HAL
open science

Quelques applications de l'optimisation numérique aux problèmes d'inférence et d'apprentissage

Hariprasad Kannan

To cite this version:

Hariprasad Kannan. Quelques applications de l'optimisation numérique aux problèmes d'inférence et d'apprentissage. Intelligence artificielle [cs.AI]. Université Paris Saclay (COmUE), 2018. Français. NNT : 2018SACLC067 . tel-01996910

HAL Id: tel-01996910

https://theses.hal.science/tel-01996910v1

Submitted on 28 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université PARIS-SACLAY

CentraleSupélec

NNT : 2018SACLC067

Quelques applications de l'optimisation numérique aux problèmes d'inférence et d'apprentissage

Thèse de doctorat de l'Université Paris-Saclay
préparée à CentraleSupélec

Ecole doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Mathematiques et Informatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 28 septembre, 2018, par

Hariprasad Kannan

Composition du Jury :

Josiane Zerubia Directrice de Recherche, Inria Sophia Antipolis	Présidente
Laurent Najman Professeur, Université Paris-Est	Rapporteur
M. Pawan Kumar Professeur, University of Oxford	Rapporteur
Nikos Komodakis Professeur, École des Ponts ParisTech	Examinateur
Yuliya Tarabalka Chargée de Recherche, Inria Sophia Antipolis	Examinatrice
Nikos Paragios Professeur, CentraleSupélec	Directeur de thèse
Frédéric Chazal Directeur de Recherche, Inria Saclay	Invité

Thèse de doctorat

université PARIS-SACLAY

CentraleSupélec

NNT : 2018SACLC067

Few Applications of Numerical Optimization in Inference and Learning

PhD thesis of Paris-Saclay University
prepared at CentraleSupélec

Doctoral School n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
PhD specialty: Mathematiques et Informatique

Thesis presented and defended at Gif-sur-Yvette, on 28th septembre, 2018, by

Hariprasad Kannan

Composition of the Jury :

Josiane Zerubia Director of Research, Inria Sophia Antipolis	Chair
Laurent Najman Professor, Université Paris-Est	Reviewer
M. Pawan Kumar Associate Professor, University of Oxford	Reviewer
Nikos Komodakis Associate Professor, École des Ponts ParisTech	Examiner
Yuliya Tarabalka Researcher, Inria Sophia Antipolis	Examiner
Nikos Paragios Professor, CentraleSupélec	Advisor
Frédéric Chazal Director of Research, Inria Saclay	Invited member

Thèse de doctorat

Few Applications of Numerical Optimization in Inference and Learning

Abstract

Numerical optimization and machine learning have had a fruitful relationship, from the perspective of both theory and application. In this thesis, we present an application oriented take on some inference and learning problems. Linear programming relaxations are central to maximum a posteriori (MAP) inference in discrete Markov Random Fields (MRFs). Especially, inference in higher-order MRFs presents challenges in terms of efficiency, scalability and solution quality. In this thesis, we study the benefit of using Newton methods to efficiently optimize the Lagrangian dual of a smooth version of the problem. We investigate their ability to achieve superior convergence behavior and to better handle the ill-conditioned nature of the formulation, as compared to first order methods. We show that it is indeed possible to obtain an efficient trust region Newton method, which uses the true Hessian, for a broad range of MAP inference problems. Given the specific opportunities and challenges in the MAP inference formulation, we present details concerning (i) efficient computation of the Hessian and Hessian-vector products, (ii) a strategy to damp the Newton step that aids efficient and correct optimization, (iii) steps to improve the efficiency of the conjugate gradient method through a truncation rule and a pre-conditioner. We also demonstrate through numerical experiments how a quasi-Newton method could be a good choice for MAP inference in large graphs. MAP inference based on a smooth formulation, could greatly benefit from efficient sum-product computation, which is required for computing the gradient and the Hessian. We show a way to perform sum-product computation for trees with sparse clique potentials. This result could be readily used by other algorithms, also. We show results demonstrating the usefulness of our approach using higher-order MRFs. Then, we discuss potential research topics regarding tightening the LP relaxation and parallel algorithms for MAP inference.

