p(x)$ factorizes according to G, then G is an I-Map of $p(x)$.
2. If G is an I-Map of $p(x)$, then $p(x)$ factorizes according to G.

For the second statement, we need an additional assumption for undirected graphs: $p(x)$ is *positive*, which is also known as the *Hammersley-Clifford theorem* (Koller et al. 2009, Theorem 4.2).

Is $I(G) = I(p)$?

In addition to the above equivalence, one may be interested in whether the statement $I(G) \supseteq I(p)$ also holds true given that G is an I-Map of $p(x)$ and $p(x)$ factorizes over G. If so, the set of conditional independences encoded in G is equivalent to that encoded in $p(x)$. Unfortunately, this desired property is not true in general as shown by a counterexample in Koller et al. (2009, Example 3.3), where the CI assertion $A \perp\!\!\!\perp B$ is not refected in the graph $A \to B$, even if $p(A, B) = p(A)p(B|A)$ does factorize according to the graph.

If $p(x)$ factorizes according to G, $p(x)$ is specified by the set of potential functions $\{\psi_a\}_{a \in \mathcal{A}}$ by eq.(2.1). The CI assertions outside $I(G)$ can then be viewed as the set of constraints on the potential functions, which can be written as polynomial equalities about the potential functions (Koller et al. 2009, Exercise 3.13). A basic property of polynomial functions is that it is either identically zero or non-zero almost everywhere. Hence, the set of potential functions that satisfy the CI outside $I(G)$ has measure zero. Hence, for almost all instances of $p(x)$ that factorize according to G, we have $I(G) = I(p)$. For a more rigorous analysis, please refer to Koller et al. (2009, Section 3.3.2). Back to the counterexample aforementioned, the CI outside $I(A \to B)$ is given by $p(B|A) = p(B)$. Then, the set $\{p(A, B) \mid p(B|A) = p(B)\}$ has measure zero. This makes sense since $I(A \to B) = I(p)$ except for a special case when $p(B|A) = p(B)$.

Comparison between BNs and MRFs

To finalize this section, I make a comparison between the Bayesian networks and the Markov random fields. There is no concise conclusion on which model is more powerful. In fact, they introduce relatively different CI assumptions. For example, MRFs cannot precisely represent the CIs induced by the v-structures, while BNs cannot precisely represent the CIs induced by cyclic structures (Murphy 2012, Section 19.2.3). A special case is the *chordal* graph, which can be perfectly modeled by either BNs or MRFs. In addition, we say a marginalization is *closed* for $p(x)$ if $p(x)$ factorizes in G implies $p(x_{V\setminus i})$ factorizes in the induced graph by removing node i and connecting all neighbors of i. This however only holds for the leafs in the case of BNs. A comparison between Bayesian networks and Markov random fields is shown in Table 2.1.

	Bayesian networks	Markov random fields
Factorization	$p(x) = \prod_{i=1}^{n} p(x_i \mid x_{\mathrm{pa}(i)})$	$p(x) = \frac{1}{Z} \prod_{a \in \mathcal{A}} \psi_a(x_a)$
Independence	d-separation	Separation
Marginalization	Not closed in general, only when marginalizing leaf nodes	Closed

Table 2.1: Comparison of some properties of Bayesian networks and Markov random fields.

2.3 Inference

Given a probability distribution $p(x)$ defined by a PGM, our focus will be solving one or more of the following probabilistic query problems, which are generically referred to as *inference* problems:

1. Computing the density function $p(x)$ via eq.(2.1): one needs to estimate the value of the partition function Z.
2. Computing the marginal distribution $p(x_A)$ over a particular subset $A \subset V$ of nodes.
3. Computing the conditional distribution $p(x_A|x_B)$ for two disjoint subsets A, B.
4. Computing the mode of $p(x)$.

The first three problems are called *probabilistic inference* or *marginal inference*, since they all involve computing summations of the form $\sum_{x_S} \prod_{a \in \mathcal{A}} \psi_a(x_a)$, specifically, the quantities read as

1. $p(x) = \frac{\prod_{a \in \mathcal{A}} \psi_a(x_a)}{\sum_{x'} \prod_{a \in \mathcal{A}} \psi_a(x'_a)}$, the density function.

2. $p(x_A) = \frac{\sum_{x \setminus x_A} \prod_{a \in \mathcal{A}} \psi_a(x_a)}{\sum_{x'} \prod_{a \in \mathcal{A}} \psi_a(x'_a)}$, the marginal distribution.

3. $p(x_A|x_B) = \frac{\prod_{a \in \mathcal{A}} \psi_a(x_a)}{\sum_{x' \setminus x'_B} \prod_{a \in \mathcal{A}} \psi_a(x'_a)}$, the conditional marginal distribution.

On the other hand, the problem of computing the mode is relatively easier, which by definition can be expressed as

$$\arg\max_x p(x) = \arg\max_x \psi(x). \tag{2.3}$$

Computing the mode is sometimes known as *decoding* or *maximum a posteriori (MAP) inference* or *most probable explanation (MPE) inference*.

Although these inference problems are NP-hard in the worst case, exact inference is possible for certain special cases, such as when the graph is a tree (i.e., an undirected graph without cycles or a directed graph in which each node has a single parent) or when the (loopy) graph whose *tree decomposition* satisfies the *running intersection property*, meaning that any subset of nodes containing a given variable forms a connected component. For a tree-structured PGM, the *sum-product belief propagation* algorithm yields exact marginals for each clique. A variant called *max-product belief propagation* can be used for solving the MAP inference problem. From an algorithmic point of view, sum-product belief propagation is a form of *nonserial dynamic programming* (Bertele and Brioschi 1972) while max-product belief propagation is a standard dynamic programming algorithm with the *optimal substructure* induced by the tree-structured graph. These two algorithms have a computational complexity that scales only linearly in the number of nodes. The same algorithms can also be extended to the tree decomposition case with running intersection property, which is known as the *junction tree algorithm* (see for example Koller et al. (2009, Chapter 10) or Wainwright (2008, Chapter 2) for a complete description), and the computational complexity also depends on the *treewidth* of the graph.

However, there are many PGMs, e.g., the grid-structured graphs for semantic image segmentation, for which the treewidth is infeasibly large. Therefore, the exact inference may not be practical any more. Many smart ideas for approximate inference have been proposed, including *variational* methods (Jordan et al. 1999) and *Monte Carlo* methods (Robert and Casella 2013). The former casts the inference as an optimization by minimizing some divergence (e.g., the *Kullback-Leibler* divergence) between an approximate distribution and the intractable true distribution. The approximate distribution is then used as a proxy for computing the quantities of interest for probabilistic queries (such as the mean and the mode of the population). The latter is based on a simple idea: generating samples from a *proposal* distribution, which are then processed to compute any quantity of interest. The most popular Monte Carlo method is called *Markov chain Monte Carlo* (MCMC), including implementations such as the *Gibbs sampling* and the *Metropolis–Hastings* algorithm (Metropolis et al. 1953), which is guaranteed to work in high-dimensional spaces. By generating sufficient samples, the approximate quantities are able to achieve any level of accuracy. That being said, both the

bias and the variance of MCMC estimate approaches zero as the Markov chain runs longer and longer. On the other hand, variational methods are considered if the error due to the variance is higher than that of the bias within a short amount of time. This is usually the case for the inference and learning problems of PGMs. Because of that variational methods are preferred over Monte Carlo methods. Moreover, comparing to MCMC, variational inference is easier to scale to large data. I will therefore only cover variational inference in this chapter.

2.3.1 Marginal inference

If exact inference for a PGM is computationally intractable, we can alternatively search for a tractable PGM that approximates the original PGM as close as possible. For instance, if we remove all other conditional independences except for those contained in a spanning tree of the graph, we obtain an approximate PGM with tractable inference. This strategy is critical since, under a certain time budget, approximate inference may be the only afordable choice for large-scale machine learning problems.

The key idea behind the variational view of inference is that we perform inference on an approximate distribution that remains the closest to the original PGM in terms of some distance measures, such as KL divergence, Jensen-Shannon divergence, Wasserstein distance and so on.

Exponential family formulation for inference

Consider a PGM$= (p(x), G)$ with $p(x) = \frac{1}{Z}\prod_{a\in\mathcal{A}}\psi_a(x_a)$. It turns out that a special parameterization of $\psi_a(x_a)$ offers a different interpretation of the inference. Specifically,

$$\psi_a(x_a;\theta_a) = \exp\Big(\langle\theta_a, \phi_a(x_a)\rangle\Big) \tag{2.4}$$

with both θ_a and $\phi_a(x_a)$ in $\mathbb{R}^{d_a}$ ($d_a \geq 1$ is a hyperparameter). As an example, $\phi_i(x_i)$ is usually provided as a $|\mathcal{X}_i|$-dimensional *one-hot* vector, denoted by e_{x_i}, with 1 for the x_i-th element and 0 elsewhere; and $\phi_a(x_a) = \otimes_{i\in c}\phi_i(x_i)$ is the result of a series of tensor products. Then, we can interpret $\theta_a = (\log\psi_a(x_a))_{x_a\in\mathcal{X}_a}$ to be the vector of log-potentials. In this chapter, we assume $\phi_a(x_a)$ is not learnable. However, this assumption can be relaxed in the case of inference, since we can always absorb ϕ_a into θ_a and let $\psi_a(x_a) = \exp(\langle\theta_a, e_{x_a}\rangle)$.

With this parameterization of ψ, we obtain an *exponential family* form for $p(x)$ as

$$p_\theta(x) = \exp\Big(\langle\theta,\phi(x)\rangle - F(\theta)\Big) = \exp\Big(\sum_{a\in\mathcal{A}}\langle\theta_a,\phi_a(x_a)\rangle - F(\theta)\Big), \tag{2.5}$$

where $\theta := (\theta_a)_{a\in\mathcal{A}}$ is the *natural parameter*; $\phi(x) := (\phi_a(x_a))_{a\in\mathcal{A}}$ is called the *sufficient statistics*; and $F(\theta) = \log Z$ is the *log-partition function* in which we explicited the dependency on θ. Note that exponential family is indeed a strict

subset of the distributions that factorize according to G, among which it achieves the maximum entropy.

The different interpretation of the inference is that the variational inference for exponential family with Kullback-Leibler (KL) divergence is equivalent to approximating the log-partition function. To give a formal justification, we will first go through some properties of the log-partition function $F(\theta)$.

Lemma 2.3.1 (Wainwright 2008, Proposition 3.1)**.** *The log-partition function associated with any regular exponential family satisfies*

1. $\frac{\partial F(\theta)}{\partial \theta_a} = \mathbb{E}_{p_\theta}\Big[\phi_a(X_a)\Big]$ *for all* $a \in \mathcal{A}$.
2. $F(\theta)$ *is convex in* $\theta \in \Theta$ *and strictly convex if* F *defines a minimal exponential family.*

Lemma 2.3.2. *The Legendre-Fenchel transformation of* $F(\theta)$ *is the negative entropy of the form*

$$H(q) = \int_{\mathcal{X}} q(x) \log q(x) \nu(dx). \tag{2.6}$$

Proof. Recall that the Legendre-Fenchel transformation of $F(\theta)$ is

$$F^*(q) = \sup_{\theta \in \mathrm{dom}(F)} [\int_{\mathcal{X}} \langle \theta, \phi(x) \rangle q(x) \nu(dx) - F(\theta)]. \tag{2.7}$$

Taking derive with respect to θ yields the condition

$$q(x) = \frac{\exp\langle \theta, \phi(x) \rangle}{\int_{\mathcal{X}} \exp\langle \theta, \phi(x) \rangle \nu(dx)}. \tag{2.8}$$

Thus,

$$\log q(x) = \langle \theta, \phi(x) \rangle - F(\theta). \tag{2.9}$$

Using the above equality and the fact that $\int_{\mathcal{X}} q(x)\nu(dx) = 1$, eq.(2.7) can be rewritten as

$$F^*(q) = \sup_{\theta \in \mathrm{dom}(F)} [\int_{\mathcal{X}} q(x)\Big(\log q(x) + F(\theta)\Big)\nu(dx) - F(\theta)] \tag{2.10}$$

$$= \int_{\mathcal{X}} q(x) \log q(x) \nu(dx). \tag{2.11}$$

It concludes the proof. □

Computing $F(\theta)$ as a KL minimization

We show in the following theorem that the KL minimization is related to the Legendre-Fenchel transformation of $F(\theta)$.

Theorem 2.3.1. *Minimizing $D_{KL}(q\|p_\theta)$ over a convex set $\mathcal{P}$ is equivalent to finding an optimal solution to the problem*

$$\max_{q\in\mathcal{P}} \Big[\langle\theta, \mathbb{E}_q[\phi(X)]\rangle + H(q)\Big]. \tag{2.12}$$

If $p_\theta \in \mathcal{P}$, the above optimal objective is exactly equal to $F^{**}(\theta) \equiv F(\theta)$.

Proof. Substituting the definition of $p_\theta(x)$, we observe that

$$D_{KL}(q\|p_\theta) = F(\theta) - \mathbb{E}_q\Big[\log \frac{\exp\Big(\langle\theta, \phi(X)\rangle\Big)}{q(X)}\Big] \tag{2.13}$$

$$= F(\theta) - \mathbb{E}_q[\langle\theta, \phi(X)\rangle] - H(q) \geq 0. \tag{2.14}$$

Since $F(\theta)$ does not depend on $q(x)$, we have that

$$\max_{q\in\mathcal{P}} \mathbb{E}_q[\langle\theta, \phi(X)\rangle] + H(q) \Leftrightarrow \min_{q\in\mathcal{P}} D_{\mathrm{KL}}\Big(q\|p_\theta\Big). \tag{2.15}$$

Since the KL divergence is strictly convex in q for θ fixed, if $p_\theta(x) \in \mathcal{P}$, we have $\min_{q\in\mathcal{P}} D_{KL}(q\|p) = 0$, and the argmin is attained at $q^*(x) = p_\theta(x)$, yielding

$$F(\theta) = \mathbb{E}_{q^*}[\langle\theta, \phi(X)\rangle] + H(q^*). \tag{2.16}$$

Note that $H(q) = -F^*(q)$. Since $F(\theta)$ is convex in θ by Lemma 2.3.1, we have $F(\theta) = F^{**}(\theta)$. □

Theorem 2.3.1 offers a variational formulation for computing $F(\theta)$, which is the key idea that casts the marginal inference as an optimization problem. More importantly, it derives a *dual* representation for $F(\theta)$ with the dual variable $q \in \mathcal{P}$. Denote by Δ is the probability simplex. It is easy to see that if $\mathcal{P} = \Delta$, Theorem 2.3.1 holds since $p_\theta \in \Delta$, that is,

$$F(\theta) = \max_{q\in\Delta} \Big[\langle\theta, \mathbb{E}_q[\phi(X)]\rangle + H(q)\Big]. \tag{2.17}$$

Note that q is indeed a functional corresponding to a *primal* functional $\theta \equiv \log\psi$, which means that the same result holds even if we did not use the linear parameterization $\theta(x) = \log\psi(x) = \langle\theta, \phi(x)\rangle$. Since now we specify the primal variable $\theta = (\theta_a)_{a\in\mathcal{A}}$ to be a vector with the same dimensionality as that of $\phi(x)$, we expect the dual variable to be a vector as well rather than a functional.

Denote by μ the dual variable, which is also known as the *mean parameter* of the exponential family. In fact, the primal and the dual is coupled by the so-called *moment matching condition*:

$$\mu_a = \frac{\partial F(\theta)}{\partial\theta_a} = \mathbb{E}_{p_\theta}[\phi_a(X_a)], \forall a \in \mathcal{A}, \tag{2.18}$$

which is given by the *Fenchel-Young's inequality* when the equality holds. If we were using functional primal and dual, namely, θ and q respectively, the above condition became

$$q(x) = \frac{\partial F(\theta)}{\partial \theta(x)} = p_\theta(x). \tag{2.19}$$

Note that, for all $\theta \in \Theta := \{\theta \mid F(\theta) < +\infty\}$, there is an induced dual variable μ by the moment matching condition. This induces a space of μ, called *marginal polytope*, and denoted by

$$\mathcal{M} := \{(\mu_a)_{a\in\mathcal{A}} \mid \exists \theta \in \Theta \text{ s.t. } \mathbb{E}_{p_\theta}[\phi_a(X_a)] = \mu_a, \forall a \in \mathcal{A}\}. \tag{2.20}$$

With the same reasoning, we can define the "functional" version of the marginal polytope to be

$$\mathcal{Q} := \{q \in \Delta \mid \exists \theta \in \Theta \text{ s.t. } q(x) = p_\theta(x), \forall x \in \mathcal{X}\}. \tag{2.21}$$

Due to the coupling between the primal and the dual, in eq.(2.17), the space of q can actually be $\mathcal{Q}$, which is a subset of Δ. Thus, by the definition of $\mathcal{Q}$, we have

$$F(\theta) = \max_{\eta\in\Theta} \Big[\langle \theta, \mathbb{E}_{p_\eta}[\phi(X)]\rangle + H(p_\eta)\Big]. \tag{2.22}$$

Similarly, we have, in the next theorem, a dual representation for $F(\theta)$ in terms of the marginal polytope.

Theorem 2.3.2 (Wainwright 2008, Theorem 3.4)**.** *For any μ belonging to the interior of $\mathcal{M}$ (denote by $\mathcal{M}^\circ$), there exists $\theta \in \Theta := \{\theta \mid F(\theta) < +\infty\}$, such that the moment matching condition holds, namely,*

$$\mu_a = \mathbb{E}_{p_\theta}[\phi_a(X_a)], \forall a \in \mathcal{A}. \tag{2.23}$$

For any natural parameter $\theta \in \Theta$ satisfying the above condition, the Fenchel conjugate of $F(\theta)$ is defined by

$$F^*(\mu) = \begin{cases} -H(p_\theta) & \text{if } \mu \in \mathcal{M}^\circ \\ \lim_{\nu\in\mathcal{M}^\circ\to\mu} F^*(\nu) & \text{if } \mu \in \mathcal{M} \setminus \mathcal{M}^\circ \\ +\infty & o.w. \end{cases} \tag{2.24}$$

Hence, the log-partition function $F(\theta)$ has a variational representation in terms of Fenchel duality:

$$F(\theta) = \max_{\mu\in\mathcal{M}} \Big[\langle \theta, \mu\rangle - F^*(\mu)\Big]. \tag{2.25}$$

Intuitively, the variational representation of $F(\theta)$ in eq.(2.25) is itself a maximization over the marginal polytope. It in general does not reduce the computational complexity of the inference problem, since both $\mathcal{M}$ and $F^*(\mu)$ are inherently intractable. In particular, $\mathcal{M}$ has an exponential number of linear constraints;

$F^*(\mu)$ depends on the negative entropy of $p_\theta(x)$, but which in general lacks an explicit form to compute in terms of μ. Furthermore, since F is not strictly convex, unless $p(x)$ belongs to a *minimal exponential family*, meaning that there does not exist a nonzero θ such that $\langle \theta, \phi(x) \rangle =$ constant for any x, $\theta(\mu)$ is in general not uniquely determined. All these entanglements render difficulty to formulate the Fenchel conjugate of $F(\theta)$ explicitly.

There are special cases where the variational representation of $F(\theta)$ is amenable, such as PGMs induced by degenerate graphs (no edges) and tree-structured graphs, in which cases both $\mathcal{M}$ and F^* take special forms. The ideas, such as *mean-field* variational inference and *Bethe* variational inference, rely on extending the special forms of $\mathcal{M}$ and F^* to general PGMs, which are categorized as approximate marginal inference.

Mean-field inference

The idea of mean-field approximation first appeared in statistical field theory (Parisi 1988). The mean-field variational inference has been sucessfully applied to Bayesian inference and other probabilistic reasonings (Blei et al. 2017). In the context of PGMs, the mean-field approximation arises from ignoring certain types of dependencies between random variables.

Theorem 2.3.3. *Consider the following PGM* $= \Big(p_\theta(x), G = (V, E, \mathcal{A})\Big)$*:*

- $p_\theta(x) = \exp\Big(\langle \theta, \phi(x) \rangle - F(\theta)\Big)$ *belonging to a minimal exponential family, and* $\theta \in \Theta := \{\theta \mid F(\theta) < +\infty\}$*;*
- *Suppose that* $T = (V, E_T, \mathcal{A}_T)$ *is a subgraph of the original graph* G*, such that* T *is acyclic, in other words,* $E_T \subset E, \mathcal{A}_T = V \cup E_T \subset \mathcal{A}$*.*
- *Denote by* Θ_T *the set of natural parameters induced by* T*, namely,* $\Theta_T = \{\theta \in \Theta \mid \theta_a = 0 \text{ if } a \notin \mathcal{A}_T\}$*.*

The mean-field variational inference for such a PGM picks an approximate distribution $q(x)$ *by solving*

$$\min_{q \in \mathcal{Q}_{mf}} D_{KL}(q \| p_\theta), \tag{2.26}$$

where $\mathcal{Q}_{mf} := \{q \in \Delta \mid \exists \theta \in \Theta_T \text{ s.t. } q(x) = p_\theta(x), \forall x \in \mathcal{X}\}$ *and* Δ *is the probability simplex. This is equivalent to approximating* $F(\theta)$ *by*

$$F_{\mathrm{mf}}(\theta) := \max_{\mu \in \mathcal{M}_{\mathrm{mf}}} \Big[\langle \theta, \mu \rangle - F^*_{\mathrm{mf}}(\mu)\Big] \quad \text{with} \tag{2.27}$$

$$\mathcal{M}_{\mathrm{mf}} := \{(\mu_a)_{a \in \mathcal{A}} \mid \exists \theta \in \Theta_T \text{ s.t. } \mathbb{E}_{p_\theta}[\phi_a(X_a)] = \mu_a, \forall a \in \mathcal{A}\}, \tag{2.28}$$

$$F^*_{\mathrm{mf}}(\mu) := -\sum_{ij \in E_T} H(q_{ij}) + \sum_{i \in V} (d(i) - 1) H(q_i), \tag{2.29}$$

where q_i, q_{ij} *satisfies* $\mathbb{E}_{q_a}[\phi_a(X_a)] = \mu_a$ *for all* $i, ij \in \mathcal{A}_T$*. Moreover,* $F_{\mathrm{mf}}(\theta) \leq F(\theta)$*.*

Proof. By Theorem 2.3.1, we have

$$\min_{q\in\mathcal{Q}_{\mathrm{mf}}} D_{\mathrm{KL}}(q\|p_\theta) \Leftrightarrow \max_{q\in\mathcal{Q}_{\mathrm{mf}}} \Big[\langle\theta, \mathbb{E}_q[\phi(X)]\rangle + H(q)\Big]. \tag{2.30}$$

By the definition of $\mathcal{Q}_{\mathrm{mf}}$, there must exists a $\eta \in \Theta_T$, such that $q = p_\eta$. Since T is acyclic, there exists a junction tree representation of $p_\eta(x)$ (Wainwright 2008, eq.(2.12)):

$$p_\eta(x) = \frac{\prod_{ij\in E_T} q_{ij}(x_{ij})}{\prod_{i\in V} q_i(x_i)^{d(i)-1}}, \tag{2.31}$$

where $q_a(x_a)$ is the marginal distribution of x_a, i.e., $q_a(x_a) = p_\eta(x_a)$, for all $i, ij \in \mathcal{A}_T$, and $d(i)$ is the number of edges in E_T connected to node i. Hence,

$$H(q) = H(p_\eta) = \sum_{ij\in E_T} H(q_{ij}) - \sum_{i\in V}(d(i)-1)H(q_i). \tag{2.32}$$

By Theorem 2.3.2, since η also belongs to Θ, there exists a $\mu \in \mathcal{M}^\circ$ such that $\mathbb{E}_q[\phi_a(X_a)] = \mu_a, \forall a \in \mathcal{A}$. Note that the mapping defined by this equation between q and μ is one-to-one, since $q \equiv p_\eta$ belongs to a minimal exponential family. This established a change of variable for the RHS optimization in eq.(2.30):

$$\max_{\mu,q} \Big[\langle\theta,\mu\rangle + H(q)\Big] \tag{2.33}$$

$$\text{s.t.} \quad \mu \in \mathcal{M}^\circ \cap \{\mu \mid \mathbb{E}_q[\phi(X)] = \mu\} \quad \text{and} \quad q \in \mathcal{Q}_{\mathrm{mf}}. \tag{2.34}$$

Since q can be seen as a function of μ, the equivalence follows by absorbing the above constraints in eq.(2.34) into a single constraint encoded by $\mathcal{M}_{\mathrm{mf}}$, leaving a single variable μ.

The mean-field approximation $F_{\mathrm{mf}}(\theta)$ yields a lower bound of the true value of $F(\theta)$ simply because $\mathcal{M}_{\mathrm{mf}}$ is a subset of $\mathcal{M}$. □

Note that the variational problem in eq.(2.27) involves solving equations $\mathbb{E}_{q_a}[\phi_a(X_a)] = \mu_a$ for all $a \in \mathcal{A}_T$, which is in general nontrivial. For a special case where $\mathcal{X}_a$ is discrete, $\mathcal{A} = V \cup E$ and $\phi_a(x_a)$ is an one-hot encoded tensor such that $\phi_i(x_i) = e_{x_i}$ and $\phi_{ij}(x_{ij}) = \phi_i(x_i)\phi_j(x_j)^\top$, we have that

$$\mathbb{E}_{q_a}[\phi_a(X_a)] = \mu_a \Leftrightarrow q_a = \mu_a.$$

In this case, the mean parameter $\mu_a(x_a)$ is indeed the marginal probability of x_a. If in addition, we choose $T = (V, E_T = \emptyset, \mathcal{A}_T = V)$, then

$$F^*_{\mathrm{mf}}(\mu) = -\sum_{i\in V} H(\mu_i),$$

and the optimization in eq.(2.27) can be rewritten as

$$\max_{\mu} \Big[\langle\theta,\mu\rangle + \sum_{i\in V} H(\mu_i)\Big] \tag{2.35}$$

$$\text{s.t.} \quad \forall a \in \mathcal{A}\colon \mu_a \in \Delta \text{ and } \forall ij \in E\colon \mu_{ij} = \mu_i\mu_j^\top. \tag{2.36}$$

This is a concave optimization over a non-convex set, which is typically optimized by a coordinate ascent scheme. For the optimization in eq.(2.35), the update rule is

$$\mu_i(x_i) \propto \exp\Big(\langle \theta_i, \phi_i(x_i)\rangle\Big) \exp\Big(\sum_{j\in \text{neighbors}(i)} \langle \theta_{ij}, \phi_i(x_i)\mu_j^\top\rangle\Big). \tag{2.37}$$

Loopy belief propagation

Although the belief propagation algorithm was originally designed for acyclic graphs, it was found by Pearl (1988) that the same propagation rules can be extended to general graphs with loops, resulting in an iterative algorithm called *loopy belief propagation* (LBP). For a discrete PGM with density $p(x) = \frac{1}{Z}\prod_{i\in V}\psi_i(x_i)\prod_{ij\in E}\psi_{ij}(x_i,x_j)$, LBP involves the following updates:

$$m_{i\to j}(x_j) = \sum_{x_i} \psi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in \text{neighbors}(i)\setminus j} m_{k\to i}(x_i), \tag{2.38}$$

$$\tilde{\mu}_i(x_i) \propto \psi_i(x_i) \prod_{j\in \text{neighbors}(i)} m_{j\to i}(x_i), \tag{2.39}$$

where $m_{i\to j} \in \mathbb{R}^{|\mathcal{X}_j|}$ is the *message* sending from node i to node j; and $\tilde{\mu}_i$ is an estimate of the marginal probability of node i. At each iteration of LBP, each node will send a message to each of its neighbors. The iterations continue until $(\mu_i)_{i\in V}$ converges. Note that LBP does not guarantee the convergence of $\tilde{\mu}$. Thus, in practice, a heuristic called *damping* is used to facilitate the convergence of LBP, which uses damped messages of the form

$$\tilde{m}_{i\to j}(x_j) \leftarrow \lambda m_{i\to j}(x_j) + (1-\lambda)\tilde{m}_{i\to j}(x_j) \tag{2.40}$$

rather than the original messages in the iterations. For a more general PGM with higher order factors, we usually apply LBP on its factor graph. Please refer to the monograph by Nowozin et al. (2011, Section 3.2) for a full description.

Yedidia et al. (2003) shows that LBP amounts to solving the fixed-point equations induced by the KKT conditions of the *Bethe approximation* of $F(\theta)$. Thus, if LBP converges, it converges to a fixed point of the KKT system.

To formally introduce the Bethe approximation of $F(\theta)$, the *local consistency polytope* should be introduced first, which is defined by

$$\mathcal{L} := \{(\tau_a)_{a\in\mathcal{A}} \mid \forall a\in\mathcal{A}: \tau_a \in \Delta, \forall i\in a, x_i\in\mathcal{X}_i : \sum_{x_a\setminus x_i} \tau_a(x_a) = \tau_i(x_i)\}. \tag{2.41}$$

Intuitively, in the context of PGMs, the local consistency polytope contains all *pseudo marginals* that are locally consistent. They are pseudo simply because the real marginals not only satisfy these constraints but also many more marginalization constraints. On the other hand, for an exponential family, the marginal polytope contains real marginals if $\phi_a(x_a) = e_{x_a}, \forall a\in\mathcal{A}, x_a\in\mathcal{X}_a$. The relationship between these two polytopes is characterized by the following lemma.

Lemma 2.3.3 (Wainwright 2008, Proposition 4.1)**.** *For any PGM* $= \big(p_\theta(x), G = (V, E, \mathcal{A})\big)$ *such that* $p_\theta(x) \propto \exp\Big(\sum_{a\in\mathcal{A}} \langle \theta_a, e_{x_a}\rangle\Big)$*, the marginal polytope is a subset of the local consistency polytope, that is,* $\mathcal{M} \subseteq \mathcal{L}$*. If, in addition,* G *is acyclic,* $\mathcal{M} = \mathcal{L}$*.*

The Bethe approximation of $F(\theta)$ is based on the observation that $\mathcal{L}$ can be used to approximate $\mathcal{M}$, which is specified as follows.

$$F_{\text{Bethe}}(\theta) := \max_{\mu\in\mathcal{L}} \Big[\langle \theta, \mu\rangle - F^*_{\text{Bethe}}(\mu)\Big] \quad \text{with} \tag{2.42}$$

$$F^*_{\text{Bethe}}(\mu) := -\sum_{a\in\mathcal{A}} H(\mu_a) + \sum_{i\in V} d(i) H(\mu_i), \tag{2.43}$$

where $d(i)$ is the number of factor nodes connected to i in the factor graph (note that $V \subset \mathcal{A}$). Comparing to eq.(2.25), it involves two kinds of approximation:

- The marginal polytope $\mathcal{M}$ is replaced by the local consistency polytope $\mathcal{L}$.
- The Fenchel conjugate of $F(\theta)$ is approximated by $F^*_{\text{Bethe}}(\mu) = -H(q)$, where $q \in \mathcal{Q}_{\text{Bethe}} := \Big\{q \mid \exists \tau \in \mathcal{L} \text{ s.t. } q(x) = \frac{\prod_{a\in\mathcal{A}} \tau_a(x_a)}{\prod_{i\in V} \tau_i(x_i)^{d(i)}}, \forall x \in \mathcal{X}\Big\}$.

Unlike $F_{\text{mf}}(\theta)$ lower bounds $F(\theta)$, the Bethe approximation F_{Bethe} does not yield a bound, since $\mathcal{L}$ is an outer bound of $\mathcal{M}$, which only produces a variational relaxation to the original maximization problem.

The Bethe variational inference has many variants based on different entropy approximations:

- Based on the insight that the Bethe entropy approximation is induced from trees, the *Kikuchi* variational inference proposed by Yedidia et al. (2005) (also known as the *generalized belief propagation*) strengthens the approximation by exploiting the more general junction trees to derive the local consistencies as well as the entropy approximation.
- The Bethe objective turns out to be non-concave although the local consistency polytope is a convex set. Consequently, the Bethe variational inference may converge to a local optima depending on the initial conditions. Therefore, a line of research focused on concave entropy approximation has been proposed. For example, the tree-reweighted (TRW) entropy approximation proposed by Wainwright et al. (2005b) replaces the Bethe entropy approximation by a convex combination of tree based entropies. In general, for an entropy approximation of the form

$$H(\mu) := \sum_{a\in\mathcal{A}} c_a H(x_a), \tag{2.44}$$

  Heskes (2006) shows that $H(\mu)$ is concave if $(c_a)_{a\in\mathcal{A}}$ (also known as *counting numbers*) can be reparameterized by $(\alpha_a \geq 0)_{a\in\mathcal{A}}$ and $(\alpha_{a,i} \geq 0)_{a\in\mathcal{A}, i\in a}$ such

that

$$\forall i \in V: \ c_i = \alpha_i - \sum_{a \ni i} \alpha_{a,i}, \tag{2.45}$$

$$\forall a \in \mathcal{A}: \ c_a = \alpha_a + \sum_{i \in a} \alpha_{a,i}. \tag{2.46}$$

London et al. (2015) shows that if in addition α_a is strictly positive for all a belongs to $\mathcal{A}$, $H(\mu)$ is strongly concave. However, the Bethe entropy approximation often outperforms the TRW entropy approximation (Meshi et al. 2009), which means that convexifying the entropy approximation may have unexpected negative impacts. One solution proposed by Meshi et al. (2009) is that the counting numbers should be tuned to be as close as possible to that of the Bethe counterpart, while they satisfy the sufficient conditions given by Heskes (2006).

To conclude, the loopy belief propagation is indeed an iterative algorithm to solve the KKT system of the Bethe approximation of the log-partition function. It explains why LBP does not always converge in practice even with the damping trick. Moreover, by viewing it this way, many existing algorithms from (convex) optimization can be applied here with a global convergence guarantee. For example, the *Frank-Wolfe algorithm* has been applied to TRW and other convexified Bethe formulations (Belanger et al. 2013, Krishnan et al. 2015), where each subproblem amounts to solving a MAP inference (with an off-the-shelf MAP solver). Alternatively, one can employ a *stochastic subspace descent* (Richtárik and Takáč 2017) with each subspace corresponding to a spanning tree, where each subproblem amounts to solving a marginal inference for the subspace only (which can be solved exactly by the sum-product belief propagation for trees). In the extreme case when the subspace reduces to a single factor, the algorithm is known as the *block-coordinate proximal gradient method* (Shalev-Shwartz and Zhang 2016, Beck and Tetruashvili 2013). Note that, unlike LBP, all these algorithms have a convergence guarantee to a stationary point. For strongly concave entropy approximations, these algorithms can even be shown to be linearly convergent [2].

2.3.2 MAP inference

In general, the MAP inference is considered easier than marginal inference, since it can be seen from eq.(2.3) that the partition function is not involved in the MAP inference. For a PGM with exponential family density, we will see in the following theorem that the difference between the MAP inference and the marginal inference is quite small.

[2]Since the entropy term does not have a Lipschitz gradient on the boundary of $\mathcal{L}$, the proof of the linear convergence has to follow the stochastic mirror descent framework (Beck and Teboulle 2003, Bolte et al. 2018, Zhang and He 2018) using properties of Bregman divergence. This is relatively straightforward, but to the best of my knowledge there is no published work dedicated to this.

Theorem 2.3.4 (Wainwright 2008, Theorem 8.1). *For any PGM* $= \Big(p_\theta(x), G = (V, E, \mathcal{A})\Big)$ *such that* $p_\theta(x) \propto \exp\Big(\sum_{a\in\mathcal{A}}\langle\theta_a, e_{x_a}\rangle\Big)$. *The MAP inference has the following alternative formulations:*

$$\max_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}}\langle\theta_a, e_{x_a}\rangle = \max_{\mu\in\mathcal{M}}\langle\theta,\mu\rangle = \lim_{\beta\to+\infty}\frac{F(\beta\theta)}{\beta} \tag{2.47}$$

The first implication from eq.(2.47) is that the only difference between marginal inference and MAP inference for exponential family is the entropy term. Then, it is natural to hypothesize that anything works for marginal inference may also work for MAP inference. Indeed, many algorithms for MAP inference rely on bounding the marginal polytope by the local consistency polytope. The resulting formulation is called *linear programming relaxation* (LPR):

$$\max_{\mu\in\mathcal{L}}\Big[\langle\theta,\mu\rangle \equiv \sum_{a\in\mathcal{A}}\langle\theta_a,\mu_a(x_a)\rangle\Big], \tag{2.48}$$

which is lower bounded by $\max_{\mu\in\mathcal{M}}\langle\theta,\mu\rangle$. The following proposition describes what we should expect from LPR.

Proposition 2.3.5 (Wainwright 2008, Proposition 8.3). *The extrem points of* $\mathcal{L}$ *and* $\mathcal{M}$ *are related as follows:*

1. *All extreme points of* $\mathcal{M}$ *are of the form* $(e_{x_a})_{a\in\mathcal{A}}$ *given that* $x \in \mathcal{X}$, *and each one is also an extreme point of* $\mathcal{L}$.

2. *For any graph with cycles,* $\mathcal{L}$ *also includes additional extreme points with fractional elements that lie strictly outside* $\mathcal{M}$.

If we obtained a fractional solution μ by LPR, an additional *rounding* scheme is needed to obtain the MAP solution (e.g., the heuristic suggested by Ravikumar and Lafferty (2006)).

The LPR for MAP inference can be viewed as a counterpart of the Bethe approximation of $F(\theta)$ for marginal inference. However, the equivalence established in eq.(2.47) does not hold for the Bethe case. That is,

$$\max_{\mu\in\mathcal{L}}\langle\theta,\mu\rangle \neq \lim_{\beta\to+\infty}\frac{F_{\text{Bethe}}(\beta\theta)}{\beta}. \tag{2.49}$$

This is because the exchange of the "limit" and the "max" is possible only if F^*_{Bethe} is convex, which is obviously not the case for Bethe approximation. Consequently, the max-product LBP does not solve the LPR problem unless the graph is tree-structured (Wainwright 2008, Section 8.4.2). However, the TRW approximation does not have such a restriction, thus the TRW version of max-product LBP does find a fixed point of LPR's KKT system (Wainwright et al. 2005a).

Since LPR is indeed a linear programming over the local consistency polytope, several authors proposed alternative algorithms for either the primal or the dual problem. Due to the entanglement in $\mathcal{L}$ (in particular, the marginalization

constraints), only a few works optimize the LPR problem in the primal directly. Yanover et al. (2006) show that TRW max-product LBP is significantly faster than general-purpose linear programming solvers. Ravikumar et al. (2010) present a double-looped proximal point algorithm for LPR, where each inner loop involves a Bregman projection onto $\mathcal{L}$ being equivalent to performing a TRW sum-product LBP.

On the other hand, the dual of LPR, with the dual variable $\delta := (\delta_{ai})_{a\in\mathcal{A},i\in a}$, has a nice decomposition property in the primal variable μ:

$$\min_{\delta} \sum_{a\in\mathcal{A}} \max_{x_a\in\mathcal{X}_a} \Big(\langle \theta_a, e_{x_a} \rangle + \sum_{b\ni a} \langle \delta_{ba}, e_{x_a} \rangle - \sum_{i\in a} \langle \delta_{ai}, e_{x_i} \rangle \Big). \tag{2.50}$$

Several authors proposed efficient algorithms for solving the dual problem, which are identified by the same name called *dual decomposition*. Specifically, Komodakis et al. (2007) applied subgradient descent on δ achieving a $\mathcal{O}(\frac{1}{\epsilon^2})$ time complexity to obtain a ϵ-accurate solution; Werner (2007), Globerson and Jaakkola (2007a), Sontag and Jaakkola (2009) applied block-coordinate descent on δ by choosing different blocks. Please refer to Sontag et al. (2011) for a survey on dual decomposition. The problem with the block-coordinate scheme is that the objective function of LPR is non-smooth, which may get stuck at a non-stationary point even when the objective is convex. To overcome this limitation, Nesterov's smoothing (Nesterov 2005) has been applied to the dual of LPR (Johnson and Willsky 2008, Hazan and Shashua 2010, Savchynskyy et al. 2011). Note that this is equivalent to computing $\frac{F(\beta\theta)}{\beta}$ with $\beta > 0$. In this case, Meshi et al. (2012) has proved a $\mathcal{O}(\frac{1}{\beta\epsilon})$ time complexity for block-coordinate methods. By further tweaking the primal of LPR to be ρ-strongly concave, Meshi et al. (2015a) has shown that a linear convergence rate (i.e. a $\mathcal{O}\big(\frac{1}{\beta\rho}\log\frac{1}{\epsilon}\big)$ time complexity) can be achieved.

In addition, for a discrete PGM, the MAP inference can be cast as a combinatorial optimization. For example, for a binary MRF with *submodular*[3] potentials, the MAP inference can be solved exactly by *GraphCuts* (Boykov et al. 2001). Please refer to Kappes et al. (2013) for other discrete minimization techniques.

2.4 Learning

A PGM is specified by the pair $(p(x), G)$. The notion of learning in PGM is usually referred to as learning the parameters of $p(x)$ rather than learning the conditional independences represented by G. To avoid ambiguity, the former is called *parameter estimation*, and the latter is called *structure learning*. To narrow down the discussion, in this section, I consider only the frequentist setting of parameter estimation and assume that the structure is fixed. I will begin with a special case of parameter estimation for exponential family and then continue on the more general structured output learning.

[3]We say that $f : 2^S \to \mathbb{R}$ is submodular if for any $A \subset B \subset S$ and $x \in S$, we have $f(A\cup\{x\}) - f(A) \geq f(B\cup\{x\}) - f(B)$.

2.4.1 Maximum likelihood estimation of exponential family

Recall that a PGM with the exponential family form is naturally parameterized by its natural parameters. Consider the independent and identically distributed (iid) setting for parameter estimation, where the dataset $D := \{x_i\}_{i=1}^n$ consists of iid data points. The maximum likelihood estimation (MLE) is achieved by maximizing the log-likelihood with respect to the natural parameter, which is an instance of empirical risk minimization with the objective function

$$\hat{L}(\theta; D) := -\frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i) \tag{2.51}$$

$$= -\langle \theta, \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)\rangle + F(\theta). \tag{2.52}$$

Denote by $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$ the *empirical mean parameter*. If $\hat{\mu} \in \mathcal{M}^\circ$, there exists a $\theta_{\hat{\mu}}$ induced by $\hat{\mu}$ such that

$$\hat{\mu} = \mathbb{E}_{p_{\theta_{\hat{\mu}}}}[\phi(x)] \tag{2.53}$$

by Lemma 2.3.2. Thus, minimizing $\hat{L}(\theta; D)$ amounts to computing the entropy since

$$H(p_{\theta_{\hat{\mu}}}) = \max_{\theta\in\Theta} \langle \theta, \hat{\mu}\rangle - F(\theta) \tag{2.54}$$

$$= \langle \theta_{\hat{\mu}}, \hat{\mu}\rangle - F(\theta_{\hat{\mu}}). \tag{2.55}$$

In practice, we find $\theta_{\hat{\mu}}$ by gradient descent with the gradient

$$\nabla L(\theta; D) = -\hat{\mu} + \mathbb{E}_{p_\theta}[\phi(x)] = -\hat{\mu} + \mu. \tag{2.56}$$

These highlight the connection between MLE and maximum entropy principle. The optimality condition of MLE is exactly the moment matching condition.

2.4.2 Structured output learning

As we have mentioned in the beginning of this chapter, the structured prediction is a general learning problem for estimating the mapping between the high-dimensional input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. In many cases, we are only interested in the structure (i.e. conditional independences) within the output y rather than the structure within the input x, since the latter is often hard to understand from a human perspective. Thus, for structured prediction, we estimate the parameters of the conditional distribution $p(y|x)$ rather than the joint distribution $p(x, y)$. The associated PGM is known as *conditional random field* (CRF) (Lafferty et al. 2001). It should be noted that CRF is not the only model for structured prediction. Since our focus is graphical model, we assume the structured output space decomposes according to the graph given by the CRF, that is, $\mathcal{Y} := \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n$.

The potential functions of $p(y|x)$ are usually parameterized as a form of $\exp\Big(\theta_a(x, y_a)\Big)$, and the density of CRF is given by

$$p_\theta(y|x) = \frac{1}{Z(x,\theta)} \exp\Big(\sum_{a\in\mathcal{A}} \theta_a(x, y_a)\Big), \tag{2.57}$$

where $Z(x,\theta) = \int_{\mathcal{Y}} \exp\Big(\sum_{a\in\mathcal{A}} \theta_a(x, y_a)\Big)\nu(dy)$. To simplify the notations, I will denote $\theta(x, y) := \sum_{a\in\mathcal{A}} \theta_a(x, y_a)$. Note that $\theta_a(x, y_a)$ has no restriction in its form of parameterization. However, most of existing algorithms are designed for the following linear parameterization:

$$\theta_a(x, y_a) = \langle \theta_a, \phi_a(x, y_a)\rangle \tag{2.58}$$

with $\theta_a \in \mathbb{R}^{d_a}$ the actual parameter to be learned for each factor a and $\phi_a(x, y_a)$ the *feature representation* with respect to the input x and the label y_a. I will stick with this linear parameterization in the following, although this may not be necessary. To reduce the number of parameters, several papers, e.g., Pletscher et al. (2009), also considered parameter sharing between factors, which can be hard coded into the parameterization of $\theta(x, y)$.

The relationship between CRF and MRF is sometimes called the discriminative-generative pair (Ng and Jordan 2002), since one could instead define a MRF with density $p(x, y)$ using exactly the same set of potentials for $p(y|x)$. One example is *logistic regression* v.s *naive Bayes*. Both are special PGMs with *null graph*.

To learn a CRF from a dataset $D := \{(x_i, y_i)\}_{i=1}^n$, the maximum likelihood estimation is always a good choice. The objective of MLE reads as

$$\frac{1}{n} \log p_\theta(y_i|x_i) = \frac{1}{n} \sum_{i=1}^n \Big[\theta(x_i, y_i) - \log \sum_{y\in\mathcal{Y}} \exp\Big(\theta(x_i, y)\Big)\Big]. \tag{2.59}$$

It is characterized by a difference of two terms, where the second term is exactly the log-partition function– the main computational difficulty as discussed in the previous sections.

Alternatively, if we know how to evaluate the prediction $\hat{y}$ with respect to the ground truth y, namely, the task-specific loss $\ell(y, \hat{y})$ is provided, then we can directly minimize the expected task-specific loss empirically:

$$\min \frac{1}{n} \sum_{i=1}^n \ell(y_i, \hat{y}_i). \tag{2.60}$$

This formulation implicitly depends the prediction $\hat{y}$, which is usually chosen to be the result of the MAP inference

$$\hat{y} = \arg\max_{y\in\mathcal{Y}} \theta(x, y). \tag{2.61}$$

Thus, from a computational point of view, the direct loss minimization is more economical, as the log-partition function is not involved.

In case $\hat{y} \mapsto \ell(y, \hat{y})$ is non-differentiable and the $y \mapsto \arg\max_{y \in \mathcal{Y}} \theta(x, y)$ is discontinuous, the *max-margin Markov network* Taskar et al. (2003) can be used to remove the arg max from the learning objective. The idea was further extended by Tsochantaridis et al. (2005), Joachims et al. (2009) leading to following two convex upper bounds for $\ell(y, \hat{y})$:

$$\text{Margin rescaling}: \quad \ell(y, \hat{y}) \leq \ell(y, \hat{y}) + \theta(x, \hat{y}) - \theta(x, y) \tag{2.62}$$

$$\leq \max_z \Big[\ell(y, z) + \theta(x, z)\Big] - \theta(x, y) \tag{2.63}$$

$$\text{Slack rescaling}: \quad \ell(y, \hat{y}) \leq \ell(y, \hat{y}) \cdot \Big(1 + \theta(x, \hat{y}) - \theta(x, y)\Big) \tag{2.64}$$

$$\leq \max_z \Big[\ell(y, z) \cdot \Big(1 + \theta(x, z) - \theta(x, y)\Big)\Big] \tag{2.65}$$

The original max-margin training in (Taskar et al. 2003, Tsochantaridis et al. 2005) was in fact motivated differently from the above bounds. The key idea is as follows. If we can obtain zero prediction error on D, then the following inequalities hold:

$$\forall i \in \{1, \ldots, n\}, y \in \mathcal{Y} : \theta(x_i, y_i) - \theta(x_i, y) \geq \gamma, \tag{2.66}$$

where $\gamma := \min_i \Big[\theta(x_i, y_i) - \max_{y \in \mathcal{Y}} \theta(x_i, y)\Big] \geq 0$ defines the margin. Since both the direction and the magnitude of θ have an impact on the margin, we thus would like to fix the norm of θ and maximize the margin γ, which yields the following optimization problem

$$\max_{\gamma, \theta:\ \|\theta\|_2 = 1} \quad \text{s.t.} \quad \forall i \in \{1, \ldots, n\}, y \in \mathcal{Y} : \theta(x_i, y_i) - \theta(x_i, y) \geq \gamma. \tag{2.67}$$

The above problem can be equivalently expressed as a convex quadratic programming of the form

$$\min_{\theta} \frac{1}{2}\|\theta\|_2^2 \quad \text{s.t.} \quad \forall i \in \{1, \ldots, n\}, y \in \mathcal{Y} : \theta(x_i, y_i) - \theta(x_i, y) \geq 1. \tag{2.68}$$

In case the zero prediction error cannot be obtained, a slack variable ξ_i can be introduced for each data point:

$$\min_{\theta, \xi \geq 0} \frac{\lambda}{2}\|\theta\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \tag{2.69}$$

$$\text{s.t.} \quad \forall i \in \{1, \ldots, n\}, y \in \mathcal{Y} : \theta(x_i, y_i) - \theta(x_i, y) \geq 1 - \xi_i. \tag{2.70}$$

The margin rescaling and slack rescaling are then introduced naturally since we do not want to punish all errors equally. Specifically, the constraints become

$$\text{Margin rescaling}: \quad \theta(x_i, y_i) - \theta(x_i, y) \geq \ell(y_i, y) - \xi_i \tag{2.71}$$

$$\text{Slack rescaling}: \quad \theta(x_i, y_i) - \theta(x_i, y) \geq 1 - \frac{\xi_i}{\ell(y_i, y)}. \tag{2.72}$$

I will only focus on the *margin-rescaling* bound since it is the most commonly used bound in the literature. An in-depth discussion on the *slack-rescaling* bound can be found in the paper by Choi et al. (2016). Since the RHS of eq.(2.63) can be interpreted as a natural generalization of the hinge loss in *support vector machine* (SVM) (Hazan et al. 2010), the method proposed by Tsochantaridis et al. (2005) is usually called *structured output SVM* (SSVM). The objective of the margin-rescaled SSVM reads as

$$\frac{1}{n}\sum_{i=1}^{n}\max_{z}\Big[\ell(y_i,z)+\theta(x_i,z)\Big]-\theta(x_i,y_i)+\frac{\lambda}{2}\|\theta\|_2^2. \tag{2.73}$$

As a special case when $\ell \equiv 0$, we retrieve the *structured perceptron* (Collins 2002). One may notice that, in eq.(2.73), there exists n max operations in the form of

$$\max_{z}\Big[\ell(y_i,z)+\theta(x_i,z)\Big], \tag{2.74}$$

which is called *loss augmented MAP inference*. Ideally, we would like the task-specific loss to be decomposable according to the factors, then eq.(2.74) reduce to normal MAP inference, since the task-specific loss can be folded into the original potentials to form the new potentials. There are a few non-decomposable exceptions, such as the *intersection-over-union* (IoU) loss (Blaschko and Lampert 2008, Ranjbar et al. 2013) and the *area-under-the-curve* (AUC) loss (Rosenfeld et al. 2014), where efficient subroutines for the loss-augmented inference (Blaschko and Lampert 2008) or proper surrogates of the loss (Yu and Blaschko 2018) are available.

In general, the SSVM training is more costly than MLE training. The difference is that the MLE involves computing the softmax functions (i.e., "log-sum-exp") while the SSVM involves only computing the max functions. Recall that this difference also appears between the MAP inference and the marginal inference.

In the following, I will review several representative algorithms for SSVM, some of which have been implemented in open-source projects such as *SVMStruct* (Joachims et al. 2009) and *Shogun Machine Learning Toolbox* (Sonnenburg et al. 2017).

Cutting plane algorithm

Joachims et al. (2009) emphasize that eq.(2.69) with margin rescaling can be viewed as a quadratic programming (QP) with $n|\mathcal{Y}|$ constraints. This version of SSVM is referred to as the *n-slack margin-rescaled SSVM*. Since $|\mathcal{Y}|$ has an exponential number of elements, it is too cumbersome for standard QP solvers to handle so many constraints. Akin to support vectors in the plain SVM, we expect only a small fraction of the constraints to be active.

The cutting plane algorithm proposed by Kelley (1960) is designed exactly for addressing this case. It maintains a working set W_i for each data point i, which is initialized as an empty set. For each inner iteration, a constraint is added for each data point if the corresponding loss-augmented margin exceeds the current slack

variable by more than ϵ. Then, in the outer loop, the QP problem is solved with respect to the current constraints stored in the working set. The cutting plane algorithm for SSVM is summarized as follows.

Initialize a working set W_i for each data point; $\xi = 0$.
for all $t = 1, \dots, T$ **do**
  for all $i = 1, \dots, n$ **do**
    $\hat{y}_i = \arg\max_z \Big[\ell(y_i, z) + \theta(x_i, z)\Big]$.
    if $\ell(y_i, \hat{y}_i) + \theta(x_i, \hat{y}_i) - \theta(x_i, y_i) > \xi_i + \epsilon$ **then**
      $W_i = W_i \cup \{\hat{y}_i\}$.
      Update θ, ξ by solving (2.69) with constraints only in W_i for all i.
    end if
  end for
end for

If the loss-augmented MAP inference can be solved efficiently, the outer loop is guaranteed to converge quickly. The following theorem shows that only a polynomial number of constraints needed for any desired precision ϵ.

Theorem 2.4.1 (Tsochantaridis et al. 2005, Theorem 18)**.** *Assuming that* $\theta(x, y) = \langle \theta, \phi(x, y) \rangle$. *Let* $\bar{R} := \max_i \max_{y \in \mathcal{Y}} \|\phi(x_i, y_i) - \phi(x_i, y)\|, \bar{\ell} := \max_i \max_{y \in \mathcal{Y}} \ell(y_i, y)$, *and for any* $\epsilon > 0$, *the cutting plane algorithm for (2.69) terminates after adding at most*

$$\max\left\{\frac{2n\bar{\ell}}{\epsilon}, \frac{8\bar{\ell}\bar{R}^2}{\lambda\epsilon^2}\right\}$$

constraints to the working sets.

A variant called *one-slack SSVM* is also considered by Joachims et al. (2009), which involves solving n loss augmented inference per iteration, but in total it requires fewer iterations.

Stochastic subgradient descent

Note that the cutting plane algorithm still keeps a double-loop structure in the algorithm. The key computational gain in the cutting plane algorithm is the working set, which enables each inner loop to operate a subset of data points in contrast with a standard subgradient descent who needs to accumulate subgradients from all data points. To go beyond the idea of working set, one can consider a *stochastic subgradient descent* (SSD) algorithm. In fact, Shalev-Shwartz et al. (2011) sucessfully applied SSD to SVM showing a sublinear convergence speed. SSD has also been applied to SSVM as a baseline in Lacoste-Julien et al. (2013b).

The algorithm is actually much simpler than the cutting plane algorithm. It also works pretty well in practice. Consider a linear parameterization $\theta(x, y) = \langle \theta, \phi(x, y) \rangle$, where $\theta \in \mathbb{R}^d$. By Danskin's theorem for minimax problems, the subgradient of (2.73) is

$$\frac{1}{n}\sum_{i=1}^{n} \phi(x_i, \hat{y}_i) - \phi(x_i, y_i), \tag{2.75}$$

where $\hat{y}_i$ is the result of loss augmented inference on data point i. Then, the SSD for SSVM takes the form of

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta_t}{n}\Big(\phi(x_i, \hat{y}_i) - \phi(x_i, y_i)\Big) \tag{2.76}$$

with the step size η_t at iteration t satisfying

$$\sum_{t=1}^{\infty} \eta_t = \infty, \sum_{t=1}^{\infty} \eta_t^2 < \infty. \tag{2.77}$$

This is called *Robbins-Monro conditions* for convergence guarantee.

Block-coordinate Frank-Wolfe algorithm

It is a common practice in the SVM literature to optimize the dual problem rather than the primal one, since the dual variable at optimum is much sparser than the primal variable (thanks to the hinge loss). By using this property, one can make the classifier much more memory efficient, especially when the number of data points is significantly less than the number of parameters. In case $\theta(x, y)$ is convex in θ, the primal objective is also convex in θ, then maximizing the dual problem is equivalent to minimizing the primal problem.

To simplify the discussion, I assume that $\theta(x, y) = \langle \theta, \phi(x, y) \rangle$. In other words, $\theta(x, y)$ is linear in θ. With this assumption, the dual problem is derived in the following proposition.

Proposition 2.4.2. *Denote by* $\alpha \in \mathbb{R}^{n|\mathcal{Y}|} := (\alpha_i(y))_{i\in[n], y\in\mathcal{Y}}$ *the vector of dual variables, the dual problem of (2.73) is given by*

$$\max_{\alpha:\, \forall i,\, \alpha_i \in \Delta} D(\alpha) := -\frac{\lambda}{2}\|A\alpha\|^2 + \langle b, \alpha \rangle \tag{2.78}$$

where the matrix $A \in \mathbb{R}^{d\times n|\mathcal{Y}|} := \frac{1}{\lambda n}\Big(\phi(x_i, y_i) - \phi(x_i, y)\Big)_{i\in[n], y\in\mathcal{Y}}$ *and the vector* $b \in \mathbb{R}^{n|\mathcal{Y}|} := \frac{1}{n}\Big(\ell(y_i, y)\Big)_{i\in[n], y\in\mathcal{Y}}$.

Proof. The convex optimization of (2.73) can be rewritten as

$$\min_{\theta} \max_{(z_i)_{i\in[n]}\in\mathcal{Y}^n} \sum_{i\in[n]} \langle -\lambda A_i^\top \theta + b_i, e_{z_i} \rangle + \frac{\lambda}{2}\|\theta\|^2, \tag{2.79}$$

where $(A_i)_{i\in[n]} \equiv A$ and $(b_i)_{i\in[n]} \equiv b$. Since the strong duality holds, switching the order of the min and the max yields an equivalent problem. Note that minimizing over θ is now an unconstrained convex problem, which has a closed form solution: $\theta^*(z) = \sum_{i\in[n]} A_i e_{z_i}$. Substituting this back to the objective yields the dual problem in eq.(2.78). □

The Frank-Wolfe algorithm was originally designed by Frank and Wolfe (1956) for convex optimization problems of the form $\min_{x\in\mathcal{C}} f(x)$ over a convex set $\mathcal{C}$, where the convex function f is assumed to be continuously differentiable. At each

iteration, let $x^{(k)}$ be the current solution, the Frank-Wolfe algorithm updates along the *descent direction* $x^{(k)} - s^*$, where s^* is obtained by minimizing the *linearization* of f at $x^{(k)}$ over $\mathcal{C}$, namley, $s^* = \arg\min_{x \in \mathcal{C}} \langle s, \nabla f(x^{(k)}) \rangle$.

The following proposition shows that the Frank-Wolfe algorithm is a good choice for solving (2.78).

Proposition 2.4.3. *Let* $s^* = \arg\max_{s:\, \forall i,\, s_i \in \Delta} \langle s, \nabla D(\alpha) \rangle$. *Then,* $s^* \equiv (z_i^*)_{i \in [n]}$, *where* $z_i^* = \arg\max_{y \in \mathcal{Y}} \Big[\ell(y_i, y) + \langle A\alpha, \phi(x_i, y) \rangle \Big]$.

Proof. First, we identify that

$$\arg\max_{y \in \mathcal{Y}} \Big[\ell(y_i, y) + \langle A\alpha, \phi(x_i, y) \rangle \Big] = \arg\max_{y \in \mathcal{Y}} \langle -\lambda A_i^\top A\alpha + b_i, e_y \rangle \tag{2.80}$$

using the notations introduced in Proposition 2.4.2. Note that the gradient $\nabla D(\alpha) = -\lambda A^\top A\alpha + b$. Then, by stacking the RHS of the above equation for all $i \in [n]$, we obtain $\arg\max_{s:\, \forall i,\, s_i \in \Delta} \langle s, \nabla D(\alpha) \rangle$. □

Lacoste-Julien et al. (2013b) apply the block-coordinate Frank-Wolfe (BCFW) algorithm for solving (2.78), which consists of the following steps:

1: Initialize $\alpha_i^{(0)} \in \Delta$ for all i.
2: **for all** $k = 1, \ldots, K$ **do**
3: Pick i at random in $\{1, \ldots, n\}$.
4: Let $z_i^* = \arg\max_{y \in \mathcal{Y}} \Big[\ell(y_i, y) + \langle A\alpha^{(k)}, \phi(x_i, y) \rangle \Big]$.
5: Let $\gamma = \frac{2n}{k+2n}$.
6: Update $\alpha_i^{(k+1)} = (1 - \gamma)\alpha_i^{(k)} + \gamma e_{z_i^*}$.
7: **end for**

The above algorithm can be made more practical in several aspects. By using the representer theorem, one can compute a candidate value for the primal variable:

$$\theta = A\alpha = \sum_{i=1}^n A_i \alpha_i = \frac{1}{\lambda n} \sum_{i=1}^n \sum_{y \in \mathcal{Y}} \alpha_i(y) \Big(\phi(x_i, y_i) - \phi(x_i, y) \Big), \tag{2.81}$$

which is important since maintaining α may not always be possible even it is a sparse vector. Instead, we can keep n local copies of θ: $w_i = A_i \alpha_i$ for all $i \in [n]$, such that $w_i^{(0)} = 0$ and $\theta^{(t)} = \sum_{i=1}^n w_i^{(t)}$. `line 6` is then equivalent to the primal update

$$w_i^{(t+1)} = (1 - \gamma) w_i^{(t)} + \frac{\gamma}{\lambda n} \Big(\phi(x_i, y_i) - \phi(x_i, y) \Big). \tag{2.82}$$

Besides, since $D(\alpha)$ is quadratic, the value of γ can be chosen by line search in a closed form. See Lacoste-Julien et al. (2013b, Algorithm 4) for the version of BCFW with line search.

Dual decomposition for SSVM

Previous algorithms introduced in this section are designed for general structured output learning. In the case of PGMs, they did not attempt to leverage the particular structure encoded in the model. Instead, they have to invoke a subroutine for lossed augmented MAP inference at each iteration, which has shown to be NP-hard in Section 2.3. The question of whether approximate inference can be used during learning has been studied by Kulesza and Pereira (2008), Finley and Joachims (2008). A counterexample given by Kulesza and Pereira (2008) shows that not all approximations yield convergent SSVM learning. However, the approximate inference based on linear programming relaxation (LPR) has empirically and theoretically demonstrated good performance (Finley and Joachims 2008) as long as the same inference scheme is used in training and testing. Further theoretical analysis for LPR based loss augmented inference have been conducted by Kulesza and Pereira (2008), Meshi et al. (2019). In particular, Kulesza and Pereira (2008) gave a PAC-Bayes bound and Meshi et al. (2019) provided a theoretical explanation to the observation that linear programming relaxations are often tight in practice.

Recall that LPR can be efficiently solved by *dual decomposition* based algorithms (Komodakis et al. 2007, Werner 2007, Globerson and Jaakkola 2007a, Sontag and Jaakkola 2009). It turns out that the same technique can be used to blend the inference and learning in the sense that the inference part and the learning part are updated in an alternating fashion. This basically summarized the idea behind the work of Meshi et al. (2010), Hazan and Urtasun (2010), Komodakis (2011b). Specifically, for the PGM $= \big(p_\theta(y|x), G = (V, E, \mathcal{A})\big)$, the SSVM learning objective in eq.(2.73) is upper bounded by

$$\frac{1}{n}\sum_{i=1}^{n} \max_{\mu \in \mathcal{L}} \Big[\sum_{a \in \mathcal{A}} \langle -\lambda A_{ia}^\top \theta_a + b_{ia}, \mu_a \rangle \Big] + \frac{\lambda}{2}\|\theta\|_2^2, \tag{2.83}$$

where A_{ia} and b_{ia} are defined similarly as A and b introduced in Proposition 2.4.2:

$$A_{ia} \in \mathbb{R}^{d_a \times |\mathcal{Y}_a|} := \frac{1}{\lambda n}\Big(\phi_a(x_i, y_{ia}) - \phi_a(x_i, y_a)\Big)_{y_a \in \mathcal{Y}_a},$$
$$b_{ia} \in \mathbb{R}^{|\mathcal{Y}_a|} := \frac{1}{n}\Big(\ell_a(y_{ia}, y_a)\Big)_{y_a \in \mathcal{Y}_a}.$$

It should be noticed that the task-specific loss $\ell(y_i, y)$ is assumed to be decomposable according to G. This allow us to take advantage of the underlying structure of the problem.

To derive the dual problem, I associate each equality constraint in $\mathcal{L}$ a Lagrangian multiplier $\delta_{as}^{(i)}$, and let

$$\delta^{(i)} := (\delta_{as}^{(i)})_{a \in \mathcal{A}, s \in a}, \quad \eta^{(i)}(\theta) := (-\lambda A_{ia}^\top \theta_a + b_{ia})_{a \in \mathcal{A}}.$$

Note that a superscript is added to distinguish different data points. The dual objective of (2.83) reads as

$$\frac{1}{n}\sum_{i=1}^{n} \min_{\delta^{(i)}} \sum_{a \in \mathcal{A}} \max_{y_a \in \mathcal{Y}_a} \Big[\langle \eta^{(i)}(\theta), e_{y_a} \rangle + \sum_{c \ni a} \langle \delta_{ca}, e_{y_a} \rangle - \sum_{s \in a} \langle \delta_{as}, e_{y_s} \rangle \Big] + \frac{\lambda}{2}\|\theta\|_2^2, \tag{2.84}$$

which is naturally an upper bound of (2.83), thus it is also an upper bound of the original SSVM objective.

The above objective function only involves $\delta^{(i)}$ and θ. We can compute an approximate stochastic gradient with respect to θ by taking a few block-coordinate descent steps on $\delta^{(i)}$. Note that the coordinate descent updates on $\delta^{(i)}$ can be derived in closed form, but the detailed form depends on the block sampling strategy. A comparison on different strategies can be found in Sontag et al. (2011).

To sum up, the dual decomposition learning algorithm takes the following steps:

1: Initialize $\delta = 0, \theta = 0$.
2: **for all** $t = 1, \ldots, T$ **do**
3: Pick i at random in $\{1, \ldots, n\}$.
4: Perform a few (usually ≤ 10) block-coordinate updates on $\delta^{(i)}$.
5: Perform a stochastic subgradient descent on θ, where the stochastic gradient is estimated based on the updated $\delta^{(i)}$.
6: **end for**

A convergence proof for the above algorithm can be found in Meshi et al. (2010).

2.5 Conclusion

In this chapter, I covered the basic ingredients on the recent development of approximate inference and learning for probabilistic graphical models. The main difficulty comes from the computation over the exponentially large output space. In inference and learning, this difficulty is reflected in computing the value or the gradient of the log-partition function. More efficient algorithms will allow the current formulations to scale up. Besides, most of existing methods focused on convex PGMs. There are a few exceptions, such as the *structured prediction energy networks* (Belanger and McCallum 2016). It remains an open question whether the current developed techniques are applicable to non-convex PGMs.

Chapter

3 SDCA-Powered Inexact Dual Augmented Lagrangian Method for Fast CRF Learning

Abstract

We propose an efficient dual *augmented Lagrangian formulation to learn conditional random fields (CRF). Our algorithm, which can be interpreted as an inexact gradient descent algorithm on the multiplier, does not require to perform global inference iteratively, and requires only a fixed number of stochastic clique-wise updates at each epoch to obtain a sufficiently good estimate of the gradient w.r.t. the Lagrange multipliers. We prove that the proposed algorithm enjoys global linear convergence for both the primal and the dual objectives. Our experiments show that the proposed algorithm outperforms state-of-the-art baselines in terms of speed of convergence.*

3.1 Introduction

Learning in graphical models has historically relied on the computation of the (sub)gradient of the log-likelihood w.r.t. to the canonical parameters, which requires to solve a MAP or probabilistic inference problem at each iteration. This approach is slow given that the inference problem is itself computationally expensive. The difficulty of inference and learning in graphical models is related to the fact that the log-partition function is in general intractable.

Recent progress on the optimization problems whose objective is a large finite sum of convex terms has shown that they could be optimized very efficiently by stochastic algorithms that sample one term at a time (Roux et al. 2012, Shalev-Shwartz and Zhang 2016, Defazio et al. 2014b). It turns out that the dual objective of the maximum likelihood estimation of CRF (a.k.a. the maximum entropy principle) decomposes additively over all cliques if a decomposable entropy surrogate is used. Even though this dual formulation has a potential to take advantage of stochastic algorithms, and can be optimized without resorting to solve a global inference on the entire graph per iteration, all dual parameters (i.e. *mean parameters*) are coupled by the *marginal polytope* constraints, which are in general intractable. Even its most commonly used relaxation, namely the *local consistency polytope*, is itself in practice difficult to optimize over. Recently, Meshi et al. (2015ba) proposed to replace the marginalization constraints, which are part of the local consistency polytope, by quadratic penalty terms. The relaxed problem has then only separable constraints over the cliques that makes it possible to use efficient block coordinate optimization schemes.

Following these ideas, we consider a dual formulation for CRF learning in which the marginalization constraints are replaced by an augmented Lagrangian term, and the intractable Shannon entropy is replaced by a quadratic surrogate so that stochastic dual coordinate ascent (SDCA) can be used to optimize over the mean parameters, with similar guarantees as in Shalev-Shwartz and Zhang (2016). We finally show that by periodically updating the Lagrangian multipliers as we are optimizing the relaxed dual, we can gradually enforce the marginalization constraints, while retaining global linear convergence. In terms of the primal problem associated with the Lagrange multipliers, our algorithm is an inexact gradient descent algorithm using stochastic approximation of the multiplier gradients.

Our paper is organized as follows. We review CRF learning in Section 3.3. A dual augmented Lagrangian formulation is presented in Section 3.4. The proposed algorithm is presented in Section 3.5, followed by its convergence analysis in Section 3.6. Finally, we present experiments on three applications in Section 3.7 (Most notations used in the paper can be found in Appendix 3.F).

3.2 Related Work

Due to the independent interest of inference problem in discrete graphical models, in particular in computer vision, a significant amount of work has been devoted to

develop efficient approximate inference algorithms (Komodakis et al. 2007, Sontag et al. 2008, Savchynskyy et al. 2011, Martins et al. 2015). However, the learning problem is not necessarily easier (can even fail to converge) with an approximate inference approach as the subroutine (Kulesza and Pereira 2007).

There is a large body of research on efficient algorithms for structured learning. For the max-margin formulation, the fastest algorithms to date rely on block coordinate Frank-Wolfe updates (Lacoste-Julien et al. 2013a, Meshi et al. 2015b, Tang et al. 2016). Using dual decomposition in the inner inference problem, Meshi et al. (2010), Hazan and Urtasun (2010), Komodakis (2011a) proposed to solve the classical saddle-point formulation for structured learning problem with algorithms that alternate between message passing and model parameter updates. Going further Meshi et al. (2015b), Yen et al. (2016) work on a purely dual formulation to enable clique-wise updates. For maximum likelihood learning, exponentiated gradient and its block variants can be applied (Collins et al. 2008). Other recent work have relied on incremental algorithms (Schmidt et al. 2015) and the fact that the Gauss-Southwell rule can be applied efficiently for coordinate descent in some forms of graphical models (Nutini et al. 2015).

The BCMM algorithm of Hong et al. (2014) which uses stochastic block coordinate updates inside ADMM inspired our approach. But our algorithm performs multiple passes over all blocks before updating the multiplier; and we prove stronger convergence rates.

We list related structured learning methods with their main characteristics in Table 1 in Appendix 3.B.5.

Yen et al. (2016) is the most similar work to ours: the proposed algorithm constructs greedily an (initially sparse) working set of cliques, which is incremented at each epoch, while we perform stochastic updates on all cliques and possibly several passes over the data between each update of all Lagrange multipliers. Also, our work is leveraging the connection with SDCA, and we prove both linear convergence in the primal and the dual whereas Yen et al. (2016) prove only linear convergence in the dual.

It is noteworthy that the work of Gidel et al. (2018), which was published at the same time as this work at AISTATS 2018, studies a related algorithmic framework based on the Frank-Wolfe algorithm for the same type of convex optimization problems. The main difference is that they take Frank-Wolfe updates for the inner loop with a specific analysis to Frank-Wolfe. Besides, they consider applications beyond the scope of probabilistic graphical models.

3.3 CRF Learning

A discrete *conditional random field* (CRF) is a family of conditional distributions over a vector of discrete random variables $Y := (Y_1, \dots, Y_m)$ given the observation X. The form of the CRF is assumed to be a product of *local functions* (a.k.a. *factors* or *clique functions*) that each depends on only a small number of random variables (i.e. a *clique*). If there exists multiple cliques that share the same local function, then we group cliques by *clique types*. Specifically, let $w_\tau \in \mathbb{R}^{d_\tau}$ be the

parameter vector associated with the clique type $\tau \in \mathcal{T}$, where $\mathcal{T}$ is the set of clique types. Let $\mathcal{C}$ denote the set of all cliques, and $\mathcal{C}_\tau$ the set of cliques of type $\tau \in \mathcal{T}$. Note that each clique c has a unique clique type, which we denote by τ_c. With these notations the density function of the CRF can be written as

$$p(y|x;w) := \frac{1}{Z(x,w)} \prod_{\tau\in\mathcal{T}} \prod_{c\in\mathcal{C}_\tau} \exp\Big(\langle w_\tau, \phi_c(x,y_c)\rangle\Big),$$

where $w = (w_\tau)_{\tau\in\mathcal{T}}$; we denoted $Z(x,w)$ the *partition function* and $\phi_c(x,y_c) \in \mathbb{R}^{d_{\tau_c}}$ the *feature map* for clique c. Since all random variables are discrete, we use a one-hot vector $y_i \in \mathcal{Y}_i := \{u \in \{0,1\}^{k_i} \colon \|u\|_1 = 1\}$ to represent the value of Y_s. Here k_i is the cardinality of $\mathcal{Y}_i$. For a clique c, the value for the corresponding random variables is $y_c = \otimes_{i\in c}\, y_i \in \mathcal{Y}_c := \bigotimes_{i\in c} \mathcal{Y}_i$, where $\otimes$ (resp. $\bigotimes$) denotes the tensor product of vectors (resp. of spaces). Similarly, $y \in \mathcal{Y}$ is of the form $y = \otimes_{i\in\mathcal{V}}\, y_i$. W.l.o.g., we consider in the paper only cliques of size at most 2, that is $\mathcal{C} = \mathcal{V} \cup \mathbb{E}$, with $\mathcal{V}$ and $\mathbb{E}$ respectively the set of nodes and of edges of the graph; the framework generalizes easily to higher-order cliques. Notations used in the paper are listed in Appendix 3.F.

3.3.1 CRF as exponential family

Given a sample $(x^{(n)}, y^{(n)})$, for each clique c, let $\eta_c^{(n)}(w) := [\langle w_{\tau_c}, \phi_c(x^{(n)}, y_c)\rangle \colon y_c \in \mathcal{Y}_c]$; then a *natural parameter* for the exponential family form of the conditional distribution $p(y \mid x^{(n)})$ is $\eta^{(n)}(w) := [\eta_c^{(n)}(w) \colon c \in \mathcal{C}]$. The associated *sufficient statistics* is $T(y) := [y_c \colon c \in \mathcal{C}]$, and $\langle \eta^{(n)}(w), T(y)\rangle = \sum_c \langle \eta_c^{(n)}(w), y_c\rangle$. With these notations, $p(y \mid x^{(n)})$ has the exponential family form:

$$p\Big(y \mid \eta^{(n)}(w)\Big) = \exp\Big[\langle \eta^{(n)}(w), T(y)\rangle - F\Big(\eta^{(n)}(w)\Big)\Big],$$

where $F(\eta) := \log \sum_y \exp\langle \eta, T(y)\rangle = \log Z(x^{(n)}, w)$ is the log-partition function.

Given i.i.d. samples $\{(x^{(n)}, y^{(n)})\}_{1\le n\le N}$, the maximum likelihood estimator for w is computed by the maximizing $\sum_n \log p(y^{(n)} \mid x^{(n)}; w)$. Using the exponential family representation, we can rewrite this problem in two equivalent forms:

$$\max_w \sum_{n=1}^{N} \Big[\langle \eta^{(n)}(w), T(y^{(n)})\rangle - F\Big(\eta^{(n)}(w)\Big)\Big],$$

and $\min_w \sum_{n=1}^N F\Big(\theta^{(n)}(w)\Big)$ with $\theta^{(n)}(w)$ another natural parameter obtained via the affine transformation $\theta^{(n)}(w) = \eta^{(n)}(w) - \langle \eta^{(n)}(w), T(y^{(n)})\rangle \mathbf{1}$. Alternatively, by defining $\Psi^{(n)}$ as a sparse block matrix with $|\mathcal{T}| \times |\mathcal{C}|$ blocks, whose (τ_c, c)-th block is the matrix $\Psi_c^{(n)} \in \mathbb{R}^{d_{\tau_c} \times k_c}$ with

$$\Psi_c^{(n)} = [\phi_c(x^{(n)}, y_c) - \phi_c(x^{(n)}, y_c^{(n)}) \colon y_c \in \mathcal{Y}_c],$$

we have $\theta_c^{(n)}(w) = \Psi_c^{(n)\intercal} w_{\tau_c}$ and $\theta^{(n)}(w) = \Psi^{(n)\intercal} w$.

W.l.o.g., we assume $N = 1$ and drop the superscript (n) from now on, since one may view N graphs as a single large graph with several connected components.

Regularized maximum likelihood estimation with a regularization constant $\lambda > 0$ is thus formulated as

$$\min_w F\Big(\theta(w)\Big) + \frac{\lambda}{2}\|w\|_2^2. \tag{3.1}$$

In order to extend this formulation to cover as well max-margin learning (i.e., structured SVMs), we consider the loss-augmented CRF learning introduced by Pletscher et al. (2010) and Hazan and Urtasun (2010), which leads to a slightly generalized formulation:

$$\min_w \gamma F\Big(\tfrac{1}{\gamma}\theta_\ell(w)\Big) + \frac{\lambda}{2}\|w\|_2^2, \tag{3.2}$$

where $\theta_\ell(w) := \theta(w) + \ell$ is then the natural parameter, with $\ell = \Big[[\ell_c(y_c^\star, y_c) \colon y_c \in \mathcal{Y}_c] \colon c \in \mathcal{C}\Big]$ the user-defined loss and $\gamma \in (0, +\infty)$ the temperature hyperparameter. For a derivation for the loss-augmented CRF see Appendix 3.A.

It is well known that the cost of gradient descent to optimize either eq.(3.1) or eq.(3.2) (for $\gamma > 0$) is prohibitive since $\nabla_{w_\tau} F(\theta(w)) = \sum_{c \in \mathcal{C}_\tau} \Psi_c \, \mathbb{E}_\theta[Y_c]$ involves an expectation over the exponentially large space $\mathcal{Y}$. To exploit the underlying structure of the function F it is useful to work on the dual problem. Indeed, since F is convex, it has a variational representation based on conjugate duality:

$$F(\theta) = \max_\mu \langle \mu, \theta \rangle - F^*(\mu),$$

where F^* is the Fenchel conjugate of F, and the dual variable μ called the *mean parameter* is defined by $\mu = (\mu_c)_{c \in \mathcal{C}}$ with $\mu_c = \mathbb{E}_\theta[Y_c]$. The set of valid mean parameters form the so called *marginal polytope* $\mathcal{M}$, which is defined as the convex hull of $\{T(y) : y \in \mathcal{Y}\}$. Moreover, if let $H_{\text{Shannon}}(\mu)$ denote the Shannon entropy of a CRF with mean parameter μ, it is a classical result (Wainwright 2008, Thm 3.4) that

$$F^*(\mu) = -H_{\text{Shannon}}(\mu) + \iota_{\mathcal{M}}(\mu),$$

where $\iota_{\mathcal{M}}(\mu)$ equal to 0 if $\mu \in \mathcal{M}$ and $+\infty$ otherwise.

3.4 Relaxed Formulations

In this section, we derive general relaxed dual, primal and corresponding saddle-point formulations for the CRF learning problem: first, we use the classical local polytope relaxation (Sec. 3.4.1). Second, we further relax the marginalization constraints via an augmented Lagrangian (Sec. 3.4.2). Third, we propose a surrogate for the entropy, which is decomposable, and retains good properties even when the aforementioned constraints are relaxed (Sec. 3.4.3). The resulting formulation is convex and is amenable to fast optimization algorithm that are presented in Section 3.5.

3.4.1 Classical local polytope relaxation

Both $\mathcal{M}$ and $H_{\text{Shannon}}(\mu)$ are in general intractable due to the exponentially large structured-output space $\mathcal{Y}$ and they are typically replaced by decomposable surrogates.

It is common to relax $\mathcal{M}$ to the *local consistency polytope* (Wainwright 2008)

$$\mathcal{L} := \Big\{\mu \in \mathcal{I} : \sum_{y_j \in \mathcal{Y}_j} \mu_{ij}(y_i, y_j) = \mu_i(y_i), \forall \{i,j\} \in \mathbb{E}, \forall y_i\Big\},$$

where $\mathcal{I}$ denotes the Cartesian product of *simplex constraints* on each clique. Note that $\mathcal{L} \supseteq \mathcal{M}$, since any set of true marginals must satisfy the simplex constraints and the *marginalization constraints*, but not vice versa. Equivalently, if we define $A_i = I_{k_i} \otimes \mathbf{1}_{k_i}^\intercal$, the equality constraints can be written in a matrix form as $\mu_i - A_i \mu_{ij} = 0$ for all $\{i,j\} \in \mathbb{E}$. Combining all equations, we have $A\mu = 0$, where A is a $|\mathbb{E}| \times |\mathcal{C}|$ block matrix (see Appendix 3.F). So, we have equivalently $\mathcal{L} = \mathcal{I} \cap \{\mu : A\mu = 0\}$.

Since H_{Shannon} is also intractable for graphs with large tree-width, we will use an approximation H_{Approx} which will be constructed so as to be defined and concave on the whole set $\mathcal{I}$. We propose several entropy approximations suited to our needs in Section 3.4.3.

Definition 3.4.1. *Let $F_{\mathcal{I}}$ and $F_{\mathcal{L}}$ be the counterparts of F obtained by relaxing $\mathcal{M}$ to $\mathcal{I}$ and $\mathcal{L}$ respectively, which, in other words, are the Fenchel conjugates of $F_{\mathcal{I}}^*$ and $F_{\mathcal{L}}^*$ when these are defined by H_{Approx}:*

$$F_{\mathcal{I}}(\theta_\ell) := \max_{\mu} \langle \mu, \theta_\ell \rangle - F_{\mathcal{I}}^*(\mu),$$
$$F_{\mathcal{L}}(\theta_\ell) := \max_{\mu} \langle \mu, \theta_\ell \rangle - F_{\mathcal{L}}^*(\mu),$$

where $F_{\mathcal{I}}^*(\mu) := -H_{\text{Approx}}(\mu) + \iota_{\mathcal{I}}(\mu)$ and $F_{\mathcal{L}}^*(\mu) := F_{\mathcal{I}}^*(\mu) + \iota_{\{A\mu=0\}}$.

Replacing F with $F_{\mathcal{L}}$ in eq.(3.2) yields the relaxed primal

$$P(w) := \gamma F_{\mathcal{L}}\Big(\tfrac{1}{\gamma}\theta_\ell(w)\Big) + \frac{\lambda}{2}\|w\|_2^2. \tag{3.3}$$

The corresponding dual objective function is given by

$$D(\mu) := \langle \mu, \ell \rangle - \gamma F_{\mathcal{L}}^*(\mu) - \frac{1}{2\lambda}\|\Psi\mu\|_2^2. \tag{3.4}$$

See Appendix 3.B.1 for a derivation.

3.4.2 A dual augmented Lagrangian

It is difficult to optimize $D(\mu)$, since the optimization requires some form of projection onto $\mathcal{L}$, which can be shown to be equivalent to perform graph-wise marginal inference (Collins et al. 2008). The difficulty is due to the coupling equality constraint $A\mu = 0$. Meshi et al. (2015b) proposed to relax $\iota_{\{A\mu=0\}}$ by a quadratic

term $\frac{1}{2\rho}\|A\mu\|_2^2$, which corresponds to employ the *penalty method* (Bertsekas 1982). They argue that it is not crucial to enforce exact $A\mu = 0$ in learning, since the relaxed problem works well in practice and enables an efficient optimization with only clique-wise updates. However, the penalty method is known to have issues associated with the choice of ρ: unless we use a carefully designed scheduling to update ρ, for a reasonably small ρ, the algorithm will be slow; on the other hand, using a large fixed value of ρ degrades the problem to independent logistic regression problems, and, thereby, leads to suboptimal solutions.

Instead, we propose to solve problem eq.(3.4) as a saddle problem of the form $\max_\mu \min_\xi D_\rho(\mu, \xi)$ where D_ρ is the augmented Lagrangian

$$\begin{aligned} D_\rho(\mu, \xi) := &\Big[\langle \ell, \mu\rangle - \gamma F_{\mathcal{I}}^*(\mu) + \langle \xi, A\mu\rangle\Big] \\ &- \Big[\frac{1}{2\rho}\|A\mu\|_2^2 + \frac{1}{2\lambda}\|\Psi\mu\|_2^2\Big], \end{aligned} \tag{3.5}$$

with ξ is the Lagrangian multiplier and $\rho > 0$.

Using duality again, we can derive an associated relaxed primal objective

$$\tilde{P}_\rho(w, \delta, \xi) := \gamma F_{\mathcal{I}}\Big(\frac{\theta_\ell(w) + A^\intercal\delta}{\gamma}\Big) + \frac{\lambda}{2}\|w\|_2^2 + \frac{\rho}{2}\|\delta - \xi\|_2^2,$$

so that $\min_{(w,\delta)} \tilde{P}_\rho(w, \delta, \xi)$ is a primal problem associated with the dual problem $\max_\mu D_\rho(\mu, \xi)$.

Strong duality between these two problems yields a representer theorem

$$w^\star = -\frac{1}{\lambda}\Psi\mu^\star, \quad \delta^\star = \xi^\star - \frac{1}{\rho}A\mu^\star \tag{3.6}$$

which provides a duality gap

$$\mathrm{gap}(w, \delta, \mu, \xi) := \tilde{P}_\rho(w, \delta, \xi) - D_\rho(\mu, \xi)$$

for the convergence of the maximization of $D_\rho(\mu, \xi)$ with respect to μ. Moreover, it is easy to check that $\min_{\xi,\delta} \tilde{P}_\rho(w, \delta, \xi) = P(w)$ because $\min_\delta F_{\mathcal{I}}(\theta(w) + A^\intercal\delta) = F_{\mathcal{L}}(\theta(w))$ for any w (see Appendix 3.B.2). This shows that $w^\star$ defined in eq.(3.6) is also an optimum of the original primal problem $\min_w P(w)$. As a consequence, if a sequence μ^t converges to $\mu^\star$ then the corresponding $w^t = -\frac{1}{\lambda}\Psi\mu^t$ converges to a solution of eq.(3.2). For more details, see Appendix 3.B.

3.4.3 Gini entropy surrogate

We seek a concave entropy surrogate H_{Approx} that decomposes additively on the cliques. Since the constraint $A\mu = 0$ is relaxed, we need a surrogate well defined on the whole set $\mathcal{I}$. The Bethe entropy (Yedidia et al. 2005) is generally non-concave. Its concave counterparts, such as the tree-reweighted entropy (Wainwright et al. 2005b) or the region-based entropy (Yedidia et al. 2005, London et al. 2015), are only concave on the local consistency polytope, but non-concave on $\mathcal{I}$.

Moreover, a generic difficulty with these entropies is that they do not have Lipschitz gradients, which prevents the direct application of proximal methods with usual quadratic proximity terms. We thus propose a coarse but convenient entropy surrogate of the form:

$$H_{\text{Approx}}(\mu) = \sum_{c \in \mathcal{C}} h_c(\mu_c) \quad \text{with} \quad h_c(\mu_c) := (1 - \|\mu_c\|_2^2).$$

Another surrogate with the same separable form is the second-order Taylor expansion of the *oriented tree-reweighted entropy* (OTRW, Globerson and Jaakkola 2007b) around the uniform distribution. This surrogate is also concave on $\mathcal{I}$ (although not strongly concave) and smooth. Preliminary experiments however did not show that using this more sophisticated entropy improved the results. See Appendix 3.C for more details.

3.5 Algorithm

Given the form of the entropy surrogate proposed, D_ρ decomposes as a sum of convex separable terms over the block associated to cliques plus a smooth term:

$$\begin{aligned} D_\rho(\mu, \xi) &= -\sum_{c \in C} f_c^*(\mu_c) - r(\mu) \qquad \text{with} \\ f_c^*(\mu_c) &:= -\gamma h_c(\mu_c) + \iota_{\triangle_c}(\mu_c) \\ r(\mu) &:= -\langle A^\intercal \xi + \ell, \mu \rangle + \frac{1}{2\lambda}\|\Psi\mu\|^2 + \frac{1}{2\rho}\|A\mu\|^2, \end{aligned} \tag{3.7}$$

where $\triangle_c := \{\mu_c \in \mathbb{R}_+^{d_c} \mid \mu_c^\intercal \mathbf{1} = 1\}$ is the canonical simplex. It can thus be maximized efficiently by a block-coordinate proximal scheme, such as the proximal stochastic dual coordinate descent (SDCA, Shalev-Shwartz and Zhang 2016), which has linear convergence guarantees both in the primal and the dual.

To solve $\min_\xi \max_\mu D_\rho(\mu, \xi)$ we thus propose an algorithm similar to the block coordinate method of multipliers (BCMM) of Hong et al. (2014): perform dual stochastic block coordinate ascent (SDCA) on the variables μ_c to partially maximize $D_\rho(\mu, \xi)$ in μ and regularly take a gradient descent step in ξ. Our algorithm, is an inexact dual augmented Lagrangian (IDAL) method, in the sense that it is an inexact gradient descent algorithm on the function $\xi \mapsto d(\xi) := \max_\mu D_\rho(\mu, \xi)$. To be precise, if at epoch t, ξ takes the value ξ^t and $\hat{\mu}^{t-1}$ is the value of μ from the previous epoch, Algorithm 2 takes T_{in} stochastic block-coordinate proximal gradient steps on μ to obtain $\hat{\mu}^t$. Denoting L_c the Lispchitz constant of r w.r.t. μ_c, μ_c is then updated by a partial gradient step, and an application of the proximal operator of $\frac{1}{L_c} f_c^*$. Then, by Danskin's theorem[1], applied to equation eq.(3.5), we have that $A\hat{\mu}^t$ is an approximate gradient of $d(\xi^t)$, and so, Algorithm 1 updates ξ with $\xi^{t+1} = \xi^t - \frac{1}{L_d} A\hat{\mu}^t$, where L_d is the Lispchitz constant of $d(\xi)$. As for the stopping criteria,

[1] Bertsekas (See e.g. 1999, Prop. B.25)

Algorithm 1 IDAL scheme

Input: $T_{\text{in}}, T_{\text{ex}}, \epsilon$
Initialize: $\hat{\mu}_c^0 = \frac{1}{k_c}\mathbf{1}$ for all $c \in \mathcal{C}$ and $\xi^1 = 0$
for $t = 1, \ldots, T_{\text{ex}}$ **do**
    $\hat{\mu}^t = \mathcal{A}(\hat{\mu}^{t-1}, T_{\text{in}}, t)$
    Stop if $G_t \leq \epsilon$ and $\|A\hat{\mu}^t\|^2 \leq \epsilon$
    $\xi^{t+1} = \xi^t - \frac{1}{L_d} A\hat{\mu}^t$
end for
Output: $\hat{\mu}^{T_{\text{ex}}}, \xi^{T_{\text{ex}}}$

Algorithm 2 SDCA version of $\mathcal{A}(\mu, T_{\text{in}}, t)$

$\mu^{t,0} = \mu$
for $s = 1, \ldots, T_{\text{in}}$ **do**
    Draw a clique c uniformly at random
    $\mu_c^{t,s} = \text{Prox}_{\frac{1}{L_c} f_c^*}\left(\mu_c^{t,s-1} - \frac{1}{L_c}\nabla_{\mu_c} r(\mu^{t,s-1})\right)$
    $\mu_{-c}^{t,s} = \mu_{-c}^{t,s-1}$
end for
Output: $\mu^{t,T_{\text{in}}}$

we use $G_t := \text{gap}(w(\hat{\mu}^t), \delta(\hat{\mu}^t, \xi^t), \hat{\mu}^t, \xi^t) \leq \epsilon$ and $\|A\hat{\mu}^t\|^2 \leq \epsilon$, where $w(\hat{\mu}^t), \delta(\hat{\mu}^t, \xi^t)$ are defined via the representer theorem eq.(3.6) (see Appendix 3.B.4).

3.6 Convergence Analysis

In this section, we study the convergence rate of our algorithm. First, we show that if we use an iterative and linearly convergent algorithm $\mathcal{A}$ to approximately solve $\min_\mu D_\rho(\mu, \xi)$, and if we use warm starts, that is, following the notations of the previous section, we use $\hat{\mu}^{t-1}$ as the initial value to solve $\min_\mu D_\rho(\mu, \xi^t)$, then running $\mathcal{A}$ for a fixed number of iterations is sufficient to guarantee global linear convergence in the primal and in the dual. We show that SDCA or simple block-coordinate proximal gradient descent are applicable as the algorithm $\mathcal{A}$.

3.6.1 Conditions for global linear convergence

To study the convergence, we consider:

- $\bar{\mu}^t := \mu^\star(\xi^t) = \text{argmax}_\mu D_\rho(\mu, \xi^t)$.
- $\mu^{t,s}$, the value of μ after s inner steps at epoch t.
- $\hat{\mu}^t := \mu^{t,T_{\text{in}}}$ the value of μ at the end of epoch t.
- D_ρ-suboptimality: $\Delta_t^s := D_\rho(\bar{\mu}^t, \xi^t) - D_\rho(\mu^{t,s}, \xi^t)$, with at the end of each epoch $\hat{\Delta}_t := \Delta_t^{T_{\text{in}}} = \Delta_{t+1}^0$.

- d-suboptimality: $\Gamma_t := d(\xi^t) - d(\xi^\star)$.

Theorem 3.6.1 (Linear convergence of the outer iteration)**.** *Let $\mathcal{A}$ be an algorithm that approximately solves $\max_\mu D_\rho(\mu, \xi^t)$ in the sense that*

$$\exists \beta \in (0,1), \qquad \mathbb{E}[\hat{\Delta}_t] \leq \beta\, \mathbb{E}[\Delta_t^0].$$

Then, $\exists \kappa \in (0,1)$ characterizing $d(\xi)$ and $C > 0$, such that, if $\lambda_{\max}(\beta)$ is the largest eigenvalue of the matrix

$$M(\beta) = \begin{bmatrix} 6\beta & 3\beta \\ 1 & 1-\kappa \end{bmatrix},$$

then after T_{ex} iterations of Algorithm 1 we have

$$\left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_{T_{\text{ex}}}] \\ \mathbb{E}[\Gamma_{T_{\text{ex}}}] \end{matrix} \right\| \leq C\, \lambda_{\max}(\beta)^{T_{\text{ex}}} \left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_0] \\ \mathbb{E}[\Gamma_0] \end{matrix} \right\|.$$

The proof is deferred to Appendix 3.D as well as the proofs of the following corollaries.

The constant κ in the theorem is of the form $\kappa = \frac{\tau}{L_d}$ with L_d the Lipschitz constant of $d(\xi)$ and τ a restricted strong convexity constant for $d(\xi)$ obtained by Hong and Luo (2017) (see Lemma 3.D.3 in Appendix 3.D.2).

Corollary 3.6.2. *If $\mathcal{A}$ is a linearly convergent algorithm with rate π and if it runs for T_{in} iterations, such that, for some β: $\lambda_{\max}(\beta) < 1$, we have $(1-\pi)^{T_{\text{in}}} \leq \beta$, then $\mathbb{E}[\hat{\Delta}_t]$ and $\mathbb{E}[\Gamma_t]$ converge linearly to 0.*

Note that linear convergence of the expectations implies that Δ_t and Γ_t converge linearly to 0 almost surely, as a classical consequence of Markov's inequality and the Borel-Cantelli lemma. We will show in the next section that when $\mathcal{A}$ is SDCA it is linearly convergent.

Note that the convergence of the gaps Δ_t and Γ_t imply the linear convergence for the augmented Lagrangian formulation, in the following sense:

Corollary 3.6.3. *Let $D_\infty(\mu) := \langle \ell, \mu \rangle - \gamma F_{\mathcal{I}}^*(\mu) - \frac{1}{2\lambda}\|\Psi\mu\|_2^2$, so that we have $D(\mu) = D_\infty(\mu) - \iota_{\{A\mu=0\}}$. If Δ_t and Γ_t converge linearly to 0, then $|D_\infty(\hat{\mu}^t) - D_\infty(\mu^\star)|$ and $\|A\hat{\mu}^t\|_2^2$ both converge to 0 linearly.*

Furthermore, if $\mathcal{A}$ is linearly convergent as in Corollary 3.6.2, the algorithm is linearly convergent in terms of the total number of inner steps (for SDCA this is the total number of clique updates) performed by algorithms $\mathcal{A}$ throughout:

Corollary 3.6.4 (Total number of inner updates)**.** *With the notations of the previous corollary, for any $\beta \in (0,1)$ such that $\lambda_{\max}(\beta) < 1$, it is possible to obtain $\mathbb{E}[\hat{\Delta}_t] \leq \epsilon$ and $\mathbb{E}[\Gamma_t] \leq \epsilon$ with a total number of inner iterations $T_{\text{tot}} := T_{\text{in}} T_{\text{ex}}$ such that*

$$T_{\text{tot}} \geq \frac{\log(\beta)}{\log \lambda_{\max}(\beta) \log(1-\pi)} \log(\epsilon).$$

We show in Appendix 3.D.5 that to have $\lambda_{\max}(\beta) < 1$ we should have $\beta = \alpha\kappa$ with $\alpha < \frac{1}{3(1+2\kappa)}$.

To reason in terms of rate, if the rate of convergence is r then we should have $T_{\text{tot}} \geq \frac{\log(\epsilon)}{\log(1-r)}$. So identifying the rate of convergence of the algorithm yields $r = 1 - \exp\left(\frac{\log(1-\pi)\log(\lambda_{\max}(\beta))}{\log(\beta)}\right)$. If α and κ are not too large, we can get a simplified expression for the rate, characterized as follows.

Corollary 3.6.5 (Convergence rate)**.** *If $\kappa < \frac{1}{2}$ and $\alpha = \frac{1}{12}$, if $T_{\text{in}} \geq \frac{\log(\alpha\kappa)}{\log(1-\pi)}$, then, there exist a constant $C' > 0$ such that after a total of $t\,T_{\text{in}} + s$ inner updates, we have*

$$\mathbb{E}[\Delta_t^s + \Gamma_t] \leq C'\left(1 - \frac{\kappa\pi}{2\log(12/\kappa)}\right)^{t\,T_{\text{in}}+s}.$$

3.6.2 Convergence results with SDCA

Given the structure of D_ρ, if the functions f_c^* in eq.(3.7) are strongly convex, a good candidate for $\mathcal{A}$ is stochastic dual coordinate ascent (SDCA). Indeed, the results of Shalev-Shwartz and Zhang (2016) show that

Proposition 3.6.6. *If $\mathcal{A}$ is SDCA, let $|\mathcal{C}|$ be the total number of cliques, σ_c the strong convexity constant of f_c^*, and L_c the Lipschitz constant of $\mu_c \mapsto r(\mu)$, then $\mathcal{A}$ is linearly convergent with rate $\pi = \min_{c\in\mathcal{C}} \frac{\sigma_c}{|\mathcal{C}|(\sigma_c+L_c)}$.*

Moreover SDCA allows us to bound the duality gap by the increase of D_ρ, which yields linear convergence in the primal.

Proposition 3.6.7. *Let $\hat{w}^t = w(\hat{\mu}^t)$. If $\mathcal{A}$ is SDCA, then*

$$\mathbb{E}[P(\hat{w}^t) - P(w^\star)] \leq \frac{1}{\pi}\mathbb{E}[\hat{\Delta}_t] + \mathbb{E}[\Gamma_t].$$

For the sake of the natural surrogates for the entropy (like the Gini-OTRW entropy proposed in Appendix 3.C), individual functions f_c^* are not strongly convex, although $-D_\rho$ is strongly convex, because the entropy surrogate is strongly concave on $\mathcal{L}$ and the term $\|A\mu\|^2$ is strongly convex on $\text{Ker}(A)^\perp$. In that case another decomposition is relevant: if σ is the strong convexity constant of $-D_\rho$, then let $\tilde{f}_c^*(\mu_c) = \iota_{\triangle_c}(\mu_c) + \sigma\|\mu_c\|_2^2$ and $\tilde{r}(\mu) = -H_{\text{Approx}}(\mu) + r(\mu) - \sigma\|\mu_c\|_2^2$. We again have $D_\rho(\mu) = -\sum_{c\in\mathcal{C}} \tilde{f}_c^*(\mu_c) - \tilde{r}(\mu)$, with $\tilde{f}_c^*$ strongly convex and $\tilde{r}$ convex and smooth. SDCA and its theory are here applicable again and guarantees that Proposition 3.6.6 and following hold. However, for the convergence in the primal a slightly different argument is needed.

Proposition 3.6.8 (Linear convergence in the primal)**.** *Let $w^{t,s} = w(\mu^{t,s})$. If $\mathcal{A}$ is a linearly convergent algorithm and the function $\mu \mapsto -H_{\text{approx}} + \frac{1}{2\rho}\|A\mu\|_2^2$ is strongly convex, then $P(w^{t,s}) - P(w^\star)$ converges to 0 linearly.*

3.6.3 Discussion

Optimization with inexact gradients (Devolder et al. 2014) and inexact proximal operators (Schmidt et al. 2011) have been shown to yield the same convergence rate as their exact counterparts, provided that errors decrease at a certain rate. Linear convergence of an inexact augmented Lagrangian method in which both inner and outer optimizations use Nesterov's accelerated gradient descent is shown in Lan and Monteiro (2016). We use the same ideas, except that we leverage the large finite sum structure of the dual problem to use randomized algorithms. The use of warm-start is also similar to its use in the meta-algorithm proposed by Lin et al. (2017), who use inexact gradient descent on the Moreau-Yosida regularization of a non-smooth objective. In our context, this approach would actually be applicable by working on $P_\rho(w, \xi)$ instead of working in the dual. An investigation in this direction is of interest but beyond the scope of this paper.

3.7 Experiments

We evaluate our algorithm IDAL on three different CRF models including 1) a simulated Gaussian mixture Potts model with grid graph and two clique types (nodes and edges); 2) a semantic segmentation model with planar graph and two clique types (nodes and edges); 3) a multi-label classification model with fully-connected graph and unique clique type for all cliques.

We compare with algorithms using only clique-wise oracles for solving $\min_\xi \max_\mu D_\rho(\mu, \xi)$, namely, the soft-constrained block-coordinate Frank-Wolfe algorithm (SoftBCFW) by Meshi et al. (2015b) and the greedy direction method of multipliers (GDMM) algorithm by Yen et al. (2016). Note that SoftBCFW in fact solves only the special case $\max_\mu D_\rho(\mu, \xi \equiv 0)$, thus it will converge to a different point than IDAL. In addition, we include a third baseline for the special case using SDCA (referred as SoftSDCA). Since SoftBCFW and GDMM have been shown outperforming other baselines such as Lacoste-Julien et al. (2013a), Meshi et al. (2010) and Hazan and Urtasun (2010), we will not make an extensive comparison for all these algorithms.

3.7.1 Setup

Gaussian mixture Potts models

This is an extension of the Potts model given observations, whose conditional density function is defined via Bayes' rule $p(y|x) \propto p(x|y)p(y)$, with $p(y)$ a Potts distribution associated with a grid graph and parameterized by $w_{\text{binary}} \in \mathbb{R}^{k^2}$, and with $p(x|y) = \prod_i p(x_i|y_i)$ assumed to factorize into independent conditional Gaussian distributions with canonical parameters $w_{\text{unary}} \in \mathbb{R}^{2k}$, i.e., $p(x_i|y_i) \propto \exp(\langle w_{\text{unary}}(y_i), [x_i, x_i^2] \rangle)$. We consider a 10×10 grid graph with node cardinality $k = 5$. To generate the data, we first draw the label y from $p(y)$, and then the observation x_i is generated from the conditional Gaussian $p(x_i|y_i)$ for each node.

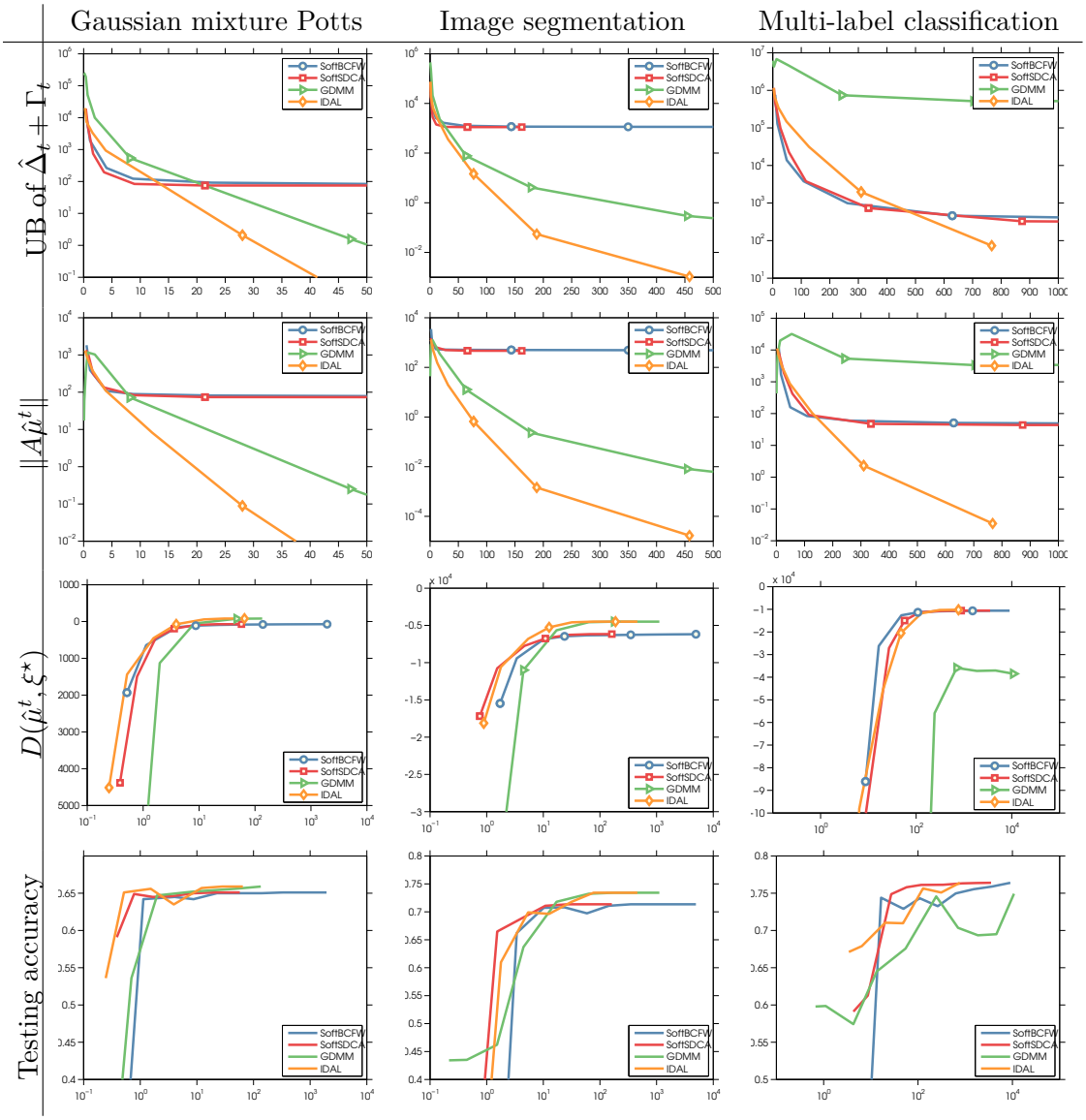


Figure 3.1: The comparison between IDAL and other baselines. For the choices of hyperparameters in terms of accuracy and speed, we set $(\lambda = 10, \rho = 1, \gamma = 1)$ for Gaussian mixture Potts, $(\lambda = 10, \rho = 1, \gamma = 10)$ for semantic segmentation and $(\lambda = 1, \rho = 0.1, \gamma = 1)$ for multi-label classification. The x-axis is running time in seconds.

The simulated dataset contains 100 samples and is equally divided for training and testing.

Semantic image segmentation

We consider a typical CRF model used in computer vision for labeling image pixels with semantic classes. The graph is built upon clustering pixels into superpixels. Each superpixel defines a node. Two superpixels with a shared boundary define an edge. The CRF model takes the form $p(y|x) \propto \exp\Big(\sum_i w_{\text{unary}}^\intercal \psi_i(x, y_i) + \sum_{i,j} w_{\text{binary}}^\intercal \psi_{ij}(x, y_i, y_j)\Big)$, where $\psi_i(x, y_i)$ measure the intra-cluster compatibility

within the superpixel i, and $\psi_{ij}(x, y_i, y_j)$ measure the inter-cluster compatibility between superpixels i and j. We conduct the experiment on the *MSRC-21* dataset introduced by Shotton et al. (2006), which has 21 classes, 335 training images and 256 testing images.

Multi-label classification

The task for this problem is assigning each input vector a set of binary target labels. It is natural to model the inter-label dependencies by CRFs that treat each label as a node in a fully connected label graph. Following Finley and Joachims (2008), we define the CRF density function as $p(y|x) \propto \exp(\sum_i w_i^\intercal \phi_i(x, y_i) + \sum_{i,j} w_{ij}^\intercal \phi_{ij}(y_i, y_j))$, where the feature maps are specified as $\phi_i(x, y_i) = y_i \otimes x$ for each node and $\phi_{ij}(y_i, y_j) = y_i \otimes y_j$ for each edge. We conduct the experiments on the *Yeast* dataset[2], which contains 1500 training samples and 917 testing samples. Each sample has 14 labels and 103 attributes.

Hyperparameters

In theory, T_{in} could be very large depending on the choice of α and the condition number π. We find that in practice only a relatively small T_{in} is needed. We empirically choose $T_{\text{in}} = \frac{1}{2}|\mathcal{C}|$. We set the number of outer iterations $T_{\text{ex}} = 3000$ and the stopping threshold $\epsilon = 10^{-3}$. The ranges of λ is pre-defined as $\{10, 1.0, 0.01, 0.001\}$ and the range of γ is $\{100.0, 10.0, 1.0, 0.001\}$. For each experiment, we choose the best λ and γ in terms of the validation accuracy and a reasonable running time (not all experiments finished in 3000 outer iterations). We set $\rho = 1.0$ or $\rho = 0.1$ as in Meshi et al. (2015b).

3.7.2 Results

To compare IDAL with GDMM, we use the criterion $P_\rho(\hat{w}^t, \hat{\delta}^t, \xi^t) - D_\rho(\hat{\mu}^t, \xi^t) + P_\rho(\hat{w}^t, \hat{\delta}^t, \xi^t) - D_\rho(\bar{\mu}^{T_{\text{ex}}}, \xi^{T_{\text{ex}}})$, which is an upper bound of the theoretical quantity $\hat{\Delta}_t + \Gamma_t$ that we analyzed. To compare IDAL with SoftBCFW, since $\xi = 0$ for SoftBCFW, we use the criterion $D_\rho(\hat{\mu}^t, \xi^\star)$, in which $\xi^\star$ is obtained from running IDAL to convergence. Besides, we also use the criteria $\|A\hat{\mu}^t\|^2$ (it measures the convergence of $d(\xi)$, since $\nabla d(\xi^t) \simeq A\hat{\mu}^t$) and the testing accuracy, which are applicable for all three algorithms. The results are shown in Figure 3.1.

There are several interesting points that we can say based on the results: 1) by tightening the marginalization constraints $A\mu = 0$, it does help to gain a better testing accuracy (e.g., IDAL gains small improvements over SoftBCFW); 2) based on the curves of $D_\rho(\mu, \xi^\star)$, we can see that it is key to approach $\mu^\star$ by first obtaining $\xi^\star$, which again shows the importance of enforcing exactness of the local consistency polytope; 3) IDAL is shown to be a faster algorithm than GDMM.

[2] http://sourceforge.net/projects/mulan/files/datasets/yeast.rar

One possible reason is that GDMM is in fact an active-set algorithm, which means the number of updated cliques at very beginning is insufficient comparing to IDAL. Based on our analysis, we have shown that the quality of the approximate gradient $A\hat{\mu}_t$ depends on T_{in}. Therefore, it is very likely that GDMM suffers from a slow convergence because of the poor gradients.

3.8 Conclusion

We proposed a relaxed dual augmented Lagrangian formulation for CRF learning, in which, thanks to dual decomposition, SDCA can be used to partially optimize over mean parameters in order to yield a sufficiently good approximation of the multiplier gradient. Our theoretical analysis shows that if warm-starts are leveraged and multiplier gradients are approximated with a linearly convergent algorithm, global linear convergence can be obtained. If SDCA is used, linear convergence is obtained both in the primal and for the convergence of the dual Lagrangian method.