Unsupervised learning is an important topic in machine learning and it could potentially help high dimensional problems like inference in graphical models. We show a general framework for unsupervised learning based on optimal transport and sparse regularization. Optimal transport presents interesting challenges from an optimization point of view with its simplex constraints on the rows and columns of the transport plan. We show one way to formulate efficient optimization problems inspired by optimal transport. This could be done by imposing only one set of the simplex constraints and by imposing structure on the transport plan through sparse regularization. We show how unsupervised learning algorithms like exemplar clustering, center based clustering and kernel PCA could fit into this framework based on different forms of regularization. We especially demonstrate a promising approach to address the pre-image problem in kernel PCA. Several methods have been proposed over the years, which generally assume certain types of kernels or have

too many hyper-parameters or make restrictive approximations of the underlying geometry. We present a more general method, with only one hyper-parameter to tune and with some interesting geometric properties. From an optimization point of view, we show how to compute the gradient of a smooth version of the Schatten p-norm and how it can be used within a majorization-minimization scheme. Finally, we present results from our various experiments.

Quelques applications de l'optimisation numérique aux problèmes d'inférence et d'apprentissage

Résumé

Les interactions entre optimisation numérique et apprentissage automatique ont apportées de nombreuses avancées tant du point de vue théorique que pratique. Cette thèse traite de problématiques applicatives d'inférence et d'apprentissage. Les relaxations en problème d'optimisation linéaire jouent un rôle central en inférence du maximum a posteriori (MAP) dans les champs aléatoires de Markov discrets. L'inférence dans les champs aléatoires de Markov d'ordre élevé représente notamment un défi en terme d'efficacité, de mise à l'échelle et de qualité de la solution. Nous étudions ici les avantages offerts par les méthodes de Newton pour résoudre efficacement le problème dual (au sens de Lagrange) d'une reformulation lisse du problème. Nous comparons ces dernières aux méthodes de premier ordre, à la fois en terme de vitesse de convergence et de robustesse au mauvais conditionnement du problème. Nous montrons qu'il est en effet possible d'obtenir une méthode de Newton à région de confiance compétitive basée sur la Hessienne pour un large panel de problèmes d'inférence MAP. Etant donnés les défis et les opportunités spécifiques à la formulation de l'inférence MAP, nous présentons en détail (i) le calcul rapide de la Hessienne ainsi que son produit par un vecteur, (ii) une stratégie pour amortir le pas de Newton visant à améliorer la précision et à diminuer le temps de calcul (iii) les différentes étapes pour améliorer l'efficacité de la méthode du gradient conjugué au travers de la règle de troncature et d'un préconditionneur. Nous démontrons également, au travers de simulations numériques, que le choix d'une méthode de quasi-Newton peut être avantageux pour l'inférence MAP dans des graphes de grandes dimensions. L'inférence MAP basée sur une formulation lisse peut grandement tirer profit d'un calcul rapide de somme-produit qui est nécessaire à la fois lors du calcul du gradient et de la hessienne. Nous détaillons une technique pour calculer la somme-produit pour des arbres contenant des cliques aux potentiels parsimonieux. Cette dernière peut également être facilement étendu à d'autres algorithmes. Nous montrons plusieurs résultats prouvant l'utilité de notre approche en utilisant des champs aléatoires de Markov d'ordre plus élevé. Nous discutons ensuite d'éventuels extensions liées à une relaxation plus stricte et à l'utilisation d'algorithmes parallèles pour l'inférence MAP.