Comparing to other baselines such as GDMM and SoftBCFW, our algorithm is faster in terms of the distance to the optimal objective function value (i.e. $\hat{\Delta}_t + \Gamma_t$) and the feasibility of the constraints $\|A\mu\|_2^2$.

It would be of interest to investigate the use of the same dual augmented Lagrangian formulation for both inference and learning, since according to Wainwright (2006), this should improve the performance.

In future work, we intend to investigate applications to other problems in machine learning, the use of Nesterov acceleration or quasi-Newton methods for multiplier updates, or the connection to other approaches based on Moreau-Yosida regularization.

Appendix

3.A Loss-Augmented CRF

In order to extend our learning formulation so as to encompass as well max-margin structured learning (i.e., structured SVM) in additional to maximum likelihood learning, we show in this section that our formulation can be generalized to cover the loss-augmented CRF learning introduced by Pletscher et al. (2010) and Hazan and Urtasun (2010).

The loss-augmented CRF $p_\gamma(y \mid y^\star, x)$ is an extension of the standard CRF with additional user-defined loss functions $\ell_c(y_c^\star, y_c)$ for all cliques and an extra temperature hyperparameter $\gamma \in (0, +\infty)$. We introduce a modified natural parameter $\eta_\ell(w) := \eta(w) + \ell$ (similarly we have θ_ℓ) that includes the loss term $\ell = \big[[\ell_c(y_c^\star, y_c) \colon y_c \in \mathcal{Y}_c] \colon c \in \mathcal{C}\big]$. The density function of the loss-augmented CRF then takes the form

$$p_\gamma(y \mid y^\star, x; w) = \exp\Big(\langle \eta_\ell(w)/\gamma, T(y)\rangle - F(\eta_\ell(w)/\gamma)\Big). \tag{3.8}$$

A justification for the form of the loss-augmented CRF is based on a rationale that distinguishes the label to predict y (which is essentially true unknown label) from the label provided by the annotation $y^\star$. The assumption made is then that, given y_c, the annotation $y_c^\star$ is independent of x and $y_{c'}$ for $c' \neq c$. This entails that $p(y, y^\star|x) = p_\gamma(y \mid y^\star, x) \propto p(y \mid y^\star) p_\gamma(y \mid x)$, which yields the above form for $p_\gamma(y \mid y^\star, x; w)$ by Bayes' rule for $p(y \mid y^\star) \propto \exp(\sum_{c\in\mathcal{C}} \ell_c(y_c^\star, y_c))$.

For learning, we use a rescaled maximum likelihood objective (i.e., multiplied by γ) of the form

$$\min_w \gamma F\Big(\tfrac{1}{\gamma}\theta_\ell(w)\Big) + \frac{\lambda}{2}\|w\|_2^2, \tag{3.9}$$

with which we can see γ only affects the entropy term in the variational representation of F, thus it plays a role to determine the learning regime. When $\gamma \to 0$, we retrieve a max-margin formulation for structured output learning, since the corresponding variational problem based on Fenchel duality is

$$\min_w \max_{\mu\in\mathcal{M}} \langle \mu, \theta_\ell\rangle + \frac{\lambda}{2}\|w\|_2^2. \tag{3.10}$$

Note that this is identical to the linear programming relaxation of the structured SVM formulation studied by Meshi et al. (2010).

It is also possible to retrieve the maximum likelihood regime by making a change of variable: $w' = w/\gamma$. Then, eq.(3.9) becomes

$$\min_{w'} F\Big(\theta(w') + \frac{1}{\gamma}\ell\Big) + \frac{\lambda\gamma}{2}\|w'\|_2^2. \tag{3.11}$$

Increasing γ decreases the effect of the loss term and simultaneously increases the effect of the regularization. The maximum likelihood regime is thus retrieved by letting $\gamma \to +\infty$ and $\lambda \to 0$.

3.B Derivations of dual, and relaxed primal and dual objectives

In this section, we derive the dual objective $D(\mu)$ of $P(w)$. Given the augmented Lagrangian $D_\rho(\mu, \xi)$, we first introduce a relaxed primal $\tilde{P}_\rho(w, \delta, \xi)$ involving a new primal variable δ whose components can be interpreted as messages exchanged between cliques in the context of marginal inference via message-passing algorithms. The partial minimization with respect to δ then yields the corresponding primal of $D_\rho(\mu, \xi)$ with respect to μ for a fixed ξ: $P_\rho(w, \xi) := \min_\delta \tilde{P}_\rho(w, \delta, \xi)$, which can be interpreted as a Moreau-Yoshida smoothing of the original objective $P_\rho(w)$.

3.B.1 Derivation of the dual objective $D(\mu)$

Given that $\theta_\ell(w) = \Psi^\intercal w + \ell$ and introducing the Fenchel conjugate of $F_\mathcal{L}$, we have

$$\begin{aligned} P(w) &= \gamma F_\mathcal{L}\Big(\frac{1}{\gamma}\theta_\ell(w)\Big) + \frac{\lambda}{2}\|w\|_2^2 \\ &= \max_{\mu\in\mathcal{L}} \Big[\langle \Psi^\intercal w + \ell, \mu\rangle - \gamma F_\mathcal{L}^*(\mu)\Big] + \frac{\lambda}{2}\|w\|_2^2. \end{aligned}$$

Given that the local polytope constraints are defined by linear inequalities, weak Slater constraint qualification are satisfied, so that strong duality holds and an equivalent dual problem in μ is obtained by switching the order of $\min_w$ and $\max_\mu$:

$$\begin{aligned} D(\mu) &= \langle \ell, \mu\rangle - \gamma F_\mathcal{L}^*(\mu) + \min_w \Big[\langle \Psi^\intercal w, \mu\rangle + \frac{\lambda}{2}\|w\|_2^2\Big] \\ &= \langle \ell, \mu\rangle - \gamma F_\mathcal{L}^*(\mu) - \lambda \max_w \Big[-\frac{1}{\lambda}\langle \Psi\mu, w\rangle - \frac{1}{2}\|w\|_2^2\Big] \\ &= \langle \ell, \mu\rangle - \gamma F_\mathcal{L}^*(\mu) - \frac{1}{2\lambda}\|\Psi\mu\|_2^2. \end{aligned}$$

3.B.2 Derivation of an extended primal $\tilde{P}_\rho(w, \delta, \xi)$

Proposition 3.B.1. *For a fixed ξ, the primal objective function of $D_\rho(\mu, \xi)$ takes the form*

$$P_\rho(w, \xi) := \min_\delta \Big[\tilde{P}_\rho(w, \delta, \xi) := \gamma F_\mathcal{I}\Big(\frac{\theta(w) + A^\intercal\delta}{\gamma}\Big) + \frac{\lambda}{2}\|w\|_2^2 + \frac{\rho}{2}\|\delta - \xi\|^2\Big].$$

Proof. Clearly, we have $D(\mu) = \min_\xi D_\rho(\mu, \xi)$. For a fixed value of ξ, consider the Lagrangian

$$L_{\rho,\xi}(\mu, \nu, \nu', w, \delta) = \langle \ell, \mu\rangle - \gamma F^*_{\mathcal{I}}(\mu) - \frac{1}{2\lambda}\|\nu\|^2 - \frac{1}{2\rho}\|\nu'\|^2 + \langle \xi, \nu'\rangle + \langle w, \Psi\mu - \nu\rangle + \langle \delta, A\mu - \nu'\rangle;$$

Then clearly $\min_{w,\delta} L_{\rho,\xi}(\mu, \nu, \nu'; w, \delta) = D_\rho(\mu, \xi)$. We compute the associated primal as

$$\begin{aligned}\tilde{P}_\rho(w, \delta, \xi) &= \max_{\mu,\nu,\nu'} L_{\rho,\xi}(\mu, \nu, \nu', w, \delta) \\ &= \max_u \Big[\langle \mu, \ell + \Psi^\intercal w + A^\intercal\delta\rangle - \gamma F^*_{\mathcal{I}}(\mu)\Big] \\ &\quad + \max_\nu \Big[\langle \nu, -w\rangle - \frac{1}{2\lambda}\|\nu\|^2\Big] + \max_{\nu'} \Big[\langle \nu', \xi - \delta\rangle - \frac{1}{2\rho}\|\nu'\|^2\Big],\end{aligned}$$

which yields the desired form of $P_\rho(w, \xi) = \min_\delta \tilde{P}_\rho(w, \delta, \xi)$ upon expliciting Fenchel conjugates. □

Proposition 3.B.2. *The following equalities hold:*

$$\min_\delta F_{\mathcal{I}}\Big(\tfrac{1}{\gamma}(\theta(w) + A^\intercal\delta)\Big) = F_{\mathcal{L}}\Big(\tfrac{1}{\gamma}\theta(w)\Big) \qquad \textit{and} \qquad \min_{\xi,\delta} \tilde{P}_\rho(w, \delta, \xi) = P(w).$$

Proof. Notice that

$$\begin{aligned}\min_\delta F_{\mathcal{I}}\Big(\tfrac{1}{\gamma}(\theta(w) + A^\intercal\delta)\Big) &= \min_\delta \max_\mu \Big(\frac{1}{\gamma}\langle \theta(w) + A^\intercal\delta, \mu\rangle + H_{\text{Approx}}(\mu) - \iota_{\mathcal{I}}(\mu)\Big) \\ &= \max_\mu \Big(\frac{1}{\gamma}\langle \theta(w), \mu\rangle + H_{\text{Approx}}(\mu) - \iota_{\mathcal{I}}(\mu) - \iota_{\{A\mu=0\}}\Big) \\ &= F_{\mathcal{L}}\Big(\tfrac{1}{\gamma}\theta(w)\Big),\end{aligned}$$

where the second equality follows by exchanging minimization and maximization (strong duality holds by Slater's conditions) and minimizing with respect to δ.

To show that $\min_{\xi,\delta} \tilde{P}_\rho(w, \delta, \xi) = P(w)$, it is easy to minimize over ξ first, which cancels out the term $\frac{\rho}{2}\|\delta - \xi\|^2$ by setting $\xi = \delta$. Then, δ only appears in $F_{\mathcal{I}}$ and the result follows from the first result. □

3.B.3 Interpretation as Moreau-Yosida smoothing

To understand the structure of $P_\rho(w, \xi)$, we shall look at $\tilde{P}_\rho(w, \delta, \xi)$. One may be interested in where does δ comes from? In fact, forming the Lagrangian of $\min_w P(w)$ with Lagrangian multiplier δ corresponding to the marginalization constraint $A\mu = 0$, we see that

$$L(w, \delta, \mu) := \langle \theta_\ell(w), \mu\rangle - \gamma F^*_{\mathcal{I}}(\mu) + \frac{\lambda}{2}\|w\|_2^2 + \langle \delta, A\mu\rangle.$$

Recall that the Moreau-Yosida regularization of a function f is defined as the infimal convolution

$$M_{\rho f}(x) = \min_z \Big[f(z) + \frac{\rho}{2}\|z - x\|^2\Big].$$

Both $P_\rho(w,\xi)$ and $D_\rho(\mu,\xi)$ have a nice interpretation in terms of the Lagrangian L and Moreau-Yosida regularization. Note that the Moreau-Yosida regularization admits the same optimum as the original function, and that it is smooth even when the original function is not. It is furthermore $\frac{\gamma\rho}{\gamma+\rho}$-strongly convex if the original function is γ-strongly convex.

Proposition 3.B.3. *$P_\rho(w,\xi)$ and $D_\rho(\mu,\xi)$ are respectively the Moreau-Yosida regularizations of $L_{\mu^\star}\colon w,\delta \mapsto \max_\mu L(w,\delta,\mu)$ and $L_{w^\star}\colon \mu,\delta \mapsto \min_w L(w,\delta,\mu)$ about δ. that is*

$$P_\rho(w,\xi) = M_{\rho L_{\mu^\star}}(w,\xi) = \min_\delta \left[\max_\mu L(w,\delta,\mu) + \frac{\rho}{2}\|\delta-\xi\|_2^2\right]$$
$$D_\rho(\mu,\xi) = M_{\rho L_{w^\star}}(\mu,\xi) = \min_\delta \left[\min_w L(w,\delta,\mu) + \frac{\rho}{2}\|\delta-\xi\|_2^2\right].$$

Proof. For $P_\rho(w,\xi)$, note that $\max_\mu L(w,\delta,\mu) \equiv \gamma F_{\mathcal{I}}\left(\frac{\theta(w)+A^\intercal\delta}{\gamma}\right) + \frac{\lambda}{2}\|w\|_2^2$. The equivalent form is immediately derived from Proposition 3.B.1.

For $D_\rho(\mu,\xi)$, note that $\min_w L(w,\delta,\mu) \equiv \langle\theta,\mu\rangle - \gamma F_{\mathcal{I}}^*(\mu) - \frac{1}{2\lambda}\|\Psi\mu\|^2 + \langle\delta, A\mu\rangle$, and $\min_\delta\langle\delta,A\mu\rangle + \frac{\rho}{2}\|\delta-\xi\|^2 \equiv \langle\xi,A\mu\rangle - \frac{1}{2\rho}\|A\mu\|_2^2$. Thus, the equivalence holds. □

Note that the penalty formulation corresponds to a special case of $\tilde{P}_\rho(w,\delta,\xi)$ and $D_\rho(\mu,\xi)$ with $\xi=0$. It introduces an additional term $\frac{\rho}{2}\|\delta\|^2$, thus making the primal strongly convex with respect to δ and the dual smoother in μ. This effect is similar to that of using Moreau-Yosida smoothing. However, the additional term $\frac{\rho}{2}\|\delta\|^2$ will never vanish, so $A\mu=0$ will never be satisfied. The more $A\mu=0$ is violated, the less the structure of CRF will be perserved.

3.B.4 Duality gaps and representer theorem

Besides, if we define $\mathrm{gap}(w,\delta,\mu,\xi) := \tilde{P}_\rho(w,\delta,\xi) - D_\rho(\mu,\xi)$ as an upper-bound estimate of the duality gap $P_\rho(w,\xi) - D_\rho(\mu,\xi)$, specifically

$$\begin{aligned}\mathrm{gap}(w,\delta,\mu,\xi) = &\left[\gamma F_{\mathcal{I}}\left(\tfrac{1}{\gamma}\theta_\ell(w) + A^\intercal\delta\right) + \gamma F_{\mathcal{I}}^*(\mu) - \langle\theta_\ell(w) + A^\intercal\delta,\mu\rangle\right]\\ &+ \left[\frac{\lambda}{2}\|w\|^2 + \frac{1}{2\lambda}\|\Psi\mu\|^2 - \langle -w,\Psi\mu\rangle\right]\\ &+ \left[\frac{\rho}{2}\|\xi-\delta\|^2 + \frac{1}{2\rho}\|A\mu\|^2 - \langle\xi-\delta,A\mu\rangle\right],\end{aligned}$$

we can see that the recovered w and δ by the optimality condition make the 2nd and 3rd term of $\mathrm{gap}(w,\delta,\mu,\xi)$ disappear. We will see later this is important in designing the algorithm to solve $\max_\mu D_\rho(\mu,\xi)$.

Finally, we give a rough picture of all the quantities that we introduced in this section, which can be easily derived from Proposition 3.B.3.

Corollary 3.B.4. *The relations between D, D_ρ, P and P_ρ could be summarized as*

$$\begin{aligned}
&D(\mu) \leq D_\rho(\mu, \xi) \leq P_\rho(w, \xi); \\
&D_\rho(\mu) \leq P(w) \leq P_\rho(w, \xi) \leq \tilde{P}_\rho(w, \delta, \xi); \\
&\max_\mu \min_\xi D_\rho(\mu, \xi) \leq \min_w P(w),
\end{aligned}$$

with equalities hold for the saddle point $(\mu^\star, w^\star, \xi^\star)$. Moreover, the first-order optimality conditions are given as

$$w^\star = -\frac{1}{\lambda}\Psi\mu^\star, \quad \delta^\star = \xi^\star - \frac{1}{\rho}A\mu^\star \tag{3.12}$$

Proof. By constructions, $D(\mu) = \min_\xi D_\rho(\mu, \xi) \leq D_\rho(\mu, \xi)$ and $P_\rho(w, \xi) = \min_\delta \tilde{P}_\rho(w, \delta, \xi) \leq \tilde{P}_\rho(w, \delta, \xi)$. Other inequalities are the consequences of Proposition 3.B.3 and the min-max inequality. Since the strong duality holds (Slater conditions satisfied and the problem is convex), we know that the equalities will hold at the saddle point.

Given the saddle point $(\mu^\star, w^\star, \xi^\star)$, to derive $w^\star, \delta^\star$ from $\mu^\star$, we know that $w^\star, \delta^\star = \arg\min_{w,\delta} \tilde{P}_\rho(w, \delta, \xi^\star)$. The result follows after computing $\nabla_w \tilde{P}_\rho(w, \delta, \xi^\star) = 0$ and $\nabla_\delta \tilde{P}_\rho(w, \delta, \xi^\star) = 0$. □

So our strategy for CRF learning is $\min_\xi \max_\mu D_\rho(\mu, \xi)$, since we know that

$$D_\rho(\mu^\star, \xi^\star) \equiv L(w^\star, \delta^\star, \mu^\star) \equiv P(w^\star).$$

Since we work on the space of μ and ξ, to compute the primal objectives or the duality gap, we can use the mapping specified by the optimality condition eq.(3.12). More precisely, we define

$$w(\mu^{t,s}) = -\frac{1}{\lambda}\Psi\mu^{t,s}, \quad \delta(\mu^{t,s}, \xi^t) = \xi^t - \frac{1}{\rho}A\mu^{t,s},$$

which is equivalent to the representer theorem. The above condition is also useful to recover intermediate $w^{t,s}$ from $\mu^{t,s}$, which allows us to test on the validation set or decide if we should stop the learning earlier.

3.B.5 Comparison with State-of-the-Art Structured Learning Methods

A number of recent works for CRF learning can be viewed as optimizing formulations which are exactly or fairly close to one of $P(w), D(\mu), P_\rho(w, \delta), \tilde{P}_\rho(w, \delta, \xi)$ or $D_\rho(\mu, \xi)$. In the following table, we compare these approaches, in terms of the optimization formulation, the convergence rate (respectively in the primal or in the dual), and the inference oracle used for computing the gradients (or blockwise gradients).

Table 3.B.1: The Comparison of Structured Learning Methods

Method	Learning Regime	Primal/Dual	Convergence	Inference Oracle
Meshi et al. (2010)	SSVM	Primal (w, δ)	Sublinear	graphwise MAP (inexact)
Hazan and Urtasun (2010)	LossAugCRF	Primal (w, δ)	Sublinear	graphwise marginal (inexact)
Lacoste-Julien et al. (2013a)	SSVM	Dual (μ)	Sublinear	graphwise MAP
Schmidt et al. (2015)	CRF	Primal (w)	Linear	graphwise marginal
Tang et al. (2016)	CRF	Dual (μ)	Sublinear	graphwise MAP
Meshi et al. (2015b)	SSVM (soft)	Dual $(\mu, \xi = 0)$	Sublinear	cliquewise MAP
Yen et al. (2016)	SSVM	Dual (μ, ξ)	Linear	cliquewise MAP
IDAL	LossAugCRF	Dual (μ, ξ)	Linear	cliquewise marginal

3.C Gini Oriented Tree-Reweighted Entropy

The Bethe entropy (Yedidia et al. 2005) is generally non-concave. Its concave counterparts, such as the tree-reweighted entropy (Wainwright et al. 2005b) or the region-based entropy (Yedidia et al. 2005, London et al. 2015), are only concave on the local consistency polytope, but non-concave on $\mathcal{I} \setminus \mathcal{L}$ (i.e., when $A\mu \neq 0$). Indeed, the Bethe entropy and its concave variants are of the form $H_{\text{Bethe}}(\mu) = \sum_{i \in \mathcal{V}} c_i H_i(\mu_i) + \sum_{\{i,j\} \in \mathcal{E}} c_{ij} H_{ij}(\mu_{ij})$, where c_i and c_{ij} are counting numbers. Even when H_{Bethe} is concave on $\mathcal{L}$, some of the c_i or c_{ij} can be negative.

The construction of the oriented tree-reweighted entropy stems from the expression of the entropy of a directed tree as the sum of of the entropy of the root and the conditional entropies of the variable at each node given their parent variable. Precisely, for an oriented tree T with the root i_0, the joint entropy can be computed as

$$H_T(Y) := H(Y_{i_0}) + \sum_{j \to i \in T} H(Y_i \mid Y_j). \tag{3.13}$$

On a general graph, if T is a (directed) spanning tree of the graph, then

$$\begin{aligned} H_T(Y) &:= H(Y_{i_0}) + \sum_{t \to i \in T} H(Y_i \mid Y_j) \\ &\geq H(Y_{i_0}) + \sum_{k=1}^{m} H(Y_{i_k} \mid Y_{i_{k-1}}, \ldots, Y_{i_0}) =: H_{\text{Shannon}}(Y). \end{aligned} \tag{3.14}$$

Thus, for any probability distribution over the set of valid directed spanning trees, in which tree T has probability ρ_T, the inequality above entails that $H_{\text{Shannon}}(Y) \leq \sum_T \rho_T H_T(Y) =: H_{\text{OTRW}}(Y)$, where $\rho_T \geq 0$ and $\sum_T \rho_T = 1$.

$H_T(Y)$ is concave since it is a sum of concave functions, and so is $H_{\text{OTRW}}(Y)$ (who is a convex combination of $H_T(Y)$). To see that, we need to prove the following fact.

Remark 3.C.1 (Concavity of the conditional entropy)**.** *The conditional entropy $H(Y_j \mid Y_i)$ is in fact a function of μ_{ij}, namely $H(Y_j \mid Y_i) = H(\mu_{ij}) - H(A_i \mu_{ij})$. Moreover, $H(Y_j \mid Y_i)$ is a concave function of μ_{ij}.*

Proof. By definition,

$$H(Y_j \mid Y_i) = \sum_{y_j, y_i} \mu_{ij}(y_j, y_i) \log \frac{\sum_{y_j} \mu_{ij}(y_j, y_i)}{\mu_{ij}(y_j, y_i)} = H(\mu_{ij}) - H(A_i \mu_{ij}).$$

To show $H(Y_j \mid Y_i)$ is concave, we compute its Hessian:

$$\begin{aligned}\frac{\partial^2 H(Y_j \mid Y_i)}{\partial \mu_{ij}^2} &= -\mathrm{diag}\Big(\mathbf{1} \oslash \mu_{ij}\Big) + A^\intercal \mathrm{diag}\Big(\mathbf{1} \oslash A\mu\Big) A \\ &= -\mathrm{diag}\Big(\mathbf{1} \oslash \mu_{ij}\Big) + \mathrm{diag}\Big(\{\frac{1}{\tilde{\mu}_i(y_i)}\mathbf{1}\mathbf{1}^\intercal\}_{y_i=1}^{k_i}\Big)\end{aligned}$$

where $\tilde{\mu}_i = A_i\mu_{ij}$, and $\oslash$ denotes entrywise division. Let's focus on the i-th block of the negative Hessian. To show that the i-th block is positive semidefinite, that is, that

$$\mathrm{diag}\Big(\Big\{\frac{1}{\mu_{ij}(y_i, y_j)}\Big\}_{1 \le y_j \le k_j}\Big) - \frac{1}{\tilde{\mu}_i(y_i)}\mathbf{1}\mathbf{1}^\intercal \succeq 0, \tag{3.15}$$

we can use the Schur complement condition for positive semidefiniteness. Let $U = \tilde{\mu}_i(y_i)$. Since $\tilde{\mu}_i(y_i) \succ 0$,

$$L - B^\intercal U^{-1} B \succeq 0 \quad \text{iff} \quad \begin{bmatrix} U & B \\ B^\intercal & L \end{bmatrix} = \begin{bmatrix} \tilde{\mu}_i(y_i) & \mathbf{1}^\intercal \\ \mathbf{1} & \mathrm{diag}\Big(\{\frac{1}{\mu_{ij}(y_i,y_j)}\}_{1 \le y_j \le k_j}\Big) \end{bmatrix} \succeq 0.$$

We also have $L = \mathrm{diag}\Big(\Big\{\frac{1}{\mu_{ij}(y_i,y_j)}\Big\}_{y_j=1}^{k_j}\Big) \succ 0$, then

$$\begin{aligned}\begin{bmatrix} U & B \\ B^\intercal & L \end{bmatrix} \succeq 0 \quad \text{iff} \quad U - BL^{-1}B^\intercal &= \tilde{\mu}_i(y_i) - \mathbf{1}^\intercal \mathrm{diag}\Big(\{\mu_{ij}(y_i, y_j)\}_{y_j=1}^{k_j}\Big)\mathbf{1} \\ &= \tilde{\mu}_i(y_i) - \tilde{\mu}_i(y_i) \succeq 0.\end{aligned}$$

Because the last inequality holds, we know eq.(3.15) must be true, which implies that the Hessian of $H(Y_j \mid Y_i)$ is negative semidefinite, thus $H(Y_j \mid Y_i)$ is concave. □

Note that $H_{\mathrm{OTRW}}(\mu)$ is concave on the entire set $\mathcal{I}$, unlike many Bethe entropy variants who are only concave in the local consistency polytope.

We define $\overrightarrow{\mathcal{E}}$ the directed edge set by expanding each edge from $\mathcal{E}$ with two directed edges, ρ_i and $\rho_{\mathrm{t}ij}$ respectively as the probabilities of i (as the root) and $i \to t$ appearing in an oriented spanning tree when the latter is drawn with probability ρ_T. Then the oriented tree-reweighted entropy takes the form

$$\begin{aligned}H_{\mathrm{OTRW}}(\mu) := &\sum_{\{i,j\} \in \mathcal{E}} \rho_{j|i}\Big[H_e(\mu_{ij}) - H_i(A_i\mu_{ij})\Big] \\ &+ \rho_{i|j}\Big[H_e(\mu_{ij}) - H_j(A_j\mu_{ij})\Big] + \sum_{i \in \mathcal{V}} \rho_i H_i(\mu_i),\end{aligned} \tag{3.16}$$

where $H_i(\mu_i) = -\sum_{y_i} \mu_i(y_i) \log \mu_i(y_i)$, $H_e(\mu_{ij}) = -\sum_{y_i, y_j} \mu_i(y_i, y_j) \log \mu_i(y_i, y_j)$ and $\rho_i, \rho_{i|j}, \rho_{j|i}$ are node/edge appearance probabilities in $[0, 1]$. H_{OTRW} is concave, since H_i is concave and it can be checked that so is $\mu_{ij} \mapsto H_e(\mu_{ij}) - H_i(A_i\mu_{ij})$

(although not strongly concave). It is easy to precompute the appearance probabilities ρ_i and ρ_{tij} via a variant of the directed matrix-tree theorem. See Koo et al. (2007) for more details.

A generic difficulty with entropies, is that H_i and H_e do not have Lipschitz gradients, which prevents the direct application of proximal methods with usual quadratic proximity terms. We thus propose to replace H_i and H_e by their second-order Taylor approximation around the uniform distribution. This yields a surrogate of the form

$$H_{\text{GTRW}}(\mu) := \sum_{\{i,j\}\in\mathcal{E}} \varepsilon\Big[k_i\rho_{j|i}\|A_i\mu_{ij}\|^2 + k_j\rho_{i|j}\|A_j\mu_{ij}\|^2\Big] - k_ik_j(\rho_{tij}+\rho_{tji})\|\mu_{ij}\|^2 + \sum_{i\in\mathcal{V}} k_i\rho_i(1-\|\mu_i\|^2), \tag{3.17}$$

where $\varepsilon = 1$. Since this function is not strongly convex w.r.t. μ_{ij} because $k_jI_{k_i} - A_i^\intercal A_i$ has a non-trivial kernel, so we also consider variants with $\varepsilon < 1$ and denote them $H_{\text{GTRW},\varepsilon}$. We call this approximation the *Gini OTRW entropy*, since it is consistent with the definition of Gini conditional entropy of Furuichi (2006).

3.D Proof of Theorem 3.6.1 and associated Corollaries

To prove Theorem 3.6.1, we first need to show $d(\xi)$ is a smooth function, and then we build up the associated lemmas which will be used in the proof of Theorem 3.6.1. Finally, in the end of this section, we prove Corollary 3.6.2 and Corollary 3.6.3 as the results to show the linear convergence in the dual and in the primal.

3.D.1 Smoothness of $d(\xi)$

Lemma 3.D.1. *$d(\xi)$ is convex and L_d-smooth, where $L_d \leq \rho$.*

Proof. By definition we have

$$D_\rho(\mu,0) = -\langle\mu,\ell\rangle + \gamma F_{\mathcal{I}}^*(\mu) + \frac{1}{2\lambda}\|\Psi\mu\|^2 + \frac{1}{2\rho}\|A\mu\|^2.$$

We then have $d(\xi) = \max_\mu D_\rho(\mu,\xi) = \max_\mu\langle\mu, A^\intercal\xi\rangle - D_\rho(\mu,0)$ so that if $J(\mu) := D_\rho(\mu,0)$, then $d(\xi) = J^*(A^\intercal\xi)$ and d is a convex function by Fenchel conjugacy.

For any ξ_1 and ξ_2, denote by μ_1 and μ_2 the minimizers of $D_\rho(\cdot,\xi_1)$ and $D_\rho(\cdot,\xi_2)$ respectively. By convexity of $d(\xi)$ and the definition of subgradient, there exists $s_1 \in \partial F_{\mathcal{I}}^*(\mu_1)$ and $s_2 \in \partial F_{\mathcal{I}}^*(\mu_2)$ such that

$$A^\intercal\xi_1 + \ell - \gamma s_1 - \frac{1}{\lambda}\Psi^\intercal\Psi\mu_1 - \frac{1}{\rho}A^\intercal A\mu_1 = 0$$
$$A^\intercal\xi_2 + \ell - \gamma s_2 - \frac{1}{\lambda}\Psi^\intercal\Psi\mu_2 - \frac{1}{\rho}A^\intercal A\mu_2 = 0$$

By convexity of $F_{\mathcal{I}}^*(\mu)$, we have

$$\langle s_1 - s_2, \mu_1 - \mu_2 \rangle \geq 0,$$

which together with the equations above yields

$$\langle A^\intercal(\xi_1 - \xi_2) - \frac{1}{\lambda}\Psi^\intercal\Psi(\mu_1 - \mu_2) - \frac{1}{\rho}A^\intercal A(\mu_1 - \mu_2), \mu_1 - \mu_2 \rangle \geq 0.$$

Hence,

$$\langle \xi_1 - \xi_2, A(\mu_1 - \mu_2) \rangle \geq \frac{1}{\lambda}\|\Psi(\mu_1 - \mu_2)\|^2 + \frac{1}{\rho}\|A(\mu_1 - \mu_2)\|^2 \geq \frac{1}{\rho}\|A(\mu_1 - \mu_2)\|^2.$$

Now substituting $\nabla d(\xi_1) - \nabla d(\xi_2) = A(\mu_1 - \mu_2)$ into the above inequality and using the Cauchy-Schwarz inequality yields

$$\|\nabla d(\xi_1) - \nabla d(\xi_2)\| \leq \rho\|\xi_1 - \xi_2\|.$$

That completes the proof. □

3.D.2 Associated lemmas for Theorem 3.6.1

We first quantify in the next two lemmas how much $D(\mu, \xi^t)$ should be minimized in μ to provide a sufficiently accurate approximate gradient that it guarantees descent on d.

Lemma 3.D.2 (Error on the gradient)**.** *Denote $\bar{\mu}^t := \mu^\star(\xi^t) = \operatorname{argmin}_\mu D(\mu, \xi^t)$; $g_t := \nabla d(\xi^t) = A\mu^\star(\xi^t)$ and $\hat{g}_t := A\hat{\mu}^t$. Let $\hat{\Delta}_t := D_\rho(\bar{\mu}^t, \xi^t) - D_\rho(\hat{\mu}^t, \xi^t)$. We have $\frac{1}{2L_d}\|\hat{g}_t - g_t\|^2 \leq \hat{\Delta}_t$, where L_d is the smoothness constant of d.*

Proof. Let $d^*(y) = \max_\xi \langle \xi, y \rangle - d(\xi)$. Then, it can easily be checked by using the definition of d and exchanging the order of maximization and minimization that $d^*(y) = \min_\mu D_\rho(\mu, 0) + \iota_{\{A\mu = y\}}$,

Since d is convex, we have $d(\xi) = \max_y \langle \xi, y \rangle - d^*(y)$, so that if $y^\star(\xi)$ is a maximizer for fixed ξ we have

$$0 \in \xi - \partial d^*(y^\star(\xi)) \Rightarrow \xi \in \partial d^*(y^\star(\xi)).$$

The strong convexity of $d^*(y)$ implies that, for all y,

$$d^*(y) - d^*(y^\star(\xi)) - \langle \xi, y - y^\star(\xi) \rangle \geq \frac{1}{2L_d}\|y - y^\star(\xi)\|^2.$$

But for any μ, we have $D_\rho(\mu, \xi) = \langle A\mu, \xi \rangle - D_\rho(\mu, 0) \leq \langle A\mu, \xi \rangle - d^*(A\mu)$, and, for $\mu^\star(\xi)$, this inequality is an equality, since we have $D_\rho(\mu^\star(\xi), \xi) = \langle y^\star(\xi), \xi \rangle - d^*(y^\star(\xi))$ and $y^\star(\xi) = A\mu^\star(\xi)$. As a consequence, setting $y = A\mu$, we have

$$D_\rho(\mu^\star(\xi), \xi) - D_\rho(\mu, \xi) \geq \frac{1}{2L_d}\|A\mu - A\mu^\star(\xi)\|^2$$

by definition of $D_\rho(\mu, \xi)$. We conclude the proof by substituting μ with $\hat{\mu}^t$ and ξ with ξ^t. □

Lemma 3.D.3 (Guaranteed decrease on d)**.** *If we take inexact gradient on ξ with a fixed step size $\frac{1}{L_d}$, namely $\xi^{t+1} = \xi^t - \frac{1}{L_d}\hat{g}_t$, then*

$$d(\xi^t) - d(\xi^{t+1}) \geq \frac{\tau}{L_d}\Gamma_t - \hat{\Delta}_t, \tag{3.18}$$

where $\tau \in (0, L_d)$ satisfying $\frac{1}{2\tau}\|g_t\|^2 \geq \Gamma_t$.

Proof. Since $d(\xi)$ is L_d-smooth, we have

$$d(\xi^{t+1}) - d(\xi^t) \leq \langle \nabla d(\xi^t), \xi^{t+1} - \xi^t \rangle + \frac{L_d}{2}\|\xi^{t+1} - \xi^t\|^2$$

Using the gradient step and $\nabla d(\xi^t) = g_t$, the above inequality can be simplified as

$$\begin{aligned} d(\xi^{t+1}) - d(\xi^t) &\leq \langle g_t, -1/L_d \hat{g}_t \rangle + \frac{L_d}{2}\|1/L_d \hat{g}_t\|^2 \\ &= \frac{1}{2L_d}\Big(\|\hat{g}_t - g_t\|^2 - \|g_t\|^2\Big). \end{aligned} \tag{3.19}$$

We notice that the error bound given by the Lemma 2.3 of Hong and Luo (2017) holds for $d(\xi)$. Specifically,

$$\exists \tau' > 0, \text{ such that } \|\nabla d(\xi)\| \geq \tau' \|\xi - \xi^\star\|.$$

Since $d(\xi)$ is L_d-smooth and $\nabla d(\xi^\star) = 0$, we have

$$d(\xi) - d(\xi^\star) \leq \frac{L_d}{2}\|\xi - \xi^\star\|^2 \leq \frac{L_d}{2\tau'}\|\nabla d(\xi)\|^2,$$

which implies

$$\frac{1}{2\tau}\|g_t\|^2 \geq \Gamma_t,$$

where $\tau = \frac{\tau'}{L_d}$. By using eq.(3.19) and the above inequality on $\|g_t\|^2$, we obtain

$$d(\xi^t) - d(\xi^{t+1}) \geq \frac{1}{2L_d}\Big(\|g_t\|^2 - \|\hat{g}_t - g_t\|^2\Big) \geq \frac{\tau}{L_d}\Gamma_t - \hat{\Delta}_t.$$

□

Since for each value of ξ^t the value and gradient of $d(\xi^t)$ need to be computed approximately by minimizing the augmented Lagrangian $D_\rho(\cdot, \xi^t)$, and since the difference between two consecutive strongly convex objectives is $D_\rho(\mu, \xi^t) - D_\rho(\mu, \xi^{t-1}) = \langle \xi^{t-1} - \xi^t, A\mu \rangle$, which is a function that converges to zero when if the sequence $\{\xi^t\}_t$ converges, a warm-restart strategy using $\hat{\mu}^t$ as the initial point to the subproblem $\max_\mu D_\rho(\mu, \xi^{t+1})$ is beneficial, as characterized by the following lemma.

Lemma 3.D.4 (Dual gap at warm start)**.** *Denote $\Delta^0_{t+1} := D_\rho(\bar{\mu}^{t+1}, \xi^{t+1}) - D_\rho(\mu^{t+1,0}, \xi^{t+1})$. If we let $\mu^{t+1,0} = \hat{\mu}^t$, then*

$$\Delta^0_{t+1} \leq (4 + \frac{2}{\omega})\hat{\Delta}_t + (1 + 2\omega)\Gamma_t, \quad \forall \omega > 0. \tag{3.20}$$

Proof. By definition, we have $D_\rho(\bar{\mu}^{t+1}, \xi^{t+1}) = D_\rho(\mu^{t+1,*}, \xi^{t+1}) = d(\xi^{t+1})$. The initial gap of μ at iteration t can then be decomposed as

$$\begin{aligned}
\Delta_{t+1}^0 &= D_\rho(\bar{\mu}^{t+1}, \xi^{t+1}) - D_\rho(\mu^{t+1,0}, \xi^{t+1}) + d(\xi^t) - d(\xi^t) - D_\rho(\hat{\mu}^t, \xi^t) + D_\rho(\hat{\mu}^t, \xi^t) \\
&= \Big[d(\xi^t) - D_\rho(\hat{\mu}^t, \xi^t)\Big] + \Big[D_\rho(\hat{\mu}^t, \xi^t) - D_\rho(\mu^{t+1,0}, \xi^{t+1})\Big] + D_\rho(\bar{\mu}^{t+1}, \xi^{t+1}) - d(\xi^t) \\
&= \Big[D_\rho(\bar{\mu}^t, \xi^t) - D_\rho(\hat{\mu}^t, \xi^t)\Big] + \Big[D_\rho(\hat{\mu}^t, \xi^t) - D_\rho(\hat{\mu}^t, \xi^{t+1})\Big] + d(\xi^{t+1}) - d(\xi^t) \\
&= \hat{\Delta}_t + \frac{1}{L_d}\|\hat{g}_t\|^2 + d(\xi^{t+1}) - d(\xi^t)
\end{aligned}$$

Again, we used the gradient step $\xi^{t+1} = \xi^t - \frac{1}{L_d}\hat{g}_t$, and recall that $A\hat{\mu}^t = \hat{g}_t$.

Now, we can bound the term $\|\hat{g}_t\|^2$ from above using the fact that

$$\begin{aligned}
\frac{1}{L_d}\|\hat{g}_t\|^2 &= \frac{1}{L_d}\Big[\|g_t\|^2 + 2\langle g_t, \hat{g}_t - g_t\rangle + \|\hat{g}_t - g_t\|^2\Big] \\
&\leq \frac{1}{L_d}\Big[(1+\omega)\|g_t\|^2 + (1+1/\omega)\|\hat{g}_t - g_t\|^2\Big],
\end{aligned}$$

where the last inequality stems from the Cauchy-Schwarz inequality $\langle g_t, \hat{g}_t - g_t\rangle \leq \|g_t\|\,\|\hat{g}_t - g_t\|$ and the fact that for any any $a, b \in \mathbb{R}$ and $\omega > 0$, we have $2ab \leq \omega a^2 + b^2/\omega$.

Combining the upper bound of $d(\xi^{t+1}) - d(\xi^t)$ from eq.(3.19), we get

$$\Delta_{t+1}^0 \leq \hat{\Delta}_t + \frac{3\omega + 2}{2\omega L_d}\|\hat{g}_t - g_t\|^2 + \frac{2\omega + 1}{2L_d}\|g_t\|^2. \tag{3.21}$$

Here, we can use again Lemma 3.D.2 and the fact that $\frac{1}{2L_d}\|g_t\|^2 \leq \Gamma_t$, which is due to the smoothness of $d(\xi)$. It follows that

$$\Delta_{t+1}^0 \leq \Big(4 + \tfrac{2}{\omega}\Big)\,\hat{\Delta}_t + \Big(1 + 2\omega\Big)\,\Gamma_t, \quad \forall \omega > 0.$$

□

3.D.3 Proof of Theorem 3.6.1

Combining Lemma 3.D.3 and 3.D.4, we now show that IDAL enjoys a linear convergence rate if we take a fixed number of inner iterations to estimate the gradient.

Theorem 3.6.1 (Linear convergence of the outer iteration)**.** *Let $\mathcal{A}$ be an algorithm that approximately solves $\max_\mu D_\rho(\mu, \xi^t)$ in the sense that*

$$\exists \beta \in (0,1), \qquad \mathbb{E}[\hat{\Delta}_t] \leq \beta\,\mathbb{E}[\Delta_t^0].$$

Then, $\exists \kappa \in (0,1)$ characterizing $d(\xi)$ and $C > 0$, such that, if $\lambda_{\max}(\beta)$ is the largest eigenvalue of the matrix

$$M(\beta) = \begin{bmatrix} 6\beta & 3\beta \\ 1 & 1-\kappa \end{bmatrix},$$

then after T_{ex} iterations of Algorithm 1 we have

$$\left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_{T_{\text{ex}}}] \\ \mathbb{E}[\Gamma_{T_{\text{ex}}}] \end{matrix} \right\| \leq C\, \lambda_{\max}(\beta)^{T_{\text{ex}}} \left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_0] \\ \mathbb{E}[\Gamma_0] \end{matrix} \right\|.$$

Proof. Note that $\Gamma_{t+1} - \Gamma_t = d(\xi^{t+1}) - d(\xi^t)$. By using Lemma 3.D.3, we have an upper bound on Γ_{t+1} in terms of Γ_t and $\hat{\Delta}_t$, namely

$$\Gamma_{t+1} \leq \hat{\Delta}_t + (1-\kappa)\,\Gamma_t \qquad \text{with} \qquad \kappa = \frac{\tau}{L_d}. \tag{3.22}$$

On the other hand, we can also derive an upper bound on $\hat{\Delta}_{t+1}$ in terms of Γ_t and $\hat{\Delta}_t$. To achieve that, we relate the inner problem with Γ_t by running the steps on μ until $\mathbb{E}[\hat{\Delta}_{t+1}] \leq (1-\pi)^{T_{\text{in}}}\mathbb{E}[\Delta^0_{t+1}] \leq \beta\,\mathbb{E}[\Delta^0_{t+1}]$, which means $T_{\text{in}} \geq \frac{\log\beta}{\log(1-\pi)}$. By Lemma 3.D.4, we have

$$\mathbb{E}[\hat{\Delta}_{t+1}] \leq \beta\,\mathbb{E}[\Delta^0_{t+1}] \leq \beta\Big(4 + \tfrac{2}{\omega}\Big)\mathbb{E}[\hat{\Delta}_t] + \beta(1+2\omega)\,\mathbb{E}[\Gamma_t]. \tag{3.23}$$

Combining eq.(3.23) and eq.(3.22), and taking expectations on both sides, we get

$$\begin{bmatrix} \mathbb{E}[\hat{\Delta}_{t+1}] \\ \mathbb{E}[\Gamma_{t+1}] \end{bmatrix} \leq M \begin{bmatrix} \mathbb{E}[\hat{\Delta}_t] \\ \mathbb{E}[\Gamma_t] \end{bmatrix} \tag{3.24}$$

Since by definition, all the elements of M are positive, we can telescope a sequence of matrix multiplications to get

$$\begin{bmatrix} \mathbb{E}[\hat{\Delta}_{T_{\text{ex}}}] \\ \mathbb{E}[\Gamma_{T_{\text{ex}}}] \end{bmatrix} \leq M \begin{bmatrix} \mathbb{E}[\hat{\Delta}_{T_{\text{ex}}-1}] \\ \mathbb{E}[\Gamma_{T_{\text{ex}}-1}] \end{bmatrix} \leq \cdots \leq M^{T_{\text{ex}}} \begin{bmatrix} \mathbb{E}[\hat{\Delta}_0] \\ \mathbb{E}[\Gamma_0] \end{bmatrix} \tag{3.25}$$

Assuming the eigen decomposition of M takes the form $M = PDP^{-1}$, then $M^t = PD^tP^{-1}$. Applying norms on both sides of the vector inequality, we have

$$\left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_{T_{\text{ex}}}] \\ \mathbb{E}[\Gamma_{T_{\text{ex}}}] \end{matrix} \right\| \leq \|P\|_{\text{op}}\, \lambda_{\max}(\beta)^{T_{\text{ex}}} \|P^{-1}\|_{\text{op}} \left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_0] \\ \mathbb{E}[\Gamma_0] \end{matrix} \right\|. \tag{3.26}$$

Note that $C = \|P\|_{\text{op}}\|P^{-1}\|_{\text{op}}$ is a constant. □

3.D.4 Proofs of Corollary 3.6.2 and Corollary 3.6.3

Corollary 3.6.2. *If $\mathcal{A}$ is a linearly convergent algorithm with rate π and if it runs for T_{in} iterations, such that, for some β: $\lambda_{\max}(\beta) < 1$, we have $(1-\pi)^{T_{\text{in}}} \leq \beta$, then $\mathbb{E}[\hat{\Delta}_t]$ and $\mathbb{E}[\Gamma_t]$ converge linearly to 0.*

Proof. If the algorithm $\mathcal{A}$ for the inner loop is linearly convergent, it implies that, after T_{in} iterations,

$$\mathbb{E}[\hat{\Delta}_t] \leq (1-\pi)^{T_{\text{in}}}\mathbb{E}[\Delta^0_t] \leq \beta\,\mathbb{E}[\Delta^0_t].$$

Then, by Theorem 3.6.1, we have

$$\left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_t] \\ \mathbb{E}[\Gamma_t] \end{matrix} \right\| \leq C\, \lambda_{\max}(\beta)^{t} \left\| \begin{matrix} \mathbb{E}[\hat{\Delta}_0] \\ \mathbb{E}[\Gamma_0] \end{matrix} \right\|.$$

As $t \to +\infty$, both $\mathbb{E}[\Delta^0_t]$ and $\mathbb{E}[\Delta^0_t]$ converge to zero linearly. □

We restate Corollary 3.6.3 in a more detailed way.

Corollary 3.D.1 (Corollary 3.6.3)**.** *Let σ denote the strong convexity constant of $\mu \mapsto D_\rho(\mu, \xi)$ and L_d the smoothness constant of d. Assume that $(\|\xi_t\|_2)_{t\in\mathbb{N}}$ is almost surely bounded by a constant B. Then the squared residuals to the constraint $A\mu = 0$ satisfy*

$$\frac{1}{2}\|A\hat{\mu}^t\|_2^2 \leq 2L_d\Gamma_t + \frac{2}{\sigma}\|A\|_{\mathrm{op}}^2\hat{\Delta}_t.$$

Furthermore, if we let $D_\infty(\mu) := \langle \ell, \mu\rangle - \gamma F_{\mathcal{I}}^*(\mu) - \frac{1}{2\lambda}\|\Psi\mu\|_2^2$, so that we have $D(\mu) = D_\infty(\mu) - \iota_{\{A\mu=0\}}$, then (given that $\mu^t \in \mathcal{I}$ throughout the algorithm) the gap between the smooth part of the objective in $\hat{\mu}^t$ and at the optimum can be bounded as follows

$$|D_\infty(\hat{\mu}^t) - D_\infty(\mu^\star)| \leq B\sqrt{2L_d\,\Gamma_t} + B\frac{\|A\|_{\mathrm{op}}}{\sqrt{\sigma}}\sqrt{2\hat{\Delta}_t} + \left(1+2\frac{L_d}{\rho}\right)\Gamma_t + \left(1+2\frac{\|A\|_{\mathrm{op}}^2}{\rho\sigma}\right)\hat{\Delta}_t.$$

Finally, if Γ_t and $\hat{\Delta}_t$ converge to 0 linearly then both the residuals $\|A\hat{\mu}^t\|_2^2$ and the gap in objective value $|D_\infty(\hat{\mu}^t) - D_\infty(\mu^\star)|$ converge to 0 linearly.

Proof. For the first inequality, by Lemma 3.D.1 we know that d is an L_d-smooth function. It is then a standard result (see e.g. Nesterov 2013, Thm 2.1.5) that we therefore have $\|\nabla d(\xi^t)\|_2^2 \leq 2L_d(d(\xi^t) - d(\xi^\star)) = 2L_d\Gamma_t$. But since $\nabla d(\xi^t) = A\bar{\mu}^t$, and using the strong convexity of $\mu \mapsto D_\rho(\mu, \xi)$, we have

$$\frac{1}{2}\|A\hat{\mu}^t\|_2^2 \leq \|A\bar{\mu}^t\|_2^2 + \|A\|_{\mathrm{op}}^2\|\bar{\mu}^t - \hat{\mu}^t\|_2^2 \leq 2L_d\Gamma_t + 2\frac{\|A\|_{\mathrm{op}}^2}{\sigma}\hat{\Delta}_t.$$

For the second inequality, by definition of D_∞, we have

$$D_\rho(\mu, \xi) := D_\infty(\mu) + \langle \xi, A\mu\rangle - \frac{1}{2\rho}\|A\mu\|_2^2, \text{ and } D_\infty(\mu^\star) = D(\mu^\star).$$

But then we have

$$\begin{aligned}|D_\infty(\hat{\mu}^t) - D_\infty(\mu^\star)| &= |D_\infty(\hat{\mu}^t) - D_\rho(\hat{\mu}^t, \xi^t)| + |D_\rho(\hat{\mu}^t, \xi^t) - D_\rho(\bar{\mu}^t, \xi^t)| \\ &\quad + |D_\rho(\bar{\mu}^t, \xi^t) - D_\infty(\mu^\star)| \\ &\leq |\langle \xi^t, A\hat{\mu}^t\rangle| + \frac{1}{2\rho}\|A\hat{\mu}^t\|_2^2 + \hat{\Delta}_t + \Gamma_t,\end{aligned}$$

and

$$|\langle \xi^t, A\hat{\mu}^t\rangle| + \frac{1}{2\rho}\|A\hat{\mu}^t\|_2^2 \leq B\|A\bar{\mu}^t\| + B\|A\|_{\mathrm{op}}\|\hat{\mu}^t - \bar{\mu}^t\|_2 + \frac{1}{\rho}(\|A\hat{\mu}^t\|_2^2 + \|A\|_{\mathrm{op}}^2\|\hat{\mu}^t - \bar{\mu}^t\|_2^2),$$

which yields the result using the same inequalities as before.

Finally, to show the implications of linear convergence, by Lemma 2.3 of Hong and Luo (2017), there exists $\tau' > 0$ such that $\|\nabla d(\xi)\| \geq \tau'\|\xi - \xi^\star\|$. So that, since $\|\nabla d(\xi^t)\|_2^2 \leq 2L_d\Gamma_t$, we have that if the sequence $(\Gamma_t)_{t\in\mathbb{N}}$ is bounded then so is $(\xi^t)_{t\in\mathbb{N}}$. Letting B be a bound on $\|\xi^t\|$ the previous statements shows the results. □

3.D.5 Proofs of Corollaries 3.6.4 and Corollary 3.6.5

Corollary 3.6.4 (Total number of inner updates)**.** *With the notations of the previous corollary, for any $\beta \in (0,1)$ such that $\lambda_{\max}(\beta) < 1$, it is possible to obtain $\mathbb{E}[\hat{\Delta}_t] \leq \epsilon$ and $\mathbb{E}[\Gamma_t] \leq \epsilon$ with a total number of inner iterations $T_{\text{tot}} := T_{\text{in}} T_{\text{ex}}$ such that*

$$T_{\text{tot}} \geq \frac{\log(\beta)}{\log \lambda_{\max}(\beta) \log(1-\pi)} \log(\epsilon).$$

Proof. To guarantee that $(1-\pi)^T_{\text{in}} < \beta$ requires that $T_{\text{in}} \geq \frac{\log(1-\pi)}{\log(\beta)}$ and to guaranteed that $\lambda_{\max}(\beta)^{T_{\text{ex}}} < \epsilon$ requires similarly that $T_{\text{ex}} \geq \frac{\log(\epsilon)}{\log(\lambda(\beta))}$. Taking the product of these inequalities yields the result. □

Corollary 3.6.5 (Convergence rate)**.** *If $\kappa < \frac{1}{2}$ and $\alpha = \frac{1}{12}$, if $T_{\text{in}} \geq \frac{\log(\alpha\kappa)}{\log(1-\pi)}$, then, there exist a constant $C' > 0$ such that after a total of $t\,T_{\text{in}} + s$ inner updates, we have*

$$\mathbb{E}[\Delta^s_t + \Gamma_t] \leq C' \Big(1 - \frac{\kappa\pi}{2\log(12/\kappa)}\Big)^{t\,T_{\text{in}}+s}.$$

Proof. Using solving the quadratic formula for the largest eigenvalue of a two-by-two matrix yields

$$\lambda_{\max}(\beta) = (1 - \kappa + 6\beta) + \sqrt{(1-\kappa-6\beta)^2 + 12\beta}.$$

It is immediate to verify that $\lambda_{\max}(\beta) < 1$ if and only if $\beta < \frac{1}{3}\frac{\kappa}{1+2\kappa}$. This shows that we need to choose $\beta = \alpha\kappa$ with $\alpha < \frac{1}{3(1+2\kappa)}$. So in particular, if $\alpha < \frac{1}{9}$, then the previous inequality is satisfied for any $0 < \kappa < 1$.

Moreover, if $\kappa \leq \frac{1}{2}$ and $\alpha < \frac{1}{6}$, we have $\lambda_{\max}(\beta) = \lambda_{\max}(\alpha\kappa) < 1 - \kappa(1-6\alpha)$. Indeed, letting $x = 3\beta$, and $\alpha' = 3\alpha$, we have

$$\begin{aligned}
2\lambda_{\max}(\beta) &= (1 - \kappa + 2x) + \sqrt{(1-\kappa-2x)^2 + 4x} \\
&= 1 - \kappa + 2x + \sqrt{(1-\kappa)^2 + 4x\kappa + 4x^2} \\
&= 1 - \kappa + 2\alpha'\kappa + \sqrt{(1-\kappa)^2 + 4\alpha'\kappa^2 + 4\alpha'^2\kappa^2} \\
&\leq 1 - \kappa + 2\alpha'\kappa + \sqrt{(1-\kappa)^2 + 4\alpha'\kappa(1-\kappa) + 4\alpha'^2\kappa^2} \\
&\leq 2\,(1 - \kappa + 6\alpha\kappa).
\end{aligned}$$

Setting $\alpha = \frac{1}{12}$, given that the rate r is $r = 1 - \exp\Big(\frac{\log(1-\pi)\log(\lambda_{\max}(\beta))}{\log(\beta)}\Big)$, we have

$$r \geq 1 - (1-\pi)^{\frac{\log(1-\frac{\kappa}{2})}{\log(\frac{\kappa}{12})}} \geq \frac{\log(1-\frac{\kappa}{2})}{\log(\frac{\kappa}{12})}\,\pi \geq \frac{\kappa}{-2\log(\frac{\kappa}{12})}\,\pi,$$

where, for the second and the third inequality, we used the fact that $\log(1-z) \geq z$ respectively for $z = \pi$ and for $z = -\frac{\kappa}{2}$. □

3.E Convergence results with SDCA

In this section, we specify the detailed form of $D_\rho(\mu, \xi)$, and show how to apply the proof scheme of Shalev-Shwartz and Zhang (2016) to SDCA for the maximization of $D_\rho(\mu, \xi)$ w.r.t. μ in order to prove Proposition 1. We first write a fully decomposed expression of $D_\rho(\mu, \xi)$. We have:

$$D_\rho(\mu, \xi) = \sum_{c \in \mathcal{C}} \langle \ell_c, \mu_c \rangle - f_c^*(\mu_c) - \frac{1}{2\lambda} \sum_{\tau \in \mathcal{T}} \Big\| \sum_{c \in \mathcal{C}_\tau} \Psi_c \mu_c \Big\|^2 - \frac{1}{2\rho} \sum_{\substack{e \in \mathcal{E} \\ i \in e}} \|\mu_i - A_i \mu_e\|^2 + \sum_{\substack{e \in \mathcal{E} \\ i \in e}} \langle \xi_{ei}, \mu_i - A_i \mu_e \rangle, \tag{3.27}$$

where $-f_c^*(\mu_c) = \gamma h_c(\mu_c) - \iota_{\triangle_c}(\mu_c)$.

We assume here that the entropy surrogate used is such that h_c is σ_c-strongly concave w.r.t. μ_c.

In particular this corresponds to two possible choices:

- The naive Gini entropy, for which $h_c(\mu_c) = (1 - \|\mu_c\|_2^2)$.
- The Gini-OTRW entropy (see Appendix 3.C) for which, given positive numbers $\rho_i, \rho_{i|j}$ and $\rho_{j|i}$ for all nodes and edges, we have
  - $h_i(\mu_i) = \rho_i k_i (1 - \|\mu_i\|_2^2) \quad$ for $i \in \mathcal{V}$
  - $h_{ij}(\mu_{ij}) = h_{i|j}(\mu_{ij}) + h_{j|i}(\mu_{ij}) \quad$ for $\{i, j\} \in \mathcal{E} \quad$ with $h_{i|j}(\mu_{ij}) = k_i \rho_{i|j} (\varepsilon \|A_j \mu_{ij}\|_2^2 - k_j \|\mu_{ij}\|_2^2)$

  for $\varepsilon < 1$ which is σ_c-strongly concave in μ_c with $\sigma_i = 2 k_i \rho_i$ if $i \in \mathcal{V}$ else $\sigma_{\{i,j\}} = 2(1 - \varepsilon) k_i k_j (\rho_{t ij} + \rho_{t ji})$. (For $\varepsilon = 1$, the surrogate is not strongly concave, and a modification of the decomposition into a separable terms and a smooth term must be used to leverage strong convexity: see the discussion in Section 6.2 after Proposition 2).

The proof of convergence for SDCA is based on showing that the expected increase in dual objective provides an upper bound on a measure of duality gap. For the problem, we are considering the gap of interest is $\text{gap}(w, \delta, \mu, \xi) := \tilde{P}_\rho(w, \delta, \xi) - D_\rho(\mu, \xi)$, which is an upper bound on the duality gap $P_\rho(w, \xi) - D_\rho(\mu, \xi)$. It can

be decomposed as follows:

$$\begin{aligned}
\mathrm{gap}(w,\delta,\mu,\xi) = \;& \Big[\gamma F_{\mathcal{I}}(\ell + \Psi^\intercal w + A^\intercal\delta) + \gamma F_{\mathcal{I}}^*(\mu) - \langle \ell + \Psi^\intercal w + A^\intercal\delta, \mu\rangle\Big] \\
& + \Big[\frac{\lambda}{2}\|w\|^2 + \frac{1}{2\lambda}\|\Psi\mu\|^2 - \langle -w, \Psi\mu\rangle\Big] \\
& + \Big[\frac{\rho}{2}\|\xi-\delta\|^2 + \frac{1}{2\rho}\|A\mu\|^2 - \langle \xi-\delta, A\mu\rangle\Big] \\
= \;& \Big[\sum_{c\in\mathcal{C}} f_c^*\big(\tfrac{1}{\gamma}\tilde{\theta}_c(w,\delta)\big) + f_c^*(\mu_c) - \langle \tilde{\theta}_c(w,\delta), \mu_c\rangle\Big] \\
& + \Big[\sum_{\tau\in\mathcal{T}} \frac{\lambda}{2}\|w_\tau\|^2 + \frac{1}{2\lambda}\|\sum_{c\in\mathcal{C}_\tau}\Psi_c\mu_c\|^2 - \langle -w_\tau, \sum_{c\in\mathcal{C}_\tau}\Psi_c\mu_c\rangle\Big] \\
& + \Big[\sum_{e\in\mathcal{E}}\sum_{i\in e}\frac{\rho}{2}\|\xi_{ei}-\delta_{ei}\|^2 + \frac{1}{2\rho}\|\mu_i - A_i\mu_i\|^2 - \langle \xi_{ei}-\delta_{ei}, \mu_i - A_i\mu_i\rangle\Big],
\end{aligned} \tag{3.28}$$

where $\tilde{\theta}_c$ is defined by

$$\tilde{\theta}_c(w,\delta) := \begin{cases} \ell_i + \Psi_i^\intercal w_{\tau_i} + \sum_{e\ni s}\delta_{ei} & \text{for } c = i \in \mathcal{V}, \\ \ell_e + \Psi_e^\intercal w_{\tau_e} - \sum_{i\in e} A_i^\intercal\delta_{ei} & \text{for } c = e \in \mathcal{E}. \end{cases} \tag{3.29}$$

We now proceed to characterize the progress of the algorithm at each iteration, and to that end, we introduce appropriate notations. In particular, since ξ is fixed during the algorithm, we drop the dependance on ξ in different functions: Denote the objective of the subproblem w.r.t. clique c as

$$D_{\rho,c}(\mu_c, \mu^s_{-c}) := -f_c^*(\mu_c) - r(\mu_c, \mu^s_{-c}), \tag{3.30}$$

with r defined by

$$\begin{aligned}
r(\mu_c, \mu^s_{-c}) := \;& \frac{1}{2\lambda}\Big\|\sum_{b\in\mathcal{C}_{\tau_c}\setminus\{c\}}\Psi_b\mu_b^s + \Psi_c\mu_c\Big\|^2 \\
& + \begin{cases} \sum_{e\ni c}\frac{1}{2\rho}\|\mu_i - A_i\mu_e^s\|^2 - \langle \mu_i, \sum_{e\ni i}\xi_{ei} + \ell_i\rangle, & c = i \in \mathcal{V}, \\ \sum_{i\in e}\frac{1}{2\rho}\|\mu_i^s - A_i\mu_e\|^2 - \langle \mu_e, -\sum_{i\in e}A_i^\intercal\xi_{ei} + \ell_e\rangle, & c = e \in \mathcal{E}. \end{cases}
\end{aligned}$$

It is straightforward to show that r is convex and smooth with cliquewise smoothness constants

$$\begin{aligned}
L_i &= \frac{1}{\lambda}\mathrm{eig}_{\max}(\Psi_i^\intercal\Psi_i) + \frac{|\{e : e\ni i\}|}{\rho},\ i\in\mathcal{V} \\
L_e &= \frac{1}{\lambda}\mathrm{eig}_{\max}(\Psi_e^\intercal\Psi_e) + \frac{1}{\rho}\sum_{i\in e}\mathrm{eig}_{\max}(A_i^\intercal A_i),\ e\in\mathcal{E}.
\end{aligned}$$

The proof of convergence hinges on the following key lemma.

Lemma 3.E.1. *Taking one of the following updates on μ_c with μ_{-c} fixed:*

- $\mu_c^{s+1} = \arg\max_{\mu_c} D_{\rho,c}(\mu_c, \mu_{-c}^s)$.
- *or, if $u \in \partial f_c(\tilde{\theta}_c(w^s, \delta^s))$, where f_c is the conjugate function of f_c^*,*

  solve $\hat{\alpha} = \arg\max_{\alpha \in [0,1]} D_{\rho,c}(\mu_c^s + \alpha(u - \mu_c^s); \mu^s)$, *and set* $\mu_c^{s+1} = \mu_c^s + \hat{\alpha}(u - \mu_c^s)$.

Then, with $\pi = \min_{c \in \mathcal{C}} \frac{\sigma_c}{|\mathcal{C}|(\sigma_c + L_c)}$, the following inequality holds

$$\mathbb{E}_c[D_\rho(\mu^{s+1}, \xi) - D_\rho(\mu^s, \xi)] \geq \pi\, \mathbb{E}_c[\tilde{P}_\rho(w^s, \delta^s, \xi) - D_\rho(\mu^s, \xi)], \quad \forall \xi,$$

where w^s, δ^s are updated to maintain the optimality conditions:

$$w^s = -\frac{1}{\lambda}\Psi\mu^s, \quad \delta^s = \xi - \frac{1}{\rho}A\mu^s.$$

Proof. Consider the following quantity:

$$\breve{D}_{\rho,c}(\mu_c; \mu^s) := -f_c^*(\mu_c) - r(\mu^s) - \langle \nabla_{\mu_c} r(\mu^s), \mu_c - \mu_c^s \rangle - \frac{L_c}{2}\|\mu_c - \mu_c^s\|^2.$$

We have $\breve{D}_{\rho,c}(\mu_c; \mu^s) \leq D_{\rho,c}(\mu_c; \mu_{-c}^s)$, since $\mu_c \mapsto r(\mu_c, \mu_{-c}^s)$ is L_c-smooth.

First, for the update $\mu_c^{s+1} = \arg\max_{\mu_c} D_{\rho,c}(\mu_c, \mu_{-c}^s)$, we have that, for any direction $u - \mu_c^s$ and any step size $\alpha \in [0, 1]$

$$\begin{aligned} D_\rho(\mu^{s+1}, \xi) - D_\rho(\mu^s, \xi) &= D_{\rho,c}(\mu_c^{s+1}, \mu_{-c}^s) - D_{\rho,c}(\mu_c^s, \mu_{-c}^s) \\ &\geq D_{\rho,c}(\mu_c^s + \alpha(u - \mu_c^s), \mu_{-c}^s) - D_{\rho,c}(\mu_c^s, \mu_{-c}^s) \\ &\geq \breve{D}_{\rho,c}(\mu_c^s + \alpha(u - \mu_c^s); \mu^s) - D_{\rho,c}(\mu_c^s, \mu_{-c}^s). \end{aligned} \tag{3.31}$$

Showing the desired inequality for the second form of update thus implies the inequality for the first type of update. Expliciting $\breve{D}_{\rho,c}(\mu_c^s + \alpha(u - \mu_c^s); \mu^s)$, we have

$$\begin{aligned} \breve{D}_{\rho,c}(\mu_c^s + \alpha(u - \mu_c^s); \mu^s) = &- f_c^*(\mu_c^s + \alpha(u - \mu_c^s)) \\ &- r(\mu^s) - \langle \nabla_{\mu_c} r(\mu^s), \alpha(u - \mu_c^s) \rangle - \frac{\alpha^2 L_c}{2}\|u - \mu_c^s\|^2. \end{aligned} \tag{3.32}$$

Since $f_c^*(u)$ assumed σ_c-strongly convex, we have

$$f_c^*(\mu_c^s + \alpha(u - \mu_c^s)) \leq \alpha f_c^*(u) + (1 - \alpha) f_c^*(\mu_c^s) - \frac{\sigma_c}{2}\alpha(1 - \alpha)\|u - \mu_c^s\|_2^2. \tag{3.33}$$

Combining eq.(3.32) and eq.(3.33), we obtain

$$\begin{aligned} \breve{D}_{\rho,c}(\mu_c^s + \alpha(u - \mu_c^s); \mu^s) \geq &-\alpha\Big(f_c^*(u) - f_c^*(\mu_c^s) + \langle \nabla_{\mu_c} r(\mu^s), u - \mu_c^s \rangle\Big) \\ &- f_c^*(\mu_c^s) - r(\mu^s) + \Big(\frac{\sigma_c}{2}\alpha(1 - \alpha) - \frac{\alpha^2 L_c}{2}\Big)\|u - \mu_c^s\|^2. \end{aligned} \tag{3.34}$$

Now, if we choose $u \in \partial f_c\Big(-\nabla_{\mu_c} r(\mu^s)\Big)$, by Fenchel conjugacy, it follows that

$$f_c\Big(-\nabla_{\mu_c} r(\mu^s)\Big) = -f_c^*(u) - \langle \nabla_{\mu_c} r(\mu^s), u\rangle.$$

One can easily see that $\tilde{\theta}_c(w^s, \delta^s) = -\nabla_{\mu_c} r(\mu^s)$ by maintaining the optimality conditions

$$\forall c \in \mathcal{C}\colon w^s_{\tau_c} = -\frac{1}{\lambda}\sum_{b\in\mathcal{C}_\tau} \Psi_b \mu^s_b,$$
$$\forall e \in \mathcal{E}, i \in e\colon \delta^s_{ei} = \xi_{ei} - \frac{1}{\rho}(\mu^s_i - A_i \mu^s_e).$$

Thus, we can further simplify eq.(3.34) as

$$\begin{aligned}\breve{D}_{\rho,c}(\mu^s_c + \alpha(u-\mu^s_c); \mu^s) &\geq \alpha\Big(f_c(\tilde{\theta}_c(w^s,\delta^s)) + f_c^*(\mu^s_c) - \langle \tilde{\theta}_c(w^s,\delta^s), \mu^s_c\rangle\Big)\\ &\quad + D_{\rho,c}(\mu^s_c, \mu^s_{-c}) + \Big(\frac{\sigma_c}{2}\alpha(1-\alpha) - \frac{\alpha^2 L_c}{2}\Big)\|u-\mu^s_c\|^2\\ &\geq D_{\rho,c}(\mu^s_c, \mu^s_{-c}) + \alpha\Big(f_c(\tilde{\theta}_c(w^s,\delta^s)) + f_c^*(\mu^s_c) - \langle \tilde{\theta}_c(w^s,\delta^s), \mu^s_c\rangle\Big), \end{aligned} \tag{3.35}$$

provided that $\frac{\sigma_c}{2}\alpha(1-\alpha) - \frac{\alpha^2 L_c}{2} \geq 0$, that is, $0 \leq \alpha \leq \frac{\sigma_c}{\sigma_c + L_c}$.

The key observation is that

$$\mathrm{gap}(w,\delta,\mu,\xi) = \sum_{c\in\mathcal{C}} f_c(\tilde{\theta}_c(w,\delta)) + f_c^*(\mu_c) - \langle \tilde{\theta}_c(w,\delta), \mu_c\rangle \tag{3.36}$$

if we maintain the optimality conditions. By using eq.(3.36) and taking expectation $\mathbb{E}_c$ w.r.t. a uniform random choice of the clique c on both sides of eq.(3.35), we guarantee that, for $\alpha \in [0, \frac{\sigma_c}{\sigma_c+L_c}]$,

$$\begin{aligned}\mathbb{E}_c\Big[\frac{\alpha}{|\mathcal{C}|}\mathrm{gap}(w^s,\delta^s,\mu^s,\xi)\Big] &\leq \mathbb{E}_c\Big[\breve{D}_{\rho,c}(\mu^s_c + \alpha(u-\mu^s_c);\mu^s) - D_{\rho,c}(\mu^s_c,\mu^s_{-c})\Big]\\ &\leq \mathbb{E}_c[D_\rho(\mu^{s+1},\xi) - D_\rho(\mu^s,\xi)].\end{aligned}$$

So, we can choose the maximum value $\frac{\sigma_c}{\sigma_c+L_c}$ for α. It follows that

$$\mathbb{E}_c[D_\rho(\mu^{s+1},\xi) - D_\rho(\mu^s,\xi)] \geq \Big(\min_{c\in\mathcal{C}} \frac{\sigma_c}{|\mathcal{C}|(\sigma_c+L_c)}\Big)\,\mathbb{E}_c[\,\mathrm{gap}(w^s,\delta^s,\mu^s,\xi)],$$

as desired. □

We can now prove Proposition 3.6.6.

Proposition 3.6.6. *If $\mathcal{A}$ is SDCA, let $|\mathcal{C}|$ be the total number of cliques, σ_c the strong convexity constant of f_c^*, and L_c the Lipschitz constant of $\mu_c \mapsto r(\mu)$, then $\mathcal{A}$ is linearly convergent with rate $\pi = \min_{c\in\mathcal{C}} \frac{\sigma_c}{|\mathcal{C}|(\sigma_c+L_c)}$.*

Proof. Denote $\Delta_t^s := D_\rho(\bar{\mu}^t, \xi^t) - D_\rho(\mu^{t,s}, \xi^t)$. Since we update $\mu^{t,s}$ to $\mu^{t,s+1}$ using SDCA, according to Lemma 3.E.1, we have

$$\begin{aligned}\mathbb{E}_c[\Delta_t^s - \Delta_t^{s+1}] &= \mathbb{E}_c[D_\rho(\mu^{t,s+1}, \xi^t) - D_\rho(\mu^{t,s}, \xi^t)] \\ &\geq \pi\, \mathbb{E}_c[\tilde{P}_\rho(w(\mu^{t,s}), \delta(\mu^{t,s}, \xi^t), \xi^t) - D_\rho(\mu^{t,s}, \xi^t)] \\ &\geq \pi\, \mathbb{E}_c[D_\rho(\bar{\mu}^t, \xi^t) - D_\rho(\mu^{t,s}, \xi^t)] = \pi \mathbb{E}_c[\Delta_t^s],\end{aligned}$$

and $\pi = \min_{c\in\mathcal{C}} \frac{\sigma_c}{|\mathcal{C}|(\sigma_c + L_c)}$. The above inequality implies that

$$\mathbb{E}_c[\Delta_t^{s+1}] \leq (1-\pi)\, \mathbb{E}_c[\Delta_t^s] \leq (1-\pi)^{s+1}\, \mathbb{E}_c[\Delta_t^0].$$

The result follows if we set $T_{\text{in}} = s + 1$. □

3.E.1 Proof of Propositions 3.6.7 and 3.6.8

Proposition 3.6.7. *Let $\hat{w}^t = w(\hat{\mu}^t)$. If $\mathcal{A}$ is SDCA, then*

$$\mathbb{E}[P(\hat{w}^t) - P(w^\star)] \leq \frac{1}{\pi}\mathbb{E}[\hat{\Delta}_t] + \mathbb{E}[\Gamma_t].$$

Proof. Recall that $P(w^\star) = D(\mu^\star) = D_\rho(\mu^\star, \xi^\star)$ by Corollary 3.B.4.

$$\begin{aligned}P(w^{t,s}) - P(w^\star) &= P(w^{t,s}) - D_\rho(\bar{\mu}^t, \xi^t) + D_\rho(\bar{\mu}^t, \xi^t) - P(w^\star) \\ &= P(w^{t,s}) - D_\rho(\bar{\mu}^t, \xi^t) + D_\rho(\bar{\mu}^t, \xi^t) - D_\rho(\mu^\star, \xi^\star) \\ &\leq \tilde{P}(w^{t,s}, \delta^{t,s}, \xi^t) - D_\rho(\bar{\mu}^t, \xi^t) + d(\xi^t) - d(\xi^\star) \\ &\leq \tilde{P}(w^{t,s}, \delta^{t,s}, \xi^t) - D_\rho(\mu^{t,s}, \xi^t) + d(\xi^t) - d(\xi^\star) \\ &= \operatorname{gap}(w^{t,s}, \delta^{t,s}, \mu^{t,s}, \xi^t) + \Gamma_t\end{aligned}$$

If $\mathcal{A}$ is SDCA, by Lemma 3.E.1, we have

$$\begin{aligned}\mathbb{E}[P(w^{t,s}) - P(w^\star)] &= \mathbb{E}[\operatorname{gap}(w^{t,s}, \delta^{t,s}, \mu^{t,s}, \xi^t) + \Gamma_t] \\ &\leq \mathbb{E}\Big[\frac{1}{\pi}(\Delta_t^s - \Delta_t^{s+1}) + \Gamma_t\Big] \\ &\leq \frac{1}{\pi}\mathbb{E}[\Delta_t^s] + \mathbb{E}[\Gamma_t].\end{aligned}$$

Given that $\hat{\Delta}_t = \Delta_t^{T_{\text{in}}}$, the result follows by setting $s = T_{\text{in}}$. □

Proposition 3.6.8 (Linear convergence in the primal)**.** *Let $w^{t,s} = w(\mu^{t,s})$. If $\mathcal{A}$ is a linearly convergent algorithm and the function $\mu \mapsto -H_{\text{approx}} + \frac{1}{2\rho}\|A\mu\|_2^2$ is strongly convex, then $P(w^{t,s}) - P(w^\star)$ converges to 0 linearly.*

Proof. Note that, if σ is the strong convexity constant of D_ρ w.r.t. μ, then given that $P_\rho(w, \xi) = \min_\delta \tilde{P}_\rho(w, \delta, \xi)$ with

$$\tilde{P}_\rho(w, \delta, \xi) = \gamma F_{\mathcal{I}}\Big(\frac{\theta(w) + A^\intercal\delta}{\gamma}\Big) + \frac{\rho}{2}\|\delta - \xi\|^2 + \frac{\lambda}{2}\|w\|_2^2,$$

we also have

$$P_\rho(w,\xi) = \max_\mu \left[\langle \mu, \Psi^\intercal w \rangle + \gamma H_{\text{approx}}(\mu) + \langle \xi, A\mu \rangle - \frac{1}{2\rho}\|A\mu\|_2^2 \right] + \frac{\lambda}{2}\|w\|_2^2,$$

which shows that $w \mapsto P_\rho(w,\xi)$ is a function with Lipschitz gradient as the sum of $w \mapsto \frac{\lambda}{2}\|w\|_2^2$ and of the Fenchel conjugate of a strongly convex function. Let L_P be its Lipschitz smoothness constant and note that it does not depend on the value of ξ. We thus have

$$P_\rho(w^{t,s},\xi^t) - P_\rho(\bar{w}^t,\xi^t) \leq L_P \|w^{t,s} - \bar{w}^t\|_2^2.$$

Then given the representer theorem, and by strong convexity of $\mu \mapsto D_\rho(\mu,\xi)$ we have

$$\|w^{t,s} - \bar{w}^t\|_2^2 = \|\Psi(\mu^{t,s} - \bar{\mu}^t)\|_2^2 \leq \frac{1}{\sigma}\|\Psi\|_{\text{op}}^2 (D_\rho(\mu^{t,s},\xi^t) - D_\rho(\bar{\mu}^t,\xi^t))$$

So that, since $P(w^{t,s}) \leq P_\rho(w^{t,s},\xi^t)$ and $P(w^\star) = P_\rho(w^\star,\xi^\star)$, we have

$$P(w^{t,s}) - P(w^\star) \leq P_\rho(w^{t,s},\xi^t) - P_\rho(\bar{w}^t,\xi^t) + P_\rho(\bar{w}^t,\xi^t) - P_\rho(w^\star,\xi^\star) \leq \frac{L_P}{\sigma}\|\Psi\|_{\text{op}}^2 \Delta_t^s + \Gamma_t.$$

Finally, global linear convergence in the primal also follows from the linear convergence of $\hat{\Delta}_t$ and Γ_t. □

3.F Notation summary

Given the number of notations in the main paper, we summarize some of them in Tables 3.F.1,3.F.2 and 3.F.3. The block matrices Ψ and A are schematically drawn below to illustrate their structure.

$$\Psi = \begin{array}{c} \\ \tau_c \end{array} \begin{array}{c} \begin{array}{cc} & c \end{array} \\ \begin{bmatrix} \cdots & \vdots \\ & \Psi_c \\ & \end{bmatrix} \end{array} \qquad A = \begin{array}{c} \\ ij \end{array} \begin{array}{c} \begin{array}{cc} & i \end{array} \\ \begin{bmatrix} \cdots & \vdots \\ & I_{k_i} \\ & \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{c} ij \end{array} \\ \begin{bmatrix} \vdots \\ -A_i \\ \end{bmatrix} \end{array}$$

Table 3.F.1: Notations for sets

Notation	Dimension	Description
$\mathcal{C}$		the set of cliques
$\mathcal{E}$		the set of edges
$\mathcal{V}$		the set of nodes
$\mathcal{Y}_i = \mathcal{S}_k$		$\mathcal{S}_k := \{u \in \{0,1\}^k \mid \|u\|_1 = 1\}$
$\mathcal{Y}_c$	$\prod_{i \in c} k_i$	$\mathcal{Y}_c := \bigtimes_{i \in c} \mathcal{Y}_i$
$\mathcal{Y}$	$\prod_{i \in V} k_i$	$\mathcal{Y} := \bigtimes_{i \in V} \mathcal{Y}_i$
$\mathcal{T}$		the set of clique types
$\mathcal{C}_\tau$		the set of cliques of type τ
$\mathcal{M}$		the marginal polytope
$\mathcal{L}$		the local polytope
$\mathcal{I}$		$\mathcal{I} := \prod_{i \in V} \Delta_{k_i} \times \prod_{e \in E} \Delta_{k_e}$

Table 3.F.2: Notations for variables, parameters and functions

Notation	Domain	Description
k_i	$\mathbb{N}$	the number of values that Y_i can take
k_c	$\mathbb{N}$	$k_c := \|\mathcal{Y}_c\| = \prod_{i \in c} k_i$
τ	$\mathbb{N}$	the type of a clique
τ_c	$\mathbb{N}$	the type of clique c
w_τ	$\mathbb{R}^{d_\tau}$	the parameter shared by all cliques with type τ
w	$\mathbb{R}^{\sum_\tau d_\tau}$	$w := [w_\tau]_{\tau \in \mathcal{T}}$
$\phi_c(x, y_c)$	$\mathbb{R}^{d_{\tau_c}}$	the feature vector associated with the clique c given $Y_c = y_c$
$Z(x, w)$	$\mathbb{R}_{+*}$	the partition function of $p(y\|x; w)$
$\ell_c(y_c^{(n)}, y_c)$	$\mathbb{R}_+$	the user defined loss function associated with the clique c
γ	$(0, +\infty)$	the temperature parameter of the loss-augmented CRF
$\Psi_c^{(n)}$	$\mathbb{R}^{d_{\tau_c} \times k_c}$	$\Psi_c^{(n)} := \left[\phi_c(x^{(n)}, y_c) - \phi_c(x^{(n)}, y_c^{(n)})\right]_{y_c \in \mathcal{Y}_c}$
$\Psi^{(n)}$	$\mathbb{R}^{\sum_\tau d_\tau \times \sum_c k_c}$	see the drawing
$\ell_c^{(n)}$	$\mathbb{R}^{k_c}$	$\ell_c^{(n)} := \left[\ell_c(y_c^{(n)}, y_c)\right]_{y_c \in \mathcal{Y}_c}$
$\ell^{(n)}$	$\mathbb{R}^{\sum_c k_c}$	$\ell^{(n)} := [\ell_c^{(n)}]_{c \in C}$
$\theta_c^{(n)}(w)$	$\mathbb{R}^{k_c}$	$\theta_c^{(n)}(w) := \Psi_c^{(n)\intercal} w_{\tau_c} + \ell_c^{(n)}$
$\theta^{(n)}(w)$	$\mathbb{R}^{\sum_c k_c}$	$\theta^{(n)}(w) := [\theta_c^{(n)}(w)]_{c \in \mathcal{C}}$, the natural parameter
F	$\mathbb{R}^{\sum_c k_c} \to \mathbb{R}$	the log partition function of θ
$T(y)$	$\mathbb{R}^{\sum_c k_c}$	the sufficient statistics
μ_c	$\mathbb{R}^{k_c}$	the mean parameter associated with the clique c
μ	$\mathbb{R}^{\sum_c k_c}$	the mean parameter
F^*	$\mathbb{R}^{\sum_c k_c} \to \mathbb{R}$	the Fenchel conjugate of F
ι_C	$\mathbb{R}^{\sum_c k_c} \to \{0, +\infty\}$	the indicator function of set C
λ	$\mathbb{R}_+$	the coefficient of the regularizer
A_i	$\mathbb{R}^{k_i \times k_e}$	the matrix encoding the marginalization constraint for i in e.
A	$\mathbb{R}^{\sum_e \sum_{i \in e} k_i \times \sum_c k_c}$	see the matrix form

Table 3.F.3: Notations smoothness, strong convexity constant and related quantities

Notation	Description
L_d	the Lipschitz constant of $\nabla d(\xi)$
τ	the constant of PL inequality for $d(\xi)$
σ_c	the strong convexity constant of $\mu_c \mapsto -H_{\text{Approx}}(\mu)$
L_c	the Lipschitz constant of $\mu_c \mapsto \frac{1}{\lambda}\Psi^\intercal w(\mu) + \frac{1}{\rho}A^\intercal \delta(\mu, \xi^t)$

Chapter

4

A Survey on Over-Parameterization in Deep Learning: Compression and Generalization

Abstract

Deep neural networks are commonly over-parameterized, namely, the parameterization is redundant, and the number of parameters can be even larger than the number of training points. Over-parameterization has shown to be an interesting property for both optimization and generalization in the sense that it not only makes the optimization easier but also assists the learning to pursue a better generalization. In this chapter, we will review representative neural network architectures, state-of-the-art network compresssion techniques and the connection between the compression and the generalization of over-parametrized models. In addition, we will cover a related problem – knowledge distillation, which enables a relaxation of the training setting of over-parameterized models with an application on small-data-small-network training.
The material of this chapter is based on a literature review up to November 2018. Since the field of deep learning theory is evolving sufficiently fast, some opinions and arguments may not be valid anymore.

4.1 Introduction

Deep learning has made remarkable progress on pushing the edge of machine learning tasks such as achieving nearly human level image classification (Krizhevsky et al. 2012b, Sun et al. 2014), defeating world champions in strategy games (Silver et al. 2017, OpenAI 2017) and outscoring top humans in reading and comprehension tests (Alibaba 2018).

At the heart of deep learning, the key assumption is that these mappings can be well approximated by multilayered artificial neural networks, also known as deep neural networks (DNNs), which have been proven to be universal function approximators (Cybenko 1989a). Training DNNs with backpropagation (BP) dates back at least to 1980s (Rumelhart et al. 1986). An extensive review on the history of deep learning can be found in Schmidhuber (2015).

The recent success of deep learning should be attributed to the development of general-purpose processing on graphics processing units (GPGPU), stochastic gradient algorithms, and open-source software frameworks. With stochastic gradient descent (SGD) (Robbins and Monro 1985) or its variants such as ADAM (Kingma and Ba 2014), modern deep neural networks (DNNs) (Krizhevsky et al. 2012b, Szegedy et al. 2015b, Simonyan and Zisserman 2014a, He et al. 2016a) are capable of training on potentially infinitely large datasets and achieve good performance on a wide range of problems. The DNN training with backpropagation from the outputs to the inputs is also called *end-to-end* training. In other words, the learning of the data representation and the final predictor are carried out simultaneously. However, there are still many mysteries within this end-to-end framework. They have recently drawn much attention in the community. For example,

- The optimization "puzzle": given that the learning problems of DNNs are highly non-convex, it is not clear why SGD with a proper random initialization converges to a global minima (near zero training error) with high probability (Kawaguchi 2016, Wu et al. 2017). Moreover, it has been empirically observed that over-parameterized DNNs, namely, DNNs possessing hundreds of millions of parameters, are easier to train (Hinton et al. 2012, Denil et al. 2013a).

- The generalization "puzzle": classical learning theory suggests that over-parameterized models will overfit the training data. Surprisingly, empirical observations (Zhang et al. 2016, Neyshabur et al. 2018, Sagun et al. 2017) find that, in the case of DNNs, increasing the model size often leads to better generalization performance even without explicit regularization. DNNs are able to achieve zero training error even with random labels (Zhang et al. 2016) implying that the complexity of DNNs is extremely high.

For the optimization puzzle, there are several existing works identify that the non-convex problem of DNNs has geometrically elegant structure to facilitate the optimization. Kawaguchi (2016) show that, for deep linear neural networks (namely without non-linear activations), every local minima is a global minima, and every

critical point that is not global is a saddle point. This result can be extended to non-linear cases with extra conditions, such as the independent activation assumption (Kawaguchi 2016) and the "one layer has more neurons than training points" assumption (Nguyen and Hein 2018). Although these assumptions are not always realistic (exceptions exist, e.g., Nguyen and Hein (2018)'s assumption holds on VGGNet (Simonyan and Zisserman 2014a)), they do shed light on why SGD works so well. In reality, bad local minima exists. But for over-parameterized DNNs, Soudry and Hoffer (2017) show that, under mild assumptions, the volume of the basin containing bad local minima is exponentially vanishing comparing to that of the basin containing global minima as the number of data points increases. Similar results have been reported by different authors (Soudry and Carmon 2016, Haeffele and Vidal 2017, Wu et al. 2018a 2017) towards understanding the loss lanscape of DNNs. Thus, it is very likely that the optimization puzzle is due to the special structure of DNNs, and the over-parameterization induced by increasing the depth and/or the width of a neural network does not only achieve a more powerful model but also eliminate bad local minima (Lu and Kawaguchi 2017, Li et al. 2017b).

The second puzzle reveals a more interesting property of DNNs, especially of the over-parameterized ones. However, it does not entails that any DNN with a large number of parameters will generalize. Consider the extreme case where the number of hidden units goes to infinity, as considered in convex neural networks (Bengio et al. 2006, Bach 2017). Just training the top layer, which is a convex problem, will result in zero training error, as the randomly initialized hidden layer has all possible features (Soudry and Hoffer 2017, Neyshabur et al. 2018). Obviously, such an extreme case will have serious overfitting issues. On the other hand, empirically designed DNNs often have good generalization although being heavily over-parameterized. For example, the largest wide residual network considered by Zagoruyko and Komodakis (2016c) has 500× more parameters than the size of CIFAR-10 (Krizhevsky 2009a). VGGNet has even 138M parameters comparing to 1.2M images in ImageNet (Deng et al. 2009), which is already considered as one of largest scale datasets. Thus, there must exist some implicit regularizations that are common in the network architectures or in the optimizers. Many existing works attempt to explain this phenomenon by, for example, the flat minima conjecture (Hochreiter and Schmidhuber 1997b, Keskar et al. 2016, Wu et al. 2017), the PAC-Bayes theory (Dziugaite and Roy 2017, Zhou et al. 2018, Pérez et al. 2018) and the information theory perspective (Hinton and Van Camp 1993, Tishby and Zaslavsky 2015, Achille and Soatto 2017). So far, the generalization puzzle is still one of the hottest topics in deep learning.

Although over-parameterization is important from the modeling and the optimization point of view, it is not necessarily a desired property from the computation and the deployment aspects. It is not surprising that there is a huge redundancy in DNNs. One could prune out even 99% parameters without hurting the model performance (Han et al. 2015a). Therefore, a natural question emerges: can we design a compact DNN which inherits the geometry and generalization properties of its over-parameterized counterparts while only contains minimal parameters/weights

to be learned?

To answer this question, I first study the common DNN architectures in Section 4.2, the potential issues of DNNs in Section 4.3, and then review existing model compression approaches in Section 4.4. It turns out that the generalization is closely related to the compressibility of the model, which is formally characterized by different theories, such as the minimum description length (MDL) principle (Rissanen 1978, Hinton and Van Camp 1993) and the information bottleneck principle (Tishby et al. 2000, Tishby and Zaslavsky 2015), for which I briefly review in Section 4.5. Finally, in Section 4.6, I discuss the knowledge distillation method, which offers a new but probably more straightforward way to mitigate over-parameterization.

4.2 Deep Network Architectures

A neural network is a special composite function of the input signal in the sense that the functions are connected in a way resembling the network of biological neurons. A function in the neural network is sometimes referred as a (hidden) *layer*. Obviously the input and the output are the first and the last layers respectively.

The computation from the input to the output is called the *forward pass*. Accordingly, the *backward pass* occurs when we try to compute the derivatives with respect to the layers as well as the parameters associated to the functions, which is also known as the *backpropagation*– an application of chain rule.

The first neural network, called the *threshold logic*, was proposed by McCulloch and Pitts (1943). Back to the time when backpropagation firstly gained recognition (Werbos 1974, Rumelhart et al. 1986), the *multilayer perceptron* (MLP) was one of the first neural networks considered in supervised learning (Cybenko 1989a), which includes alternatively linear functions and non-linear functions such as sigmoid, tanh, rectified linear unit (ReLU) and so on.

For specific purposes, different types of architecture are proposed. The most common type is called *feedforward neural network*, such as the aforementioned MLP. On the other hand, if there exists a function in the network that the previous outputs from itself are used as inputs, the neural network is called *recurrent neural network* (Rumelhart et al. 1986), which has an advantage in modeling arbitrarily long sequences.

The recent development of neural networks is mainly focused on *convolutional neural networks* (CNNs) (LeCun et al. 1998) due to their big success in image classification (Krizhevsky et al. 2012b). CNNs were designed for images, inspired by the structure of visual cortex, but they have been extended to other kinds of data (Oord et al. 2016). Convolutional neural networks make use of the spatial information in the input signals to enable weight sharing, which makes it invariant to translations of the input. Besides, weight sharing significantly reduces the number of parameters to be learned compared to fully connected neural networks (e.g. MLPs) with the same input-output dimensions. We list several commonly used CNN architectures in the following, and summarize their characteristics in

CNN	#parameters			computation		
	size(M)	Conv%	FC%	FLOPS(G)	Conv%	FC%
LeNet-5	0.431	6.0	94.0	0.0066	87.8	12.2
AlexNet	61	3.8	96.2	0.72	91.9	8.1
VGGNet	138	6.3	93.7	2.6	96.3	3.7
NIN	7.6	100	0	1.1	100	0
GoogLeNet	6.9	85.1	14.9	1.6	99.9	0.1
ResNet-18	5.6	100	0	1.8	100	0
ResNet-50	12.2	100	0	3.8	100	0
ResNet-101	21.2	100	0	7.6	100	0

Table 1: The computation and size of common convolutional neural networks.

terms of the number of parameters and their computational costs (measured by FLOPS) in Table 1.

- The LeNet proposed by LeCun et al. (1998) is one of the first convolutional neural networks. The architecture starts with two convolutional layers, each of which followed by a subsampling layer, to build up a task specific representation from the input. Then, there are two fully connected (FC) layers taking the representation as input to output the final classification scores/probabilities.

- AlexNet (Krizhevsky et al. 2012b) can be seen as a deeper and wider extension of the LeNet, which leads to a significant improvement in the accuracy of image classification. It was considered as the "return" of deep learning. There are other key differences between AlexNet and LeNet. Rather than using saturated nonlinear activations, such as sigmoid function or tanh function, AlexNet uses the *rectified linear units* (ReLU) to mitigate the vanishing gradient problem (Bengio et al. 1994). Besides, it also replaces average-pooling operations with max-pooling operations in subsampling layers in order to avoid the blur effects. Apart from the architecture differences, it is one of the first examples showing that dropout (Hinton et al. 2012) can be used after linear or convolutional layers to avoid overfitting in overparameterized networks. AlexNet has 61 millions parameters across five convolutional layers and three fully connected layers. The spatial resolution is reduced three times before the full connected layers.

- VGGNet (Simonyan and Zisserman 2014a) is an even larger variant of AlexNet with 13 convolutional layers and 5 max-pooling layers. It is the first neural network using small convolution filters (e.g. 3×3 in spatial dimensions), which is based on an empirical observation that large *receptive*

fields [1] can be preserved by stacking multiple small convolutions. However, VGGNet has 138 millions of parameters across 13 convolutional layers and 3 fully connected layers in order to achieve a deep architecture, which makes it one of the largest architectures in the literature.

- The network in network (NIN) (Lin et al. 2013a) starts a trend of fully convolutional neural networks (Springenberg et al. 2014, Szegedy et al. 2015b, He et al. 2016a). It is easy to verify that a fully connected layer with input dimension $C \cdot H \cdot W$ and output dimension K is equivalent to a convolutional layer including K convolutional kernels of shape $C \times H \times W$. Thus, fully connected layers can be implemented by convolutions unless there are computational concerns. A big advantage is that fully convolutional CNNs do not require inputs to have uniform spatial dimensions. If the input tensor is $C \times H \times W$, we get a $K \times 1 \times 1$ output tensor. When the width or the height of the input tensor is larger, the output tensor has spatial dimensions larger than 1. To obtain a $K \times 1 \times 1$ output tensor, the *global average-pooling* over the spatial dimensions is introduced. NIN sheds light on the special case of convolution when $H = W = 1$ (i.e. 1×1 convolution) can be seen as applying MLP on each bin of the input tensor whose shape is $C \times 1 \times 1$. This insight has been very influential on recent neural network architectures.

- GoogLeNet (Szegedy et al. 2015b) was one of the first attempts to reduce the computational overhead without sacrificing the performance. The *inception* module (a.k.a. the bottleneck layer) was first introduced in GoogLeNet. Inspired by NIN, GoogLeNet uses 1×1 convolutions to reduce 4 times the number of input channels before going through an computationally expensive convolutional layer. This saved a large number of operations while maintaining the performance. There are several extensions of GoogLeNet, such as Inception V2 (Ioffe and Szegedy 2015b) and V3 (Szegedy et al. 2016). Apart from the architectural differences, the *batch normalization* (BN) is introduced as an alternative regularizer to weight decay, dropout etc. BN accelerates the training by reducing internal covariate shifts. It also alleviates the problems of vanishing/exploding gradients.

- ResNet (He et al. 2016a) inherits the best elements of NIN and GoogLeNet, which is now one of the most commonly used architectures. The key ingredient introduced in this architecture is the *skip connection*, which further alleviates the issues of vanishing/exploding gradients. Moreover, it is shown by Orhan and Pitkow (2017) that the skip connections eliminate saddle points thus the training of ResNet can potentially converge to a better local minima. As a result, He et al. (2016a) demonstrated that there is no big technical problems on training a neural network with a thousand layers. Zagoruyko

[1] Informally, it is the region in the image that involves in the computation of a neuron.

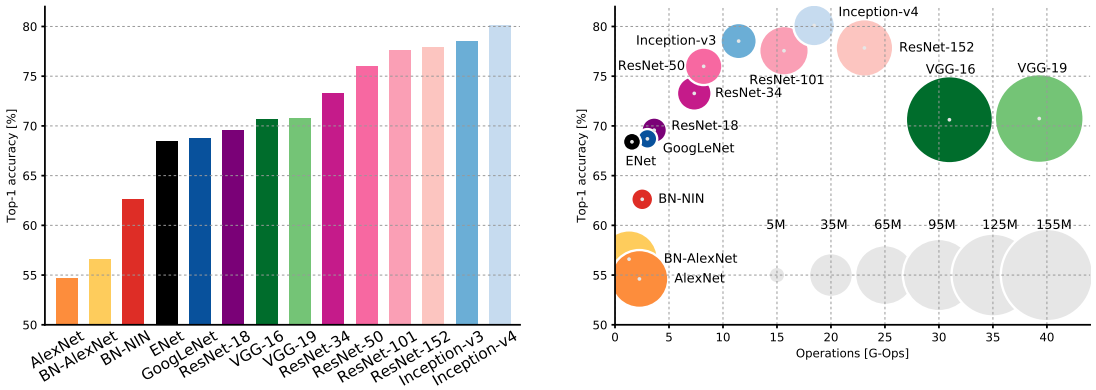


Figure 1: Left: The comparison of top-1 accuracy on the ImageNet dataset. **Right**: Top-1 accuracy vs. giga floating point operations per second. The top-1 accuracy is calculated on the ImageNet dataset with a single forward pass. The number of parameters for a network is demonstrated by the size of the disk, which is ranged from $5M$ to $155M$. **Source**: Figure 1 & 2 in Canziani et al. (2016).

and Komodakis (2016c) studied the trade-off between increasing the number of layers and increasing the number of parameters per layer, and proposed a variant of ResNet called Wide ResNet.

For a more complete discussion on neural network architectures, we refer the reader to the survey by Canziani et al. (2016). We show in Figure 1 a summary picture taken from their paper.

4.3 Memory and Energy Issues with Over-Parameterized DNNs

From Figure 1 and Table 1, we can see that all commonly used deep neural networks contain millions of parameters, and require gigas of FLOPS for a single forward pass. This situation has been mitigated in recent architectures by replacing fully connected layers with convolutional layers and by using convolutions with small kernels. In general, fully connected layers introduce more parameters but require less FLOPS comparing to convolutional layers. However, none of the designs of neural networks can avoid over-parameterization completely, even with a good balance between the number of parameters and the FLOPS.

Inevitably, over-parameterization leads to inefficiency in memory and energy. Moreover, the resulting model is hard to interpretable. Applications such as embedded systems, internet of things and mobile phones are examples where computers have limited processors, memory and batteries. The goal of such resource limited applications is quite different from that of achieving human-level performance. Deploying deep neural networks on embedded systems (Venieris et al. 2018) has become a recent trend towards intelligent mobile computing. However, it is infeasible to design an architecture for each embedded system from scratch.

One valid solution is to adaptively compress the universal model. Specifically, according to the memory and computation requirements of a given application, we reduce the storage of network parameters or eliminate less important parameters. This is generally called *model compression* in deep learning.

Energy efficiency is another desideratum from an implementation perspective. Regarding this perspective, Max Welling shared his opinion in his keynote talk "Intelligence per kilowatthour" at ICML 2018 (Welling 2018):

> *In the 19th century the world was revolutionized because we could transform energy into useful work. The 21st century is revolutionized due to our ability to transform information (or data) into useful tools. Driven by Moore's law and the exponential growth of data, artificial intelligence is permeating every aspect of our lives. But intelligence is not for free, it costs energy, and therefore money. Evolution has faced this problem for millions of years and made brains about a* $100\times$ *more energy efficient than modern hardware (or, as in the case of the sea-squirt, decided that it should eat its brain once is was no longer necessary). I will argue that energy will soon be one of the determining factors in AI. Either companies will find it too expensive to run energy hungry ML tools (such as deep learning) to power their AI engines, or the heat dissipation in edge devices will be too high to be safe. The next battleground in AI might well be a race for the most energy efficient combination of hardware and algorithms.*

We reuse an interesting plot in Figure 2 from Welling (2018) illustrating the development of deep neural networks and the corresponding energy consumptions. With such a rapid growth, by 2025, the hardware can host a neural network with as many parameters as neurons in the human brain. However, digital computers are not comparable with its biological counterpart in terms of energy efficiency. Moreover, artificial neural networks usually only target on a single task. Without model compression, we are simply wasting the resource of the earth.

4.4 Model Compression Approaches

Before discussing what is the best way to adaptively compress a neural network, one may ask: is it possible to compress a pretrained network without sacrificing its performance significantly? The answer is very likely, since it can be seen from Table 1 and Figure 1 that the performance of a neural network is not fully determined by the number of parameters. For example, ResNet-18 has only $5.6M$ parameters comparing to VGGNet which has $138M$ parameters. However, their top-1 accuracy and FLOPS are almost identical.

Denil et al. (2013a) shows that there is significant redundancy in the parameterization and that up to 95% parameters can be predicted from a small portion of learned parameters. That is, parameters are not equally important. Li et al. (2018), Giryes et al. (2016) argue that the key to learn a good representation of the data is the architecture itself. With a good network architecture, a large portion

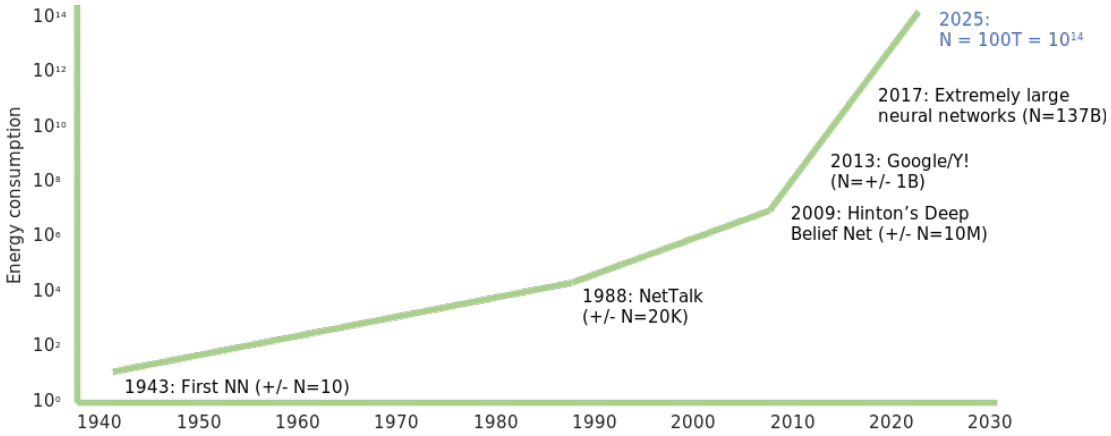


Figure 2: Energy consumption of deep neural networks. We will soon reach the capacity of the human brain. **Source**: Welling (2018).

of parameters can be even random values. All of these observations shed light on why model compression is possible. More intuitions and explanations will be discussed in Section 4.5.

We now provide a brief survey for model compression, classifying these approaches in terms of their main characteristics.

Low-rank approximation

Not surprisingly, there are a large amount of weights close to zeros in a learned neural network. Thus, the sparsity may be the first property to be explored in order to reduce the model size. Given a pretrained neural network, the weights are presented as matrices or tensors, Denton et al. (2014a), Jaderberg et al. (2014a) explored the idea of low-rank approximation as a postprocessing step to optimize the weight storage. Specifically, the original weight matrices/tensors are sparsely reconstructed through a matrix/tensor decomposition. For example, Denton et al. (2014a) perform a singular value decomposition for each convolutional layer, and fine-tune the upper layers until the prediction performance is restored. They achieve a speedup in convolutional layers by a factor of 2 to 3 times and a reduction in parameters in fully connected layers by a factor of 5 to 10 times.

Weight pruning

Since not all weights are equally important, some weights can be pruned according to some "saliency" criterion, which has been explored in the 90s by LeCun et al. (1990a), Hassibi et al. (1993). Standard critera include magnitude based thresholding, that is,

$$|w| < \text{threshold},$$

and the minimal decrease in training loss (LeCun et al. 1990a, Hassibi et al. 1993):

$$\min_q \Big\{ \min_{\delta w} \Delta\text{loss}(\delta w) \mid \delta w_q + w_q = 0 \Big\}.$$

Weight pruning can also be done in an iterative way by gradually building a mask of important weights (Han et al. 2015b). Retraining intervenes at each iteration to update the masked weights. Alternatively, the mask can be a part of the optimization. This was an idea proposed by Guo et al. (2016). They iteratively solve

$$\min_{w,T} L(w \odot T),$$

where the mask T is a function of w; $\odot$ denotes element-wise product.

It is also possible to prune out a whole convolutional filter. Liu et al. (2017) associate a scaling factor (reused from a batch normalization layer) to each channel in the convolutional layers. Then, a sparsity inducing regularization is imposed on these scaling factors during training to automatically identify unimportant channels. The channels with small scaling factor values will be pruned, followed by finetuning to achieve comparable (or even higher) accuracy as a normally trained full network.

Weight sharing

Weight sharing plays an important role in convolutional neural networks. This idea has been further extended to design even more efficient convolutional modules that directly build up efficient CNNs rather than compressing a pretrained one.

Flattened CNNs (Jin et al. 2014) allows for an acceleration of the feedforward pass by replacing each $C \times H \times W$ convolution with $C \times 1 \times 1$, $1 \times H \times 1$, and $1 \times 1 \times W$ convolutions, which can be seen as applying SVD on the original weights, but this is trained in an end-to-end manner rather than through post-processing.

SqueezeNet (Iandola et al. 2016a) achieves AlexNet-level accuracy with 50 times fewer parameters. It employs strategies such as replacing 3×3 filters by 1×1 filters; decreasing the number of input channels for 3×3 filters; downsampling late in the network so that convolution layers have large activation maps.

Other efficient CNNs including MobileNet (Howard et al. 2017a), ResNext (Xie et al. 2017) and ShuffleNet (Ma et al. 2018), which are built upon efficient CNN modules, such as *grouped convolution*, *depthwise convolution*, *channel shuffle* and so on.

Note that all the aforementioned efficient CNNs can be combined with other compression methods with a minor loss of accuracy.

Weight generated by a compact neural network

As mentioned by Denil et al. (2013a), it is possible to accurately predict the remaining weights from only a few weights. This is due to the fact that weights in learned networks tend to be structured. Inspired by this observation, a line of research consists of several different ways to form a weight generator to achieve model compression or relevant goals, which itself is a small network (i.e. a meta model), but it generates the actual weights for a larger network for the main task.

- The fast-weight network was proposed by (Schmidhuber 1992) for MLPs, in which one network produces context-dependent weight changes for a

second network. A subsequent work by Gomez and Schmidhuber (2005) demonstrated practical applications for the fast-weight network in the context of artificial control, where a generator network is learned by evolution algorithms.

- The HyperNetworks (Ha et al. 2017a) was proposed to allow non-shared weights for RNNs, since the weight sharing with recurrence was the main issue that causes gradient vanishing and exploding problems. The hypernetwork generates weights adaptively according to the layer-embedding vectors.

- The deep fried convnets (Yang et al. 2015) uses a special weight generator, reparameterizing a fully connected layer with d inputs and n outputs by the so called *adaptive fastfood transform*, which is a generalization of the *fastfood transform* for approximating kernels. It reduces the storage and the computation costs from $\mathcal{O}(nd)$ to $\mathcal{O}(n)$ and $\mathcal{O}(n \log d)$ respectively.

- The dynamic filter networks by Jia et al. (2016) was proposed in the context of video prediction and stereo vision. It depends on two related inputs. One input is used to predict sample specific and location specific information, forming the weights (i.e. filters) of the convolution, which will be applied on another input.

- The Bayesian neural networks (Blundell et al. 2015, Lacoste et al. 2017) can also be seen as a particular example of the weight generation, since it actually learns a distribution of the weights rather than a point estimate. In this case, the weight generator is exactly the distribution, generating weights amounts to sampling weights.

Depending on how small the meta network is, the actual weights to be learned is much less than learning the main network directly. It is certainly not the best method to yield the best compression rate, since the size of the meta network is not adaptive to different scenarios. Comparing to post-processing methods, such as low-rank approximation and weight pruning, this method is able to train all the modules in an end-to-end manner.

It is worth mentioning that the weight generating idea also has a wide application in meta learning (Gordon et al. 2018, Garnelo et al. 2018a, Bertinetto et al. 2016, Gidaris and Komodakis 2018).

Quantization

It has been recognized back in the 90s that the weights and activations of the neural networks do not take values in the whole real axis, thus they are not necessarily stored in the floating point format (Fiesler et al. 1990, Balzer et al. 1991). Quantization is the process of constraining an input to take value from a discrete set, given the input originally ranges over a large set of values. In the case of deep learning, quantization has been applied to weights, activations and gradients through rounding (Courbariaux et al. 2015, Wu et al. 2018b), vector

quantization (Gong et al. 2014, Chen et al. 2015) or optimization with discrete-set constraints (Rastegari et al. 2016, Carreira-Perpinán and Idelbayev 2017), which is aimed at achieving the same level of accuracy as that of their full-precision counterparts.

In particular, the quantization can be applied either in the postprocessing phase to coarsen the pretrained weights or in an end-to-end manner to directly obtain discrete weights and/or activations. The former is easier to implement. For example, Gong et al. (2014) implemented vector quantization for the fully connected layers, which was able to reduce the memory up to 24 times while keeping the drop of the top-5 accuracy within 1%. Han et al. (2015a) used a similar vector quantization to achieve the state-of-the-art model compression in accordance with magnitude-based weight pruning and entropy encoding in an iterative manner.

The end-to-end training of quantized neural networks with gradient based methods is more challenging due to the presence of discrete weights. It turns out that passing through a quantization layer is somehow equivalent to sampling from a discrete distribution. The latter has been widely studied in deep neural networks (Hinton et al. 2012, Kingma et al. 2015). As an example, BinaryConnect (Courbariaux et al. 2015) uses quantization in the forward pass and treats it as an identity function in the backward pass. This idea is also known as the *straight through gradient estimator* (Bengio et al. 2013). Although it is a biased estimator, it has been shown to be an useful trick in practice. The same idea can also be applied to activations leading to fully quantized neural networks (Courbariaux et al. 2016, Hubara et al. 2017).

Apart from being memory efficient, the quantization also enables energy efficient computations. For example, the dot product between two binary vectors amounts to counting the number of 1's in the logical conjunction of these two vectors. The bitwise operations can be carried out using arithmetic logic, which are much less costly than floating point operations.

4.5 Towards Understanding Generalization via Compression

In previous sections, we have seen that deep neural networks are typically designed to be over-parameterized, which renders potential limitations in applications due to memory and/or energy constraints. What is more critical is that a model with thousands of parameters are considered to have high complexity. This is a bad news in light of the classical learning theory, since the classical generalization bounds are based on measures of model complexity. However, deep learning has impressive generalization performance for a wide range of problems. This indicates that there must be some form of inductive bias or implicit regularization along with the network architectures or with the stochastic optimizers, which facilitates the learning converging to good solutions with high probability.

We have also reviewed model compression methods for deep learning, many

of which are able to reduce a large number of parameters without sacrificing the generalization. This means that the compression does not ruin the recipe of deep learning. Han et al. (2015a) show that the compressed model can sometimes be even better than the original model. Therefore, we may interpret the compression as a particular regularization if we incorporate it during the training.

It turns out that the connection between compression and generalization has already been exploited in different contexts. From a data communication point of view, the idea of understanding generalization through compression has been employed by Hinton and Van Camp (1993) via the minimum description length (MDL) theory (Rissanen 1978). From an optimization point of view, the flat minima conjecture (Hochreiter and Schmidhuber 1997b) associates the model compressibility with the geometry of the optimization landscape. More recently, compression scheme has been embedded into the probably approximately correct (PAC) learning framework to achieve PAC bounds (Arora et al. 2018) and PAC-Bayes bounds (Dziugaite and Roy 2017, Zhou et al. 2018, Pérez et al. 2018). Based on information theory, Tishby and Zaslavsky (2015) established the information-theoretical foundation for representation learning, where they interpret SGD as an implicit regularizer. Although being agnostic to the learning problem, it guides the learning biases towards achieving compressed representations.

4.5.1 The generalization puzzle

Due to the success of deep learning, many researchers attempt to answer the following question: *why does deep learning generalizes so well even without explicit regularizations?* The classical intuitive explanation (especially for CNNs) is that deep learning builds up a hierarchical feature representation in analogy to the primary visual cortex of the mammalian brain; the generalization comes from this bio-inspired "inductive bias", which leads to a special hypothesis space.