L'apprentissage non-supervisé est un domaine de l'apprentissage automatique d'un intérêt particulier pour les problèmes de grande dimension comme l'inférence dans les modèles graphiques. Nous exposons donc un cadre général pour l'apprentissage non-supervisé basé sur le transport optimal et les régularisations parcimonieuses. Le transport optimal soulève des défis intéressants du point de vue de l'optimisation, de par ses contraintes de type simplexe sur les lignes et les colonnes du plan de transport. En s'inspirant du transport optimal, nous présentons une méthode pour formuler des problèmes d'optimisation avantageux en imposant

un unique ensemble de contraintes de type simplexe, ainsi qu'une structure sur le plan de transport au travers d'une régularisation parcimonieuse. Nous montrons alors que les algorithmes d'apprentissage non supervisé, tels que les méthodes de partitionnement des données et l'ACP à noyau, rentrent dans ce cadre selon les différentes regularisations. Nous exhibons notamment une approche prometteuse pour résoudre le problème de la préimage dans l'ACP à noyau. Bien que de nombreuses méthodes similaires aient été proposées durant les dernières années, celles-ci ne fonctionnent que pour certains noyaux particuliers, possèdent de trop nombreux hyperparamètres ou encore imposent des approximations trop restrictives sur la géométrie sous-jacente du problème. Nous construisons dans cette thèse une méthode plus générale avec un unique hyper-paramètre à régler et qui possède des propriétés géométriques intéressantes. Du point de vue de l'optimisation, nous décrivons le calcul du gradient d'une version lisse de la norme p de Schatten et comment cette dernière peut être utilisée dans un schéma de majoration-minimisation. Enfin, nous discutons les résultats de nos diverses simulations numériques.

கல்வி கரையில; கற்பவர் நாள்சில
மெல்ல நினைக்கிற் பிணிபல - தெள்ளிதின்
ஆராய்ந் தமைவுடைய கற்பவே நீரொழியப்
பாலுண் குருகிற் றெரிந்து.

-Nāladi
circa 800 A.D.

Learning is a shoreless sea; the learner's time is limited;
Prolonged study is beset with many ills;
With clear discrimination learn what's good for you;
Like the swan that leaves the water, to drink the milk.

Acknowledgments

I would like to thank my advisor, Nikos Paragios, for giving me the great opportunity to pursue a PhD. Several years back, I had come across the book "Handbook of Mathematical Models in Computer Vision" and it felt like a nice foundation to someone working in an applied area like computer vision. I have fond memories of reading the chapter written by him. Being on the technical path has been a long-standing dream and I am happy that I ended up pursuing my PhD with the person who wrote that chapter and also, co-edited that book.

I would like to thank my labmates for their help in various ways - getting a place in the creche for my son, applying for carte vitale, obtaining CAF, finding a place to stay, making phone calls in French and calling me for lunch. I thank Alp, Arthur, Enzo, Eugene, Evgenios, Guillaume, Jiaqian, Khue, Maria, Marie-caroline, Maxim, Mihir, Puneet, Pritish, Rafael, Siddharth, Stavros (both), Stefan, Vivien and Wacha. The help of my labmates has been critical, to say the least. I thank the administrators of my lab, Natalia, Jana and Alexandra, for all their support to take care of various challenges that keep coming up. At this point, I thank the administrators and technical staff at CentraleSupélec, which includes Antony, Prof. Duc and Mme. Batalie. I thank the other researchers in the lab, who added to the dynamic atmosphere in the lab: Edouard, Émilie, Eva, Fragkiskos, Iasonas, Marc, Matthew, Prof. Pesquet and Prof. Talbot.

I thank my thesis reviewers, Prof. Laurent Najman and Prof. Pawan Kumar, for their time and feedback. I appreciate their effort and their specific inputs. It has helped me as a researcher. I thank the rest of the members of the jury, Prof. Josiane Zerubia, Prof. Nikos Komodakis and Prof. Yuliya Tarabalka, for their support, which I will always appreciate. I thank Prof. Komodakis for the technical discussions. I thank Prof. Francis Bach for his time to discuss technical matters. It is nice to see his dedicated effort towards work. I thank the people who introduced me to him.

I thank my parents for their love. They have been instrumental in shaping my thoughts about studies during my childhood. I thank my in-laws for helping to take care of my son during crucial junctures. I thank my wife's manager, Mr. S. V. Desai, for his support. I thank my wife and my son for the various ways in which they have supported me in this pursuit.

Contents

1 Introduction **3**

2 LP Relaxations for MAP Inference **5**
2.1 Notation and Terminology 6
2.2 MAP inference 6
2.3 Exact inference in hypertrees 9
2.4 LP relaxation based approach 10
2.5 Efficient dual based formulation 15
2.6 Unconstrained dual 18

3 Optimization Algorithms for MAP Inference **21**
3.1 Supergradient based optimization 21
3.2 Smooth accelerated gradient method 22
3.3 Coordinate Maximization Methods 25
3.4 Augmented Lagrangian Methods 28
3.4.1 AD3 29
3.5 Estimating primal variables 31
3.6 Lipschitz constant of the smooth dual 33
3.7 Dual decomposition for large graphs 34
3.8 Efficient belief propagation in trees 36

4 Newton Methods and Linear System Solvers **39**
4.1 Linear System Solvers 41
4.2 The Conjugate Gradient Method 42
4.3 Convergence of CG and Preconditioning 45
4.4 Truncated Newton methods 47
4.4.1 Truncating CG iterations 47
4.5 Line search Newton 48
4.6 Trust-region Newton 50
4.7 The Steihaug Method 51
4.8 Damping matrix based approach 52
4.9 Quasi-Newton methods 53
4.10 Preconditioning 55
4.10.1 Diagonal preconditioner 56
4.10.2 Block diagonal preconditioner 56
4.10.3 Incomplete Cholesky Preconditioner 56
4.10.4 Multigrid Preconditioner 57
4.10.5 Quasi-Newton preconditioner 57
4.10.6 Combinatorial preconditioner 58
4.11 Hessian-vector products 58

5 Newton Methods for Inference in Higher-order Markov Random Fields 59
5.1 Ill-conditioning and affine invariance . . . 59
5.1.1 Hessian related computations . . . 59
5.1.2 Damping matrix approach . . . 61
5.1.3 Forcing sequence for CG truncation . . . 62
5.1.4 Clique based Preconditioner . . . 62
5.1.5 Backtracking search . . . 63
5.1.6 Annealing schedule and stopping condition . . . 63
5.1.7 TRN-MRF algorithm and parameter settings . . . 64
5.2 Quasi-Newton approach for large graphs . . . 64
5.3 Experiments . . . 65
5.3.1 Higher-order Stereo . . . 68
5.4 Discussion . . . 68

6 Some thoughts on continuous relaxation based MAP inference 71
6.1 Notes about Newton methods . . . 71
6.1.1 Stochastic Newton . . . 71
6.1.2 Partial separability and quasi-Newton methods . . . 71
6.1.3 Projected Newton for small and medium graphs . . . 72
6.2 Preconditioning first order optimization methods . . . 72
6.3 Numerical linear algebra and Graphical models . . . 73
6.4 Tightening the relaxation . . . 73
6.5 Parallel algorithms to solve the LP relaxation . . . 74
6.5.1 Enforcing node consensus in distributed setting . . . 76
6.5.2 A note on asynchronous projected gradient for MAP inference 77

7 Unsupervised learning: a perspective based on optimal transport and sparse regularization 79
7.1 Archetypal analysis . . . 81
7.2 A loss encouraging reduced output variance . . . 83
7.3 K-medoids clustering . . . 84
7.4 Center based clustering . . . 86
7.5 Kernel PCA and the pre-image problem . . . 87
7.5.1 Our Approach . . . 91
7.5.2 Optimization . . . 93
7.5.3 Out-of-sample extension . . . 94
7.5.4 Robust pairwise cost . . . 95
7.6 Experiments . . . 96
7.7 Concluding remarks . . . 99

8 Conclusion 101

9 Appendix **103**
9.1 Convex Optimization 103
9.1.1 Halfspace 103
9.1.2 Dual norm 103
9.1.3 Fenchel conjugate 103
9.1.4 α strong convexity 103
9.1.5 L smoothness 103
9.1.6 Strict convexity 104
9.1.7 Prox function for the simplex 104
9.1.8 M norm of a vector 104
9.2 Numerically stable sum-product computation 104
9.3 Gradient of Variational Majorant 105

Bibliography **107**

List of Figures

2.1 (a) A graph with pair-wise edges: nodes $= \{v_1, v_2, v_3, v_4, v_5\}$ and edges $= \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_4, v_5\}\}$ (b) A hypergraph with hyperedges: nodes $= \{v_1, v_2, v_3, v_4, v_5\}$ and hyperedges $= \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_4, v_5\}\}$ 5