A recent empirical result shown by Zhang et al. (2016) challenges this explanation, for which completely different opinions[2] have been expressed. Some of the reviewers found these results not surprising at all, but others viewed them as thought-provoking.

To test the capacity of common DNNs, Zhang et al. (2016) conduct *randomization tests* on CIFAR-10 (Krizhevsky 2009a) and ImageNet (Deng et al. 2009) with three network architectures: Inception V3 (Szegedy et al. 2016), AlexNet (Krizhevsky et al. 2012b) and a 2-layer MLP with 512 hidden units, which means either the labels or the images are corrupted via the following ways:

- *Random labels*: all the labels are replaced with random ones.
- *Shuffled pixels*: a random permutation of the pixels is chosen and then the same permutation is applied to all the images in both the training and testing set.

[2] https://openreview.net/forum?id=Sy8gdB9xx

(a) (b) (c)

Figure 3: (a) Fitting random labels and random pixels on CIFAR10. Training losses of various experiments decrease to near zero. Source: Figure 1 in Zhang et al. (2016). (b) Experiments with ResNet18 architectures of different sizes on CIFAR-10. Even when after network is large enough to completely fit the training data(reference line), the test error continues to decrease for larger networks. Source: Figure 1 in Neyshabur et al. (2018). (c) The data (5 points) is generated from $y = x^3 - 2x^2 + 1 + \mathcal{N}(0, 0.1)$. MLP do not overfit even with 12 layers which is a network containing 18000 parameters. As a comparison, the dashed line shows an overfitted solution. Source: Figure 1 in Wu et al. (2017).

- *Random pixels*: a different random permutation is applied to each image independently.
- *Gaussian*: A Gaussian distribution (with matching mean and variance to the original image dataset) is used to generate random pixels for each image.

The result on CIFAR-10 is shown in Figure 3(a). In the case where all labels are random, the inputs and the labels are independent. However, DNNs are still able to achieve almost zero training errors, although the convergence time is much slower. Of course, DNNs overfit the train-set but this is considered unusual since DNNs rarely overfit real datasets. An immediate implication from this experiment is that DNNs have high capacity. That is, the hypothesis space of DNNs admits high complex solutions.

In practice, DNNs generalize well on real datasets. For example, Neyshabur et al. (2018) show that (in Figure 3(b)), on CIFAR10 with ResNet18, training with increasing number of parameters leads to a decrease in the testing error. In this case, both the training and testing errors converge to fix values, which suggests that being over-parameterized does not affect the generalization capability. This is not a standalone experiment, but an observation practically verified by a large number of network architectures (Novak et al. 2018). Similarly, Wu et al. (2017) argue that the reason why DNNs do not overfit in general is because learning tends to converge to a low complexity solution. They empirically verify that (in Figure 3(c)) over-parameterized MLPs do not overfit on a three-order polynomial regression with only 5 data points. Although there are many different complex solutions, learned DNNs favor simple solutions that agree with the train-set regardless of the number of layers.

To summarize, we leanrned several conclusions from these experiments:

- DNNs have high capacity to fit almost any dataset.

- DNNs do not overfit on real datasets as the number of parameters increasing, which is possibly due to implicit regularization or simply because adding more parameters does not necessarily increase the model complexity.
- DNNs tend to converge to simple solutions.

In order to understand the implications from these conclusions, we may turn to the classical framework in the learning theory– probably approximately correct (PAC) learning.

Denote by $R(f) := \mathbb{E}_{x,y}[\ell(f(x), y)]$ the risk of $f \in \mathcal{H}$ wrt a loss function $\ell(\cdot, \cdot)$, and $\hat{R}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$ is the empirical risk, where $\mathcal{H}$ is the hypothesis space. The PAC learning framework states that, with high probability, the generalization error bound for any $f \in \mathcal{H}$ takes the form of

$$R(f) - \hat{R}(f) \leq \mathcal{O}\Big(\frac{\text{complexity}(\mathcal{H})}{\sqrt{n}}\Big), \tag{4.1}$$

where complexity($\mathcal{H}$) is a complexity measure of the hypothesis space $\mathcal{H}$, such as the VC dimension (Blumer et al. 1989) or the Rademacher complexity (Koltchinskii 2001). Harvey et al. (2017) give a near-tight estimate of the VC dimension for DNNs with ReLU activations:

$$\text{VC-dimension}(\mathcal{H}) = \mathcal{O}\Big(LE\log(E)\Big) \tag{4.2}$$

with L the number of layers and E the number of edges/links in the network. Combining the conclusions we learned from the last section, one may notice that the classical PAC bounds can hardly explain the generalization of deep learning. Several possible reasons are listed as below.

- The VC-dimension is dependent of the number of parameters. However, this leads to a loose generalization bound, cannot explain the actual behaviors. Moreover, when the VC-dimension is much larger than the number of data points in the train-set, we can no longer identify a hypothesis with good generalization, since the hypothesis space can *shatter* a set that is larger than the train-set. In other words, there always exists two hypotheses predict exactly the same in the train-set, but they disagree on the test-set.
- The PAC learnability holds for the entire hypothesis space and for any distribution of the data. Thus, the universal statement also holds for the hypothesis with the highest complexity. In practice, the obtained solutions are in fact much simpler making the theoretical results less informative.

Given this situation, there are many attempts recently towards solving this puzzle. Among these attempts, the flat minima conjecture (Hochreiter and Schmidhuber 1997b, Keskar et al. 2016, Wu et al. 2017) is one of the well established conjectures based on the insights of the special geometry of the loss landscape. From a stochastic perspective, the PAC-Bayes explanations (Dziugaite and Roy 2017, Zhou et al. 2018, Pérez et al. 2018) and the information theoretical explanations (Hinton and Van Camp 1993, Tishby and Zaslavsky 2015, Achille and Soatto

2017) also shed light on the generalization puzzle. In fact, all of these conjectures and explanations are either directly linked to model compression or derived from the compressibility.

4.5.2 Sharpness: the bridge between compressibility and generalization

Keskar et al. (2016) observes that there is a degradation in the performance when using SGD with large batch size. They empirically find that large-batch SGD tends to converge to sharp minima, which are minima in which Hessians have large spectral norms. The sharpness is a widely used geometrical property in optimization. For example, it is used by Łojasiewicz (1993), Kurdyka (1998) to define the Kurdyka-Łojasiewicz condition. In deep learning, Hochreiter and Schmidhuber (1997b) has associated sharpness with generalization in terms of the *minimum description length* (MDL) principle (see Section 4.5.3): sharp minima correspond to weights which have to be specified with high precisions, while flat minima on the other hand only need to be determined with low precisions. In the terminlogy of MDL, the weights with low precision need fewer bits to describe (i.e. are of low complexity), thus, they have better generalization. Intuitively, low-precision weights are more robust to noise and dataset/domain shifts. As an example (Figure 4), if there is a small shift between the training function and the testing function, who generate the training data and the testing data respectively, a sharp minimum on the training function may be a point with high value on the testing function. This is exactly a case of overfitting.

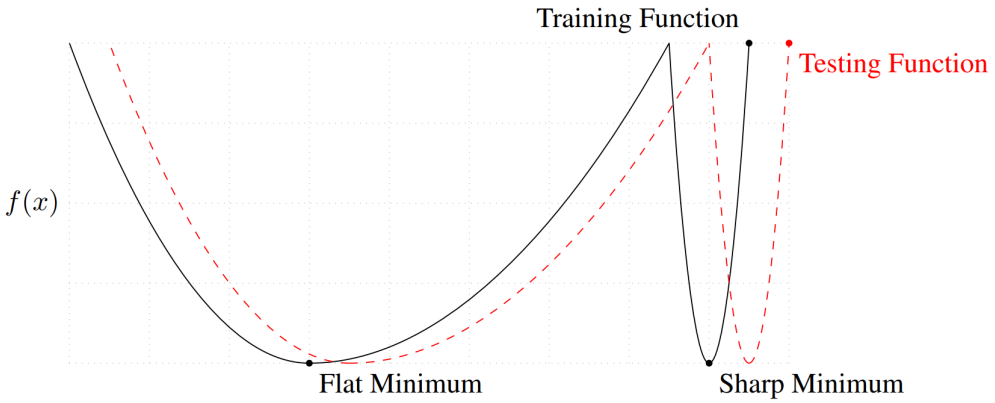


Figure 4: An intuitive example shows that flat minima generalizes better. Source: Figure 1 in Keskar et al. (2016).

The sharpness has more implications. Wu et al. (2017) show that flat minima imply smoother classifiers. Here, smoothness is in terms of the sensitivity to input changes. Thus, it also entails low complexity.

Theorem 4.5.1 (Wu et al. (2017), Corollary 1)**.** *For 2-layer neural networks of*

the form

$$f(x) := \sum_{k=1}^{K} a_k \sigma(w_k^\top x + b_k) \quad s.t. \quad \|w\|_F \leq W \tag{4.3}$$

with W a positive constant and $\sigma(\cdot)$ the ReLU activation function, denote by $\hat{R}(f)$ the empirical risk wrt $f(x)$, then

$$2\mathbb{E}\|\nabla_x f(x)\|^2 \leq \|\nabla_c^2 \hat{R}(f)\|_F^2 + W, \tag{4.4}$$

for $f(x)$ trained on a large enough dataset.

This theorem shows that if we train a 2-layer ReLU DNN to convergence, and the minimum is flat (i.e. $\|\nabla_c^2 \hat{R}(f)\|_F^2$ is small), then the returned DNN has low complexity. A similar implication has been demonstrated by Achille and Soatto (2017), where they show that the mutual information between a flat minimum and the true labels is low. On the contrary, the loss landscape of the learning with the random labels does not contain flat minima, since the mutual information between any solution and the random labels is predictably high.

The sharpness based on the magnitude of the eigenvalues of the Hessian is too expensive for large DNNs. Keskar et al. (2016) propose an alternative definition based on the sensitivity wrt weights. The definition is simplified by Dinh et al. (2017):

$$\frac{\max_{w' \in B(w,\epsilon)} \hat{R}(f_{w'}) - \hat{R}(f_w)}{1 + \hat{R}(f_w)} \simeq \max_{w' \in B(w,\epsilon)} \hat{R}(f_{w'}) - \hat{R}(f_w), \tag{4.5}$$

where $B(w, \epsilon)$ is an Euclidean ball centered at the minimum w with radius ϵ. The similar equal holds since $\hat{R}(f_w)$ is near zero at the minimum w.

However, both definitions of sharpness are sensitive to reparameterizations, as pointed out by Dinh et al. (2017). One can reparameterize the neural networks without changing the inputs and outputs while making the sharp minima to flat minima. Thus, sharpness alone cannot be a local measure to predict generalization.

Denote by $R(f)$ the risk with respect to the scoring function f. Specifying $q(w) = \mathcal{N}(\mu, \sigma^2 I)$, with probability $1 - \delta$ over the training set, the PAC-Bayes generalization bound in Theorem 4.5.2 can be rewriten as

$$\mathbb{E}_{q(w)}[R(f_w)] \leq \mathbb{E}_{q(w)}[\hat{R}(f_w)] + \mathcal{O}\left(\sqrt{\frac{D_{\mathrm{KL}}(q\|\pi)}{n}}\right), \tag{4.6}$$

$$\leq \hat{R}(f_\mu) + \underbrace{\max_{w \in B(\mu,\sigma)} [\hat{R}(f_w)] - \hat{R}(f_\mu)}_{\text{sharpness}} + \mathcal{O}\left(\sqrt{\frac{D_{\mathrm{KL}}(q\|\pi)}{n}}\right). \tag{4.7}$$

This result was first observed by Neyshabur et al. (2017). They also verified empirically that the generalization depends on both the sharpness and the KL divergence between the posterior $q(w)$ and the prior $\pi(w)$.

4.5.3 MDL: a lossless-compression-induced supervised learning framework

The minimum description length (MDL) principle (Rissanen 1978) is a model selection framework. It is based on the insights that the goal of statistical inference may be cast as trying to find regularity in the data, and the regularity may be identified as the ability to compress (Grunwald 2004). MDL is related to the Occam's razor. The latter only suggests that among all the hypotheses that are compatible with the evidence, the best hypothesis is the simplest one. MDL embodies Occam's razor by viewing machine learning as lossless data compression, where the objective is a trade-off between the data fitting (i.e. the compatibility with the evidence) and the model complexity. This is also known as the two-part codes in the literature.

Hinton and Van Camp (1993) utilizes the MDL as a bridge to explicitly link the regularizations of DNN training with the lossless data compression. They consider a communication game associated with the supervised learning framework. Suppose that a sender observes both the inputs $\mathbf{x} := \{x_i\}_{i=1}^n$ and the corresponding labels $\mathbf{y} := \{y_i\}_{i=1}^n$, while a receiver only observes the inputs. Denote by $D := (\mathbf{x}, \mathbf{y})$. A further required assumption is that the data are iid sampled. In order to send $\mathbf{y}$, rather than sending the raw data, they communicate in terms of a decoder (i.e. the model), which is known a priori by both sides with the form of $p(y|x, w)$, where w is the message (i.e. the parameter). Note that x, y, w are assumed to be integer-valued for the sake of simplicity in the communication game. The steps of the game are shown as follows.

1. The sender encodes D to a message $\hat{w}$ and compress $\hat{w}$ losslessly to a bitstream using entropy coding (e.g. Huffman coding) according to a prior distribution $p(w)$, which is agreed a priori by both sides.

2. The sender transmits the bitstream of $\hat{w}$ to the receiver. The code length is $-\log p(\hat{w})$.

3. The receiver decodes the bitstream and recovers $\hat{w}$ given access to $p(w)$. The receiver now obtain the model $p(y|x, \hat{w})$.

4. The sender transmits every y_i via a lossless compression using entropy coding according to $p(y_i|x_i, \hat{w})$. Each has a code length $-\log p(y_i|x_i, \hat{w})$.

5. The receiver reconstructs y_i with the decoder $p(y_i|x_i, \hat{w})$.

The only issue now is what is a good criterion for encoding $\mathbf{y}$ in the first step? From a data transmission viewpoint, a good criterion should be linked with the communication cost. That is, the code length should be as short as possible. This idea leads to an optimization of the form

$$\min_w -\sum_{i=1}^n \log p(y_i|x_i, w) - \log p(w), \tag{4.8}$$

which computes the optimal code length for the communication game. By defining $p(w)$, for example, as $\mathcal{N}(0, \lambda^2 I)$, we obtain exactly the ℓ_2 regularized maximum log-likelihood estimation with the regularization coefficient $\lambda > 0$, which encourages w to take values in the ℓ_2 ball. The argmin according to eq.(4.8), namely $\hat{w}$, is also known as the *maximum a posterior* estimate in statistics. Thus, we may call the above communication game *MAP coding*.

Alternatively, instead of picking the MAP code $\hat{w}$, we may sample w from an auxilary distribution $q(w|D)$, which is constructed by the sender given D. Let $\tilde{w}$ denote the sample drawn from $q(w|D)$, which carries auxilary information to be transmitted between the sender and the receiver. According to the MAP coding, the code length would be

$$-\sum_{i=1}^{n} \log p(y_i|x_i, \tilde{w}) - \log p(\tilde{w}). \tag{4.9}$$

Having sent $\mathbf{y}$, the receiver now has all the ingredients to recover $q(w|D)$ by the same algorithm used by the sender. Then, the receiver can decode the auxilary message according to $q(w|D)$. If the transmission of the auxilary message was conducted separately, the extra cost $-\log q(\tilde{w}|D)$ has to be paid. However, since the main message and the auxilary message are sent via a single communication, we have saved $-\log q(\tilde{w}|D)$ bits back. In other words, the actual communication cost in total is

$$-\sum_{i=1}^{n} \log p(y_i|x_i, \tilde{w}) - \log p(\tilde{w}) + \log q(\tilde{w}|D). \tag{4.10}$$

Now, taking into account the sampling, the only way to optimize the total communication cost in average is to manipulate the auxilary distribution $q(w|D)$, such that the expected eq.(4.10) is minimized. In other words, we choose $q(w|D)$ to be

$$\operatorname{argmin}_q \Big[-\sum_{i=1}^{n} \mathbb{E}_{q(w)}[\log p(y_i|x_i, w)] + D_{\mathrm{KL}}(q(w)\|p(w)) \Big]. \tag{4.11}$$

This version of the communication game is referred to as *bits-back coding*, which was originally proposed by Hinton and Van Camp (1993). Comparing to the MAP coding, the bits-back coding in average saves $H\big(q(w|D)\big)$ bits. Note that the "bits-back" argument is just a way to motivate that sending a distribution of the weights is much more efficient than sending a point estimate in terms of the communication cost. The auxilary task does not necessarily exist as a real task. The auxilary distribution $q(w|D)$ in fact have different names in different contexts. In Bayesian inference, it is called the *variational posterior*, while in PAC-Bayesian theory, it is called the *Gibbs predictor*.

Intuitively, the analogy between lossless data transmission and supervised learning is based on the insight that the data encoder and the supervised learner are both mechanisms to abstract information. If the data transmission favors simple and precise encoding, so does the supervised learning. Thus, MDL is a good principle to align compression with generalization guiding the learning towards

low complexity solutions. As can be seen from (4.11), very noisy weights can be communicated very cheaply, but they also cause extra variance in the data fitting term, making it more expensive to communicate.

The objective function in eq.(4.11) is variously known as the free energy in thermodynamics or the negative evidence lower bound (ELBO) in variational inference. In the context of approximate Bayesian inference (Graves 2011, Blundell et al. 2015), it can be derived from

$$\theta^* = \arg\min_\theta D_{\mathrm{KL}}\Big(q(w|\theta)\|p(w|D)\Big) \tag{4.12}$$

$$= \arg\min_\theta \int dw q(w|\theta) \log \frac{q(w|\theta)}{p(w)p(D|w)} \tag{4.13}$$

$$= \arg\min_\theta -\mathbb{E}_{q(w|\theta)}[\log p(D|w)] + D_{\mathrm{KL}}(q(w|\theta)\|p(w)) \tag{4.14}$$

In practice, the data fitting term is intractable due to the expectation over w, which is a high-dimensional vector. So, as a solution, Monte Carlo integration can be applied, but the optimization will suffer from high-variance gradients. Alternatively, if $q(w|\theta)$ is parameterized as a distribution in the location-scale family, the reparameterization trick (Kingma and Welling 2013) can be used. It employs a change of variables: $w = f(\phi, \epsilon)$, where ϕ is the variational parameter; $\epsilon \sim \mathcal{N}(0, I)$ is a noise variable. Then, the first term in eq.(4.14) can be rewriten as $-\mathbb{E}_{p(\epsilon)}[\log p(D|f(\phi, \epsilon))]$. The same reparameterization trick can be applied to the KL term in eq.(4.14). But that is not necessary, as KL divergence sometimes has closed form (e.g. for Gaussians). By applying this trick, we obtain unbiased stochastic gradients wrt the variational parameter ϕ, transforming it to an optimization problem that is solvable by stochastic gradient descent (SGD).

Variational Bayes with sparsity inducing prior explicitly align model compression with generalization. By imposing sparsity inducing priors, such the mixture of Gaussians (Ullrich et al. 2017a) and continuous relaxations of the spike-and-slab distribution (Louizos et al. 2017), the learned weights are easier to be pruned or quantized afterwards, since they are explicitly guided towards sparse or clustered structures while targeting on good performance on the training data. In practice, even with a standard Gaussian prior, variational Bayes provides a natural way to prune weights by examining the signal-to-noise-ratio (SNR) $\frac{|\mu|}{\sigma}$ (Graves 2011, Blundell et al. 2015).

So far we have reasoned the relationship between compression and generalization through the MDL intuition with the noisy weights. There is in fact a more straightforward justification from the PAC-Bayesian learning theory. Moreover, one can easily identify the connection between MDL and PAC-Bayesian bounds.

Let us first introduce some notations. The risk associated with a scoring function $f_w(x, y)$ and a posterior $q(w)$ is given by

$$R(q, f) = \mathbb{E}_{w\sim q(w)}\mathbb{E}_{(x,y)\sim p^*(x,y)}[f_w(x, y)], \tag{4.15}$$

where p^* is the underlying data distribution, which is unknown and independent of any model. Let $(\mathbf{x}, \mathbf{y}) \in (\mathcal{X} \times \mathcal{Y})^n$ be an iid sample of size n drawn from the

data distribution. The empirical risk is defined by

$$\hat{R}(q, f) = \mathbb{E}_{w \sim q(w)} \frac{1}{n} \sum_{i=1}^{n} [f_w(y_i, x_i)]. \tag{4.16}$$

The PAC Bayesian theory (McAllester 2003) provides a probably approximately correct bound on the generalization error, which is a data-driven bound that are computed on the sample $(\mathbf{x}, \mathbf{y})$, and holds uniformly over all encoder $q(w)$ which is a valid posterior distribution.

Theorem 4.5.2 (McAllester (2003))**.** *For a sample* $(\mathbf{x}, \mathbf{y})$ *of size* n *drawn iid from the data distribution, with probability at least* $1 - \delta$ *over the choice of* $(\mathbf{x}, \mathbf{y})$*, we have, for all valid posterior* $q(w)$*, scoring function* $f_w(x, y)$ *and prior* $\pi(w)$*,*

$$R(q, f) \leq \hat{R}(q, f) + \sqrt{\frac{D_{\mathrm{KL}}(q \| \pi) - \log \delta + \log n + 2}{2n - 1}}. \tag{4.17}$$

The quality of the generalization bound depends on the number of data points and the KL divergence between $q(w)$ and $\pi(w)$. Moreover, it is easy to see that the right hand side of eq.(4.17) resembles the objective function of variational Bayes, if we consider a specical case of the scoring function:

$$f_w(x, y) = -\log p(y|x, w) - \log \sum_{y'} e^{-f_w(x, y')}. \tag{4.18}$$

In fact, Germain et al. (2016) has identified that minimizing the PAC-Bayesian bound is equivalent to maximizing the Bayesian marginal likelihood.

In order to directly relate generalization error with model compression, one could define a prior reflecting the model size. For example, Zhou et al. (2018) considered a prior

$$\pi(w) = \frac{1}{Z} u(|w|) 2^{-|w|} \quad \text{with} \quad Z = \sum_{w} u(|w|) 2^{-|w|} \tag{4.19}$$

based on the model size $|w|$, where $u(\cdot)$ is the uniform distribution, and a posterior $q(w)$ being the delta distribution with a point mass at w. The KL divergence is then upper bounded by

$$D_{\mathrm{KL}}(q \| \pi) \leq |w| \log 2 - \log(u(|w|)) \tag{4.20}$$

due to the fact that $Z \leq 1$. By plugging eq.(4.20) into Theorem 4.5.2, we see clearly that the generalization bound depends on the model size.

4.5.4 Information bottleneck: a lossy-compression-induced supervised learning framework

MDL is an analogy between lossless data compression and supervised learning. However, since the data usually contains noise, it is more appropriate to use a lossy data compression scheme. Tishby and Zaslavsky (2015) take a first attempt on interpreting the DNN training with SGD optimizer in terms of a lossy data compression scheme– the information bottleneck methods. It becomes one of the hottest topics in theoretical deep learning, although it also brought a controversial discussion (Saxe et al. 2018).

Information bottleneck

The information bottleneck method (Tishby et al. 2000) extends the rate-distortion framework for supervised learning or other problems involving finding the best tradeoff between preserving the relevance wrt a random variable Y and removing the irrelevance wrt another random variable X. This is done by searching a "bottleneck", such that the information that X contains about Y is squeezed through the bottleneck. The key underlying assumption is that Y must not be independent of X, namely, $I(X;Y) > 0$. In the case of supervised learning, it is common that Y is a deterministic function of X. Thus, $I(X;Y) = H(Y) - H(Y|X) = H(Y) > 0$. Specifically, the goal is to create a summary of X, the bottleneck, denoted by $\hat{X}$, such that it forms a graphical model $\hat{X} \leftarrow X \leftarrow Y$, and the distortion is defined in terms of $\hat{X}$ and Y. The proposed objective leads to an abstract optimization problem

$$\min I(\hat{X};X) - \beta I(\hat{X};Y), \tag{4.21}$$

which is a generalization of the standard rate-distortion tradeoff. $I(\hat{X};Y)$ can be seen as a "correct" measure of the distortion for this setting. To see this, we rewrite

$$\begin{aligned}
-I(\hat{X};Y) &= -\mathbb{E}_{p(x,\hat{x},y)}\Big[\log \frac{p(y|\hat{x})}{p(y)}\Big] && (4.22)\\
&= \mathbb{E}_{p(x,\hat{x})}\Big[-\sum_y p(y|x)\log\frac{p(y|\hat{x})}{p(y)} + \sum_y p(y|x)\log\frac{p(y|x)}{p(y|x)}\Big] && (4.23)\\
&= \mathbb{E}_{p(x,\hat{x})}\Big[\sum_y p(y|x)\log\frac{p(y|x)}{p(y|\hat{x})} - \sum_y p(y|x)\log\frac{p(y|x)}{p(y)}\Big] && (4.24)\\
&= \mathbb{E}_{p(x,\hat{x})}\Big[D_{\mathrm{KL}}\big(p(y|x)\|p(y|\hat{x})\big) - D_{\mathrm{KL}}\big(p(y|x)\|p(y)\big)\Big], && (4.25)
\end{aligned}$$

where the joint distribution $p(x,\hat{x},y) = p(x,\hat{x})p(y|x)$. Since we do not have the control over the interaction between X and Y, $D_{\mathrm{KL}}\big(p(y|x)\|p(y)\big)$ is a constant in the optimization. By applying Lemma 5.1.1, we specify the information bottleneck optimization as

$$\min_{p(\hat{x}|x)\in\mathcal{P}_{\hat{x}}} \min_{m(\hat{x})\in\mathcal{M}} \min_{p(y|\hat{x})\in\mathcal{P}_y} I(p(\hat{x}|x), m(\hat{x})) + \beta\mathbb{E}_{p(\hat{x}|x)m(\hat{x})} D_{\mathrm{KL}}\big(p(y|x)\|p(y|\hat{x})\big). \tag{4.26}$$

Comparing to (1.51), we see that

$$D(p(\hat{x}|x), m(\hat{x}), p(y|\hat{x})) = \mathbb{E}_{p(\hat{x}|x)m(\hat{x})} D_{\mathrm{KL}}\big(p(y|x)\|p(y|\hat{x})\big) \tag{4.27}$$

with an extended parameter $p(y|\hat{x}) \in \mathcal{P}_y$, where $\mathcal{P}_y$ is the set of valid $p(y|\hat{x})$.

To solve (4.26), assuming that we are access to $p(x)$ and $p(y|x)$, a Blahut-Arimoto-like algorithm is proposed by Tishby et al. (2000).

Proposition 4.5.3 (Tishby et al. (2000), Theorem 5)**.** *The minimization in (4.26) is performed by the convergent alternating iterations. Denoting by t the iteration step, we have*

$$p_{t+1}(\hat{x}|x) = \frac{m_t(\hat{x}) \exp\Big(- \beta D(p_t(\hat{x}|x), m_t(\hat{x}), p_t(y|\hat{x}))\Big)}{\sum_{\hat{x}'} m_t(\hat{x}') \exp\Big(- \beta D(p_t(\hat{x}'|x), m_t(\hat{x}'), p_t(y|\hat{x}'))\Big)} \tag{4.28}$$

$$m_{t+1}(\hat{x}) = \sum_x p(x) p_{t+1}(\hat{x}|x) \tag{4.29}$$

$$p_{t+1}(y|\hat{x}) = \sum_x p(y|x) p(x) \frac{p_{t+1}(\hat{x}|x)}{m_{t+1}(\hat{x})} \tag{4.30}$$

Note that the above alternating minimization algorithm is not in general a realistic algorithm, since we do not access to either $p(x)$ or $p(y|x)$ in most problems, such as the real supervised learning problems. However, the rate distortion theory and information bottleneck still provide an interesting new way to understand the generalization in machine learning.

Information bottleneck insights on generalization of DNNs

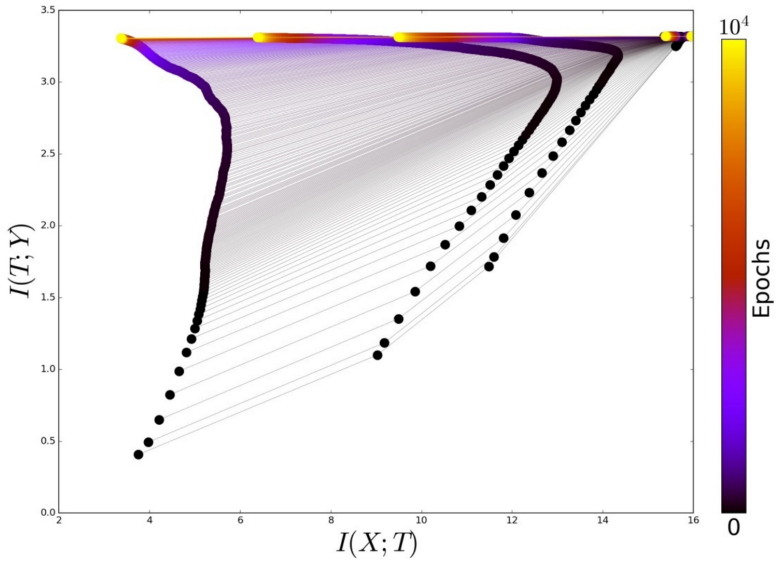


Figure 5: The trajectories of the points $(I(T;X), I(T;Y))$ at different epochs from different layers. The neural network is a CNN with ReLU activations and 5 hidden layers, which is trained on the CIFAR-10 dataset with SGD for 10^4 epochs. Each trajectory is associated with a hidden layer, and is represented by a sequence of points calculated at each epoch. The color is chosen to be directly linked to epoch. There is a clear transition between two phases, which can be seen for all trajectories. Due to the data processing inequality, lower layers always have higher mutual information. Source: the figure is taken from the comment of Naftali Tishby in the discussion at https://openreview.net/forum?id=ry_WPG-A-.

Tishby and Zaslavsky (2015) propose a hypothesis to explain the phenomenon that overfitting is not observed in highly over-parameterized neural networks: the training of deep neural networks is implicitly regularized by SGD, which biases

the learned representation of the data towards minimal sufficient statistics. This phenomenon is verified in a small-scale experiment, where they observe that there exists a two-phase transition during the learning. Denote by X, Y, T the input, the output and the representation respectively. In the first phase, the SGD has a fast convergence rate which is related to gain a sufficiency of the representation, namely, achieving high $I(T;Y)$. Later, the SGD enters an asymptotic phase which is related to the compression of the representation, namely, achieving small $I(T;X)$. An example of the evolving dynamics for all layers is shown in Figure 5. Thus, the conjecture is that SGD implicitly optimizes an objective which works similarly as that of the information bottleneck. Recent works on interpreting SGD as variational inference by Chaudhari and Soatto (2018), Mandt et al. (2017) share some interesting thoughts related to this conjecture.

Other than focusing on the noise introduced by SGD, Achille and Soatto (2018), Dai et al. (2018) relate information bottleneck with dropout (Hinton et al. 2012) through the local reparameterization trick (Kingma et al. 2015). Specifically, denote by Z_i the activation of the ith layer, the information bottleneck principle reads as

$$\min \sum_{i=1}^{L} \beta_i I(Z_i; Z_{i-1}) - I(Z_i; Y), \tag{4.31}$$

where $\beta_i >= 0$ is the coefficient to balance the minimality and sufficiency. Due to the intractable mutual information, a variational lower bound of the mutual information is used, leading to the following surrogate minimization:

$$\min \mathcal{L}\Big(\{p(z_i|z_{i-1})\}_{i=1}^{L}, \{q(z_i)\}_{i=1}^{L}, q(y|z_L)\Big) \tag{4.32}$$

$$\mathcal{L} := \sum_{i=1}^{L} \beta_i \mathbb{E}_{z_{i-1}\sim p(z_{i-1})}[D_{\mathrm{KL}}(p(z_i|z_{i-1})\|q(z_i))] - \mathbb{E}_{(x,y)\sim p^*(x,y)}\mathbb{E}_{z_L\sim p(z_L|x)}[\log q(y|z_L)],$$

where $\{q(z_i)\}_i, q(y|z_L)$ are variational distributions; $p(z_i|z_{i-1})$ comes from the decomposition of the joint distribution $p(z_i, z_{i-1}) = p(z_i|z_{i-1})p(z_{i-1})$. As a specical case, $z_0 = x$. So, $p(z_L|x) = \sum_{z\setminus z_L} \prod_i p(z_i|z_{i-1})$. It can be shown that if we specify $p(z_i|z_{i-1})$ as a conditional Gaussian via

$$z_i = (z_{i-1} \odot \epsilon_i) \cdot w_i \quad \text{with} \quad \epsilon_i \sim \mathcal{N}(0, I), \tag{4.33}$$

which is known as the Gaussian dropout, we obtain an equivalent way to inject Gaussian noise to the weights.

It is interesting to point out that, by maximizing the lower bound of the information bottleneck, i.e., (4.32), also known as the variational information bottleneck (Alemi et al. 2016), we are in fact doing exactly the same thing that MDL and variational Bayes are asking to. The equivalence is shown by the local reparameterization trick, that is, instead of introducing noise to the weights, it is more efficient to propagate noise to activations directly. This will also reduces the variance of the stochastic gradients (Kingma et al. 2015).

4.6 Transferring Knowledge from Over-Parameterized Models

We have seen that, empirically, the combination of SGD and over-parameterized DNNs is so far the fastest and most reliable way to achieve good performance in various machine learning tasks. On the other hand, we have seen in Section 4.4 that model compression methods, such as pruning and quantization, are applied either in an end-to-end fashion or by a post-processing scheme, which do not degrade the original models. However, model compression methods are unavoidably limited by the initial network architectures, which are not designed for energy efficient scenarios. As a result, compressed networks often cannot be tailored to fit mobile and embedded devices. Moreover, compression does not necessarily reduce the running time memory even the networks can be stored more efficiently. Another big issue is that over-parameterized models are often trained with huge data. Many tasks in real life are not able to be solved in that way, since gathering sufficiently large data is not always possible. Now, a natural question is *can we make use of the nice properties of the over-parameterization in deep learning if we only have limited data?*

Transfer learning (Pratt 1993, Caruana 1995) is targeted on this problem focusing on recognizing and applying relevant knowledge learned from previous tasks to new tasks. There are many methods can be used for transfer learning. The common approach is called finetuning, which amounts to training a source network on the source task and then copy the weights of the source network to initialize the target network. It is noticed that the top layers are more specific to the task itself while the lower layers are more generic (Yosinski et al. 2014). So, we usually froze low layers while finetuning, meaning that they do not change during the learning on the target task. A slightly more sophisticated idea is to regularize the distance between the weights of the source network and the weights of the target network. This idea has been successfully used by passive-aggressive algorithm for online learning (Crammer et al. 2006), adaptive SVM for computer vision (Yang et al. 2007, Aytar and Zisserman 2011), and more recently, elastic weight consolidation for overcoming catastrophic forgetting in continual learning (Kirkpatrick et al. 2017).

It turns out that both model tailoring and transfer learning can be cast as a knowledge transfer problem. In the former case, we transfer knowledge between two DNNs using the same dataset. In a slightly different scenario, namely, transfer learning, we perform the same transfer but acrossing two similar datasets.

The teacher-student framework emerged as a solution to the knowledge transfer problems, which was firstly proposed by Ba and Caruana (2014) as a model compression method. Within the same dataset, the framework is also known as knowledge distillation due to Hinton et al. (2015). Ba and Caruana (2014) focus on the regularization aspect of knowledge distillation. They argue that a shallow neural network can be as good as a deep neural network with the same number of parameters. Although shallow neural networks are more difficult to train, with the distillation loss as a regularization, they perform equally well. Hinton et al.

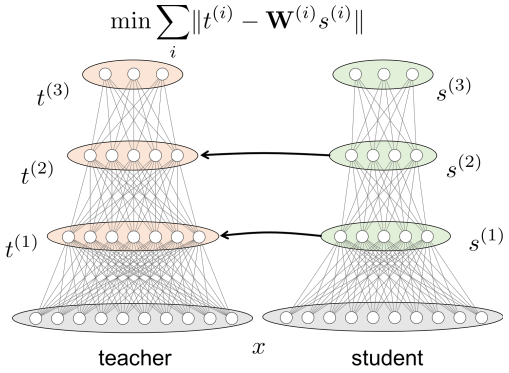


Figure 6: FitNet by Romero et al. (2014).

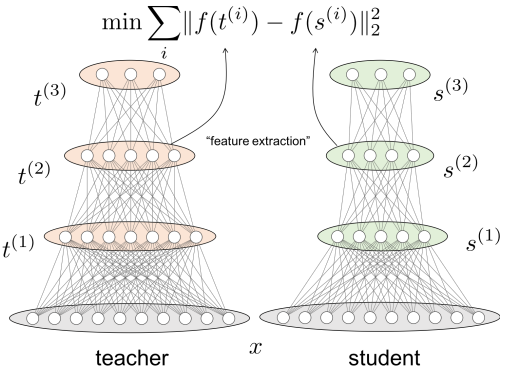


Figure 7: Attention transfer by Zagoruyko and Komodakis (2016b).

(2015) pay more attention on distilling knowledge from an ensemble of models (e.g. by dropout) into a single model as well as improving the training of an ensemble of specialist neural networks.

Concretely, we aim to train a student network (usually smaller) to mimic a pretrained teacher network (usually over-parameterized and trained with a sufficiently large dataset). The framework was originally implemented by matching the predictions of the two networks. For a classification problem, this means matching the final outputs (i.e. logits) of the two networks either via ℓ_2 distance on the log scale (Ba and Caruana 2014) or by the cross entropy loss (Hinton et al. 2015). Recently, several different implementations of the teacher-student framework have been proposed for convolutional neural networks. Moreover, rather than matching only the logits or the softmax layers, it has been shown that knowledge distillation can be applied to intermediate layers (Romero et al. 2014). Since the activation dimensions are mismatched between the teacher and the student, Zagoruyko and Komodakis (2016b) propose to extract statistics and then match those statistics. For example, matching the averages of activation maps can be interpreted as matching attention maps. Similarly, Romero et al. (2014) solve the mismatch issue by performing a linear regression on the student activation to match the teacher activation. An illustrative comparison between Romero et al. (2014) and Zagoruyko and Komodakis (2016b) is shown in Figure 6 and 7.

4.7 Conclusion

In this chapter, we have reviewed the over-parameterization design and its implications in deep learning. In particular, we have shown that over-parameterized DNNs can be compressed without significant performance drop, and have discussed the relationship between model generalization and compression. Finally, we have mentioned the teacher-student framework as a general framework to distil knowledge from over-parameterized DNNs.

Chapter

5 β-BNN: A Rate-Distortion Perspective on Bayesian Neural Networks

Abstract

We propose an alternative training framework for Bayesian neural networks (BNNs), which is motivated by viewing the Bayesian model for supervised learning as an autoencoder for data transmission. Then, a natural objective can be invoked from the rate-distortion theory. Specifically, we end up minimizing the mutual information between the weights and the dataset with a constraint that the negative log-likelihood is smaller than a certain value. The classical Blahut-Arimoto algorithm for solving this kind of optimization is infeasible due to the intractable expectations over the weights and the dataset, so we develop a new approximation to the steps of the Blahut-Arimoto algorithm. Our method exhibits some attractive properties over the conventional KL-regularized training of BNNs with fixed Gaussian prior: firstly, improved stability during optimization; secondly, a more flexible prior which can be understood from an empirical Bayes viewpoint.

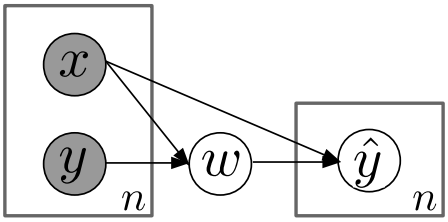


Figure 1: The graphical model of Bayesian supervised learning. Given a dataset, the supervised learning can be viewed as an autoencoding of the labels, where w is the shared latent variable.

5.1 Supervised learning via lossy compression

Given a sample $S := X \times Y$ with inputs $X := \{(x_i)\}_{i=1}^n$ and labels $Y := \{(y_i)\}_{i=1}^n$, consider a sender who would like to send Y to a receiver. Instead of transmitting the raw labels, the sender may compress the data with an encoder $q(w|S)$ and send the encoder as well as the inputs. The receiver can then reconstruct the labels by combining the encoder and a predefined decoder $p(y \mid x, w)$:

$$\hat{y}_i \sim q(y \mid x_i, S) := \int p(y \mid x_i, w) q(w|S) dw, \tag{5.1}$$

where $\hat{y}_i$ the reconstructed label for input x_i. Note that $q(y \mid x_i, S)$ resembles the *predictive distribution* in the Bayesian terminology, whose graphical model is shown in Figure 1.

This data-transmission view motivates a new learning objective for Bayesian models based on information theory. The Bayesian inference can be seen as a special case where we use the true posterior $p(w|S)$ as the encoder. However, the encoder $q(w|S)$ is not necessarily equal to the posterior $p(w|S)$. It comes from an empirical view of the joint distribution $p(S, w)$:

$$p(S, w) = q(w|S)p^*(S) \quad \text{with} \quad p^*(S) = \prod_{i=1}^{n} p^*(x_i, y_i), \tag{5.2}$$

where we denote the "true" distribution of data as $p^*(x, y)$, which is unknown and model independent. The data-transmission analogy of Bayesian inference offers us a direct way to relate data compression with model generalization. To see this, we analyze the limits of the encoder $q(w|S)$. If we force w to memorize everything in S including the noise, e.g. to have an identity map, we have to pay a high price for the communication cost. However, the encoder may not be even useful for another sample since it is too specific to a particular sample S. On the other hand, if w is independent of S, then the receiver has no way to decode the message no matter how powerful the decoder is.

The trade-off between the compression and the reconstruction can be formulated as a rate-distortion trade-off (Cover and Thomas 2012). The compression rate is measured by the mutual information $I(w; S)$[1]. The distortion function, denoted by

[1]Note that the rate is the minimum $I(\hat{S}, S)$ over the output $\hat{S}$ of the autoencoder. We

$d(w, S)$, measures the quality of the reconstruction, can be naturally defined as the negative log-likelihood. This argument is in line with the Minimum Description Length principle under lossless compression, which says that the best model among all equally good models is the one that leads to the best compression of the data. However, the proposed learning formulation is based on lossy compression. Specifically, the rate-distortion trade-off is expressed as the following optimization problem:

$$\min_{q(w|\cdot)\in\mathcal{Q}} \Big[I(w;S) \equiv I(q(w|S)) \Big] \qquad \text{s.t.} \quad \mathbb{E}_{p^*(S)}\mathbb{E}_{q(w|S)} d(w,S) \leq D \tag{5.3}$$

$$I(q(w|S)) := \mathbb{E}_{p^*(S)}\mathbb{E}_{q(w|S)} \Big[\log \frac{q(w|S)}{q(w)} \Big], \tag{5.4}$$

$$d(w,S) := -\sum_{i=1}^{n} \log p(y_i|x_i,w) = -\log p(S|w) + \text{ constant}, \tag{5.5}$$

where $\mathcal{Q}$ is the set of properly normalized pdfs. A similar rate-distortion analysis for unsupervised representation learning has been conducted by Alemi et al. (2017). Note that $q(w) = \sum_S p^*(S) q(w|S)$ is the aggregated posterior (Makhzani et al. 2015, Tomczak and Welling 2017). This coupling term renders optimization difficult. We show in the following lemma how to convert eq.(5.3) to an equivalent but more convenient problem.

Lemma 5.1.1 (10.8.1 (Cover and Thomas 2012))**.** *The mutual information has a variational form:*

$$I(X;Y) = \min_{m(y)} D_{\mathrm{KL}}(p(x,y) \| p(x)m(y)), \quad \text{where } m^*(y) = p(y) = \int p(x,y)dx.$$

By applying Lemma 5.1.1 to eq.(5.3), we rewrite the rate-distortion trade-off as:

$$\min_{m(w)} \min_{q(w|\cdot)\in\mathcal{Q}} I(q(w|S), m(w)) \qquad \text{s.t.} \quad \mathbb{E}_{p^*(S)}\mathbb{E}_{q(w|S)} d(w,S) \leq D \tag{5.6}$$

$$I(q(w|S), m(w)) := \mathbb{E}_{p^*(S)}\mathbb{E}_{q(w|S)} \Big[\log \frac{q(w|S)}{m(w)} \Big]. \tag{5.7}$$

Using the Lagrange dual function we convert eq. 5.6 into an unconstrained optimization problem:

$$F(\beta) := \min_{m(w)} \min_{q(w|\cdot)\in\mathcal{Q}} I(q(w|S), m(w)) + \beta \Big[\mathbb{E}_{p^*(S)}\mathbb{E}_{q(w|S)} d(w,S) - D \Big]. \tag{5.8}$$

Interestingly, we can see a connection with the maximum entropy principle (Jaynes 1957), where $m(w)$ term relates to the base measure of the entropy of $q(w)$.

minimize $I(w;S)$, since the decoder $p(y|x,w)$ only depends on x and w, and $I(\hat{S},S) \leq I(w;S)$ by data processing inequality.

One may also see the connection to the empirical Bayes (Robbins 1985, Kucukelbir and Blei 2014), since eq.(5.6) involves optimizing the "posterior" $q(w|S)$ and the "prior" $m(w)$ at the same time. This perspective links information theory and (empirical) Bayes at a model level rather than at an inference level. It is worth mentioning that Achille and Soatto (2017) was the first arriving the objective in eq.(5.3). However, their derivation was quite different and did not use the aforementioned variational upper bound for the mutual information.

5.2 Approximate Blahut-Arimoto Algorithm

The direct optimization of eq.(5.8) is cumbersome, since we need to parameterize a high dimensional mapping $q(w|S)$. Hence, we resort to the classical Blahut-Arimoto algorithm (Arimoto 1972, Blahut 1972) for computing $F(\beta)$, which is an alternating minimization using the following fixed point equations:

$$q(w|S) = \frac{m(w)\exp(-\beta\, d(w,S))}{\int m(v)\exp(-\beta\, d(v,S))dv} \tag{5.9}$$

$$m(w) = \sum_S p^*(S)q(w|S). \tag{5.10}$$

The iterative process defined by the Blahut-Arimoto algorithm optimizes $q(w|S)$ and $m(w)$ at the same time which, in theory, can lead to edge cases where $m(w)$ "chases" $q(w|S)$. However, in practice, this approach has proven to be effective, as shown in the empirical Bayes literature (Robbins 1985, Kucukelbir and Blei 2014) which is relevant due to our treatment of $m(w)$. In section 3, we will show these practical benefits in our scenario. As future work, we plan to address these theoretical questions exploiting the new uncovered links between distortion-rate theory and empirical Bayes.

We now continue with our inference method by making two approximations to the above equations:

1. We use a variational approximation $q(w|\theta)$ for $q(w|S)$ by solving

$$\theta(S) = \arg\min_\theta D_{\mathrm{KL}}(q(w|\theta)\|q(w|S)) \tag{5.11}$$

$$= \arg\min_\theta D_{\mathrm{KL}}(q(w|\theta)\|m(w)) + \beta\, \mathbb{E}_{q(w|\theta)}\Big[d(w,S)\Big]. \tag{5.12}$$

   Following Blundell et al. (2015), we parameterize $q(w|\theta)$ as a Gaussian distribution.

2. We approximate $m(w) \simeq \sum_S p^*(S)q(w|\theta(S)) \simeq \frac{1}{K}\sum_{k=1}^K q(w|\theta(B_k)) =: \tilde{m}(w)$, where B_k is a bootstrap sample of size n_b drawn from the empirical distribution $p_S(x,y) = \frac{1}{n}\sum_{i=1}^n \delta(x_i = x)\delta(y_i = y)$. Note that we only resample data to create B_k if $n_b > n$.

We call the resulting approach β-BNN. The detailed algorithm is shown in Algorithm 3. Note that the step for updating $q(w|\theta)$ resembles the ELBO derived by

Blundell et al. (2015) for vanilla BNNs, except that the coefficient β is now formally introduced, and instead of setting $m(w)$ to be $\mathcal{N}(0, I)$, $m(w)$ is approximated by a mixture of variational posteriors. It is clear that n_b and K determine how close we follow the classical Blahut-Arimoto steps. However, we do not want to take very large n_b and K, since both steps are approximated. Inspired by this argument, we also consider an online version, where $\tilde{m}(w)$ is updated whenever a new variational posterior is produced.

Algorithm 3 Approximate Blahut-Arimoto Algorithm

Input: S (dataset), β (coefficient), K (# mixture components), n_b (size of a bootstrap sample).
Initialize: $\Theta = \{\theta_k^{(0)} = (0, I)\}_{k=1}^K$; $\tilde{m}(w) = \frac{1}{K}\sum_{\theta\in\Theta} q(w|\theta)$.
for all $t = 1, \dots, T$ **do**
  Draw K bootstrap samples $\{B_k\}_{k=1}^K$ of size n_b from $p_S(x, y)$.
  for all $k = 1, \dots, K$ **do**
    $\theta_k^{(t)} \leftarrow n_b$ SGD steps on the loss of (5.12) initialized at $\theta_k^{(t-1)}$.
    $\Theta = \Theta \cup \{\theta_k^{(t)}\} \setminus \{\theta_k^{(t-1)}\}$.
    if do online update **or** $k = K$ **then**
      $\tilde{m}(w) = \frac{1}{K}\sum_{\theta\in\Theta} q(w|\theta)$.
    end if
  end for
end for
Output: Θ.

5.3 Experiments

We test *β-BNN*, *online β-BNN* and *vanilla BNN* (Blundell et al. 2015) on the colorful MNIST dataset (Bulten 2017), where each image is converted to RGB space and blended with a random background. We also test a *fixed-prior β-BNN*, which is a special case: $\tilde{m}(w) \equiv \mathcal{N}(0, I)$.

For this experiment, $T = 100$ is sufficient to converge; $q(w|\theta)$ and $\tilde{m}(w)$ are specified in Section 5.2; $p(y|x, w)$ is implemented by a multilayer perceptron (Linear400-ReLU-Linear400-ReLU-Linear10-Softmax); $\theta_k^{(t)}$ is obtained by running SGD over B_k once with batch size 128 and learning rate 10^{-3}. The comparison is shown in Table 1, where the prediction is $\arg\max_y p(y|x, \mathbb{E}[w])$. We allocate 10000 points as the validation set, and choose β^{-1} from 10 candidates ranging uniformly from 10^{-11} to 10^{-2}. We set $K = 5$ for the comparison as the performance only increases marginally for $K \geq 5$.

We empirically observe that fixed-prior BNNs suffer from slow convergence due to very large KL terms (about 10^6). Thus, we need to choose very small β (about 10^{-9}) to compensate, which will not be scalable for much larger networks. The convergence comparison is shown in Figure 2, where we can see that online β-BNN converges to a better local minimum and enjoys a faster convergence.

Algorithm	$\frac{1}{\beta^*}$	Accuracy
Vanilla BNN	$\frac{1}{n}$	90.05
Fixed-prior β-BNN	10^{-10}	95.86
β-BNN	10^{-5}	96.08
Online β-BNN	10^{-3}	97.12

Table 1: The comparison on colorful MNIST. We choose $n_b = 10000, K = 5$ for β-BNN. Then, at each iteration, it goes through the training set once. We choose $n_b = 128, K = 5$ for online β-BNN, and increase T to visit the same amount of data. Note that vanilla BNN corresponds to fixed-prior β-BNN with $\beta^{-1} = \frac{1}{n}$ and $K = 1$.

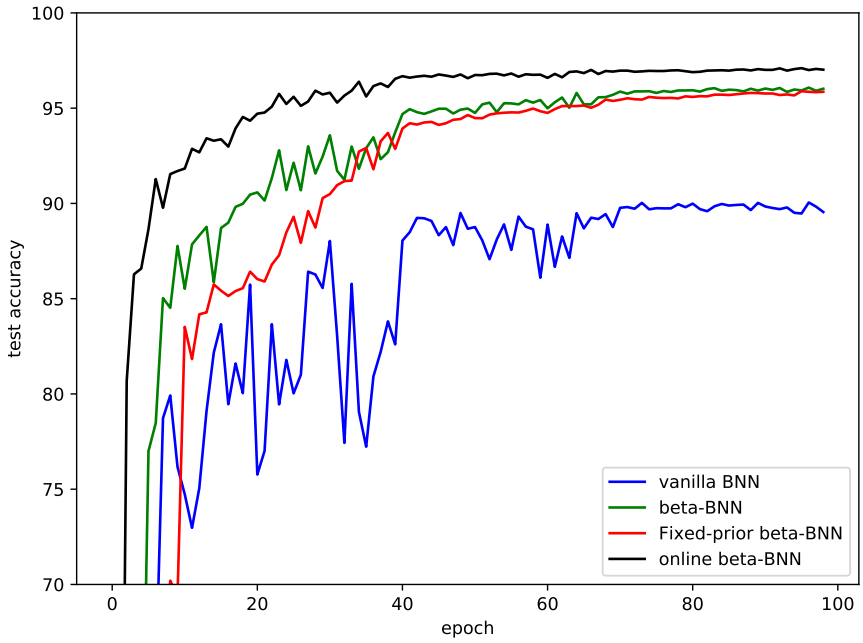


Figure 2: Testing accuracy over training epochs.

5.4 Discussion

The proposed algorithm is in fact quite similar to the training algorithm of vanilla BNNs, except that we update the "prior" every few iterations. We however did not fully take advantage of the stochasticity in eq.(5.8), which allows us to explicitly take the generalization into account. Consider two iid samples S and T drawn from the data distribution $p^*(S)$. We may take the following stochastic objective

$$\mathbb{E}_{q(w|\theta(S))}\left[\log \frac{q(w|\theta(S))}{m(w)}\right] + \beta/2\left[\mathbb{E}_{q(w|\theta(S))}d(w,S) + \mathbb{E}_{q(w|\theta(S))}d(w,T)\right] \tag{5.13}$$

as the loss function for each iteration, that is, we use S to update $q(w|\cdot)$, attaining $\theta(S)$, but take into account the performance of another sample T. We leave this extention to our future work.

Chapter

6 Empirical Bayes Transductive Meta-Learning with Synthetic Gradients

Abstract

We propose a meta-learning approach that learns from multiple tasks in a transductive setting, by leveraging unlabeled information in the query set to learn a more powerful meta-model. To develop our framework we revisit the empirical Bayes formulation for multi-task learning. The evidence lower bound of the marginal log-likelihood of empirical Bayes decomposes as a sum of local negative KL divergences between the variational posterior and the true posterior of each task. We derive a novel amortized variational inference that couples all the variational posteriors into a meta-model, which consists of a synthetic gradient network and an initialization network. The combination of local KL divergences and synthetic gradient network allows for backpropagating information from unlabeled data, thereby enabling transduction. Our results on the Mini-ImageNet and CIFAR-FS benchmarks for episodic few-shot classification significantly outperform previous state-of-the-art methods.

6.1 Introduction

While supervised learning of deep neural networks can achieve or even surpass human-level performance (He et al. 2015, Devlin et al. 2018), they can hardly extrapolate the learned knowledge beyond the domain where the supervision is provided. The problem of solving rapidly a new task after learning several other similar tasks is called *meta-learning* (Schmidhuber 1987, Bengio et al. 1991, Thrun and Pratt 1998); typically, the data is presented in a two-level hierarchy such that each data point at the higher level is itself a dataset associated with a task, and the goal is to learn a *meta-model* that generalizes across tasks. In this paper, we focus on *few-shot learning* (Vinyals et al. 2016), an instance of meta-learning problems, where a task t, in *meta-testing*, consists of an *unlabeled set* $x_t := \{x_{t,i}\}_{i=1}^{n}$ and a *labeled set* (aka *support set*) $d_t^l := \{(x_{t,i}^l, y_{t,i}^l)\}_{i=1}^{n^l}$, and the goal is to predict the corresponding labels, namely $y_t = \{y_{t,i}\}_{i=1}^{n}$, for the unlabeled set. In *meta-training*, y_t is provided as ground truth. The set $d_t := (x_t, y_t)$ is sometimes referred to as *query set*.

A particular important distinction to make is whether each task is solved in a *transductive* or *inductive* manner. The inductive setting is what was originally proposed by Vinyals et al. (2016): we use d_t^l to train a model and test it on x_t (one example at a time). Transduction, however, has the advantage of being able to see all points in x_t before making predictions. We argue that the transductive setting is more relevant to the problem since, as in semi-supervised learning, an inductive learner can always be built from a transductive one (Chapelle et al. 2006). In fact, Nichol et al. (2018) notice that most of the existing meta-learning methods follow the transductive setting unintentionally since they use x_t implicitly via the *batch normalization* (Ioffe and Szegedy 2015b).

Due to the hierarchical structure of the data, it is natural to formulate meta-learning as an instance of *hierarchical Bayes* (HB) (Good 1980, Berger 1985), or alternatively, empirical Bayes (EB) (Robbins 1985, Kucukelbir and Blei 2014). The difference is that the latter restricts the learning of meta-parameters to point estimates. In this paper, we focus on the EB model, since it largely simplifies the training and testing without losing the strength of the HB formulation.