2.2 Sepset based visualization of a clique tree, where ovals indicate cliques and rectangles indicate sepsets. 9

2.3 A cartoonish illustration of the relationship between the marginal polytope $\mathcal{M}(\mathcal{G})$ and the local polytope $\mathcal{L}(\mathcal{G})$. All the corners of $\mathcal{M}(\mathcal{G})$ are also corners of $\mathcal{L}(\mathcal{G})$. $\mathcal{L}(\mathcal{G})$ also has fractional corners. The fact that $\mathcal{L}(\mathcal{G})$ has lesser number of facets than $\mathcal{M}(\mathcal{G})$ is difficult to visualize. *Image courtesy [Wainwright 2008].* 14

3.1 Superlevel sets of two dimensional non-smooth problems: (a) Separable non-smooth part, guaranteed convergence to global optimum (b) General non-smooth function, with a shape that is not favourable to coordinate ascent. *Image courtesy G. Gordon & R. Tibshirani, Optimization 10-725 slides, CMU.* 26

3.2 A clique chain. 37

4.1 Computing the Newton direction by forming the quadratic approximation. *Image adapted from https://suzyahyah.github.io/calculus/optimization/2018/04/06/Taylor-Series-Newtons-Method.html.* 39

4.2 For an ill-conditioned problem, the quadratic approximation is a long bowl. *Image adapted from [Nocedal 2006, §4].* 51

5.1 The two components of the Hessian. Within each component, blocks of the same color have the same values. In component one, there are as many unique blocks as cliques. In component two, each row/column of blocks has the same block, corresponding to the shared node. . . . 60

5.2 Matching 1st frame to 90th frame. 67

5.3 Tsukuba and Venus results for quasi-Newton. 69

7.1 Two dimensional data with 60 points. (left) frobenius norm based loss, (right) optimal transport based loss. Optimal transport based loss penalizes variance of the output points and those points are close to or coincide with input points. 97

7.2 From left to right: [Mika 1999], [Kwok 2004], [Bakir 2004], our approach. [Mika 1999] and [Kwok 2004] use RBF kernel. [Bakir 2004] and ourselves use polynomial kernel. 97

Chapter 1

Introduction

In this thesis, we will see how numerical optimization could be applied to address some problems in computer vision and machine learning. The two types of problems that we will be looking at are (i) maximum a posteriori (MAP) inference in Markov random fields (MRFs), (ii) some topics in unsupervised learning. For various machine learning problems, numerical optimization helps with formulating ideas, designing algorithms and analysing guarantees. Some of the challenges that we come across while working with numerical optimization are: 1. the trade-off between tractability and performance while working with a convex formulation of a learning task; 2. exploiting problem structure to achieve an efficient and scalable algorithm. We hope to better appreciate these aspects through this thesis.

Many computer vision problems can be modelled using Markov Random Fields (MRFs), an undirected graphical model. Maximum a posteriori (MAP) estimation in an MRF, is the process of assigning labels to the nodes of the graph, in order to maximize the joint probability distribution. Among various approaches for performing MAP inference, the LP relaxation based approach has theoretical and practical advantages. We will understand these aspects better in chapter (2). Over the years, there have been several algorithms to solve the LP relaxation. We will get a taste for some of them in chapter (3).

In recent years, Newton methods have led to impressive results in various optimization based machine learning algorithms [Schmidt 2010,Martens 2010,Lee 2012,Scheinberg 2016]. These methods are able to move along a better direction in the landscape of the objective function by considering curvature information and have quadratic convergence rate when sufficiently close to the optimum. One of the challenges while solving any optimization problem, is the conditioning of the objective function. Intuitively, a problem is ill-conditioned if the change in the objective value, due to a perturbation in the variable value, could vary radically depending on the direction of the perturbation. This nature of the objective function is characterized by the condition number. Theoretical and empirical evidence show that first order methods are faster when the condition number is small or moderate but second order Newton methods perform much better for ill-conditioned problems [Fountoulakis 2015]. A first order method makes progress by considering only gradient information. On the other hand, a second order method considers the curvature, which is quantified by the Hessian. In MAP inference for MRFs, we will see that we will come across ill-conditioned optimization problems. We will get a better understanding of Newton methods in chapter (4) and go through our numerical experience with Newton methods for MAP inference in chapter (5). To better appreciate future research

possibilities, we will discuss various avenues in the topic of MAP inference in chapter (6).