The idea of using HB or EB for meta-learning is not new: Amit and Meir (2018) derive an objective similar to that of HB using PAC-Bayesian analysis; Grant et al. (2018) show that MAML (Finn et al. 2017) can be understood as a EB method; Ravi and Beatson (2018) consider a HB extension to MAML and compute posteriors via amortized variational inference. However, unlike our proposal, these methods do not take advantage of the unlabeled set. Roughly speaking, they construct the variational posterior as a function of the labeled set d_t^l without taking advantage of the unlabeled set x_t. The situation is similar in gradient based meta-learning methods (Finn et al. 2017, Ravi and Larochelle 2016, Li et al. 2017c, Nichol et al. 2018, Flennerhag et al. 2019) and many other meta-learning methods (Vinyals et al. 2016, Snell et al. 2017, Gidaris and Komodakis 2018), where the mechanisms used to generate the task-specific parameters rely on groundtruth labels, thus, there is no place for the unlabeled set to contribute. We argue that

this is a suboptimal choice, which may lead to overfitting when the labeled set is small and hinder the possibility of zero-shot learning (when the labeled set is not provided). An exception is Liu et al. (2018). They reuse the label propagation algorithm (Zhu et al. 2003) for transductive inference within each task and show that transduction is useful for boosting the performance.

In this paper, we propose to use synthetic gradient (Jaderberg et al. 2017) to enable transductivity, such that the variational posterior is implemented as a function of the labeled set d_t^l and the unlabeled set x_t. The synthetic gradient is produced by a neural network and learned to be a surrogate of the true gradient. The optimization process is similar to the inner gradient descent in MAML, but it iterates on the unlabeled set x_t rather than on labeled set d_t^l, since it does not rely on y_t to compute the true gradient. The labeled set for an unseen task is now optional, which is only used to generate the initialization in our case. In summary, our main contributions are the following:

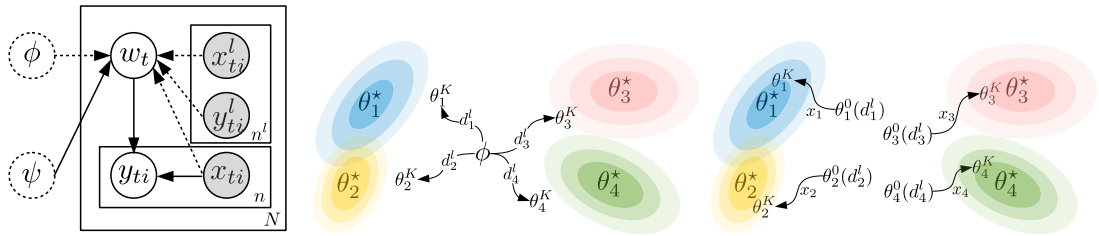
1. In section 6.2 and section 6.3, we develop a novel empirical Bayes formulation with transduction for meta-learning. To perform amortized variational inference, we propose a parameterization for the variational posterior based on synthetic gradient descent, which incoporates the contextual information from all the inputs of the query set.

2. In section 6.4, we show in theory that a transductive variational posterior yields better generalization performance. Besides, we show that the proposed empirical Bayes formulation is equivalent to the information bottleneck principle considered by Achille and Soatto (2017). We thus call our method *synthetic information bottleneck* (SIB).

3. In section 6.5, we verify our proposal empirically. Our experimental results demonstrate that our method significantly outperforms the state-of-the-art meta-learning methods on standard few-shot classification benchmarks.

6.2 Meta-learning with transductive inference

The goal of meta-learning is to train a *meta-model* on a collection of tasks, such that it works well on another disjoint collection of tasks. Suppose that we are given a collection of N tasks for training. The associated data is denoted by $\mathcal{D} := \{d_t := (x_t, y_t)\}_{t=1}^N$. In the case of few-shot learning, we are given in addition a support set d_t^l for each task. In this section, we revisit the classical empirical Bayes model for meta-learning. Then, we propose to use a transductive scheme in the variational inference by constructing the variational posterior as a function of x_t.



(a) Graphical model of EB (b) MAML (c) Our method (SIB)

Figure 1: **(a)** The generative and inference processes of the empirical Bayes model are depicted in solid and dashed arrows respectively, where the meta-parameters are denoted by dashed circles due to the point estimates. A comparison between MAML eq.(6.6) and our method (SIB) eq.(6.9) is shown in **(b)** and **(c)**. MAML is an inductive method since, for a task t, it first constructs a variational posterior $q_{\theta_t^K}$ as a function of the labeled set d_t^l, and then test on the unlabeled set x_t; while SIB constructs a better variational posterior as a function of both d_t^l and x_t: it starts from an initialization $\theta_t^0(d_t^l)$, and then yields θ_t^K by running K synthetic gradient steps on x_t.

6.2.1 Empirical Bayes model

Due to the hierarchical structure among data, it is natural to consider a hierarchical Bayes model with the marginal likelihood

$$p_f(\mathcal{D}) = \int_\psi p_f(\mathcal{D}|\psi)p(\psi) = \int_\psi \Big[\prod_{t=1}^N \int_{w_t} p_f(d_t|w_t)p(w_t|\psi)\Big]p(\psi). \tag{6.1}$$

The generative process is illustrated in Figure 1 (left, in solid arrows): first, a *meta-parameter* ψ (aka hyper-parameter) is sampled from the *hyper-prior* $p(\psi)$; then, for each task, a *task-specific parameter* w_t is sampled from the *prior* $p(w_t|\psi)$; finally, the dataset is drawn from the *likelihood* $p_f(d_t|w_t)$. Without loss of generality, we assume the log-likelihood model factorizes as

$$\begin{aligned}\log p_f(d_t|w_t) &= \sum_{i=1}^n \log p_f(y_{t,i}|x_{t,i}, w_t) + \log p(x_{t,i}|w_t), \\ &= -\frac{1}{n}\sum_{i=1}^n \ell_t\Big(\hat{y}_{t,i}(f(x_{t,i}), w_t), y_{t,i}\Big) + \text{constant},\end{aligned} \tag{6.2}$$

which is the setting advocated by Minka (2005), in which the generative model $p(x_{t,i}|w_t)$ can be safely ignored since it is irrelevant to the prediction of y_t. To simplify the presentation, we still keep the notation $p_f(d_t|w_t)$ for the likelihood of the task t and use ℓ_t to specify the discriminative model, which is also referred to as the *task-specific loss*, e.g., the cross entropy loss. The first argument in ℓ_t is the prediction, denoted by $\hat{y}_{t,i} = \hat{y}_{t,i}(f(x_{t,i}), w_t)$, which depends on the *feature representation* $f(x_{t,i})$ and the *task-specific weight* w_t.

Note that rather than following a fully Bayesian approach, we leave some random variables to be estimated by a frequentist approach, e.g., f is a *meta-parameter* of the likelihood model shared by all tasks, for which we use a point estimate. As such, the posterior inference about these variables will be largely

simplified. For the same reason, we derive the *empirical Bayes* (Robbins 1985, Kucukelbir and Blei 2014), which interprets ψ in a frequentist way:

$$p_{\psi,f}(\mathcal{D}) = \prod_{t=1}^{N} \int_{w_t} p_f(d_t|w_t) p_\psi(w_t). \tag{6.3}$$

We highlight the meta-parameters as subscripts of the corresponding distributions to distinguish from random variables. Indeed, we are not the first to formulate meta-learning as empirical Bayes. The overall model formulation is essentially the same as the ones considered by Amit and Meir (2018), Grant et al. (2018), Ravi and Beatson (2018).

6.2.2 Amortized inference with transduction

As in standard probabilistic modeling, we derive an *evidence lower bound* (ELBO) on the log version of eq.(6.3) by introducing a variational distribution $q_{\theta_t}(w_t)$ for each task with parameter θ_t:

$$\log p_{\psi,f}(\mathcal{D}) \geq \sum_{t=1}^{N} \Big[\mathbb{E}_{w_t \sim q_{\theta_t}} \Big[\log p_f(d_t|w_t) \Big] - D_{\mathrm{KL}} \Big(q_{\theta_t}(w_t) \| p_\psi(w_t) \Big) \Big]. \tag{6.4}$$

The variational inference amounts to maximizing the ELBO with respect to $\theta_1, \ldots, \theta_N$, which together with the maximum likelihood estimation of the meta-parameters, we have the following optimization problem:

$$\min_{\psi,f} \min_{\theta_1,\ldots,\theta_N} \frac{1}{N} \sum_{t=1}^{N} \Big[\mathbb{E}_{w_t \sim q_{\theta_t}} \Big[-\log p_f(d_t|w_t) \Big] + D_{\mathrm{KL}} \Big(q_{\theta_t}(w_t) \| p_\psi(w_t) \Big) \Big]. \tag{6.5}$$

However, the optimization in eq.(6.5), as N increases, becomes more and more expensive in terms of the memory footprint and the computational cost. We therefore wish to bypass this heavy optimization and to take advantage of the fact that individual KL terms indeed share the same structure. To this end, instead of introducing N different variational distributions, we consider a parameterized family of distributions in the form of $q_{\phi(\cdot)}$, which is defined implicitly by a deep neural network ϕ taking as input either d_t^l or d_t or both, that is, $q_{\phi(d_t^l)}$ or $q_{\phi(d_t^l, d_t)}$. This technique is known as *amortized* variational inference in the literature (Kingma and Welling 2013, Rezende et al. 2014).

Since d_t^l and x_t are disjoint, the inference scheme is *inductive* with a variational posterior $q_{\phi(d_t^l)}$. As an example, MAML (Finn et al. 2017) is an inductive method forcing $q_{\phi(d_t^l)}$ to be a Dirac delta distribution, where $\phi(d_t^l) = \theta_t^K$, the K-th iterate of the stochastic gradient descent

$$\theta_t^{k+1} = \theta_t^k + \eta \nabla_\theta \mathbb{E}_{w_t \sim q_{\theta_t^k}} \Big[\log p(d_t^l|w_t) \Big] \text{ with } \theta_t^0 = \phi. \tag{6.6}$$

Note that we overload ϕ to be both the learnable initialization as well as the amortization.

In this work, we consider a *transductive* inference scheme by using the entire x_t to define the variational posterior as $q_{\phi(x_t)}$. Replacing each q_{θ_t} by $q_{\phi(x_t)}$, eq.(6.5) can be written as

$$\min_{\psi,f} \min_{\phi} \frac{1}{N} \sum_{t=1}^{N} \Big[\mathbb{E}_{w_t \sim q_{\phi(x_t)}} \big[\log p_f(d_t|w_t) \big] - D_{\mathrm{KL}} \Big(q_{\phi(x_t)}(w_t) \| p_\psi(w_t) \Big) \Big]. \tag{6.7}$$

It is also possible to define the variational posterior as $q_{\phi(x_t,y_t)} \equiv q_{\phi(d_t)}$. However, to be consistent with testing, we do not take y_t as input since for which we do not have access to during testing.

In fact, nothing prevents us to come up with an even better variational posterior $q_{\phi(x_t,d_t^l)}$, shown in dashed arrows in Figure 1 (a), which is again transductive by definition. In a nutshell, the meta-model includes f, ψ from empirical Bayes and the amortization ϕ for inference.

6.3 Variational inference with synthetic gradients

It is however non-trivial to design a network architecture to implement the amortization $\phi(x_t)$ directly since x_t is itself a dataset. The strategy adopted by *neural processes* (Garnelo et al. 2018b) is to aggregate the information from all individual examples via a permutation invariant function. However, as pointed out by Kim et al. (2019), such a strategy tends to underfit x_t because the aggregation does not necessarily attain the most relevant information for identifying the task-specific parameter. Attentive neural process (Kim et al. 2019) may alleviate this issue with a time complexity of $O(n^2)$ while the original neural processes only need $O(n)$ time. We instead design a neural network $\phi(x_t)$ to parameterize the optimization process of θ_t. More specifically, consider a stochastic gradient descent on θ_t for optimizing eq.(6.5) with step size η:

$$\theta_t^{k+1} = \theta_t^k - \eta \, \nabla_{\theta_t} D_{\mathrm{KL}} \Big(q_{\theta_t^k}(w) \,\|\, p_{\psi,f}(w \,|\, d_t) \Big). \tag{6.8}$$

We would like to parameterize this optimization dynamics up to the K-th step via $\phi(x_t)$, such that $q_{\theta_t^K}$ is a good approximation of the optimum $q_{\theta_t^\star}$. It consists of parameterizing

> (a) the **initialization** θ_t^0 and (b) the **gradient** $\nabla_{\theta_t} D_{\mathrm{KL}}(q_{\theta_t}(w_t) \,\|\, p_{\psi,f}(w_t|d_t))$.

By doing so, θ_t^K becomes a function of ϕ, ψ and x_t[1], we therefore realize $q_{\phi(x_t)}$ as $q_{\theta_t^K}$.

[1] θ_t^K is also dependent of f. We deliberately remove this dependency to simplify the update of f.

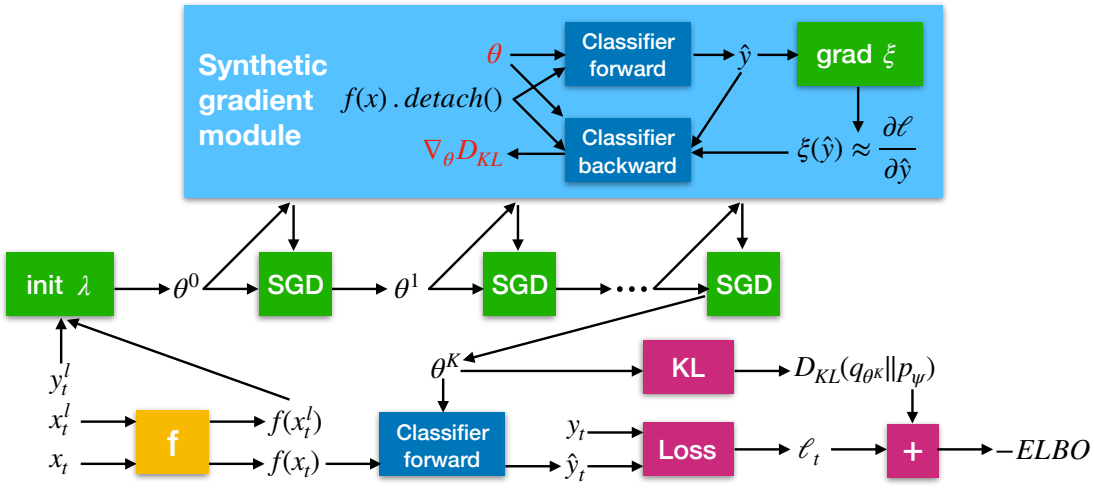


Figure 2: The computation graph to compute the negative ELBO, where the input and output of the synthetic gradient module are highlighted in red. The detach() is used to stop the back-propagation down to the feature network. Note that we do not include every computation for simplicity.

For (a), we opt to either let $\theta_t^0 = \lambda$ to be a global data-independent initialization as in MAML (Finn et al. 2017) or let $\theta_t^0 = \lambda(d_t^l)$ with a few supervisions from the support set, where λ can be implemented by a permutation invariant network described in Gidaris and Komodakis (2018). In the second case, the features of the support set will be first averaged in terms of their labels and then scaled by a learned vector of the same size.

For (b), the fundamental reason that we parameterize the gradient is because we do not have access to y_t during the meta-testing phase. Note that we are able to follow eq.(6.8) in meta-training to obtain $q_{\theta_t^\star}(w_t) \propto p_f(d_t|w_t)p_\psi(\mathrm{w}_t)$. To make a consistent parameterization in both meta-training and meta-testing, we thus discard y_t when constructing the variational posterior. Regarding the true gradient, a key observation is that, under a reparameterization $w_t = w_t(\theta_t, \epsilon)$ with $\epsilon \sim p(\epsilon)$,

$$\nabla_{\theta_t} D_{\mathrm{KL}}\Big(q_{\theta_t} \| p_{\psi,f}\Big) = \mathbb{E}_\epsilon\Big[\frac{1}{n}\sum_{i=1}^{n} \frac{\partial \ell_t(\hat{y}_{t,i}, y_{t,i})}{\partial \hat{y}_{t,i}} \frac{\partial \hat{y}_{t,i}}{\partial w_t} \frac{\partial w_t(\theta_t, \epsilon)}{\partial \theta_t}\Big] + \nabla_{\theta_t} D_{\mathrm{KL}}\Big(q_{\theta_t} \| p_\psi\Big),$$

where all the terms can be computed without y_t except for $\frac{\partial \ell_t}{\partial \hat{y}_{t,i}}$, thus, we introduce a deep neural network $\xi(\hat{y}_{t,i})$ to synthesize it. The idea of synthetic gradients was originally proposed by Jaderberg et al. (2017) to parallelize the back-propagation. Here, the purpose of $\xi(\hat{y}_{t,i})$ is to update θ_t regardless of the groundtruth labels, which is slightly different from its original purpose. Besides, we do not introduce an additional loss to force $\xi(\hat{y}_{t,i})$ to approximate $\frac{\partial \ell_t}{\partial \hat{y}_{t,i}}$ since $\xi(\hat{y}_{t,i})$ will be learned to yield a reasonable θ_t^K even without mimicking the true gradient.

To sum up, we have derived a particular implementation of $\phi(x_t)$ by parameterizing the ideal mean-field update, namely eq.(6.8), with respect to the query set d_t, where the meta-model ϕ includes an initialization network λ and a synthetic gradient network ξ, such that the output of the amortization $\phi(x_t)$ is θ_t^K – the

Algorithm 4 Variational inference with synthetic gradients for empirical Bayes

Input: the dataset $\mathcal{D}$; the step size η; the number of inner iterations K; pretrained f.
Initialize the meta-models ψ, and $\phi = (\lambda, \xi)$.
while not converged **do**
    Sample a task t and the associated dataset d_t (plus optionally the support set d_t^l).
    Compute the initialization $\theta_t^0 = \lambda$ or $\theta_t^0 = \lambda(d_t^l)$.
    for $k = 1, \ldots, K$ **do**
        Compute θ_t^k via eq.(6.9).
    end for
    Compute $w_t = w_t(\theta_t^K, \epsilon)$ with $\epsilon \sim p(\epsilon)$.
    Update $\psi \leftarrow \psi - \eta \nabla_\psi D_{\mathrm{KL}}(q_{\theta_t^K(\psi)} \| p_\psi)$.
    Update $\phi \leftarrow \phi - \eta \nabla_\phi D_{\mathrm{KL}}(q_{\phi(x_t)} \| p_f \cdot p_\psi)$.
    Optionally, update $f \leftarrow f + \eta \nabla_f \log p_f(d_t | w_t)$.
end while

K-th iterate of the following update:

$$\theta_t^{k+1} = \theta_t^k - \eta \left[\mathbb{E}_\epsilon \left[\frac{1}{n} \sum_{i=1}^{n} \xi(\hat{y}_{t,i}) \frac{\partial \hat{y}_{t,i}}{\partial w_t} \frac{\partial w_t(\theta_t^k, \epsilon)}{\partial \theta_t} \right] + \nabla_{\theta_t} D_{\mathrm{KL}} \left(q_{\theta_t^k} \| p_\psi \right) \right]. \tag{6.9}$$

The overall algorithm is depicted in Algorithm 4. We also make a side-by-side comparison with MAML shown in Figure 1. Rather than viewing eq.(6.9) as an optimization process, it may be more precise to think of it as a part of the computation graph created in the forward-propagation. The computation graph of the amortized inference is shown in Figure 2,

As an extension, if we were deciding to estimate the feature network f in a Bayesian manner, we would have to compute higher-order gradients as in the case of MAML. This is inpractical from a computational point of view and needs technical simplifications (Nichol et al. 2018). By introducing a series of synthetic gradient networks in a way similar to Jaderberg et al. (2017), the computation will be decoupled into computations within each layer, and thus becomes more feasible. We see this as a potential advantage of our method and leave this to our future work[2].

6.4 Connection to information bottleneck

In this section, we study the empirical Bayes model in its abstract form, which turns out have a connection to a variant of the information bottleneck principle Achille and Soatto (2017), Tishby et al. (2000).

[2]We do not insist on Bayesian estimation of the feature network because most Bayesian versions of CNNs underperform their deterministic counterparts.

Abstract form of empirical Bayes

From eq.(6.3), we see that the empirical Bayes model implies a simpler joint distribution since

$$\log p_{\psi,f}(w_1, \ldots, w_N, \mathcal{D}) = \sum_{t=1}^{N} \log p_f(d_t|w_t) + \log p_\psi(w_t), \tag{6.10}$$

which is equal to the log-density of N iid samples drawn from the joint distribution

$$p(w, d, t) \equiv p_{\psi,f}(w, d, t) = p_f(d|w, t)p_\psi(w)p(t)^{3}$$

up to a constant if we introduce a random variable to represent the task and assume $p(t)$ is an uniform distribution. Thus, this joint distribution embodies the *generative process* of empirical Bayes. Correspondingly, there is another graphical model of the joint distribution characterizes the *inference process* of the empirical Bayes:

$$q(w, d, t) \equiv q_\phi(w, d, t) = q_\phi(w|d, t)q(d|t)q(t),$$

where $q_\phi(w|d, t)$ is the abstract form of the variational posterior with amortization, includes both the inductive form and the transductive form. The coupling between $p(w, d, t)$ and $q(w, d, t)$ is due to $p(t) \equiv q(t)$ as we only have access to tasks through sampling.

We are interested in the case that the number of tasks $N \to \infty$, such as the few-shot learning paradigm proposed by Vinyals et al. (2016), in which the objective of eq.(6.7) can be rewritten in an abstract form of

$$\mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\Big[\mathbb{E}_{q(w|d,t)}\big[-\log p(d|w, t)\big] + D_{\mathrm{KL}}\big(q(w|d, t)\|p(w)\big)\Big]. \tag{6.11}$$

In fact, optimizing this objective is the same as optimizing eq.(6.7) from a stochastic gradient descent point of view.

The learning of empirical Bayes with amortized variational inference can be understood as a variational EM in the sense that the E-step amounts to aligning $p_{\psi,f}(w|d, t)$ with $q_\phi(w|d, t)$ by tuning ϕ while the M-step amounts to adjusting the likelihood $p_f(d|w, t)$ and the prior $p_\psi(w)$.

Connection to information bottleneck

The following theorem shows the connection between eq.(6.11) and the information bottleneck principle. Before going into details, we first introduce a few notations: the *entropy* is defined as $H_p(\mathrm{x}) := \mathbb{E}_{p(\mathrm{x})}[-\log p(\mathrm{x})]$; the *mutual information* is given by $I_p(\mathrm{x}; \mathrm{y}) := D_{\mathrm{KL}}\big(p(\mathrm{x}, \mathrm{y})\|p(\mathrm{x})p(\mathrm{y})\big)$; the *cross entropy* is defined as $H_{p,q}(\mathrm{x}) := \mathbb{E}_{p(\mathrm{x})}[-\log q(\mathrm{x})]$.

[3]When the context is clear, we ignore the parameters and write $p_{\psi,f}(w, d, t) = p(w, d, t)$.

Theorem 6.4.1. *Given distributions $q(w|d,t)$, $q(d|t)$, $q(t)$, $p(w)$ and $p(d|w,t)$, we have*

$$\mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\Big[\mathbb{E}_{q(w|d,t)}\big[-\log p(d|w,t)\big] + D_{KL}\big(q(w|d,t)\|p(w)\big)\Big] \geq I_q(w;d|t) + H_{q,p}(d|w,t)$$

Proof. Denote by $q(w|t) := \mathbb{E}_{q(d|t)}q(w|d,t)q(d|t)$ the aggregated posterior of task t. eq.(6.11) can be decomposed as

$$\mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\Big[\mathbb{E}_{q(w|d,t)}\big[-\log p(d|w,t)\big] + D_{\mathrm{KL}}\big(q(w|d,t)\|p(w)\big)\Big] \tag{6.12}$$

$$= \mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\mathbb{E}_{q(w|d,t)}\left[\log \frac{q(w|d,t)q(w|t)}{p(d|w,t)p(w)q(w|t)}\right] \tag{6.13}$$

$$= \mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\mathbb{E}_{q(w|d,t)}\left[\log \frac{q(w|d,t)}{q(w|t)}\right] + \mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\mathbb{E}_{q(w|d,t)}\big[-\log p(d|w,t)\big]$$

$$+ \mathbb{E}_{q(t)}\mathbb{E}_{q(d|t)}\mathbb{E}_{q(w|d,t)}\left[\log \frac{q(w|t)}{p(w)}\right] \tag{6.14}$$

$$\geq I_q(w;d|t) + H_{q,p}(d|w,t). \tag{6.15}$$

The inequality is because $D_{\mathrm{KL}}(q(w|t)\|p(w)) \geq 0$ for all t's. □

eq.(6.15) is an extention of the information bottleneck principle (Achille and Soatto 2017) for multi-task setting, which, together with the synthetic gradient based variational posterior, inspire the name "synthetic information bottleneck" of our method.

6.5 Experiments

In this section, we first validate our method on few-shot learning, and then on zero-shot learning. Note that many meta-learning methods, such as MAML, cannot do zero-shot learning since they rely on the support set.

6.5.1 Few-shot classification

We compare SIB with state-of-the-art methods on few-shot classification problems. Our code is available at `https://bit.ly/2CHHR3F`.

Setup

Datasets

We choose standard benchmarks of few-shot classification for this experiment. Each benchmark is composed of disjoint training, validation and testing classes. **MiniImageNet** is proposed by Vinyals et al. (2016), which contains 100 classes, split into 64 training classes, 16 validation classes and 20 testing classes, where each class consists of 600 image-label pairs and each image is of size 84×84. **CIFAR-FS** is proposed by Bertinetto et al. (2018), which is created by dividing the original

Method	Backbone	MiniImageNet, 5-way 1-shot	MiniImageNet, 5-way 5-shot	CIFAR-FS, 5-way 1-shot	CIFAR-FS, 5-way 5-shot
MatchingNet (Vinyals et al. 2016)	Conv-4-64	44.2%	57%	–	–
MAML (Finn et al. 2017)	Conv-4-64	48.7±1.8%	63.1±0.9%	58.9±1.9%	71.5±1.0%
ProtoNet (Snell et al. 2017)	Conv-4-64	49.4±0.8%	68.2±0.7%	55.5±0.7%	72.0±0.6%
Relation Net (Sung et al. 2018)	Conv-4-64	50.4±0.8%	65.3±0.7%	55.0±1.0%	69.3±0.8%
GNN (Satorras and Bruna 2017)	Conv-4-64	50.3%	66.4%	61.9%	75.3%
R2-D2 (Bertinetto et al. 2018)	Conv-4-64	49.5±0.2%	65.4±0.2%	62.3±0.2%	77.4±0.2%
TPN (Liu et al. 2018)	Conv-4-64	55.5%	69.9%	–	–
(Gidaris et al. 2019)	Conv-4-64	54.8±0.4%	**71.9±0.3%**	63.5±0.3%	**79.8±0.2%**
SIB η=1e-3, K=0	Conv-4-64	50.0±0.4%	67.0±0.4%	59.2±0.5%	75.4±0.4%
SIB η=1e-3, K=3	Conv-4-64	**58.0±0.6%**	70.7±0.4%	**68.7±0.6%**	77.1±0.4%
SIB η=1e-3, K=0	Conv-4-128	53.62 ± 0.79%	71.48 ± 0.64%	–	–
SIB η=1e-3, K=1	Conv-4-128	58.74 ± 0.89%	74.12 ± 0.63%	–	–
SIB η=1e-3, K=3	Conv-4-128	62.59 ± 1.02%	75.43 ± 0.67%	–	–
SIB η=1e-3, K=5	Conv-4-128	**63.26 ± 1.07%**	**75.73 ± 0.71%**	–	–
TADAM (Oreshkin et al. 2018)	ResNet-12	58.5±0.3%	76.7±0.3%	–	–
SNAIL (Santoro et al. 2017)	ResNet-12	55.7±1.0%	68.9±0.9%	–	–
MetaOptNet-RR (Lee et al. 2019)	ResNet-12	61.4±0.6%	77.9±0.5%	72.6±0.7%	84.3±0.5%
MetaOptNet-SVM	ResNet-12	62.6±0.6%	78.6±0.5%	72.0±0.7%	84.2±0.5%
CTM (Li et al. 2019)	ResNet-18	64.1±0.8%	**80.5±0.1%**	–	–
(Qiao et al. 2018)	WRN-28-10	59.6±0.4%	73.7±0.2%	–	–
LEO (Rusu et al. 2019)	WRN-28-10	61.8±0.1%	77.6±0.1%	–	–
(Gidaris et al. 2019)	WRN-28-10	62.9±0.5%	79.9±0.3%	73.6±0.3%	**86.1±0.2%**
SIB η=1e-3, K=0	WRN-28-10	60.6±0.4%	77.5±0.3%	70.0±0.5%	83.5±0.4%
SIB η=1e-3, K=1	WRN-28-10	67.3±0.5%	78.8±0.4%	76.8±0.5%	84.9±0.4%
SIB η=1e-3, K=3	WRN-28-10	69.6±0.6 %	78.9±0.4%	78.4±0.6%	85.3±0.4%
SIB η=1e-3, K=5	WRN-28-10	**70.0±0.6%**	79.2±0.4%	**80.0±0.6%**	85.3±0.4%

Table 1: Average classification accuracies (with 95% confidence intervals) on the test-set of MiniImageNet and CIFAR-FS. For evaluation, we sample 2000 and 5000 episodes respectively for MiniImageNet and CIFAR-FS and test three different architectures as the feature extractor: Conv-4-64, Conv-4-128 and WRN-28-10. We train SIB with learning rate 0.001 and try different numbers of synthetic gradient steps K.

CIFAR-100 into 64 training classes, 16 validation classes and 20 testing classes; each image is of size 32×32.

Network architectures

Following Gidaris and Komodakis (2018), Qiao et al. (2018), Gidaris et al. (2019), we implement f by a 4-layer convolutional network (Conv-4-64 or Conv-4-128[4]) or a WideResNet (WRN-28-10) (Zagoruyko and Komodakis 2016c). We pretrain the feature network $f(\cdot)$ on the 64 training classes for a stardard 64-way classification. We reuse the feature averaging network proposed by Gidaris and Komodakis (2018) as our initialization network $\lambda(\cdot)$, which basically averages the feature vectors of all data points from the same class and then scales each feature dimension differently by a learned coefficient. For the synthetic gradient network

[4]Conv-4-64 consists of 4 convolutional blocks each implemented with a 3×3 convolutional layer followed by BatchNorm + ReLU + 2×2 max-pooling units. All blocks of Conv-4-64 have 64 feature channels. Conv-4-128 has 64 feature channels in the first two blocks and 128 feature channels in the last two blocks.

$\xi(\cdot)$, we implement a three-layer MLP with hidden-layer size $8k$. Finally, for the predictor $\hat{y}_{ij}(\cdot, w_i)$, we adopt the cosine-similarity based classifier advocated by Chen et al. (2019) and Gidaris and Komodakis (2018).

Evaluation metrics

In few-shot classification, a task (aka episode) t consists of a *query set* d_t and a *support set* d_t^l. When we say the task t is *k-way-n^l-shot* we mean that d_t^l is formed by first sampling k classes from a pool of classes; then, for each sampled class, n^l examples are drawn and a new label taken from $\{0, \ldots, k-1\}$ is assigned to these examples. By default, each query set contains $15k$ examples. The goal of this problem is to predict the labels of the query set, which are provided as ground truth during training. The evaluation is the average classification accuracy on the tasks created with testing classes.

Training details

We run SGD with batch size 8 for 40000 steps, where the learning rate is fixed to 10^{-3}. During training, we freeze the feature network. To select the best hyper-parameters, for each dataset, we sample 1000 tasks from the validation classes and reuse them at each training epoch. We use these validation tasks to select the best meta-model and then use it for the final evaluation on the tasks sampled from testing classes.

Comparison to state-of-the-art meta-learning methods

In Table 1, we show a comparison between the state-of-the-art approaches and several variants of our method (varying K or $f(\cdot)$). Apart from SIB, TPN (Liu et al. 2018) and CTM (Li et al. 2019) are also transductive methods.

First of all, comparing SIB ($K = 3$) to SIB ($K = 0$), we observe a clear improvement, which suggests that, by taking a few synthetic gradient steps, we do obtain a better variational posterior as promised. For 1-shot learning, SIB (when $K = 3$ or $K = 5$) significantly outperforms previous methods on both MiniImageNet and CIFAR-FS. For 5-shot learning, the results are also comparable. It should be noted that the performance boost is consistently observed with different feature networks, which suggests that SIB is a general method for few-shot learning.

However, we also observe a potential limitation of SIB: when the support set is relatively large, e.g., 5-shot, with a good feature network like WRN-28-10, the initialization θ_t^0 may already be close to some local minimum, making the updates later less important.

For 5-shot learning, SIB is sligtly worse than CTM (Li et al. 2019) and/or Gidaris et al. (2019). CMT (Li et al. 2019) can be seen as an alternative way to incorporate transduction – it measures the similarity between a query example and the support set while making use of intra- and inter-class relationships. Gidaris et al. (2019) uses in addition the self-supervision as an auxilary loss to learn a

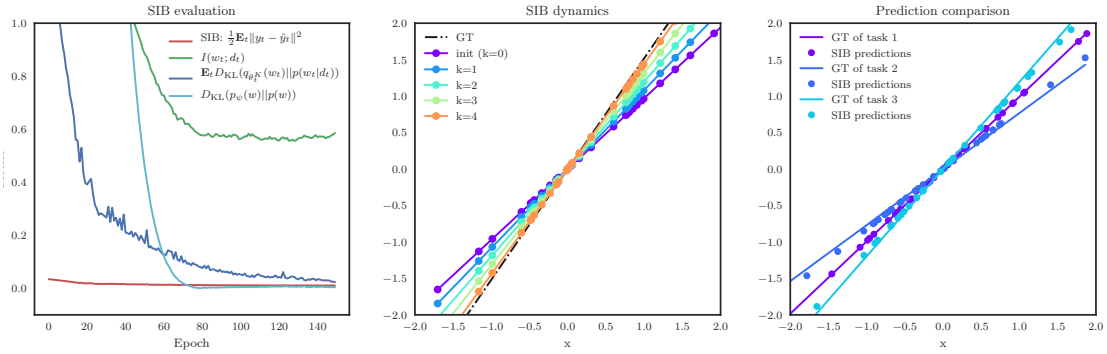


Figure 3: Left: the mean-square errors on D_{test}, $\mathbb{E}_t D_{\mathrm{KL}}(q_{\theta_t^K}(w_t)\|p(w_t|d_t))$, $D_{\mathrm{KL}}(p_\psi(w)\|p(w))$ and the estimate of $I(w;d) \approx \mathbb{E}_t D_{\mathrm{KL}}(q_{\theta_t^K}(w_t)\|p_\psi(w_t))$. **Middle**: the predicted y's by $y = \theta_t^k x$ for $k = 0, \ldots, 4$. **Right**: the predictions of SIB.

richer and more transferable feature model. Both ideas are complementary to SIB. We leave these extensions to our future work.

6.5.2 Zero-shot regression: spinning lines

Since our variational posterior relies only on x_t, SIB is also applicable to zero-shot problems (i.e., no support set available). We first look at a toy multi-task problem, where $I(w_t; d_t)$ is tractable.

Denote by $D_{\text{train}} := \{d_t\}_{t=1}^N$ the train set, which consists of datasets of size n: $d = \{(x_i, y_i)\}_{i=1}^n$. We construct a dataset d by firstly sampling iid Gaussian random variables as inputs: $x_i \sim \mathcal{N}(\mu, \sigma^2)$. Then, we generate the weight for each dataset by calculating the mean of the inputs and shifting with a Gaussian random variable ϵ_w: $w = \frac{1}{n}\sum_i x_i + \epsilon_w$, $\epsilon_w \sim \mathcal{N}(\mu_w, \sigma_w^2)$. The output for x_i is $y_i = w \cdot x_i$. We decide ahead of time the hyperparameters $\mu, \sigma, \mu_w, \sigma_w$ for generating x_i and y_i. Recall that a weighted sum of iid Gaussian random variables is still a Gaussian random variable. Specifically, if $w = \sum_i c_i x_i$ and $x_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, then $w \sim \mathcal{N}(\sum_i c_i \mu_i, \sum_i c_i^2 \sigma_i^2)$. Therefore, we have $p(w) = \mathcal{N}(\mu + \mu_w, \frac{1}{n}\sigma^2 + \sigma_w^2)$. On the other hand, if we are given a dataset d of size n, the only uncertainty about w comes from ϵ_w, that is, we should consider x_i as a constant given d. Therefore, the posterior $p(w|d) = \mathcal{N}(\frac{1}{n}\sum_{i=1}^n x_i + \mu_w, \sigma_w^2)$. We use a simple implementation for SIB:

- the variational posterior is realized by

$$q_{\theta_t^K}(w) = \mathcal{N}(\theta_t^K, \sigma_w), \ \theta_t^{k+1} = \theta_t^k - 10^{-3}\sum_{i=1}^n x_i \xi(\theta_t^k x_i), \text{ and } \theta_t^0 = \lambda \in \mathbb{R}; \tag{6.16}$$

- ℓ_t is a mean squared error, implies that $p(y|x, w) = \mathcal{N}(wx, 1)$;
- $p_\psi(w)$ is a Gaussian distribution with parameters $\psi \in \mathbb{R}^2$;
- the synthetic gradient network ξ is a three-layer MLP with hidden size 8.

Method	Art	Cartoon	Sketch	Photo	Average
JiGen (Carlucci et al. 2019)	84.9%	81.1%	79.1%	98.0%	85.7%
Rot (Xu et al. 2019)	88.7%	86.4%	74.9%	98.0%	87.0%
SIB-Rot $K = 0$	85.7%	86.6%	80.3%	98.3%	87.7%
SIB-Rot $K = 3$	**88.9%**	**89.0%**	**82.2%**	**98.3%**	**89.6%**

Table 2: Multi-source domain adaptation results on PACS with ResNet-18 features. Three domains are used as the source domains keeping the fourth one as target.

In the experiment, we sample 240 tasks respectively for both D_{train} and D_{test}. We learn SIB and BNN on D_{train} for 150 epochs using the ADAM optimizer (Kingma and Ba 2014), with learning rate 10^{-3} and batch size 8. Other hyperparameters are specified as follows: $n = 32, K = 3, \mu = 0, \sigma = 1, \mu_w = 1, \sigma_w = 0.1$. The results are shown in Figure 3. On the left, both $D_{\text{KL}}(q_{\theta_t^K}(w_t)\|p(w_t|d_t))$ and $D_{\text{KL}}(p_\psi(w)\|p(w))$ are close to zero indicating the success of the learning. More interestingly, in the middle, we see that $\theta_t^0, \theta_t^1, \ldots, \theta_t^4$ evolves gradually towards the ground truth, which suggests that the synthetic gradient network is able to identify the descent direction after meta-learning.

6.5.3 Zero-shot classification: unsupervised multi-source domain adaptation

A more interesting zero-shot multi-task problem is unsupervised domain adaptation. We consider the case where there exists multiple source domains and a unlabeled target domain. In this case, we treat each minibatch as a task. This makes sense because the difference in statistics between two minibatches are much larger than in the traditional supervised learning. The experimental setup is similar to few-shot classification described in Section 6.5.1, except that we do not have a support set and the class labels between two tasks are the same. Hence, it is possible to explore the relationship between class labels and *self-supervised* labels to implement the initialization network λ without resorting to support set. We reuse the same model implementation for SIB as described in Section 6.5.1. The only difference is the initialization network. Denote by $z_t := \{z_{t,i}\}_{i=1}^n$ the set of self-supervised labels of task t, the initialization network λ is implemented as follows:

$$\theta_t^0 = \lambda - \eta \nabla_\theta L_t\Big(\hat{z}_t\Big(\hat{y}_t(f(x_t), w_t(\theta, \epsilon)), f(x_t)\Big), z_t\Big), \tag{6.17}$$

where λ[5] is a global initialization similar to the one used by MAML; L_t is the self-supervised loss, $\hat{z}_t$ is the set of predictions of the self-supervised labels. One may argue that $\theta_t^0 = \lambda$ would be a simpler solution. However, it is insufficient since the gap between two domains can be very large. The initial solution yielded by eq.(6.17) is more dynamic in the sense that θ_t^0 is adapted taking into account the information from x_t.

[5] λ is overloaded to be both the network and its parameters.

In terms of experiments, we test SIB on the PACS dataset (Li et al. 2017a), which has 7 object categories and 4 domains (Photo, Art Paintings, Cartoon and Sketches), and compare with state-of-the-art algorithms for unsupervised domain adaptation. We follow the standard experimental setting (Carlucci et al. 2019), where the feature network is implemented by ResNet-18. We assign a self-supervised label $z_{t,i}$ to image i by rotating the image by a predicted degree. This idea was originally proposed by Gidaris et al. (2018) for representation learning and adopted by Xu et al. (2019) for domain adaptation. The training is done by running ADAM for 60 epochs with learning rate 10^{-4}. We take each domain in turns as the target domain. The results are shown in Table 2. It can be seen that SIB-Rot ($K = 3$) improves upon the baseline SIB-Rot ($K = 0$) for zero-shot classification, which also outperforms state-of-the-art methods when the baseline is comparable.

6.6 Conclusion

We have presented an empirical Bayesian framework for meta-learning. To enable an efficient variational inference, we followed the amortized inference paradigm, and proposed to use a transductive scheme for constructing the variational posterior. To implement the transductive inference, we make use of two neural networks: a synthetic gradient network and an initialization network, which together enables a synthetic gradient descent on the unlabeled data to generate the parameters of the amortized variational posterior dynamically. We have studied the theoretical properties of the proposed framework and shown that it yields significant performance boost on MiniImageNet and CIFAR-FS for few-shot classification.

Appendix

6.A Importance of synthetic gradients

To further verify the effectiveness of the synthetic gradient descent, we implement an inductive version of SIB inspired by MAML, where the initialization θ_t^0 is generated exactly the same way as SIB using $\lambda(d_t^l)$, but it then follows the iterations in eq.(6.6) as in MAML rather than follows the iterations in eq.(6.9) as in standard SIB.

We conduct an experiment on CIFAR-FS using Conv-4-64 feature network. The results are shown in Table 6.A.1. It can be seen that there is no improvement over SIB ($K = 0$) suggesting that the inductive approach is insufficient.

		inductive SIB		**SIB**			
		Training on 1-shot		**Training on 1-shot**		**Training on 5-shot**	
		Testing on		**Testing on**		**Testing on**	
K	η	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
0	-	59.7±0.5%	75.5±0.4%	59.2±0.5%	75.4±0.4%	59.2±0.5%	75.4±0.4%
1	1e-1	59.8±0.5%	71.2±0.4%	65.3±0.6%	75.8±0.4%	64.5±0.6%	77.3±0.4%
3	1e-1	59.6±0.5%	75.9±0.4%	65.0±0.6%	75.0±0.4%	64.0±0.6%	77.0±0.4%
5	1e-1	59.9±0.5%	74.9±0.4%	66.0±0.6%	76.3±0.4%	64.0±0.5%	76.8±0.4%
1	1e-2	59.7±0.5%	75.5±0.4%	67.8±0.6%	74.3±0.4%	63.6±0.6%	77.3±0.4%
3	1e-2	59.5±0.5%	75.8±0.4%	68.6±0.6%	77.4±0.4%	67.8±0.6%	78.5±0.4%
5	1e-2	59.8±0.5%	75.7±0.4%	67.4±0.6%	72.6±0.6%	67.7±0.7%	77.7±0.4%
1	1e-3	59.5±0.5%	75.6±0.4%	66.2±0.6%	75.7±0.4%	64.6±0.6%	78.1±0.4%
3	1e-3	59.9±0.5%	75.9±0.4%	68.7±0.6%	77.1±0.4%	66.8±0.6%	78.4±0.4%
5	1e-3	59.4±0.5%	75.7±0.4%	69.1±0.6%	76.7±0.4%	66.7±0.6%	78.5±0.4%
1	1e-4	58.8±0.5%	75.5±0.4%	59.0±0.5%	75.7±0.4%	59.3±0.5%	75.7±0.4%
3	1e-4	59.4±0.5%	75.9±0.4%	58.9±0.5%	75.6±0.4%	59.3±0.5%	75.9±0.4%
5	1e-4	59.3±0.5%	75.3±0.4%	60.1±0.5%	76.0±0.4%	60.5±0.5%	76.4±0.4%

Table 6.A.1: Average 5-way classification accuracies (with 95% confidence intervals) with Conv-4-64 on the test set of CIFAR-FS. For each test, we sample 5000 episodes containing 5 categories (5-way) and 15 queries in each category. We report the results with using different learning rate η as well as different number of updates K. Note that $K = 0$ is the performance only using the pre-trained feature.

6.B Varying n

We notice that changing the size of d_t (i.e., n) during training does make a difference on testing. The results are shown in Table 6.B.1.

n	5-way, 5-shot		5-way, 1-shot	
	Validation	Test	Validation	Test
3	77.97 ± 0.34%	75.91 ± 0.66%	63.60 ± 0.52%	61.32 ± 1.02%
5	78.14 ± 0.35%	76.01 ± 0.66%	64.67 ± 0.55%	62.50 ± 1.02%
10	**78.30 ± 0.35%**	**76.22 ± 0.66%**	**65.34 ± 0.56%**	**63.22 ± 1.04%**
15	77.53 ± 0.35%	75.43 ± 0.67%	65.14 ± 0.55%	62.59 ± 1.02%
30	76.21 ± 0.35%	74.04 ± 0.67%	63.37 ± 0.53%	60.96 ± 0.98%
45	75.65 ± 0.36%	73.27 ± 0.66%	62.08 ± 0.51%	59.59 ± 0.93%

Table 6.B.1: Average classification accuracies on the validation set and the test set of Mini-ImageNet with backbone Conv-4-128. We modify the number of query images, i.e., n, for each episode to study the effect on generalization.

Chapter

7 Variational Information Distillation for Knowledge Transfer

Abstract

Transferring knowledge from a teacher neural network pretrained on the same or a similar task to a student neural network can significantly improve the performance of the student neural network. Existing knowledge transfer approaches match the activations or the corresponding hand-crafted features of the teacher and the student networks. We propose an information-theoretic framework for knowledge transfer which formulates knowledge transfer as maximizing the mutual information between the teacher and the student networks. We compare our method with existing knowledge transfer methods on both knowledge distillation and transfer learning tasks and show that our method consistently outperforms existing methods. We further demonstrate the strength of our method on knowledge transfer across heterogeneous network architectures by transferring knowledge from a convolutional neural network (CNN) to a multi-layer perceptron (MLP) on CIFAR-10. The resulting MLP significantly outperforms the-state-of-the-art methods and it achieves similar performance to the CNN with a single convolutional layer.

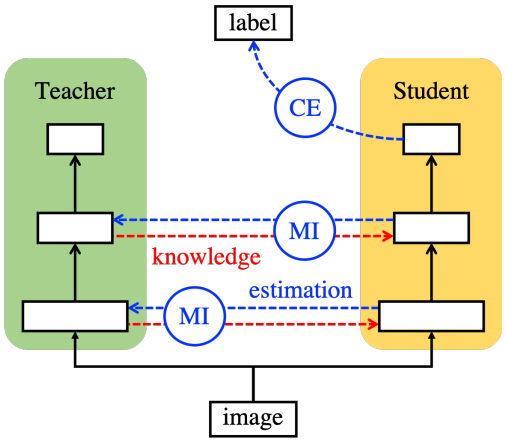


Figure 1: Conceptual diagram of the proposed knowledge transfer method. The student network efficiently learns the target task by minimizing the cross-entropy (CE) loss while retaining high mutual information (MI) with the teacher network. The mutual information is maximized by learning to estimate the distribution of the activations in the teacher network, provoking the transfer of knowledge.

7.1 Introduction

Deep neural networks (DNNs) play important roles in various computer vision tasks, *e.g.*, depth estimation (Eigen et al. 2014), pose estimation (Toshev and Szegedy 2014), optical flow (Dosovitskiy et al. 2015), object classification (He et al. 2016c), detection (Girshick 2015), and segmentation (Simonyan and Zisserman 2014b). A typical DNN approach for a computer vision task is to train a sophisticated end-to-end neural network with a large amount of labeled data. Such an approach often delivers state-of-the-art performance if a sufficient amount of data is available. However, in many cases, it is impossible to gather sufficiently large data to train a DNN. For example, in many medical image applications (Schlegl et al. 2014), the amount of available data is constrained by the number of patients of a particular disease.

A popular approach for handling such lack of data is transfer learning (Pan et al. 2010), where the goal is to transfer knowledge from the source task to facilitate learning on the target task. Typically, one considers the source task to be generic with a larger amount of available data that contains useful knowledge for learning the target task, *e.g.*, knowledge from natural image classification (Russakovsky et al. 2015) is likely to be useful for fine-grained bird classification (Welinder et al. 2010). Hinton et al. (2015) proposed the teacher-student framework for transferring such knowledge between DNNs being trained on the source and target tasks respectively. The high-level idea is to introduce an additional regularization for the DNN being trained on the target task, *i.e.*, the student network, which allows learning the knowledge existing in the DNN that was pre-trained on the source task, *i.e.*, the teacher network. While the framework was originally designed for knowledge transfer between DNNs on the same dataset, recent works (Yim et al. 2017, Zagoruyko and Komodakis 2016b) started exploiting its potential for

more general transfer learning tasks, *i.e.*, when the source data and the target data are different.

Many knowledge transfer methods have been proposed with various intuitions. Hinton *et al.* Hinton et al. (2015) and Ba and Caruana Ba and Caruana (2014) propose to match the final layers of the teacher and the student network, as the outputs from the final layer of the teacher network provide more information than raw labels. Romero *et al.* Romero et al. (2014) proposes to match intermediate layers of the student network to the corresponding layers of the teacher network. Recent works (Chen et al. 2018, Huang and Wang 2017, Yim et al. 2017, Zagoruyko and Komodakis 2016b) relax the regularization of matching the entire layer by matching carefully designed features/statistics extracted from intermediate layers of the teacher and the student networks, *e.g.*, attention maps (Zagoruyko and Komodakis 2016b) and maximum mean discrepancy (Huang and Wang 2017).

Evidently, there is no commonly agreed theory behind knowledge transfer. This causes difficulty in understanding empirical results and in developing new methods in a more principled way. In this paper, we propose variational information distillation (VID) as an attempt towards this direction in which we formulate the knowledge transfer as maximization of the mutual information between the teacher and the student networks. This framework proposes an actionable objective for knowledge transfer and allows us to quantify the amount of information that is transferred from a teacher network to a student network. Since the mutual information is computationally intractable, we employ a variational information maximization (Agakov 2004) scheme to maximize the variational lower bound instead. See Figure 1 for the conceptual diagram of the proposed knowledge transfer method. We further show that several existing knowledge transfer methods (Li and Hoiem 2017, Romero et al. 2014) can be derived as specific implementations of our framework by choosing different forms of the variational lower bound. We empirically validate our VID framework, which significantly outperforms existing methods. We observe the gap is especially large in the cases of small data and heterogeneous architectures.

In summary, the overall contributions of our paper are as follows:

- We propose variation information distillation, a principled knowledge transfer framework based on maximizing mutual information between two networks based on the variational information maximization technique.

- We demonstrate that VID generalizes several existing knowledge transfer method. In addition, our implementation of the framework empirically outperforms the state-of-the-art knowledge transfer methods on various knowledge transfer experiments, including knowledge transfer between (heterogeneous) DNNs on the same dataset or on different datasets.

- In particular, we demonstrate that heterogeneous knowledge transfer between convolutional neural networks (CNN) and multilayer perceptrons (MLP) is possible on CIFAR-10. Our method yields a student MLP that significantly outperforms the best-reported MLPs (Lin et al. 2015, Urban et al. 2017) in the literature.

7.2 Variational information distillation (VID)

In this section, we describe VID as a general framework for the knowledge transfer in the teacher-student framework. Specifically, consider training a student neural network on a target task, given another teacher neural network pre-trained on a similar (or related) source task. Note that the source task and the target task can be the same. The underlying assumption is that the layers in the teacher network have been trained to represent certain attributes of given inputs that exist in both the source task and the target task. For a successful knowledge transfer, the student network must learn how to incorporate the knowledge of such attributes to its own learning.

From a perspective of information theory, knowledge transfer can be expressed as retaining high mutual information between the layers of the teacher and the student networks. More specifically, consider an input random variable $\boldsymbol{x}$ drawn from the target data distribution $p(\boldsymbol{x})$ and K pairs of layers $\mathcal{R} = \{(\mathcal{T}^{(k)}, \mathcal{S}^{(k)})\}_{k=1}^{K}$, where each pair $(\mathcal{T}^{(k)}, \mathcal{S}^{(k)})$ is selected from the teacher network and the student network respectively. Feedforwarding the input $\boldsymbol{x}$ through the networks induces K pairs of random variables $\{(\boldsymbol{t}^{(k)}, \boldsymbol{s}^{(k)})\}_{k=1}^{K}$ which indicate activations of the selected layers, *e.g.*, $\boldsymbol{t}^{(k)} = \mathcal{T}^{(k)}(\boldsymbol{x})$. The mutual information between the pair of random variables $(\boldsymbol{t}, \boldsymbol{s})$ is defined by:

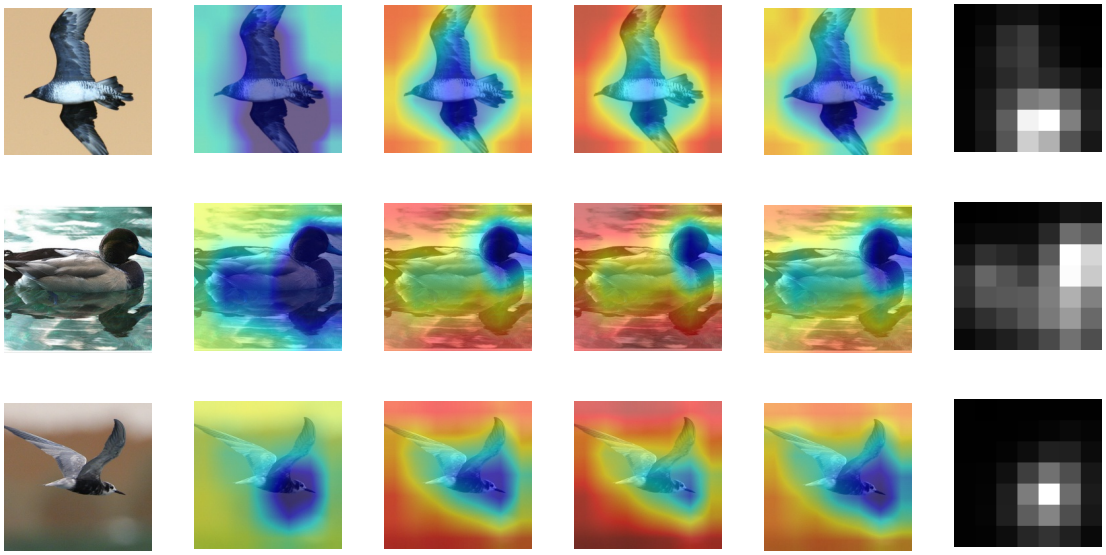
$$\begin{aligned} I(\boldsymbol{t}; \boldsymbol{s}) &= H(\boldsymbol{t}) - H(\boldsymbol{t}|\boldsymbol{s}) \\ &= -\mathbb{E}_{\boldsymbol{t}}[\log p(\boldsymbol{t})] + \mathbb{E}_{\boldsymbol{t},\boldsymbol{s}}[\log p(\boldsymbol{t}|\boldsymbol{s})], \end{aligned} \tag{7.1}$$

where the entropy $H(\boldsymbol{t})$ and the conditional entropy $H(\boldsymbol{t}|\boldsymbol{s})$ were derived from the joint distribution $p(\boldsymbol{t}, \boldsymbol{s})$. Note that the joint distribution $p(\boldsymbol{t}, \boldsymbol{s})$ is a result of aggregation over the layers with input $\boldsymbol{x}$ sampled from the empirical distribution $p(\boldsymbol{x})$. Intuitively, the definition of $I(\boldsymbol{t}; \boldsymbol{s})$ can be understood as a reduction in uncertainty about the knowledge of the teacher encoded in its layer $\boldsymbol{t}$ when the the student layer $\boldsymbol{s}$ is known.

We now define the following loss function which aims to learn a student network for the target task while encouraging high mutual information with the teacher network:

$$\mathcal{L} = \mathcal{L}_{\mathcal{S}} - \sum_{k=1}^{K} \lambda_k I(\boldsymbol{t}^{(k)}, \boldsymbol{s}^{(k)}), \tag{7.2}$$

where $\mathcal{L}_{\mathcal{S}}$ is the task-specific loss function for the target task and $\lambda_k > 0$ is the hyper-parameter introduced for regularization of the mutual information. Equation eq.(7.2) needs to be minimized with respect to the parameters of the student network. However, the minimization is hard since the computation of the exact mutual information is intractable. We instead propose a variational lower bound for each mutual information term $I(\boldsymbol{t}; \boldsymbol{s})$, in which we define a variational



(a) input (b) 0-th epoch (c) 40-th epoch (d) 160-th epoch (e) no transfer (f) magnitude of $t_{h,w}$

Figure 2: Plots for the heat maps corresponding to the variational distribution evaluated for spatial dimensions of the intermediate layer in the teacher network, *i.e.*, $\log q(\boldsymbol{t}_{h,w}|\boldsymbol{s}) = \sum_c \log q(t_{c,h,w}|\boldsymbol{s})$. Each figure corresponds to (a) original input image, (b, c, d) log-likelihood $\log q(\boldsymbol{t}_{h,w}|\boldsymbol{s})$ that was normalized and interpolated to fit the spatial dimension of the input image (red pixels correspond to high probability), (d) log-likelihood of variational distribution optimized for the student network trained without any knowledge transfer applied and (f) magnitude of the layer $\boldsymbol{t}$ averaged for each spatial dimensions.

distribution $q(\boldsymbol{t}|\boldsymbol{s})$ that approximates $p(\boldsymbol{t}|\boldsymbol{s})$:

$$\begin{aligned} I(\boldsymbol{t};\boldsymbol{s}) &= H(\boldsymbol{t}) - H(\boldsymbol{t}|\boldsymbol{s}) \\ &= H(\boldsymbol{t}) + \mathbb{E}_{\boldsymbol{t},\boldsymbol{s}}[\log p(\boldsymbol{t}|\boldsymbol{s})] \\ &= H(\boldsymbol{t}) + \mathbb{E}_{\boldsymbol{t},\boldsymbol{s}}[\log q(\boldsymbol{t}|\boldsymbol{s})] + \mathbb{E}_{\boldsymbol{s}}[D_{\mathrm{KL}}(p(\boldsymbol{t}|\boldsymbol{s})||q(\boldsymbol{t}|\boldsymbol{s}))] \\ &\geq H(\boldsymbol{t}) + \mathbb{E}_{\boldsymbol{t},\boldsymbol{s}}[\log q(\boldsymbol{t}|\boldsymbol{s})], \end{aligned} \tag{7.3}$$

where the expectations are over the distribution $p(\boldsymbol{t},\boldsymbol{s})$ and the last inequality is due to the non-negativity of the Kullback-Leiber divergence $D_{\mathrm{KL}}(\cdot)$. Such a technique is known as the *variational information maximization* (Agakov 2004). Finally, we obtain VID by applying the variational information maximization to each mutual information term $I(\boldsymbol{t}^{(k)},\boldsymbol{s}^{(k)})$ in eq.(7.2), leading to a minimization of the following loss function:

$$\widetilde{\mathcal{L}} = \mathcal{L}_{\mathcal{S}} - \sum_{k=1}^{K} \lambda_k \mathbb{E}_{\boldsymbol{t}^{(k)},\boldsymbol{s}^{(k)}}[\log q(\boldsymbol{t}^{(k)}|\boldsymbol{s}^{(k)})]. \tag{7.4}$$

The objective $\widetilde{\mathcal{L}}$ is jointly minimized over the parameters of the student network and the variational distribution $q(\boldsymbol{t}|\boldsymbol{s})$. Note that the entropy term $H(\boldsymbol{t})$ has been removed from the equation eq.(7.3) since it is constant with respect to the parameters to be optimized. Alternatively, one could interpret the objective eq.(7.4) as jointly training the student network for the target task and maximization of the

conditional likelihood to fit the activations of the selected layers from the teacher network. By doing so, the student network obtains the "compressed" knowledge required for recovering activations of the selected layers in the teacher network.

7.2.1 Algorithm formulation

We further specify our framework by choosing a form made for the variational distribution $q(\boldsymbol{t}|\boldsymbol{s})$. In general, we employ a Gaussian distribution with heteroscedastic mean $\boldsymbol{\mu}(\cdot)$ and homoscedastic variance $\boldsymbol{\sigma}$ as the variational distribution $q(\boldsymbol{t}|\boldsymbol{s})$, *i.e.*, the mean $\boldsymbol{\mu}(\cdot)$ is a function of $\boldsymbol{s}$ and the standard deviation $\boldsymbol{\sigma}$ is not. Next, the parameterization of $\boldsymbol{\mu}(\cdot)$ and $\boldsymbol{\sigma}$ is further specified by the type of layer corresponding to $\boldsymbol{t}$. When $\boldsymbol{t}$ corresponds to intermediate layer of the teacher network with spatial dimensions indicating channel, height and width respectively, *i.e.*, $\boldsymbol{t} \in \mathbb{R}^{C\times H\times W}$, our choice of variational distribution is expressed as follows:

$$\begin{aligned} -\log q(\boldsymbol{t}|\boldsymbol{s}) &= -\sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W}\log q(t_{c,h,w}|\boldsymbol{s}) \\ &= \sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W}\log\sigma_c + \frac{(t_{c,h,w}-\mu_{c,h,w}(\boldsymbol{s}))^2}{2\sigma_c^2} + \text{constant}, \end{aligned} \tag{7.5}$$

where $t_{c,h,w}$ denote scalar components of $\boldsymbol{t}$ indexed by (c,h,w). Further, $\mu_{c,h,w}$ represents the output of a single unit from the neural network $\boldsymbol{\mu}(\cdot)$ consisting of convolutional layers and the variance is ensured to be positive using the softplus function, *i.e.*, $\sigma_c^2 = \log(1+\exp(\alpha_c)) + \epsilon$ where $\alpha_c \in \mathbb{R}$ being the parameter to be optimized and $\epsilon > 0$ is minimum variance introduced for numerical stability. Typically, one can choose $\boldsymbol{s}$ from the student network with similar hierarchy and spatial dimension as $\boldsymbol{t}$. When spatial dimension of two layers are equal, 1×1 convolutional layers are typically used for efficient parameterization of $\boldsymbol{\mu}(\cdot)$. Otherwise, convolution or transposed convolution with larger kernel size could be used to match the spatial dimensions.

We additionally consider the case when the layer $\boldsymbol{t} = \mathcal{T}^{(\text{logit})}(\boldsymbol{x}) \in \mathbb{R}^N$ corresponds to the logit layer of the teacher network. Here, our choice of the variational distribution is expressed as follows:

$$\begin{aligned} -\log q(\boldsymbol{t}|\boldsymbol{s}) &= -\sum_{n=1}^{N}\log q(t_n|\boldsymbol{s}) \\ &= \sum_{n=1}^{N}\log\sigma_n + \frac{(t_n-\mu_n(\boldsymbol{s}))^2}{2\sigma_n^2} + \text{constant}, \end{aligned} \tag{7.6}$$

where t_n indicates the n-th entry of the vector $\boldsymbol{t}$, μ_n represents the output of a single unit of neural network $\boldsymbol{\mu}(\cdot)$ and σ_n is parameterized by softplus function to enforce positivity. For this case, the corresponding layer $\boldsymbol{s}$ in the student network is the penultimate layer $\mathcal{S}^{(\text{pen})}$ instead of the logit layer in order to match the hierarchy of two layers without being too restrictive on the output of the student network. Furthermore, we found that using a simple linear transformation for the

parameterization of the mean function was sufficient in practice, *i.e.*, $\boldsymbol{\mu}(\boldsymbol{s}) = \mathbf{W}\boldsymbol{s}$ for some weight matrix $\mathbf{W}$.

The aforementioned implementations turned out to perform satisfactorily during the experiments. We also considered using heteroscedastic variance $\boldsymbol{\sigma}(\cdot)$, but it gave unstable training with ignorable improvements. Other types of parameterizations such as a heavy-tailed distribution or the mixture density network (Bishop 1994) could be used to gain additional performance, but we leave it for future exploration.

See Figure 2 for an illustration of the training VID using the implementation based on equation eq.(7.5). Here, we display the change in the evaluated log-likelihood of the variational distribution aggregated over channels, *i.e.*, $\log q(\boldsymbol{t}_{h,w}|\boldsymbol{s}) = \sum_c \log q(t_{c,h,w}|\boldsymbol{s})$, given input $\boldsymbol{x}$ (Figure 2(a)) throughout the VID training process. One observes that the student network is trained gradually for the variational distribution to estimate the density of the intermediate layer from the teacher network (Figure 2(b), 2(c) and 2(d)). As a comparison, we also optimize the variational distribution for the student network trained without knowledge transfer, (Figure 2(e)). For this case, we observe that this particular instance of the variational distribution fails to obtain high log-likelihoods, indicating low mutual information between the teacher and the student networks. Interestingly, the parts that correspond to the background achieve higher magnitudes compared to that of the foreground in general. Our explanation is that the output of layers corresponding to the background that mostly corresponds to zero activations (Figure 2(f)) and contains less information, being a relatively easier target for maximizing the log-likelihood of the variational distribution.

7.2.2 Connections to existing works

The infomax principle

We first describe the relationship between our framework and the celebrated *infomax principle* (Linsker 1989) applied to representation learning (Vincent et al. 2010), stating that "good representation" is likely to contain much information in the corresponding input. Especially, such a principle has been successfully applied to semi-supervised learning for neural networks by maximizing the mutual information between the input and output of the intermediate layer as a regularization to learning the target task, *e.g.*, learning to reconstruct input based on autoencoders Rasmus et al. (2015). Interestingly, our framework can be viewed similarly as an instance of semi-supervised learning with modification of the infomax principle: layers of the teacher network contain important information for the target task, and a good representation of the student network is likely to retain much of their information. One recovers the traditional semi-supervised learning based on the infomax principle when we set $\boldsymbol{t}^{(k)} = \boldsymbol{x}$ in the equation eq.(7.2).

Generalizing mean squared error matching

Next, we explain how existing knowledge transfer methods based on mean squared error matching can be seen as a specific instance of the proposed framework.

In general, the methods will be induced from the equation eq.(7.4) by making a specific choice of the layers $\mathcal{R} = \{(\mathcal{T}^{(k)}, \mathcal{S}^{(k)})\}_{k=1}^{K}$ for knowledge transfer and parameterization of heteroscedastic mean $\boldsymbol{\mu}(\cdot)$ in the variational distribution:

$$-\log q(\boldsymbol{t}|\boldsymbol{s}) = \sum_{n=1}^{N} \frac{(t_n - \mu_n(\boldsymbol{s}))^2}{2} + \text{constant}. \tag{7.7}$$

Note that the equation eq.(7.7) corresponds to a Gaussian distribution with unit variance over every dimension of the layer in the teacher network. Ba and Caruana Ba and Caruana (2014) showed that knowledge can be transferred between the teacher and the student networks that were designed for the same task, by matching the output of logit layers $\mathcal{T}^{(\text{logit})}, \mathcal{S}^{(\text{logit})}$ from the teacher and the student networks with respect to mean squared error. Such a formulation is induced from the equation eq.(7.7) by letting $\mathcal{R} = \{(\mathcal{T}^{(\text{logit})}, \mathcal{S}^{(\text{logit})})\}$, and $\boldsymbol{\mu}(\boldsymbol{s}) = \boldsymbol{s}$ in the equation eq.(7.7). This was later extended for knowledge transfer between the teacher and the student networks designed for different tasks by Li and Hoiem Li and Hoiem (2017), through adding an additional linear layer on top of the penultimate layer $\mathcal{S}^{(\text{pen})}$ in the student network to matching with logit layer $\mathcal{T}^{(\text{logit})}$ in the teacher network. This is induced similarly from the equation eq.(7.7) by letting $\mathcal{R} = \{(\mathcal{T}^{(\text{logit})}, \mathcal{S}^{(\text{pen})})\}$, and $\boldsymbol{\mu}(\cdot)$ being a linear transformation, *i.e.*, $\boldsymbol{\mu}(\boldsymbol{s}) = \mathbf{W}\boldsymbol{s}$. Next, Romero *et al.* Romero et al. (2014) proposed a knowledge transfer loss for minimizing the mean squared error between intermediate layers from the teacher and the student networks, with additional convolutional layer introduced for adapting different dimension size between each pair of matched layers. This is recovered from the regularization term in the equation eq.(7.7) by choosing layers for the knowledge transfer to be intermediate layers of the teacher and the student networks, and $\boldsymbol{\mu}(\cdot)$ being a linear transformation corresponding to a single 1×1 convolutional layer.

In general, the described methods are similar to our implementation of the framework as they all use Gaussian distribution as the variational distribution. However, our method differs mainly in two ways: (a) allowing to use more flexible nonlinear functions for heteroscedastic mean and (b) modeling different variance for each dimension in the variational distribution. This allows transferring mutual information in a more flexible manner without wasting much capacity of the model. Especially, modeling unit variance for all dimensions of the layer $\boldsymbol{t}$ in the teacher network could be highly restrictive for the student network. To illustrate, the layer of the teacher network might include an activation t_n that contains information irrelevant to the task of the student network, yet requires much capacity for regression of $\mu_n(\boldsymbol{s})$ to t_n. This would raise over-regularization issues, *i.e.*, wasting the majority of the student network's capacity on trying to fit such a unit. Instead, modeling high homoscedastic variance σ_n for such dimension make its contribution ignorable to the overall loss, allowing one to "filter" out such unit in an efficient way.

Comparison with feature matching

Besides the knowledge transfer methods based on mean squared error matching, several works (Chen et al. 2018, Huang and Wang 2017, Yim et al. 2017, Zagoruyko and Komodakis 2016b) have proposed to indirectly match the handcrafted features extracted from intermediate layers. More specifically, Zagoruyko and Komodakis (2016b) proposed to match the "attention maps" generated from activations from the layers. Huang and Wang (2017) later generalized the attention map to the maximum mean discrepancy of the activations. Yim et al. (2017) proposed to match the feature called Flow of Solution Procedure (FSP) defined by the Gram matrix of layers adjacent in the same network. Chen et al. (2018) considered matching the reconstructed input image from the intermediate layers of the teacher and the student networks. These methods could be seen as smartly avoiding the aforementioned over-regularization issue by filtering out information in the teacher network using expert knowledge. However, such methods potentially lead to suboptimal results when the feature extraction method is not apt for the particular knowledge transfer task and may discard important information from the layer of the teacher network in an irreversible way.

7.3 Experiments

We demonstrate the performance of the proposed knowledge transfer framework by comparing VID to state-of-the-art knowledge transfer methods on image classification. We apply VID to two different locations: (a) VID between intermediate layers of the teacher and the student network (VID-I) and (b) VID between the logit layer of the teacher network and the penultimate layer of the student network (VID-LP). For comparison, we consider the following knowledge transfer methods: the original knowledge distillation (KD) Hinton et al. (2015), learning without forgetting (LwF) Li and Hoiem (2017), hint based transfer (FitNet) (Zagoruyko and Komodakis 2016b), activation-based attention transfer (AT) Zagoruyko and Komodakis (2016b) and polynomial kernel-based neural selectivity transfer (NST) Huang and Wang (2017). Note that we consider FitNet as a regularization for training the student network (Zagoruyko and Komodakis 2016b) instead of a stage-wise training procedure as first proposed in (Romero et al. 2014). We compare knowledge transfer methods for knowledge transfer between same and different datasets, which is commonly referred to as the knowledge distillation and transfer learning tasks respectively.

In all the experiments, we select the same pairs of intermediate layers for knowledge transfer based on VID-I, FitNet, AT and NST. Similarly, the same pairs of layers for knowledge transfer are used for LwF and VID-LP. All the hyper-parameters of both our and existing methods are chosen according to the performance on a validation set, which is 20% of the training set. We carefully pick the set of candidate values of hyper-parameters such that all the values proposed in the original works are included. The presented performances are the average of three repeated runs. More details about experiments are included in

M	5000	1000	500	100
Teacher	94.26	-	-	-
Student	90.72	84.67	79.63	58.84
KD	91.27	86.11	82.23	64.24
FitNet	90.64	84.78	80.73	68.90
AT	91.60	87.26	84.94	73.40
NST	91.16	86.55	82.61	64.53
VID-I	**91.85**	**89.73**	**88.09**	**81.59**
KD + AT	91.81	87.34	85.01	76.29
KD + VID-I	91.7	88.59	86.53	78.48

Table 1: Experimental results (test accuracy) of knowledge distillation on the CIFAR-10 dataset from teacher network (WRN-40-2) to student network (WRN-16-1) with varying number of data points per class (denoted by M).

the supplementary material. The implementation of the algorithm will be made publicly available shortly.

7.3.1 Knowledge distillation

We first compare knowledge transfer methods on the traditional knowledge distillation task, where a student network is trained on the same task as the teacher network. By distilling the knowledge from a large teacher network into a small student network, we can speed up the computation for prediction. We further investigate two problems for this task: whether we can benefit from knowledge transfer in the small data regime and how much performance we lose by reducing the size of the student network? Note that we do not evaluate the performance of VID-LP and LwF as they are designed for transfer learning. When applied, KD, VID-LP and LwF delivered similar performance.

Reducing training data

Knowledge transfer can be a computationally expensive task. Given a pre-trained teacher network on the whole training data set, we explore the possibility of using a small portion of the training set for knowledge transfer. We demonstrate the effect of a reduced training set by applying knowledge distillation on CIFAR-10 (Krizhevsky 2009b) with four different sizes of training data. We employ wide residual networks (WRN) (Krizhevsky 2009b) for the teacher network (WRN-40-2) and the student network (WRN-16-1), where the teacher network is pre-trained on the whole training set of CIFAR-10. Knowledge distillation is applied to four different sizes of training set: 5000 (the full size), 1000, 500, 100 data points per class.

We compare VID-I with KD, FitNet, AT and NST. We also provide performances of the teacher network (Teacher) and the student network trained without any knowledge transfer (Student) as baselines. We choose four pairs of inter-

(d, w)	(40,2)	(16, 2)	(40, 1)	(16, 1)
Teacher	74.16	-	-	-
Student	74.34	70.42	68.79	65.46
KD	75.80	72.87	70.99	66.03
FitNet	74.29	70.89	68.66	65.38
AT	74.76	71.06	69.85	65.31
NST	74.81	71.19	68.00	64.95
VID-I	75.25	73.31	71.51	66.32
KD + AT	75.86	73.13	71.4	67.07
KD + VID-I	**76.11**	**73.69**	**72.16**	**67.19**

Table 2: Experimental results (test accuracy) of knowledge distillation on the CIFAR-100 dataset from the teacher network (WRN-40-2) to the student networks (WRN-d-w) with varying factor of depth d and width w.

mediate layers similarly to (Zagoruyko and Komodakis 2016b), each of which is located at the end of a group of residual blocks. We implemented VID-I using two 1×1 convolutional layers with hidden channel size as twice of the output channel size. The results are shown in Table 1. Our method, VID-I, outperforms other knowledge transfer methods consistently across all regimes. The performance gap increases as the size of dataset get smaller, *e.g.*, VID-I only drops 10.26% of accuracy even when 100 data points per each class are provided to the student network. There is a 31.88% drop without knowledge transfer and a 15.52% drop for the best baseline, *i.e.*, KD + AT.

Varying the size of the student network

The size of the student network gives a trade-off between the speed and the performance in knowledge transfer. We evaluate the performance of knowledge transfer methods on different sizes of the student network. The teacher network (WRN-40-2) is pre-trained on the whole training set of CIFAR-100. A student network with four choices of size, *i.e.*, WRN-40-2, WRN-16-2, WRN-40-1, WRN-16-1, is trained on the whole training set of CIFAR-100. We compare our VID-I with KD, FitNet, AT and NST along with the Teacher and Student baselines. The choices of intermediate layers are the same as the previous experiment.

The results are shown in in Table 1. As also noticed by Furlanello et al. (2018), the student network with the same size as the teacher network outperforms the teacher network with all the knowledge transfer methods. One observes that VID-I consistently outperforms FitNet, AT and NST, which correspond to the same choice of layers for knowledge transfer. It also outperforms KD except for the case when the structure of the student network is identical to that of the teacher network, *i.e.*, WRN-40-2, where two methods can be combined to yield the best performance.

M	≈80	50	25	10
Student	48.13	37.69	27.01	14.25
Finetuned	70.97	66.04	58.13	47.91
LwF	63.43	51.79	41.04	22.76
FitNet	71.34	60.45	54.78	36.94
AT	58.21	48.66	43.66	27.01
NST	55.52	46.34	33.21	20.82
VID-LP	67.91	58.51	47.09	31.94
VID-I	71.34	63.66	60.07	**50.97**
LwF + FitNet	70.97	60.37	54.48	38.73
VID-LP + VID-I	**71.87**	**65.75**	**61.79**	50.37

(a) MIT-67, ResNet-34 to ResNet-18

M	≈80	50	25	10
Student	53.58	43.96	29.70	15.97
Finetuned	65.97	58.51	51.72	39.63
LwF	60.90	52.01	41.57	27.76
FitNet	70.90	64.70	54.48	40.82
AT	60.90	52.16	42.76	25.60
NST	55.60	46.04	35.22	21.64
VID-LP	68.88	61.64	50.22	39.25
VID-I	**72.01**	**67.01**	**59.33**	**45.90**
LwF + FitNet	70.52	64.10	54.63	40.15
VID-LP + VID-I	71.72	66.49	58.96	45.89

(b) MIT-67, ResNet-34 to VGG-9

Table 3: Experimental results (test accuracy) of transfer learning from the teacher network (ResNet-34) to the student network (ResNet-18/VGG-9) for the MIT-67 dataset with varying number of data points per class (denoted by M). We use $M \approx M_{\text{avg}}$ to denote the setting where the number of data points per class is non-uniform and M_{avg} in average.

7.3.2 Transfer learning

We evaluate knowledge transfer methods on transfer learning. The teacher network is a residual network (ResNet-34) (He et al. 2016c) pre-trained on the ImageNet dataset (Russakovsky et al. 2015). We apply transfer learning to improve the performance of two separate image classification tasks. The first task is a fine-grained bird species classification based on the CUB-200-2011 dataset (Welinder et al. 2010), which contains 11,788 images in total for 200 bird species. The second task is an indoor scene classification based on the MIT-67 dataset (Quattoni and Torralba 2009), which contains 15,620 images for 67 classes of indoor scenes. For both tasks, there are a relatively few numbers of images per class, which can significantly benefit from knowledge transfer from the ImageNet classification task. To evaluate the performance at various levels of data scarcity, we subsample both

M	≈29.95	20	10	5
Student	37.22	24.33	12.00	7.09
Finetuned	76.69	71.00	59.25	44.07
LwF	55.18	42.13	26.23	14.27
FitNet	66.63	56.63	46.68	31.04
AT	54.62	41.44	28.90	16.55
NST	55.01	41.87	23.76	15.63
VID-LP	65.59	54.12	39.20	27.86
VID-I	**73.25**	**67.20**	**56.86**	**46.21**
LwF + FitNet	68.69	58.81	48.86	31.30
VID-LP + VID-I	69.71	63.94	52.87	41.12

(a) CUB-200-2011, ResNet-34 to ResNet-18

M	≈29.95	20	10	5
Student	44.59	32.10	15.69	9.66
Finetuned	60.96	51.86	46.88	39.98
LwF	52.18	38.05	25.57	13.93
FitNet	68.96	61.52	48.04	32.89
AT	56.28	43.96	28.33	13.98
NST	56.55	44.95	28.43	14.66
VID-LP	66.82	55.94	38.10	30.47
VID-I	**71.51**	**65.69**	53.29	38.09
LwF + FitNet	70.56	62.44	47.36	30.52
VID-LP + VID-I	70.00	65.14	**53.78**	**38.76**

(b) CUB-200-2011, ResNet-34 to VGG-9

Table 4: Experimental results (test accuracy) of transfer learning from the teacher network (ResNet-34) to the student network (ResNet-18/VGG-9) for the CUB-200-2011 dataset with varying number of data points per class (denoted by M). We use $M \approx M_{\text{avg}}$ to denote the setting where the number of data points per class is non-uniform and M_{avg} in average.

datasets into three different sizes (50, 25, 10 per class for MIT-67 and 20, 10, 5 per class for CUB-200-2011) and compare the knowledge transfer methods.

We evaluate the knowledge transfer methods in two scenarios: a smaller student network of the same architecture (ResNet-18) and different architecture (VGG-9) (Simonyan and Zisserman 2014b). We compare our VID-I and VID-LP with LwF, FitNet, AT and NST. We evaluate the performance of the student network without transfer learning (Student) as a baseline. For the teacher and the student network with ResNet architecture, we choose the outputs of the third and fourth groups of residual blocks (from the input) as the intermediate layers for knowledge transfer. In the case of the VGG-9 student network, we choose the fourth and fifth max-pooling layers as the intermediate layers for knowledge transfer, which corresponds to the same spatial dimension as the intermediate layers selected from the teacher network. For applying VID-I to the ResNet-18 student network, we

Network	MLP-4096	MLP-2048	MLP-1024
Student	70.60	70.78	70.90
KD	70.42	70.53	70.79
FitNet	76.02	74.08	72.91
VID-I	**85.18**	**83.47**	**78.57**
Urban *et al.* Urban et al. (2017)		74.32	
Lin *et al.* Lin et al. (2015)		78.62	

Table 5: Experimental result (test accuracy) of distillation on CIFAR-10 from the convolutional teacher network (WRN-40-2) to the fully connected student network (MLP-h) with varying size of hidden dimensions h.

use two 1×1 convolutional layers with the size of intermediate channels as half of the output channel size. When the student network is VGG-9, a single 1×1 convolutional layer without non-linearity is used.

The results are shown in Table 3 and 4. The knowledge transfer from ResNet-34 to VGG-9 gives a very similar performance to the transfer from ResNet-34 to ResNet-18 for all the knowledge transfer methods. This shows that knowledge transfer methods are robust against small architecture changes. Our methods outperform other knowledge transfer methods in all regions of comparison. Both VID-I and VID-LP outperforms baselines that correspond to the same choice of layers for knowledge transfer. For the MIT-67 dataset, we observe that our algorithm outperforms even the finetuning method, which requires pre-training of the student network on the source task.

7.3.3 Knowledge transfer from CNN to MLP

The transfer learning experiments show the robustness of the knowledge transfer method against small architecture changes. This leads to an interesting question: whether a knowledge transfer method can work between two completely different network architectures. A solution to this question can open a new direction of knowledge transfer and potentially offer solutions to many problems, *e.g.*, speeding up prediction of recurrent neural networks (RNNs) by transferring knowledge from a RNN to a CNN, speeding up prediction of CNN on CPU or low-energy device by transferring knowledge from a CNN to a multi-layer perceptron (MLP). In this paper, we evaluate the performance of knowledge transfer from CNN to MLP on CIFAR-10.

There is a well-known performance gap between CNN and MLP on CIFAR-10 (Lin et al. 2015, Urban et al. 2017). The state-of-the-art performance on CIFAR-10 with MLP is 78.62% with initialization from auto-encoders by Lin et al. (2015) and 74.32% using knowledge distillation by Urban et al. (2017). Urban et al. (2017) also trained a single convolutional layer achieving the performance of 84.6% using knowledge distillation.

We apply the knowledge transfer methods in the knowledge distillation setting

as mentioned in Section 7.3.1. We use a teacher network with convolutional layers (WRN-40-2) pre-trained on CIFAR-10. We use a MLP with five fully connected hidden layers as the student network, constructed by stacking one linear layer, three bottleneck linear layers and one linear layer in sequence. Each is followed by a non-linearity activation in between. Here, the bottleneck layer indicates a composition of two linear layers without non-linearity that is introduced to speed up learning by reducing the number of parameters. All the hidden layers have the same h number of units and the bottleneck linear layer is composed of two linear layers with a size of $h \times \frac{h}{4}$ and $\frac{h}{4} \times h$.

The knowledge transfer between intermediate layers is defined between the outputs of four residual groups of the teacher network and the outputs of the first four fully connected layers of the student network. We compare VID-I with KD and FitNet since these knowledge transfer methods do not rely on spatial structures. For the same reason, AT and NST are not applicable to multilayer perceptrons. VID-I is implemented with multiple transposed convolutional layers without non-linearities. Specifically, the inputs for the variational distributions, *i.e.*, the hidden layers of the MLP is treated as a tensor with 1×1 spatial dimensions. Single transposed convolutional layer with a 4×4 kernel, unit stride and zero padding is followed by multiple transposed convolutional layers with a 4×4 kernel, two strides, and single padding in order to match the spatial dimension of the corresponding layer of the teacher network for knowledge transfer. More details on implementations of the student network and the auxiliary distribution are in the supplementary material.

The results are shown in Table 5. Both FitNet and VID-I improve the performance comparing the baseline of directly training the intermediate layers of the student network. VID-I significantly outperforms FitNet on MLPs with different sizes. Furthermore, MLP-4096 outperforms the the state-of-the-art performance with MLP reported by Lin et al. (2015) (78.62%) and Urban et al. (2017) (74.32%) significantly. More importantly, our method bridges the performance gap between CNN (84.6% using one convolutional layer (Urban et al. 2017)) and MLP shown in previous works.

7.4 Conclusion

In this work, we proposed the VID framework for effective knowledge transfer by maximizing the variational lower bound of the mutual information between two neural networks. The implementation of our algorithm is based on Gaussian observation models and is empirically shown to outperform other benchmarks in the distillation and transfer learning tasks. Using more flexible recognition models, *e.g.*, Kingma et al. (2016), for accurate maximization of mutual information and alternative estimation of mutual information, *e.g.*, Belghazi et al. (2018), are of future interest. Despite our principled approach, there exist many unanswered questions in knowledge transfer, such as which attributes of the teacher network should be chosen as a target for transfer learning? which layers of the student network should we transfer the knowledge into? We hope to be able to answer

this from an information theoretic point of view in the future.

Chapter

8 Exploring Weight Symmetry in Deep Neural Networks

Abstract

We propose to impose symmetry in neural network parameters to improve parameter usage and make use of dedicated convolution and matrix multiplication routines. Due to significant reduction in the number of parameters as a result of the symmetry constraints, one would expect a dramatic drop in accuracy. Surprisingly, we show that this is not the case, and, depending on network size, symmetry can have little or no negative effect on network accuracy, especially in deep overparameterized networks. We propose several ways to impose local symmetry in recurrent and convolutional neural networks, and show that our symmetry parameterizations satisfy universal approximation property for single hidden layer networks. We extensively evaluate these parameterizations on CIFAR, ImageNet and language modeling datasets, showing significant benefits from the use of symmetry. For instance, our ResNet-101 with channel-wise symmetry has almost 25% fewer parameters and only 0.2% accuracy loss on ImageNet. Code for our experiments is available at https://github.com/hushell/deep-symmetry

8.1 Introduction

For a long time neural networks had a capacity problem: making them have too many parameters for a limited amount of data would dramatically affect their generalization capabilities. Thus, several regularization techniques were developed, for example, early stopping (Weigend and Huberman 1990) and L^2 regularization. The advent of batch normalization (Ioffe and Szegedy 2015a), skip-connections (Hochreiter and Schmidhuber 1997a, Srivastava et al. 2015, He et al. 2016b), and overall architecture search helped to mitigate this problem very recently, so that increasing capacity no longer hurts the accuracy, but even before that it was known that "unimportant" connections between neurons can be removed, resulting in a network with significantly fewer parameters and little or no drop in performance. Optimal brain damage (LeCun et al. 1990b) proposed a second order method to prune such connections. Soft weight sharing (Nowlan and Hinton 1992, Ullrich et al. 2017b) can be used to very effectively reduce the number of parameters in a trained network. Denil et al. (2013b) showed that it is possible to learn part of filters and predict the rest. In fact, parameter sharing is one of the most important features of convolutional neural networks, where sharing is built into convolutional layer, and was thoroughly explored. Recurrent neural networks also heavily rely on sharing parameters over multiple time steps. It is also the key element in training siamese and triplet networks.

More recently, trained on massive amounts of data image recognition neural networks were quickly increasing in the number of parameters, starting from AlexNet (Krizhevsky et al. 2012a) and VGG (Simonyan and Zisserman 2015). Network-In-Network (Lin et al. 2013b) proposed to stack MLPs which share local receptive fields, removing the need of massive fully-connected layers, and reducing the number of parameters to achieve the same accuracy as AlexNet by several times. This technique was further adopted by Szegedy et al. (2015a) in their Inception architectures. Downsampling and upsampling can also be seen as a way of parameter sharing in SqueezeNet (Iandola et al. 2016b). More recently, Highway (Srivastava et al. 2015) and later ResNet (He et al. 2016b) proposed to add skip-connections, which allowed training of very deep networks with improved accuracy. It was later shown by Wide ResNet (Zagoruyko and Komodakis 2016a) that the number of parameters was key to their accuracy, and depth was complementary. Either very deep, or very wide residual networks could be trained with massive number of parameters, without suffering from decreased accuracy. After that, parameter sharing exploration continued on ResNet architectures. For example, Boulch (2017) proposed to share some portion of convolutional filters in each group of residual block, for example every second convolution, achieving relatively small performance loss.

Not all parameter sharing approaches brought both performance improvement and reduction in parameters, and there appears to be some trade-off between parameter sharing and computational efficiency. As an extreme case of parameter sharing with significant performance loss, HyperNetworks (Ha et al. 2017b), proposed to have another network to generate filters. Among methods improving

computational efficiency, a good example is MobileNet (Howard et al. 2017b), which suggested to reparameterize each 3×3 convolution as a pair of depth-wise convolution and 1×1 convolution, can also be seen as a way of sharing parameters. Their work was later extended to grouped 1×1 convolution by ShuffleNet (Zhang et al. 2018), which further reduced the complexity of residual block.

Another approach to reducing number of parameters is post processing, when a trained networks is modified, either with or without additional fine tuning. A number of low rank approximation by tensor decomposition approaches were proposed, such as works of Jaderberg et al. (2014b) and Lebedev et al. (2016). For example, Denton et al. (2014b) used low-rank decomposition of the weight matrices to reduce the effective number of parameters in the network. Such approaches tend to be less applicable as more parameter-effective architectures are being invented. A very effective way of reducing parameters in a trained network is deep compression proposed by Han et al. (2016), with a reduction of parameters of dozens of times, achieved by pruning, weight sharing, quantization and compression. Weight sharing locations are determined by weight values in a trained network, similar in value neurons share the same weight. Common approach today is to train a massive network, and reduce it later. This can even improve results over single-time trained network (Han et al. 2017).

Recurrent neural networks are known to be significantly overparameterized as well. For example, Kim and Rush (2016) show that it is possible to reduce neural machine translation model size by 13× with an insignificant BLEU score drop by doing teacher-student knowledge distillation. Several works (Merity et al. 2016, Inan et al. 2016, Kim et al. 2016) focus on reducing number of parameters in language modeling tasks.

All of the above suggest that architectures and methods used for training deep neural networks are suboptimal, and could be significantly improved by organizing parameter sharing from the start. It would be interesting to learn it, and a few attempts were made, e.g. in DCT space with hashing (Chen et al. 2016b) and FFT space (Mathieu et al. 2014). Also, automatic architecture search approaches, such as (Zoph et al. 2018), do not handle sharing.

Despite so much work on reducing neural network parameters, surprisingly, very little attention has been given to weight symmetry. To the best of our knowledge, it has not been used in the context of parameter sharing so far. We propose a number of possible approaches to enforce symmetry on weights *from scratch*, i.e. not using a pretrained network. Imposed correctly, such symmetry can significantly reduce the number of parameters and computational complexity by sacrificing little or no accuracy, both at train and test time. Post-processing methods such as deep compression could be applied later. Also, our parameterizations are simple to implement in any modern framework with automatic differentiation. Besides, specialized routines for symmetric matrix multiplication and convolution could be used in both training and testing (Goto and Van De Geijn 2008, Nath et al. 2011, Igual et al. 2009).

We believe that our findings are surprising and uncover interesting properties of deep neural networks, which could led to further advances in understanding and

efficiency. Our contributions are summarized below.

- We propose an effective use of symmetry for model compression for the first time;
- Explore various ways of imposing symmetry constraints;
- Experimentally show that symmetry can successfully be imposed in various architectures and datasets, with no or little loss in accuracy;
- We show that our symmetric parameterization is generic and can be applied to both image classification and natural language processing tasks;

8.2 Symmetric reparameterizations

In this section we introduce several ideas to impose axial symmetry on the weights of convolutional / fully-connected layers, which prunes out a large fraction of redundant parameters and gains computational speed-up for both training and testing. Throughout this section we consider all operations applied to matrices, which can be easily extended to multidimensional tensors by block symmetry.

8.2.1 Motivation

Real symmetric matrices have only real eigenvalues, while rotation matrices whose rotation angles are positive have at least one complex eigenvalue. Thus, restricting the weight matrix to be symmetric for some layers is equivalent to forcing these layers to learn non-rotational transformations. In a deep neural network, this restriction, which can be considered as an inductive bias, should not hurt the overall performance, since different layers are encouraged to focus on particular transformations, and then a good representation of the input is built up by compositing all of them.

On the other hand, introducing symmetric weights for network compression have multiple benefits:

- Symmetric matrix multiplications could deliver potential speedup over generic BLAS routines.
- Training from scratch: it avoids additional fine tuning compared with post-processing methods (Jaderberg et al. 2014b, Lebedev et al. 2016, Han et al. 2016).
- Easy to combine with other compression methods (e.g. with ShaResNet (Boulch 2017)).

8.2.2 Soft constraints

We first try to see if it is possible to enforce symmetry constraints in a soft manner during training, which has the advantage that training procedure remains very similar to standard supervised training. We start by adding a penalty term to the training loss, which leads to a soft constraint on $\boldsymbol{W}$:

$$\mathcal{L}(\boldsymbol{W}) + \rho \left\| \text{vec}(\boldsymbol{W}) - \text{vec}(\boldsymbol{W}^\top) \right\|_p, \tag{8.1}$$

where $\mathcal{L}(\boldsymbol{W})$ is a task-specific loss with respect to $\boldsymbol{W}$; ρ is a hyper-parameter to control the slackness of the constraints $\boldsymbol{W} = \boldsymbol{W}^\top$; $\text{vec}(\boldsymbol{W})$ is an operator that vectorizes the matrix into a column vector. For the norm of the penalty term, we use either $p = 2$ or $p = 1$. However, L^1-norm turns out to be slightly more effective given that it promotes sparsity, which would possibly result in a larger number of exactly symmetric weights.

Due to the soft constraints, the resulting matrix is not guaranteed to be symmetric and so at test time we use only the upper triangular part.

8.2.3 Hard constraints

Alternatively, we can make use of a specific parameterization by a linear operator $T\colon \boldsymbol{W} \mapsto \hat{\boldsymbol{W}}$ for some layer in the neural network that explicitly encode symmetry, where $\boldsymbol{W}$ is the actual weight and $\hat{\boldsymbol{W}}$ is the constructed weight for that layer. We are interested in the case where $\boldsymbol{W}$ has fewer elements than $\hat{\boldsymbol{W}}$, which enables a reduction in weights while keeping exactly the same network architecture as if it is fully-parameterized. Note that we choose T to be nonparametric, thus the only additional burden of training is to forward and backward propagate through T; while in testing, since $\hat{\boldsymbol{W}}$ is symmetric, we only need to store the upper triangular of the learned $\hat{\boldsymbol{W}}$, which saves almost half of the space.

To this end, we propose different instantiations of $T(\boldsymbol{W})$ in terms of different ways to construct symmetric matrices.

Triangular parameterization

One of the simplest way to impose hard symmetry constraint is to define the linear operator T as a sum of the upper triangular matrix of $\boldsymbol{W} \in \mathbb{R}^{N\times N}$, its transpose $\boldsymbol{W}^\top$ and the diagonal vector $\boldsymbol{v} \in \mathbb{R}^N$:

$$\hat{\boldsymbol{W}} = T(\text{triu}(\boldsymbol{W}), \boldsymbol{v}) := \text{diag}(\boldsymbol{v}) + \text{triu}(\boldsymbol{W}) + \text{triu}(\boldsymbol{W})^\top, \tag{8.2}$$

where $\text{triu}(\boldsymbol{W})$ returns the upper triangular part of the matrix $\boldsymbol{W}$. Note that we use the full matrix $\boldsymbol{W}$ in the above expression just to simplify the notation. In fact, the triangular parameterization stores only the upper triangular of $\boldsymbol{W}$ and the main diagonal $\boldsymbol{v}$, while elements of the lower triangular will never be touched. Thus, the number of parameters in this parameterization is $\frac{1}{2}N(N+1)$, reduced by almost 2 times compared to full parameterization. Note that $(\text{triu}(\boldsymbol{W}), \boldsymbol{v})$ should be initialized from the same distribution as if $\boldsymbol{W}$ is learned directly. Due to strong sharing, the gradient with respect to $\boldsymbol{W}$ will be twice higher in magnitude, so the learning rate needs to be adjusted accordingly.

Average parameterization

Let us also consider a redundant, but more straightforward formulation of the reparameterization T, in which we keep $N \times N$ matrix $\boldsymbol{W}$ as the actual weight, and define T as a sum of $\boldsymbol{W}$ and its transpose $\boldsymbol{W}^\top$ divided by two:

$$\hat{\boldsymbol{W}} = T(\boldsymbol{W}) := \frac{1}{2}(\boldsymbol{W} + \boldsymbol{W}^\top). \tag{8.3}$$

Although this is a redundant parameterization to obtain a symmetric matrix, we wanted to explore its performance given that it is known that overparameterized networks are often easier to optimize / train (e.g., directly training compact networks is much more challenging compared to first training overparameterized ones and then properly pruning their parameters). In this case, the learning rate remains the same since the gradient has been scaled by definition. As in triangular parameterization, $\boldsymbol{W}$ is initialized from the same distribution as basic non-symmetric parameterization. The number of parameters is the same as non-symmetric parameterization at training time, but $\frac{1}{2}N(N+1)$ at testing time due to the fact that $\hat{\boldsymbol{W}}$ is symmetric.

Eigen parameterization

In addition, we consider a more generic eigen parameterization, inspired by the fact that any symmetric matrix has an eigen decomposition. Given a matrix $\boldsymbol{V} \in \mathbb{R}^{N \times R}$ and a vector $\lambda \in \mathbb{R}^R$ as actual weights, T is defined by

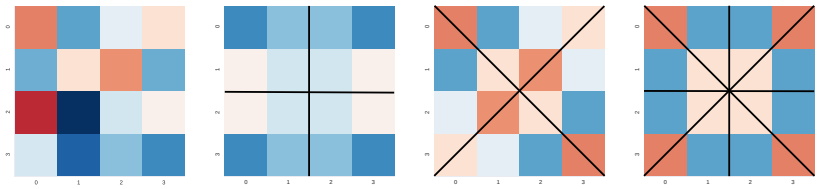
$$\hat{\boldsymbol{W}} = T(\boldsymbol{V}, \lambda) := \boldsymbol{V}\mathrm{diag}(\lambda)\boldsymbol{V}^\top. \tag{8.4}$$

Ideally, $\boldsymbol{V}$ has to be an orthogonal matrix, but we relax this constraint due to a heavy computation of performing projected stochastic gradient descent. We still initialize $\boldsymbol{V}$ and λ from an eigen decomposition of the initial full weight matrix. The number of parameters in such relaxed parameterization is $N(R+1)$ at training time, and $\frac{1}{2}N(N+1)$ at testing time. We empirically choose $R = N/2$ as it yields similar performance as the case of $R = N$.

Note that a similar approach was suggested by Denil et al. (2013b), where $\hat{\boldsymbol{W}}$ is parameterized by matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ with columns of $\boldsymbol{U}$ forming a dictionary of basis functions.

LDL parameterization

There is a close relationship between the eigen decomposition of a matrix and its LDL decomposition. Recall that the LDL decomposition factorizes a matrix as a product of an unit lower triangular matrix $\boldsymbol{L}$ (meaning that all elements on the diagonal are 1's), a diagonal matrix $\boldsymbol{D}$ and the transpose of $\boldsymbol{L}$. The advantage of using LDL decomposition over eigen decomposition is that it is much easier to maintain a valid decomposition of $\hat{\boldsymbol{W}}$ during training. We thus also consider



(a) (b) (c) (d)

Figure 1: N-way parameterizations. (a) Original 4×4 weight matrix. (b) 4-way blocking: $\boldsymbol{V}$ is the bottom-right block; $\hat{\boldsymbol{W}} = \text{reflect}_{-}(\text{reflect}_{|}(\boldsymbol{V}))$. (c) 4-way triangulizing: $\boldsymbol{V}$ is the top triangle; $\hat{\boldsymbol{W}} = \text{reflect}_{/}(\text{reflect}_{\backslash}(\boldsymbol{V}))$. (d) 8-way triangulizing: $\boldsymbol{V}$ is the top-left triangle; $\hat{\boldsymbol{W}} = \text{reflect}_{/}(\text{reflect}_{\backslash}(\text{reflect}_{|}(\boldsymbol{V})))$.

a LDL parameterization with additional assumptions[1] on $\hat{\boldsymbol{W}}$. Specifically, the reparameterization T with actual weights $\boldsymbol{L}$ and $\boldsymbol{D}$ is given by

$$\hat{\boldsymbol{W}} = T(\boldsymbol{L}, \boldsymbol{D}) := \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^\top, \tag{8.5}$$

where $\boldsymbol{L}$, $\boldsymbol{D}$ are restricted to be unit lower triangular matrix and diagonal matrix respectively.

N-way symmetry parameterization

Inspired by triangular parameterization, which can be viewed as an axial symmetry about the main diagonal, we consider a more general N-way parameterization with respect to multiple axes of symmetry, where N denotes the number of repeated parts. Thus, previously introduced parameterizations are 2-way symmetries.

Given $\boldsymbol{V}$ as the actual weight, which is considered in general to be smaller than the required weight matrix $\boldsymbol{W}$, we create a symmetrized version of $\boldsymbol{W}$ by a composition of linear transformations

$$\hat{\boldsymbol{W}} = T_1 \circ T_2 \circ \ldots \circ T_k(\boldsymbol{V}), \tag{8.6}$$

where $T_j(\cdot)$ is a basic linear operator which does one of the following things: translation, reflection, rotation, tessellation etc. In fact, the form of N-way symmetry is quite flexible. We consider only the cases where symmetry leads to efficient computations in testing. In this work, we focus on two N-way parameterizations: *blocking* and *triangulizing*. The details are listed as follows.

- Blocking: Given $\boldsymbol{V}$ as a $\frac{M}{\sqrt{N}} \times \frac{M}{\sqrt{N}}$ matrix, N-way blocking can be obtained by a series of reflections (denoted by $\text{reflect}_{\text{axis}}(\cdot)$) to include mirrors.
- Triangulizing: Given $\boldsymbol{V}$ as an isosceles right triangle with area $\frac{M^2}{N}$, $\hat{\boldsymbol{W}}$ is obtained by a series of reflections about different axes.

We demonstrate several examples of N-way symmetries in Figure 1.

[1] If $\hat{\boldsymbol{W}}$ is symmetric and it factorizes as $\hat{\boldsymbol{W}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{U} = \boldsymbol{U}^\top\boldsymbol{D}\boldsymbol{L}^\top$, then by uniqueness, it follows that $\hat{\boldsymbol{W}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^\top$. However, $\hat{\boldsymbol{W}}$ has a LU decomposition if $\hat{\boldsymbol{W}}$ satisfies a particular rank condition studied by Okunev and Johnson (2005). Thus, we in fact assumes $\hat{\boldsymbol{W}}$ is better conditioned.

8.2.4 Combining with other methods

It is not surprising that hard-constrained symmetry parameterization can be viewed as a special weight sharing method. Nevertheless, symmetry parameterization is capable of taking advantage of special matrix computation routines to speed up both training and testing, which is not the case for unstructured weight sharing methods such as duplicating randomly picked elements to form $\hat{\boldsymbol{W}}$.

Symmetry parameterization can also be complementary to other parameter reduction or weight sharing methods. For example, we force weight sharing not only between residual blocks within the same stage (Boulch 2017) but also within the same residual block using symmetry parameterizations. We show in Table 3 that triangular parameterization can be combined with ResNeXt (Xie et al. 2016) and MobileNet (Howard et al. 2017b), which have already been designed to take advantage of weight sharing. Besides, post-processing methods (Han et al. 2016) can be applied on the trained symmetric weights with fine tuning, thus further reducing the number of parameters. We show experiments on combing channel-wise symmetry with other parameter reduction methods (e.g. ShaResNet (Boulch 2017)) in Section 8.4.1.

8.3 Implementations of block symmetry

The proposed symmetry parameterization is a generic parameter reduction method, which can be easily adapted to various network architectures. For example, convolutional / linear layers in feedforward networks (such as VGG (Simonyan and Zisserman 2015), ResNeXt (Xie et al. 2016), MobileNet (Howard et al. 2017b) etc.) and in recurrent networks (such as LSTM (Hochreiter and Schmidhuber 1997a), GRU (Cho et al. 2014) etc.) can be symmetrized. In the case of grouped convolution in ResNeXt, block symmetry can be imposed on each group kernel: for instance, suppose that filters are of shape $g \times N \times N \times k \times k$, where g is the number of groups, we can impose symmetry on dimensions of $N \times N$. However, symmetry is not directly applicable to DenseNet (Huang et al. 2016), as the number of filters grows with every layer. We conduct some experiments along this direction in Section 8.4.1.

8.3.1 Imposing symmetry in convolutional neural networks

We denote by $(\boldsymbol{W}, \tilde{\boldsymbol{W}})$ the whole set of parameters of the convolutional neural network, where $\boldsymbol{W}$ is the subset of parameters to be symmetrized, and $\tilde{\boldsymbol{W}}$ is the subset of free parameters. To be more specific, we assume there are totally L convolutional layers being reparameterized to equip symmetric parameters. That is, $\boldsymbol{W} := \{\boldsymbol{W}^l\}_{l=1}^L$ with $\boldsymbol{W}^l$ satisfying certain symmetric properties: $\boldsymbol{W}^l \in \mathbb{R}^{N_o \times N_i \times K_h \times K_w}$ is constructed so that the number of input channels is equal to the number of outputs channels (i.e. $N_i = N_o = N$) and the spatial domain is a square

(i.e. $K_h = K_w = K$). We propose to impose symmetry on slices of $\boldsymbol{W}^l$, namely, on the slice $\boldsymbol{W}_i^l$, which is a square matrix.

Depending on which direction to slice the tensor, we have *channel-wise symmetry* and *spatial symmetry*. In general, we can write $\boldsymbol{W}^l := \{\boldsymbol{W}_i^l\}_{i \in I}$. For channel-wise symmetry, $\boldsymbol{W}_i^l$ is a $N \times N$ symmetric matrix, and $I := \Big\{(k_h, k_w) \mid k_h, k_w \in \{1, \dots, K\}\Big\}$; For spatial symmetry, $\boldsymbol{W}_i^l$ is a $K \times K$ symmetric matrix, and $I := \Big\{(k_i, k_o) \mid k_i, k_o \in \{1, \dots, N\}\Big\}$. Since these two symmetries are not exclusive, we can indeed impose both at the same time.

In theory, both channel-wise and spatial symmetries will reduce the freedom of layers, and their ability to approximate functions. Enforcing them to a shallow network can be problematic, since it may significantly reduce the expressive power of the network. Deep highway and residual networks, on the other hand, are more robust to symmetric weights, since they have many layers that are capable to make relatively small changes and iteratively improve the representation of the input (Greff et al. 2017). In addition, spatial symmetry can be further motivated by the success of scattering networks (Bruna and Mallat 2013), whose filters are fixed as wavelets and constructed to enjoy certain symmetric properties.

We show in experiments that the proposed symmetries can be applied to several modern deep neural networks without suffering a significant drop in both training and testing accuracy.

8.3.2 Imposing symmetry in recurrent neural networks

We chose perhaps the most popular RNN variant, LSTM, which is known to be overparameterized, to experiment with symmetry. In a LSTM cell, the weight matrices between the hidden unit and gates (input, forget, output) as well as the weight matrix between the hidden unit and itself are square, so it is immediately valid to apply aforementioned symmetry parameterizations. This reduces about 25% parameters from the standard LSTM. We show in Section 8.4.3 that the symmetrized LSTM works as well as the standard LSTM in language modeling.

8.4 Experiments

This section is composed as follows. We start with CIFAR experiments, where we first test various symmetry parameterizations on wide residual networks (WRN) by Zagoruyko and Komodakis (2016a). After determining which parameterizations work best, we determine in which layers symmetry can be applied. We then test it with WRN of different widths and depths to determine the best configuration in terms of parameter reduction, computational complexity and simplicity. We also apply the proposed symmetrization to other architectures, and show that the conclusions drawn from CIFAR are able to transfer to larger datasets (ImageNet-1K dataset). Finally, we apply symmetrization to language modeling tasks.

We emphasize that our goal here is not to show state-of-the-art accuracy, but to show that very simple symmetry constraints can be used to significantly reduce

the number of parameters in various network architectures.

There are two common ResNet variants are considered as baselines in the following experiments: *basic* blocks and *bottleneck* blocks (He et al. 2016b). Basic blocks have two 3×3 convolutional layers and a parallel residual connection; bottleneck block is a combination of 1×1, 3×3 and 1×1 convolutional layers. Similar to WRN, we refer to WRN-n-k-blocktype as the network of depth n, width k (number of channels multiplier), and blocktype meaning either basic or bottleneck.

Code for all our experiments is available at https://github.com/hushell/deep-symmetry.

8.4.1 CIFAR experiments

The results of various experiments on CIFAR-10/100 datasets are presented below.

Symmetry parameterizations

We first compare various symmetry parameterizations on CIFAR-10 with WRN-16-1-bottleneck, which is a relatively small network enabling us to perform quick experiments. The median validation errors over 5 runs are reported in Table 1 as well as a comparison in terms of the number of parameters needed in training and testing.

symmetry parameterization	#parameters train	#parameters test	CIFAR-10
baseline (non-symmetric)	0.219M	0.219M	8.49
L^1 soft constraints	0.219M	0.172M	8.61
channelwise-triangular	0.172M	0.172M	8.84
channelwise-average	0.219M	0.172M	8.83
channelwise-eigen	0.173M	0.173M	10.23
channelwise-LDL	0.172M	0.172M	9.15
spatial-average	0.219M	0.187M	9.70
spatial&channelwise-average	0.219M	0.156M	10.20

Table 1: Various parameterizations on CIFAR-10 with WRN-16-1-bottleneck. We show median error over 5 runs and the numbers of parameters used in training and testing.

All symmetry parameterizations have certain drop in performance comparing to the baseline. We observe that channel-wise triangular / average parameterizations have the lowest drop. Eigen parameterization does not work well in this experiment, which is possibly a consequence of $\boldsymbol{V}$ is not forced to be an orthogonal matrix. On the other hand, LDL parameterization as an alternative attains a better result. We also test soft channel-wise L^1-norm (see eq. 8.1), where the number of parameters remains the same in training, and reduced at test time by using upper triangular weights only. The slackness is controlled by the coefficient ρ. For a large ρ, the soft-constrained symmetrization is slightly better than hard-constrained symmetrizations.

Spatial symmetry parameterizations do not work as well as channel-wise symmetry. This is expected since the size of spatial dimensions is much smaller compared with channel dimensions (i.e. $N_i = N_o > k_h = k_w$). Thus, the constraints imposed on spatial dimensions are much harsher making the learning much more difficult.

Based on the aforementioned analysis, we choose triangular parameterization (in terms of validation accuracy, parameter reduction and simplicity both at train and test time) as the main method to conduct all experiments further in this section.

Wide Residual Networks with symmetry

In this section we compare symmetrized WRN to its wider and thinner counterparts, and discuss the choice of network architecture. We use the terms *conv0*, *conv1* and *conv2* to refer to the first, the second and the third convolutional layers respectively (basic blocks have only *conv0* and *conv1*). For our experiments we choose the bottleneck and constrain the mid-bottleneck 3×3 *conv1* convolution to be symmetric, keeping 1×1 *conv0* and *conv2* unconstrained. This choice is because the approximating power of the residual block is least reduced, as real eigenvalues of *conv1* are rotated by the surrounding convolutions, resulting in overall rich parameterization.

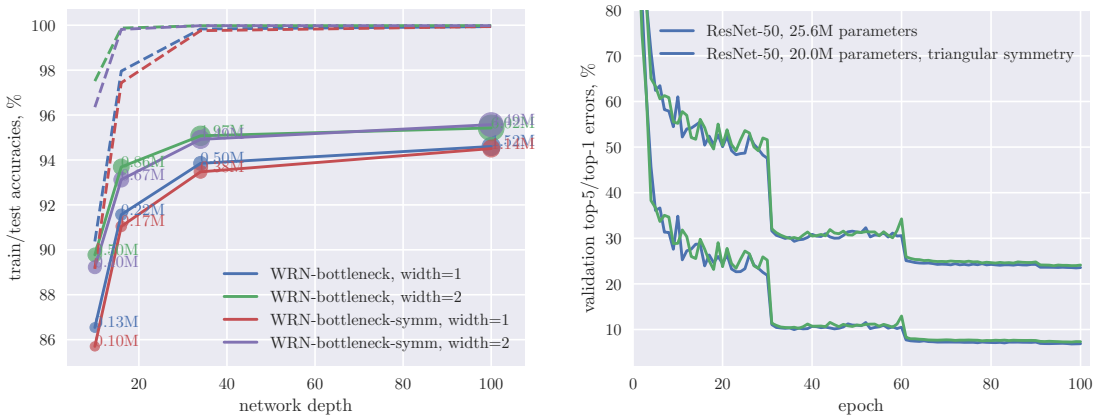
We present results for WRN-40-1-bottleneck and WRN-40-2-bottleneck trained on CIFAR in Table 2. We also train thinner networks with the same number of parameters to compare with their symmetric variants. On both datasets symmetric parameterization compares favorably to both wider and thinner non-symmetric counterparts in terms of accuracy and number of parameters.

base network	symmetry location	width	#params	CIFAR-10	CIFAR-100
WRN-40-1-bottleneck	none	1	0.59M	6.12	26.86
	none	0.875	0.45M	6.29	28.36
	conv1	1	0.45M	6.24	27.66
WRN-40-2-bottleneck	none	2.0	2.34M	4.95	22.51
	none	1.75	1.79M	5.12	23.18
	conv1	2.0	1.76M	4.96	22.98

Table 2: CIFAR test error (median of 5 runs) of triangular channel-wise parameterization on WRN-40 with bottleneck layers. *conv0* and *conv1* refer to the first and the second convolutional layers respectively.

We further illustrate this in Fig. 2a, where we show training and validation accuracy of WRN with respect to various depths and widths. WRN has a lower accuracy in shallower networks when the training accuracy does not reach 100%, that is, the network struggles to fit into training data. In such cases symmetry constraints damage both training and validation accuracy significantly. Here we also notice that symmetry constraints cause much smaller accuracy drop in networks which are able to fit into training data perfectly, having almost 100%

training accuracy. We hereafter refer to such networks as *overparameterized*, and we should note that overparameterization should not be confused with overfitting, that is, overparameterized network do not suffer from poor generalization.



(a) WRN of various depth and width with bottleneck blocks and triangular symmetry. Dash (solid) lines denote train (val) accuracy respectively (medians over 5 runs).

(b) Convergence curves of top-1 (top lines) and top-5 (bottom lines) validation errors of ResNet-50 and its triangularly symmetrized variant on ImageNet.

Figure 2: Image classification results for ResNet with symmetric filters.

Other architectures

In this section we show that triangular symmetry works as well on other architectures and larger networks. We pick a residual network variant, ResNeXt, which reduces the number of parameters and computational complexity in ResNet by using grouped 3×3 convolution in bottleneck block. For large networks we use WRN-28-10, both basic and bottleneck variants, and a simple feedforward VGG. We also apply symmetry to MobileNets, a popular architecture for mobile devices. Even though being feedforward, it compares favorably to smaller residual networks such as ResNet-18 and ResNet-34, with significant reduction in parameters needed to achieve the same accuracy.

Results are presented in Table 3. We put triangular symmetry constraint on the second layer in each residual block of WRN-basic-28-10, and on 3×3 convolutional layers in WRN-bottleneck-28-10. In both cases, there are no drops in accuracy, as expected in overparameterized networks which easily achieve 100% training accuracy. Triangular parameterization works well even with ResNeXt, which has much less parameters in 3×3 layers. That is also the case for VGG, which, in contrast to others, does not have 1×1 or depth-wise convolutions between layers with symmetry. We believe the fact that there is no big reduction in accuracy is due to the existence of redundant parameters. In MobileNet, we parameterize all square $1\times$ layers, and observe relatively small drop in accuracy.

Overall, we observe that overparameterized networks can easily benefit from symmetry parameterizations in terms of the number of parameters and potential speedup from more efficient implementation. It is surprising to see that triangular

network	symmetry	#params	CIFAR-10	CIFAR-100
ResNeXt-16-2-4		0.57M	6.93	28.3
ResNeXt-16-2-4	✓	0.53M	7.18	28.86
MobileNet		3.2M	7.6	31.05
MobileNet	✓	2.0M	7.91	31.48
VGG		20M	6.11	25.75
VGG	✓	10.8M	6.19	26.8
WRN-28-10-basic		36.5M	3.99	18.7
WRN-28-10-basic	✓	26.8M	3.97	19.1
WRN-28-10-bottleneck		39.8M	3.96	18.94
WRN-28-10-bottleneck	✓	30.2M	3.77	18.79

Table 3: Triangular symmetry applied to various architectures on CIFAR. Triangular channelwise symmetry is imposed on every second convolution in residual blocks in WRN and ResNeXt, and in all square convolutions in MobileNet and VGG. Median test accuracy of 5 runs is reported.

parameterization works well even with ResNeXt and MobileNet, which have already been designed to enjoy weight sharing. Among all these experiments, VGG with triangular parameterization reduces almost half of the parameters (i.e. 9.2 million), yet the accuracy still remains almost the same.

Importance of batch normalization

One might notice that batch normalization Ioffe and Szegedy (2015a) could potentially be a symmetry-breaking component when combined with symmetric convolutional or linear layers, as it can be viewed as an affine transform of each feature plane, or a diagonal fully connected layer. We, however, successfully apply symmetric parameterization to networks without batch normalization.

To test the influence of batch normalization on symmetric parameterization, we trained Network-In-Network on CIFAR with and without batch normalization, convergence plots are presented on Fig. 3. As can be seen, the difference in accuracies is very similar in both cases. If batch normalization played a significant role in improving symmetric parameterization, Network-In-Network without batch normalization (left) and triangular symmetry would have a higher accuracy drop. We use learning rate of 0.1 to train the networks with batch normalization, and of 0.01 without. Also, for triangular parameterization without batch normalization we reduce learning rate on upper triangular part by 2 to compensate for gradient magnitude increase due to sharing.

Combining symmetric parameterization with ShaResNet

We discussed the possibility of combining symmetry parameterizations with other parameter reduction methods in Section 8.2.4. Here, we conduct an experiment that combines 2-way channelwise symmetry with ShaResNet Boulch (2017), which is a weight sharing method that all *conv1* of residual blocks (bottleneck block) within a group (i.e. layers between two dimensionality reduction convolutions)

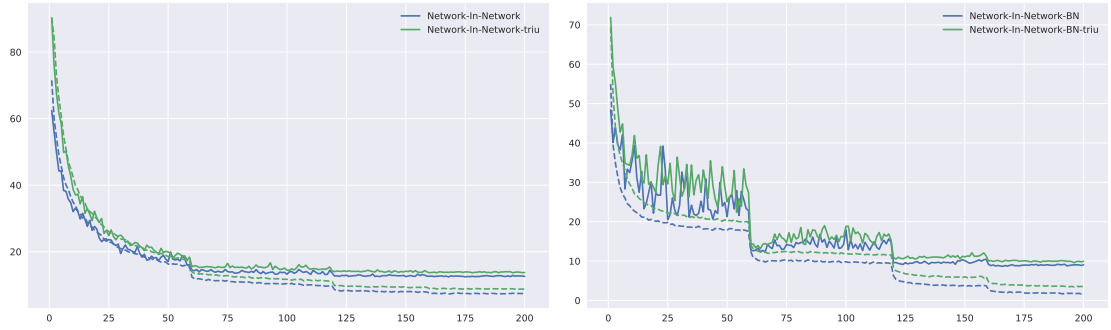


Figure 3: Combining symmetric parameterization without batch normalization (left) and with (right), Network-In-Network. Training accuracy in shown by dashed lines, validation - solid. Accuracy drop is similar in both cases.

share the same weights. The results are shown in Table 4. It can be seen that the combination further reduces the number of parameters, while the testing performance is only slightly affected. In particular, with the bottleneck architecture, triangular symmetry pluses ShaResNet reduce about 33% of parameters and the performance drop is less than 1%.

Table 4: Combining ShaResNet Boulch (2017) with 2-way channelwise symmetry. Test errors (mean/std/median over 5 runs) are compared for different symmetry parameterizations on CIFAR-10 using WRN-16-1-bottleneck with symmetric *conv1*.

network	share	symmetry	#params	mean	std	median
WRN-16-1-bottleneck		none	0.219M	8.40	0.24	8.49
	✓	none	0.171M	8.64	0.15	8.63
		triangular	0.172M	8.76	0.19	8.84
	✓	triangular	0.147M	9.49	0.34	9.36
	✓	average	0.171M	9.35	0.23	9.25

N-way symmetries

As discussed in Section 8.2.3, it is possible to push forward the triangular parameterization to a more general N-way triangular parameterization. In addition to triangulizing and blocking, we also consider a chunking implementation (a naive N-way weight sharing: given V as a $M \times \frac{M}{N}$ matrix. We define by $\text{tile}_{N\times}(V)$ a transforming function to tile/copy V N times to construct $\hat{W} = \text{tile}_{N\times}(V))$ as an example to show that carefully designed N-way symmetries yield better performance than a naive N-way weight sharing. Recall that standard triangular parameterization is equivalent to 2-way triangulizing. In this section, we test our proposals including chunking, blocking and triangulizing to achieve N-way channelwise symmetry. For chunking, we examine the cases of $N = 2, 4, \ldots, 64$. If the number of channels in a convolutional layer is less than N, we simply set N equal to the number of channels. For blocking and triangulizing, it is non-intuitive

to come up with N-way symmetry for $N > 4$, so we only test the cases of $N = 4, 16$ and $N = 2, 4, 8$ respectively.

In our experiments, as shown in Fig. 4, chunking causes the steepest linear decrease in validation accuracy. Triangular parameterizations work fine even in the case of 8-way symmetry, which reduces almost $\frac{3}{8}$ percentage of parameters from 2-way symmetry while only suffer about 1% decrease in accuracy.

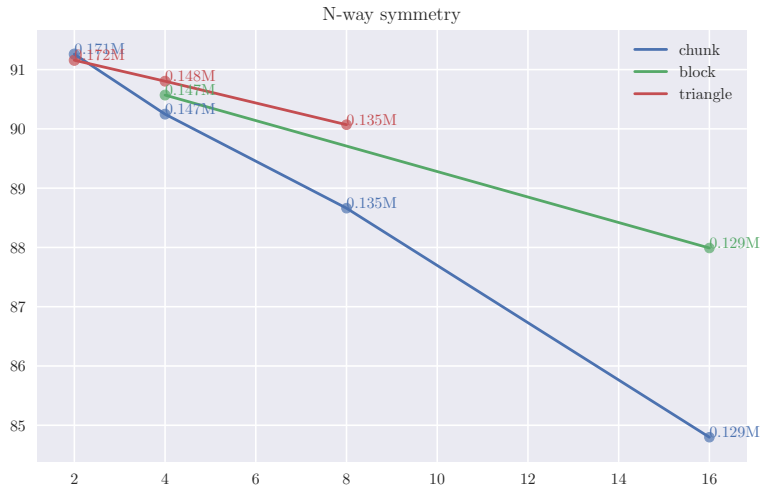


Figure 4: N-way sharing (chunking) v.s. N-way symmetries (blocking, triangulizing) on CIFAR-10 with WRN-16-1-bottleneck. x-axis represents N. y-axis represents the accuracy.

8.4.2 ImageNet experiments

In this section we present ImageNet results for networks with triangular symmetry and without, to check if our conclusions from CIFAR transfer to larger dataset.

We start with results for relatively small networks, MobileNet and ResNet-18. In MobileNet we impose triangular symmetry on all square 1×1 convolutional layers. ResNet-18 has basic block architecture and doesn't have enough parameters to fit into training data well, so we impose symmetry on every second 3×3 convolutional layer in block. Both MobileNet and ResNet-18 are relatively shallow networks and have 28% less parameters with triangular symmetry, so, as expected, the drop in accuracy is significant, see Table 5. Still, constrained MobileNet has only 3M parameters and achieves almost the same accuracy with ResNet-18.

As for large networks, we trained ResNet-50 and ResNet-101 with bottleneck block architecture and triangular symmetry on all 3×3 layers. As on CIFAR, drop in accuracy is much smaller for these: a reduction of 23% parameters (which would correspond to approximately ResNet-40 for ResNet-50 and ResNet-77 for ResNet-101) causes only 0.5% accuracy drop compared to unconstrained ResNet-50, and even smaller for ResNet-101, which is about 0.2%. We show convergence curves for both ResNet-50 and its symmetric variant on Fig. 2b.

All networks were trained in the same conditions and with the same hyper-parameters. We used large mini-batch training approach as proposed in Goyal

network	symmetry	#params	top-1	top-5
MobileNet		4.2M	28.18	9.8
MobileNet	✓	3.0M	30.57	11.6
ResNet-18		11.8M	30.54	10.93
ResNet-18	✓	8.6M	31.44	11.55
ResNet-50		25.6M	23.50	6.83
ResNet-50	✓	20.0M	23.98	7.25
ResNet-101		44.7M	22.14	6.09
ResNet-101	✓	34.0M	22.36	6.35

Table 5: ImageNet results for networks with triangular symmetry parameterization. Smaller networks such as MobileNet and ResNet-18 have more significant drop in accuracy than larger ResNet-50 and ResNet-101. The latter have 23% less parameters than non-symmetric counterparts and have drops in accuracy of 0.5% and 0.2% correspondingly

et al. (2017) on 8 GeForce 1080Ti GPUs, scaling learning rate proportionally to mini-batch size. Also, we do not regularize batch normalization and depth-wise convolution parameters in MobileNet. Surprisingly, our MobileNet and ResNet baselines outperform the original networks proposed in Howard et al. (2017b) and He et al. (2016b). We plan to make our code and networks available for download online.

8.4.3 Language modeling

We test symmetry on a common language modeling Penn-Tree-Bank (Marcus et al. 1993) dataset. Our experimental setup reflects that of (Zaremba et al. 2014)[2]. We use a network with 2-layer LSTM and dropout, and test out a medium size configuration with 650 neurons (dropout 0.5), and a larger model with 1500 neurons (dropout 0.65). We try to symmetrize weights corresponding to input and hidden and gates separately and altogether, as well for each gate separately. Surprisingly, we find that adding symmetry on all hidden gates does not hurt, and even obtain slightly lower perplexity for both medium and large models. The results are presented in Table 6. We use the average parameterization $\frac{1}{2}(\boldsymbol{W} + \boldsymbol{W}^\top)$ as we notice it gives slightly better results. Also, there is no L^2-regularization applied during training, so the original $\boldsymbol{W}$ weights converge to be almost symmetric. There is also a large variation in final validation perplexity, so we train each network 5 times with different random seed and report mean±std results for all models.

For a fair comparison we also trained thinner networks with 565 and 1300 hidden neurons for medium and large networks correspondingly, so that the number of parameters is approximately equal to those of hidden-symmetrized networks. Symmetrized versions of large networks compare favourably to these networks.

We also include experimental results on a larger wikitext-2 (Merity et al. 2016)

[2] https://github.com/pytorch/examples/tree/master/word_language_model

Model	symmetry	#parameters	Penn-Tree-Bank		wikitext-2	
			validation	test	validation	test
LSTM-650		6.8M	85.38 ± 0.29	81.49 ± 0.04	99.59 ± 0.17	94.26 ± 0.21
LSTM-565		5.1M	85.48 ± 0.45	81.47 ± 0.28	100.60 ± 0.28	94.99 ± 0.24
LSTM-650	✓	5.1M	83.73 ± 0.42	79.73 ± 0.23	100.81 ± 0.45	95.43 ± 0.37
LSTM-1500		36M	81.90 ± 0.54	77.95 ± 0.41	95.59 ± 0.34	90.92 ± 0.20
LSTM-1300		27M	80.71 ± 0.14	77.42 ± 0.18	96.08 ± 0.26	90.77 ± 0.21
LSTM-1500	✓	27M	79.66 ± 0.41	75.69 ± 0.32	97.13 ± 0.77	91.97 ± 0.85

Table 6: Language modeling perplexity (lower is better) on Penn-Tree-Bank and wikitext-2 datasets. Only hidden gate weights are symmetrized with triangular parameterization. We count only parameters in RNN, skipping encoder and decoder. Mean±std results over 5 runs are reported.

dataset in Table 6, which is about 2 times larger than Penn-Tree-Bank, as well experiments with symmetry on each gate separately. We apply averaging symmetry to hidden gates of LSTM and compare to thinner networks with comparable number of parameters. In this case symmetry slightly hurts perplexity of medium model, and more significantly large model. This might be due to different dropout regularization in the networks (we use the same dropout rates as in Penn-Tree-Bank[3], which may not be the best choices for wikitext-2).

8.5 Conclusion

In this paper, we have shown that, quite surprisingly, deep neural network weights can be successfully parameterized to be symmetric without suffering a significant loss in accuracy. The proposed symmetry parameterizations could lead to potentially significant improvements for a wide range of mobile applications in terms of computational efficiency (dedicated routines for symmetric convolutions and matrix multiplication can be applied) and storage efficiency (memory requirements for storing network weights are dramatically reduced).

For future work, it would be interesting to compare with other structural weight matrices (e.g., (Zhao et al. 2017)) with computational benefits and understand what kinds of inductive bias are implied. It should also be noted that our universal approximation analysis holds only for approximating single univariate continuous functions. It remains an open question what theoretical guarantees a symmetric deep neural network can provide for more general multivariate functions.

[3]Suggested by https://github.com/pytorch/examples/tree/master/word_language_model

Conclusion

The main thread of this thesis has been devoted to the efficiency of machine learning methods. However, the emphases are different for different parts. For probabilistic graphical models, we focused on the computational efficiency, for which we have investigated linearly convergent algorithms with the presence of equality constraints. For deep neural networks, we considered several efficiency issues due to the over-parameterization. Throughout the chapters, our proposed solutions mainly arise from variational inference. Because of its flexibility, we also heavily explored the underlying structure of particular problems.

In Chapter 2, we reviewed the inference and learning problems of discrete graphical models, which are NP-hard for general graphs. The computation of learning is even heavier since each gradient step of learning involves a full inference on the entire graph. Therefore, a strong incentive as discussed in Chapter 3 is that we would like to learn discrete graphical models with block-coordinate approximate inference, where blocks correspond to cliques in the graph. We devoted a large fraction of Chapter 3 to derive this algorithm. In particular, we exploited the fact that strong duality holds for the regularized maximum likelihood estimation. The dual problem has a nice structure since 1) the primal solution can be directly retrieved from the dual solution via the representer theorem and 2) the objective can be decomposed into a non-smooth (and strongly concave) but separable part, a smooth but non-separable part, and an equality constraint. We employ an ADMM-like scheme to handle the equality constraint resulting in a min-max problem. It introduces additionally a linear term and a quadratic term absorbed by the non-smooth part and the smooth part respectively. Our contributions lie in both the algorithmic aspect and the convergence analysis. The proposed algorithm is inspired by inexact gradient descent. We control the number of inner maximization steps (by a linearly convergent subroutine) to yield a linear rate convergence for the outer minimization. Our theoretical analysis focuses on the trade-off between the inner exactness and the outer decrease, characterizing the global convergence in terms of the total number of inner iterations, which also implies a linear convergence in the primal.

The second topic of the thesis is dedicated to Bayesian deep learning. Bayesian approaches on neural networks treat the parameters or weights as random variables. We reviewed the relationship between Bayesian neural networks and compression in Chapter 4, as well as the recent understanding of the generalization in deep learning via compression. Following the lossy compression scheme, namely, the information bottleneck principle, we proposed, in Chapter 5 and Chapter 6, to use the mutual information between the weights and the dataset as a regularizer for

Bayesian supervised learning and Bayesian meta-learning. We used a variational upper bound on the mutual information, reformulated the learning problem as an alternating minimization (aka bi-level optimization). Although this formulation was not derived from Bayes' rule, the optimization variables resemble the prior and the posterior in classical Bayesian models. It can also be viewed as an empirical Bayes approach, as discussed in Chapter 6. Apart from the insights on the connections between the empirical Bayes and the information bottleneck, our main contributions are the proposed new algorithms.

For supervised learning, there is only one instance of the dataset, to introduce a mutual information regularizer, we reinterpreted the dataset as a bootstrap sample drawn from the empirical data distribution, thus it boils down to compute the mutual information between the weights and a bootstrap sample (or equivalently a minibatch). The developed algorithm, as shown in Chapter 5, is referred to as β-BNN, since the inner step resembles the SGD step for a standard BNN with the KL term weighted by β, while the outer step maintains the aggregated posterior (i.e., the prior) as a running average of previous variational posteriors.

For meta-learning, the main challenge is how to implement a meta-model capable of gathering domain knowledge rapidly. Inspired by the aforementioned alternating minimization, we proposed to parameterize the inner minimization with respect to the variational posterior as a part of the meta-model, which consists of a synthetic gradient network and an initialization network. In particular, the synthetic gradient is the key to enable the transductive inference towards a better chracterization of the inner optimization dynamics. To support this proposal, we first showed in theory that, under this setting, the generalization error of the variational posterior can be largely reduced by transduction. Besides, we demonstrated that our algorithm significantly outperforms previous state-of-the-art methods on Mini-ImageNet and CIFAR-FS.

In the last two chapters, namely, Chapter 7 and Chapter 8, we presented our works about model compression for over-parameterized neural networks. The compression is done by either making a compact student network (which would be guided by a pretrained teacher network) or taking advantage of the symmetry in the weights. We proposed a new framework in Chapter 7 for the knowledge transfer from the teacher network to the student network by maximizing a variational lower bound of the mutual information between the activations of these two networks. This idea can also be used to transfer learning when the teacher is pretrained on another task. On the other hand, we conducted multiple experiments in Chapter 8 on image classification and language modeling and verified that enforcing weight symmetry is feasible in these tasks.

Future work: integrating data-driven methods and knowledge-driven methods

Coming back to the discussion in the introduction, a good restarting point is a recent blog (Sutton 2019) posted by Richard Sutton who suggests that "*a bitter lesson*" we all should learn from the competition between data-driven AI and knowledge-driven AI:

> *"1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning."*

A few weeks later, Max Welling responds with a relatively neutral opinion on Twitter (Welling 2019):

> *"When you have sufficient data, you do not need to impose a lot of human generated inductive bias on your model. You can let the data speak. However, when you do not have sufficient data available you will need to use human-knowledge to fill the gaps."*

This is not the first debate between the supporters of data-driven methods and knowledge-driven methods. A thought-provoking summary was published by Marvin Minsky about thirty years ago in his article titled "Logical vs. Analogical or Symbolic vs. Connectionist or Neat vs. Scruffy" (Minsky 1991). His insights are of a fundamental and look-ahead character. One of the insights he shared is that since each method has its own virtues and deficiencies, an integrated system is needed to exploit the advantage of both:

> *"In favor of the top-down side, research in artificial intelligence has told us a little – but only a little – about how to solve problems by using methods that resemble reasoning. If we understood more about this, perhaps we could more easily work down toward finding out how brain cells do such things. In favor of the bottom-up approach, the brain sciences have told us something – but again, only a little – about the workings of brain cells and their connections. [...] I'll argue that the solution lies somewhere between these two extremes, and our problem will be to find out how to build a suitable bridge."*

Minsky did not simply suggest that we should combine logical methods or probabilistic graphical models with deep neural networks. What he meant "an integrated system" is beyond a naive combination. He described some properties of the integrated system:

> *"To solve typical real-world commonsense problems, a mind must have at least several different kinds of knowledge. First, we need to represent goals: what is the problem to be solved. Then the system must also possess adequate knowledge about the domain or context in which that problem occurs. Finally, the system must know what kinds of reasoning are applicable in that area. Superimposed on all of this, our systems must have management schemes that can operate different representations and procedures in parallel, so that when any particular method breaks down or gets stuck, the system can quickly shift over to analogous operations in other realms that may be able to continue the work."*

This description, however, did not provide a specific solution and obviously, none of the solutions we have had possess all of these properties. My personal take is the following. An AI system of this kind must be reconfigurable such that

- it is able to automatically set up different goals (i.e., losses or rewards) for different scenarios;
- it is able to unify the feature (or knowledge) representation for the information learned from the data or embedded through human prior knowledge.
- it embodies a general reasoning mechanism on top of the feature representations regardless of the particular problem to be solved.
- it considers robustness as equally important as its main goals.

Indeed, none of these desiderata is completely new, however, they have not been put all together to build a functional system. Finding a principle way to integrate different pieces is an interesting direction for future work.

On the other hand, an AI system, as argued by Lawrence (2019), will never replace NI systems since the evolution, not to mention its time span, is unique. As such, blending knowledge-driven methods with data-driven methods or the other way around is arguably the most straightforward solution to build an AI system comparable to its NI counterpart, which is like to stand on the shoulder of the evolution giant. However, there are many foreseeable challenges towards a reasonable implementation of such blending, among which the main technical issue is how to unify the representations learned from different sources, which have to be consistent when they refer to the same thing. Besides, it will be more efficient to employ different paradigms for learning different functionalities. For example, if we are trying to implement a football-playing AI, the data-driven methods will be more effective in learning ball touches, dribbles and so on from practical games, while the knowledge-driven methods will be more effective in learning gestures, tactics, formations and so on.

Last but not least, it will still be relevant to improve variational techniques since they are the key building blocks to realize robustness and uncertainty. For future work, one important topic is to achieve more efficient algorithms for variational inference. The current state-of-the-art algorithms are based on stochastic gradient descent, thus they inherit the limitations of gradient methods – it is not a global optimization algorithm and its convergence suffers from the oscillation near a stationary point. To overcome bad local minima, which occurs in general variational inference, one promising idea is to treat the optimization as an optimal control problem. If we learn to solve this problem, taking gradient as a part of the state information, and keeping a good balance of exploration and exploitation, we may find a better local minimum with much faster convergence rate.

Bibliography

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentangling in deep representations. *arXiv preprint arXiv:1706.01350*, 2017. (document), 4.1, 4.5.1, 4.5.2, 5.1, 2, 6.4, 6.4

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 4.5.4

David Barber Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in Neural Information Processing Systems*, 16:201, 2004. 7.1, 7.2

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016. 4.5.4

Alexander A Alemi, Ben Poole, Ian Fischer, Joshua V Dillon, Rif A Saurous, and Kevin Murphy. An information-theoretic analysis of deep latent-variable models. *arXiv preprint arXiv:1711.00464*, 2017. 5.1

Alibaba. Alibaba's ai outguns humans in reading test. https://www.bloomberg.com/news/articles/2018-01-15/alibaba-s-ai-outgunned-humans-in-key-stanford-reading-test, 2018. 4.1

Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *International Conference on Machine Learning*, pages 205–214, 2018. 6.1, 6.2.1

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972. (document), 1.3, 5.2

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018. 4.5

Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2252–2259. IEEE, 2011. 4.6

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014. (document), 4.6, 7.1, 7.2.2

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017. 4.1

Wolfgang Balzer, Masanobu Takahashi, Jun Ohta, and Kazuo Kyuma. Weight quantization in boltzmann machines. *Neural Networks*, 4(3):405–409, 1991. 4.4

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. 2

Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM J. Optim.*, 2013. 2.3.1

David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992, 2016. 2.5

David Belanger, Dan Sheldon, and Andrew McCallum. Marginal inference in mrfs using frank-wolfe. In *NIPS Workshop on Greedy Optimization, Frank-Wolfe and Friends*, 2013. 2.3.1

Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R Devon Hjelm, and Aaron Courville. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018. 7.4

Y Bengio, S Bengio, and J Cloutier. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume 2, pages 969–vol. IEEE, 1991. 6.1

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 4.2

Yoshua Bengio, Nicolas L Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. In *Advances in neural information processing systems*, pages 123–130, 2006. 4.1

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4.4

James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag New York, 1985. 6.1

Umberto Bertele and Francesco Brioschi. *Nonserial dynamic programming*. Academic Press, 1972. 2.3

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, pages 523–531, 2016. 4.4

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *ArXiv*, abs/1805.08136, 2018. 6.5.1, ??

D. P. Bertsekas. The method of multipliers for equality constraints. In *Constrained optimization and Lagrange Multiplier methods*. Athena scientific, 1982. 3.4.2

Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific, 1999. 1

Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994. 7.2.1

Richard Blahut. Computation of channel capacity and rate-distortion functions. *IEEE transactions on Information Theory*, 18(4):460–473, 1972. (document), 1.3, 1.3, 5.2

Matthew B Blaschko and Christoph H Lampert. Learning to localize objects with structured output regression. In *European conference on computer vision*, pages 2–15. Springer, 2008. 2.4.2

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. (document)

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. (document), 2.3.1

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4): 929–965, 1989. 4.5.1

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015. (document), 4.4, 4.5.3, 4.5.3, 1, 5.2, 5.3

Jérôme Bolte, Shoham Sabach, Marc Teboulle, and Yakov Vaisbourd. First order methods beyond convexity and lipschitz gradient continuity with applications to quadratic inverse problems. *SIAM Journal on Optimization*, 28(3):2131–2151, 2018. 2

Alexandre Boulch. Sharesnet: reducing residual network parameter number by sharing weights. *Proceedings of the International Conference on Learning Representations*, 2017. 8.1, 8.2.1, 8.2.4, 8.4.1, 4

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. (document)

Y Boykov, O Veksler, and R Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001. 2.3.2

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013. 8.3.1

Wouter Bulten. Getting started with gans part 2: Colorful mnist. `https://www.wouterbulten.nl/blog/tech/getting-started-with-gans-2-colorful-mnist/`, 2017. 5.3

Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016. 1, 4.2

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019. **??**, 6.5.3

Miguel A Carreira-Perpinán and Yerlan Idelbayev. Model compression as constrained optimization, with application to neural nets. part ii: Quantization. *arXiv preprint arXiv:1707.04319*, 2017. 4.4

Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems*, pages 657–664, 1995. 4.6

O Chapelle, B Schölkopf, and A Zien. A discussion of semi-supervised learning and transduction. *Semi-Supervised Learning*, pages 457–462, 2006. 6.1

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018. 4.5.4

Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016a. (document)

Shixing Chen, Caojin Zhang, and Ming Dong. Coupled end-to-end transfer learning with generalized fisher information. In *Computer Vision and Pattern Recognition*, 2018. 7.1, 7.2.2

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019. 6.5.1

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015. 4.4

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1475–1484, 2016b. ISBN 978-1-4503-4232-2. 8.1

Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*, 2014. 8.3

Heejin Choi, Ofer Meshi, and Nathan Srebro. Fast and scalable structural svm with slack rescaling. In *Artificial Intelligence and Statistics*, pages 667–675, 2016. 2.4.2

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002. 2.4.2

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *JMLR*, 9:1775–1822, 2008. 3.2, 3.4.2

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015. 4.4

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 4.4

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012. 1.3, 1.3.1, 1.3.2, 1.3, 5.1, 5.1.1

Richard T Cox. Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13, 1946. (document)

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar): 551–585, 2006. 4.6

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989a. 4.1, 4.2

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989b. (document)

Bin Dai, Chen Zhu, and David Wipf. Compressing neural networks using the variational information bottleneck. *arXiv preprint arXiv:1802.10399*, 2018. 4.5.4

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, pages 1646–1654, 2014a. (document)

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014b. 3.1

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 4.1, 4.5.1

Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013a. 4.1, 4.4, 4.4

Misha Denil, Babak Shakibi, Laurent Dinh, Marc' Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems 26*, pages 2148–2156. Curran Associates, Inc., 2013b. URL http://papers.nips.cc/paper/5025-predicting-parameters-in-deep-learning.pdf. 8.1, 8.2.3

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014a. 4.4

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems 27*, pages 1269–1277, 2014b. 8.1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 6.1

Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014. 3.6.3

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017. 4.5.2, 4.5.2

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 7.1

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011. (document)

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017. (document), 4.1, 4.5, 4.5.1

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 7.1

Emile Fiesler, Amar Choudry, and H John Caulfield. Weight discretization paradigm for optical neural networks. In *Optical interconnections and networks*, volume 1281, pages 164–174. International Society for Optics and Photonics, 1990. 4.4

T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *International Conference on Machine Learning (ICML)*, pages 304–311, 2008. 2.4.2, 3.7.1

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. (document), 6.1, 6.2.2, 6.3, **??**

Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. *International Conference on Learning Representations (ICLR)*, 2019. 6.1

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956. 2.4.2

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018. (document), 7.3.1

Shigeru Furuichi. Information theoretical properties of Tsallis entropies. *Journal of Mathematical Physics*, 47(2):023302, 2006. 3.C

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1704–1713, 2018a. 4.4

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b. 6.3

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984. (document)

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. Pac-bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems*, pages 1884–1892, 2016. 4.5.3

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018. (document), 4.4, 6.1, 6.3, 6.5.1

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1v4N2l0-`. 6.5.3

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. *arXiv preprint arXiv:1906.05186*, 2019. **??**, **??**, 6.5.1, 6.5.1

Gauthier Gidel, Fabian Pedregosa, and Simon Lacoste-Julien. Frank-wolfe splitting via augmented lagrangian method. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1456–1465. PMLR, 2018. 3.2

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 7.1

Raja Giryes, Guillermo Sapiro, and Alexander M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Trans. Signal Processing*, 64(13):3444–3457, 2016. 4.4

Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, 2007a. 2.3.2, 2.4.2

Amir Globerson and Tommi Jaakkola. Convergent propagation algorithms via oriented trees. In *UAI*, pages 133–140, 2007b. (document), 3.4.3

Faustino Gomez and Jürgen Schmidhuber. Evolving modular fast-weight networks for control. In *International Conference on Artificial Neural Networks*, pages 383–389. Springer, 2005. 4.4

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 4.4

Irving John Good. Some history of the hierarchical bayesian methodology. *Trabajos de estadística y de investigación operativa*, 31(1):489, 1980. 6.1

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Decision-theoretic meta-learning: Versatile and efficient amortization of few-shot learning. *arXiv preprint arXiv:1805.09921*, 2018. 4.4

Kazushige Goto and Robert Van De Geijn. High-performance implementation of the level-3 blas. *ACM Trans. Math. Softw.*, 35(1):4:1–4:14, July 2008. ISSN 0098-3500. 8.1

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. 8.4.2

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018. 6.1, 6.2.1

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011. (document), 4.5.3, 4.5.3

Klaus Greff, Rupesh Kumar Srivastava, and jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *Proceedings of the International Conference on Learning Representations*, 2017. 8.3.1

Peter Grunwald. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*, 2004. 4.5.3

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016. 4.4

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representation (ICLR)*, 2017a. URL `https://openreview.net/pdf?id=rkpACe1lx`. 4.4

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. *Proceedings of the International Conference on Learning Representations*, 2017b. 8.1

Benjamin D Haeffele and René Vidal. Global optimality in neural network training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7331–7339, 2017. 4.1

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a. 4.1, 4.4, 4.5

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, 2015b. 4.4

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016. 8.1, 8.2.1, 8.2.4

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J. Dally. DSD: regularizing deep neural networks with dense-sparse-dense training flow. *International Conference on Learning Representations*, 2017. 8.1

Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension bounds for piecewise linear neural networks. In *Conference on Learning Theory*, pages 1064–1068, 2017. 4.5.1

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *Neural Networks, 1993., IEEE International Conference on*, pages 293–299. IEEE, 1993. 4.4

W. Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. (document)

John Haugeland. *Artificial intelligence: The very idea*. MIT press, 1989. (document)

T. Hazan and R. Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *NIPS*, pages 838–846, 2010. 2.4.2, 3.2, 3.3.1, 3.7, 3.A, 3.B.1

Tamir Hazan and Amnon Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010. 2.3.2

Tamir Hazan, Joseph Keshet, and David A McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2010. 2.4.2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 6.1

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, 2016a. (document), 4.1, 4.2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016b. 8.1, 8.4, 8.4.2

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, 2016c. 7.1, 7.3.2

Tom Heskes. Convexity arguments for efficient minimization of the bethe and kikuchi free energies. *J. Artif. Intell. Res.(JAIR)*, 26:153–190, 2006. 2.3.1, 2.3.1

G. E. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009. doi: 10.4249/scholarpedia.5947. revision #91189. (document)

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. (document), 4.6, 7.1, 7.3

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993. (document), 4.1, 4.5, 4.5.1, 4.5.3, 4.5.3

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. (document), 4.1, 4.2, 4.4, 4.5.4

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory, 1997a. 8.1, 8.3

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997b. 4.1, 4.5, 4.5.1, 4.5.2

Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017. 3.6.1, 3.D.2, 3.D.4

Mingyi Hong, Tsung-Hui Chang, Xiangfeng Wang, Meisam Razaviyayn, Shiqian Ma, and Zhi-Quan Luo. A block successive upper bound minimization method of multipliers for linearly constrained convex optimization. *arXiv preprint arXiv:1401.7079*, 2014. 3.2, 3.5

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017a. 4.4

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017b. 8.1, 8.2.4, 8.3, 8.4.2

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016. 8.3

Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017. 7.1, 7.2.2, 7.3

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017. 4.4

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016a. 4.4

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016b. 8.1

Francisco D. Igual, Gregorio Quintana-Ortí, and Robert A. van de Geijn. Level-3 blas on a GPU : Picking the low hanging fruit. FLAME working note #37. Technical Report DICC 2009-04-01, Department of Computer Sciences, The University of Texas at Austin, April 2009. URL `http://www.cs.utexas.edu/~flame/pubs/FLAWN37.pdf`. 8.1

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462, 2016. 8.1

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015a. (document), 8.1, 8.4.1

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015b. 4.2, 6.1

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014a. 4.4

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014b. 8.1, 8.2.1

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1627–1635. JMLR, 2017. 6.1, 6.3, 6.3

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4): 620, 1957. 5.1

Edwin T Jaynes. *Probability theory: the logic of science*. Washington University St. Louis, MO, 1996. (document)

Harold Jeffreys. *Theory of Probability*. The Clarendon Press, Oxford, 1939. (document)

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016. 4.4

Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014. 4.4

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009. 2.4.2, 2.4.2, 2.4.2, 2.4.2

Jason K Johnson and Alan S Willsky. *Convex relaxation methods for graphical models: Lagrangian and maximum entropy approaches*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering . . . , 2008. 2.3.2

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013. (document)

Michael Jordan. Artificial intelligence: The revolution hasn't happened yet. `https://doi.org/10.1162/99608f92.f06c6e61`, 2019. (document)

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999. (document), 2.3

Joerg Kappes, Bjoern Andres, Fred Hamprecht, Christoph Schnorr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard Kausler, Jan Lellmann, Nikos Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1335, 2013. 2.3.2

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016. 4.1

James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960. 2.4.2

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. (document), 4.1, 4.5.1, 4.5.2, 4, 4.5.2

John Maynard Keynes. *A treatise on probability*. Courier Corporation, 1921. (document)

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019. 6.3

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *EMNLP*, 2016. 8.1

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *AAAI*, 2016. 8.1

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (document), 4.1, 6.5.2

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4.5.3, 6.2.2

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015. (document), 4.4, 4.5.4, 4.5.4

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016. 7.4

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835, 2017. 4.6

Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. (document), 2, 2.2, 2.2, 2.2, 2.2, 2.3

Andrey Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Publishing Company, New York, 1933. (document)

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001. 4.5.1

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, pages 1–8, 2007. 2.3.2, 2.4.2, 3.2

Nikos Komodakis. Efficient training for pairwise or higher order CRFs via dual decomposition. In *CVPR*, pages 1841–1848, 2011a. 3.2

Nikos Komodakis. Efficient training for pairwise or higher order CRFs via dual decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1841–1848. IEEE, 2011b. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995375`. 2.4.2

Terry Koo, Amir Globerson, Xavier Carreras Pérez, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007. 3.C

Rahul G Krishnan, Simon Lacoste-Julien, and David Sontag. Barrier frank-wolfe for marginal inference. In *Advances in Neural Information Processing Systems*, pages 532–540, 2015. 2.3.1

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012a. 8.1

Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009a. 4.1, 4.5.1

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009b. 7.3.1

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, 2012b. 4.1, 4.2, 4.5.1

Frank R Kschischang, Brendan J Frey, Hans-Andrea Loeliger, et al. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001. 2.2

Alp Kucukelbir and David M Blei. Population empirical bayes. *arXiv preprint arXiv:1411.0292*, 2014. 5.1, 5.2, 6.1, 6.2.1

Alex Kulesza and Fernando Pereira. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems*, pages 785–792, 2007. 3.2

Alex Kulesza and Fernando Pereira. Structured learning with approximate inference. In *Advances in neural information processing systems*, pages 785–792, 2008. 2.4.2

Krzysztof Kurdyka. On gradients of functions definable in o-minimal structures. In *Annales de l'institut Fourier*, pages 769–783, 1998. 4.5.2

Alexandre Lacoste, Thomas Boquet, Negar Rostamzadeh, Boris Oreshki, Wonchang Chung, and David Krueger. Deep prior. *arXiv preprint arXiv:1712.05016*, 2017. 4.4

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *ICML*, pages 53–61, 2013a. 3.2, 3.7, 3.B.1

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pages 53–61, 2013b. 2.4.2, 2.4.2, 2.4.2

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001. 2.4.2

Guanghui Lan and Renato DC Monteiro. Iteration-complexity of first-order augmented Lagrangian methods for convex programming. *Mathematical Programming*, 155(1-2): 511–547, 2016. 3.6.3

Pierre Simon Laplace. *Théorie analytique des probabilités*. Courcier, 1820. (document)

Neil Lawrence. Machine learning systems design. `http://inverseprobability.com/talks/notes/machine-learning-systems-design.html`, 2019. 8.5

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *International Conference on Learning Representations (ICLR)*, 2016. 8.1, 8.2.1

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990a. 4.4

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann, 1990b. URL `http://papers.nips.cc/paper/250-optimal-brain-damage.pdf`. 8.1

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (document), 4.2

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436, 2015. (document)

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019. **??**

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018. 4.4

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5542–5550, 2017a. 6.5.3

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017b. 4.1

Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding Task-Relevant Features for Few-Shot Learning by Category Traversal. In *CVPR*, 2019. **??**, 6.5.1

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017c. 6.1

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 7.1, 7.2.2, 7.3

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. QuickeNing: A generic quasi-Newton algorithm for faster gradient-based optimization. *arXiv preprint arXiv:1610.00960*, 2017. 3.6.3

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013a. 4.2

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013b. 8.1

Zhouhan Lin, Roland Memisevic, and Kishore Konda. How far can we go without convolution: Improving fully-connected networks. *arXiv preprint arXiv:1511.02580*, 2015. 7.1, **??**, 7.3.3

Ralph Linsker. An application of the principle of maximum information preservation to linear systems. In *Advances in neural information processing systems*, pages 186–194, 1989. 7.2.2

Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018. 6.1, **??**, 6.5.1

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2755–2763. IEEE, 2017. 4.4

Stanislas Łojasiewicz. Sur la géométrie semi-et sous-analytique. *Ann. Inst. Fourier*, 43 (5):1575–1595, 1993. 4.5.2

B. London, B. Huang, and L. Getoor. The benefits of learning with strongly convex approximate inference. In *ICML*, pages 410–418, 2015. 2.3.1, 3.4.3, 3.C

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017. 4.5.3

Haihao Lu and Kenji Kawaguchi. Depth creates no bad local minima. *arXiv preprint arXiv:1702.08580*, 2017. 4.1

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *Computer Vision and Pattern Recognition (CVPR)*, pages 122–138, 2018. 4.4

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997. (document)

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 5.1

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1): 4873–4907, 2017. 4.5.4

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19: 313–330, 1993. 8.4.3

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. AD3: Alternating directions dual decomposition for MAP inference in graphical models. *JMLR*, 16:495–545, 2015. 3.2

Michael Mathieu, Mikael Henaff, and Yann Lecun. Fast training of convolutional networks through ffts. In *International Conference on Learning Representations (ICLR2014)*, 2014. 8.1

David A McAllester. Pac-bayesian stochastic model selection. *Machine Learning*, 51(1): 5–21, 2003. 4.5.3, 4.5.2

John McCarthy. *Programs with common sense*. RLE and MIT computation center, 1960. (document)

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 4.2

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. 8.1, 8.4.3

O. Meshi, A. Jaimovich, A. Globerson, and N. Friedman. Convexifying the bethe free energy. In *UAI*, 2009. (document), 2.3.1

O. Meshi, T. Jaakkola, and A. Globerson. Convergence rate analysis of MAP coordinate minimization algorithms. In *NIPS*, 2012. 2.3.2

O. Meshi, M. Mahdavi, and A. G. Schwing. Smooth and strong: MAP inference with linear convergence. In *NIPS*, pages 298–306, 2015a. 2.3.2, 3.1

O. Meshi, N. Srebro, and T. Hazan. Efficient training of structured SVMs via soft constraints. In *AISTATS*, pages 699–707, 2015b. 3.1, 3.2, 3.4.2, 3.7, 3.7.1, 3.B.1

Ofer Meshi, David Sontag, Amir Globerson, and Tommi S Jaakkola. Learning efficiently with approximate inference via dual losses. In *ICML*, pages 783–790, 2010. 2.4.2, 2.4.2, 3.2, 3.7, 3.A, 3.B.1

Ofer Meshi, Ben London, Adrian Weller, and David Sontag. Train and test tightness of lp relaxations in structured prediction. *Journal of Machine Learning Research*, 20 (13):1–34, 2019. 2.4.2

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. (document), 2.3

Tom Minka. Discriminative models, not discriminative training. Technical report, Technical Report MSR-TR-2005-144, Microsoft Research, 2005. 6.2.1

Marvin L Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, 12(2):34–34, 1991. 8.5

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017. (document)

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. 2, 2.2, 2.2

Rajib Nath, Stanimire Tomov, Tingxing Dong, and Jack Dongarra. Optimizing symmetric dense matrix-vector multiplication on gpus. In *High Performance Computing, Networking, Storage and Analysis (SC)*, page 6, 11 2011. 8.1

Yu. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005. 2.3.2

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2013. 3.D.4

Allen Newell and Herbert Alexander Simon. Gps, a program that simulates human thought. Technical report, RAND CORP SANTA MONICA CALIF, 1961. (document)

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017. 4.5.2

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018. 4.1, 3, 4.5.1

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. 2.4.2

Quynh Nguyen and Matthias Hein. Optimization landscape and expressivity of deep cnns. In *International Conference on Machine Learning*, pages 3727–3736, 2018. 4.1

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 6.1, 6.3

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018. 4.5.1

Steven J. Nowlan and Geoffrey E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Comput.*, 4(4):473–493, July 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.4.473. URL http://dx.doi.org/10.1162/neco.1992.4.4.473. 8.1

Sebastian Nowozin, Christoph H Lampert, et al. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4): 185–365, 2011. 2.3.1

Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *ICML*, pages 1632–1641, 2015. 3.2

Pavel Okunev and Charles R Johnson. Necessary and sufficient conditions for existence of the lu factorization of an arbitrary matrix. *arXiv preprint math/0506382*, 2005. 1

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 4.2

OpenAI. Openai dota 2 1v1 bot. https://openai.com/the-international/, 2017. 4.1

Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. ??

A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017. 4.2

Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 7.1

Giorgio Parisi. *Statistical field theory*. Addison-Wesley, 1988. (document), 2.3.1

J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proc. of Cognitive Science Society (CSS-7)*, 1985. (document)

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988. (document), 2.3.1

Guillermo Valle Pérez, Ard A Louis, and Chico Q Camargo. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018. (document), 4.1, 4.5, 4.5.1

Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987. (document)

Patrick Pletscher, Cheng Soon Ong, and Joachim M. Buhmann. Spanning tree approximations for conditional random fields. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 408–415, 2009. 2.4.2

Patrick Pletscher, Cheng Soon Ong, and Joachim M. Buhmann. Entropy and margin maximization for structured output learning. In *ECML*, pages 83–98, 2010. 3.3.1, 3.A

Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993. 4.6

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018. ??, 6.5.1

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE, 2009. 7.3.2

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2847–2854. JMLR. org, 2017. (document)

Mani Ranjbar, Tian Lan, Yang Wang, Steven N Robinovitch, Ze-Nian Li, and Greg Mori. Optimizing nondecomposable loss functions in structured prediction. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):911–924, 2013. 2.4.2

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015. 7.2.2

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 4.4

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations (ICLR)*, 2018. 6.1, 6.2.1

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representation*, 2016. (document), 6.1

Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, pages 737–744. ACM, 2006. 2.3.2

Pradeep Ravikumar, Alekh Agarwal, and Martin J Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, 11(Mar):1043–1080, 2010. 2.3.2

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 6.2.2

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006. (document)

Peter Richtárik and Martin Takáč. Stochastic reformulations of linear systems: algorithms and convergence theory. *arXiv preprint arXiv:1706.01108*, 2017. 2.3.1

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978. (document), 4.1, 4.5, 4.5.3

Herbert Robbins. An empirical bayes approach to statistics. In *Herbert Robbins Selected Papers*, pages 41–47. Springer, 1985. 5.1, 5.2, 6.1, 6.2.1

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. (document)

Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985. 4.1

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013. 2.3

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 6, 4.6, 7.1, 7.2.2, 7.3

Nir Rosenfeld, Ofer Meshi, Danny Tarlow, and Amir Globerson. Learning structured models with the auc loss and its generalizations. In *Artificial Intelligence and Statistics*, pages 841–849, 2014. 2.4.2

Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pages 2663–2671, 2012. (document), 3.1

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986. (document), 4.1, 4.2

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 7.1, 7.3.2

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 2009. (document)

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJgklhAcK7`. ??

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017. 4.1

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009. (document)

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017. ??

Victor Garcia Satorras and Joan Bruna. Few-shot learning with graph neural networks. *ArXiv*, abs/1711.04043, 2017. ??

B. Savchynskyy, J. Kappes, S. Schmidt, and C Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *CVPR*, pages 1817–1823, 2011. 2.3.2, 3.2

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representation (ICLR)*, 2018. 4.5.4

Thomas Schlegl, Joachim Ofner, and Georg Langs. Unsupervised pre-training across image domains improves lung tissue classification. In *International MICCAI Workshop on Medical Computer Vision*, pages 82–93. Springer, 2014. 7.1

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Phd thesis, Technische Universitat Munchen, Germany, 1987. URL http://www.idsia.ch/~juergen/diploma.html. 6.1

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. 4.4

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 4.1

Mark Schmidt, Nicolas Le Roux, and Francis R. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *NIPS*, pages 1458–1466, 2011. 3.6.3

Mark Schmidt, Reza Babanezhad, Mohamed Ahmed, Aaron Defazio, Ann Clifton, and Anoop Sarkar. Non-uniform stochastic average gradient method for training conditional random fields. In *AIStats*, pages 819–828, 2015. 3.2, 3.B.1

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *ICML*, pages 64–72, 2014. (document)

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1-2):105–145, 2016. URL http://link.springer.com/article/10.1007/s10107-014-0839-0. 2.3.1, 3.1, 3.5, 3.6.2, 3.E

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1): 3–30, 2011. 2.4.2

Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948. 1.2

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision*, pages 1–15. Springer, 2006. 3.7.1

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017. 4.1

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 8.1, 8.3

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014a. 4.1, 4.2

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014b. 7.1, 7.3.2

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. (document), 6.1, **??**

Soeren Sonnenburg, Heiko Strathmann, Sergey Lisitsyn, Viktor Gal, Fernando J. Iglesias García, Wu Lin, Soumyajit De, Chiyuan Zhang, frx, tklein23, Evgeniy Andreev, JonasBehr, sploving, Parijat Mazumdar, Christian Widmer, Pan Deng / Zora, Giovanni De Toni, Saurabh Mahindre, Abhijeet Kislay, Kevin Hughes, Roman Votyakov, khalednasr, Sanuj Sharma, Alesis Novik, Abinash Panda, Evangelos Anagnostopoulos, Liang Pang, Alex Binder, serialhex, and Björn Esser. shogun-toolbox/shogun: Shogun 6.1.0, November 2017. URL `https://doi.org/10.5281/zenodo.1067840`. 2.4.2

David Sontag and Tommi Jaakkola. Tree block coordinate descent for map in graphical models. In *Artificial Intelligence and Statistics*, pages 544–551, 2009. 2.3.2, 2.4.2

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening LP relaxations for MAP using message passing. In *UAI*, pages 503–510, 2008. 3.2

David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual composition for inference. In *Optimization for Machine Learning*. MIT Press, 2011. 2.3.2, 2.4.2

Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016. 4.1

Daniel Soudry and Elad Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *arXiv preprint arXiv:1702.05777*, 2017. 4.1

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 4.2

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. (document)

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2377–2385. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5850-training-very-deep-networks.pdf`. 8.1

Ruslan Leont'evich Stratonovich. Conditional markov processes. In *Non-linear transformations of stochastic processes*, pages 427–453. Elsevier, 1965. (document)

Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1891–1898, 2014. 4.1

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. ??

Richard Sutton. The bitter lesson. `http://incompleteideas.net/IncIdeas/BitterLesson.html`, 2019. 8.5

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015a. 8.1

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015b. 4.1, 4.2

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. 4.2, 4.5.1

Kui Tang, Nicholas Ruozzi, David Belanger, and Tony Jebara. Bethe learning of graphical models via MAP decoding. In *AIStats*, pages 1096–1104, 2016. 3.2, 3.B.1

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 25–32. MIT Press, 2003. 2.4.2, 2.4.2

Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning (ICML)*, 2018. (document)

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998. 6.1

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015. (document), 4.1, 4.5, 4.5.1, 4.5.4, 4.5.4

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 4.1, 4.5.4, 4.5.4, 4.5.3, 6.4

Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017. 5.1

Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014. 7.1

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005. 2.4.2, 2.4.2, 2.4.2, 2.4.1

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations (ICLR)*, 2017a. 4.5.3

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *International Conference on Learning Representations*, 2017b. 8.1

Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Abdelrahman Mohamed, Matthai Philipose, Matt Richardson, and Rich Caruana. Do deep convolutional nets really need to be deep and convolutional? In *ICLR*, 2017. 7.1, **??**, 7.3.3

Leslie G Valiant. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445. ACM, 1984. (document)

Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994. 1

Stylianos I Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. Toolflows for mapping convolutional neural networks on fpgas: A survey and future directions. *ACM Computing Surveys (CSUR)*, 51(3):56, 2018. 4.3

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11 (Dec):3371–3408, 2010. 7.2.2

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016. (document), 6.1, 6.4, 6.5.1, **??**

M. J. Wainwright. Estimating the wrong graphical model: Benefits in the computation-limited setting. *JMLR*, 7(Sep):1829–1859, 2006. 3.8

M. J. Wainwright. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008. (document), 2, 2.2, 2.3, 2.3.1, 2.3.2, 2.3.1, 2.3.3, 2.3.4, 2.3.5, 2.3.2, 3.3.1, 3.4.1

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *PAMI*, 2005a. 2.3.2

Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51 (7):2313–2335, 2005b. (document), 2.3.1, 3.4.3, 3.C

A.S. Weigend and B. Huberman. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990. 8.1

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 7.1, 7.3.2

Max Welling. Intelligence per kilowatt-hour, 2018. URL https://youtu.be/5DbBQDoBNYc. 4.3, 2

Max Welling. Do we still need models or just more data and compute? https://staff.fnwi.uva.nl/m.welling/wp-content/uploads/Model-versus-Data-AI-1.pdf, 2019. 8.5

Paul Werbos. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974. 4.2

Tomas Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007. 2.3.2, 2.4.2

Chenwei Wu, Jiajun Luo, and Jason D Lee. No spurious local minima in a two hidden unit relu network. In *International Conference on Learning Representation Workshop*, 2018a. 4.1

Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017. (document), 4.1, 3, 4.5.1, 4.5.1, 4.5.2, 4.5.1

Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*, 2018b. 4.4

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 8.2.4, 8.3

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017. 4.4

Jiaolong Xu, Liang Xiao, and Antonio M López. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 7:156694–156706, 2019. ??, 6.5.3

Jun Yang, Rong Yan, and Alexander G Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007. 4.6

Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015. 4.4

Chen Yanover, Talya Meltzer, and Yair Weiss. Linear programming relaxations and belief propagation–an empirical study. *Journal of Machine Learning Research*, 7(Sep): 1887–1907, 2006. 2.3.2

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8: 236–239, 2003. 2.3.1

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005. 2.3.1, 3.4.3, 3.C

Ian En-Hsu Yen, Xiangru Huang, Kai Zhong, Ruohan Zhang, Pradeep K Ravikumar, and Inderjit S Dhillon. Dual decomposed learning with factorwise oracle for structural SVM of large output domain. In *NIPS*, pages 5024–5032, 2016. 3.2, 3.7, 3.B.1

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Computer Vision and Pattern Recognition*, pages 4133–4141, 2017. (document), 7.1, 7.2.2

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. 4.6

Jiaqian Yu and Matthew B Blaschko. The lovász hinge: A novel convex surrogate for submodular losses. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2.4.2

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016a. 8.1, 8.4

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016b. (document), 7, 4.6, 7.1, 7.2.2, 7.3, 7.3.1

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016c. (document), 4.1, 4.2, 6.5.1

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization, 2014. URL `https://arxiv.org/abs/1409.2329`. 8.4.3

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. (document), 4.1, 4.5.1, 4.5.1, 3

Siqi Zhang and Niao He. On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization. *arXiv preprint arXiv:1806.04781*, 2018. 2

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 8.1

Liang Zhao, Siyu Liao, Yanzhi Wang, Zhe Li, Jian Tang, and Bo Yuan. Theoretical properties for neural networks with weight matrices of low displacement rank. In *International Conference on Machine Learning*, pages 4082–4090, 2017. 8.5

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. Compressibility and generalization in large-scale deep learning. *arXiv preprint arXiv:1804.05862*, 2018. (document), 4.1, 4.5, 4.5.1, 4.5.3

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003. 6.1

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 8.1