We will look at some unsupervised learning problems in chapter (7), which form the second type of problems that this thesis has focused on. We would like to point out that graphical models are high dimensional by nature and unsupervised learning has an important role to play in making graphical models more scalable. In this work, we will discuss how some unsupervised learning problems could be formulated using sparse regularizers [Mairal 2014] along with a loss function based on optimal transport [Kolouri 2017]. We will see how different sparse regularizers encourage different structures in the output. We will see how the loss function based on optimal transport ties with a topic called archetypal analysis in unsupervised learning. A comparison of the optimal transport inspired loss and the loss based on Frobenius norm will be made. The algorithmic aspects for each of the optimization problems will also be discussed.

Chapter 2

LP Relaxations for MAP Inference

Many problems involve a set of interacting random variables, where the joint probability distribution is over the product space of the random variables. Even though such product spaces are very large, there will be some structure in how the random variables interact. Probabilistic graphical models [Koller 2009, Wainwright 2008, Koller 2007] model such problems, via directed or undirected graphs, where the nodes are the random variables and the interactions are represented by edges. In this thesis we will focus on *undirected* graphical models, which are referred to as Markov random fields (MRFs). Several real world problems in diverse fields such as computer vision [Blake 2011], natural language processing [Sutton 2012], communication systems [Frey 1998], sensor networks [Chen 2006], computational biology [Ma 2014] etc, could be modeled by MRFs. The random variables could take continuous or discrete values. We focus on *discrete* values or *labels* in this thesis. A graph can have edges between pairs of nodes or hyperedges which are comprised of three or more nodes. A graph with hyperedges is a hypergraph (figure 2.1). Markov random fields with hyperedges are called *higher-order* Markov random fields. In recent years, higher-order MRFs have achieved excellent results in various applications, since they model far-reaching interactions between the nodes. Development of scalable and efficient techniques for higher order models is an evolving topic [Ishikawa 2011], [Komodakis 2009], [Kohli 2009], [Fix 2014], [Kolmogorov 2015], [Vineet 2014]. You will find a bias towards higher-order MRFs in this thesis.

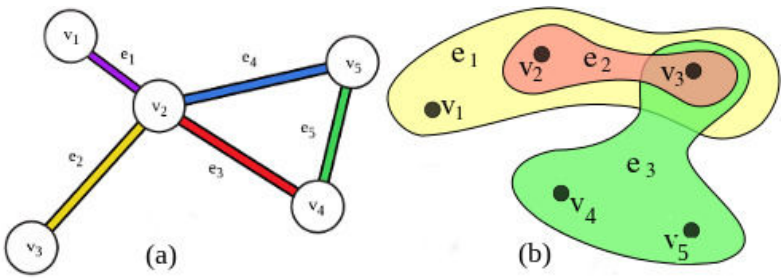

Figure 2.1: (a) A graph with pair-wise edges: nodes $= \{v_1, v_2, v_3, v_4, v_5\}$ and edges $= \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_4, v_5\}\}$ (b) A hypergraph with hyperedges: nodes $= \{v_1, v_2, v_3, v_4, v_5\}$ and hyperedges $= \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_4, v_5\}\}$

2.1 Notation and Terminology

We will clarify some notation and graph theoretic language before we go further. A graph $\mathcal{G}$ is made of nodes, with $\mathcal{V}$ denoting the set of all nodes. We will use capital letters to denote random variables. We will usually work with n discrete variables $X_1, ..., X_n$ and they correspond to the nodes in a graph. Each variable X_i takes labels from a set of labels $\boldsymbol{l}_i$ of size l_i. Generally, it is possible for each node i to have a separate set of labels, i.e., $\boldsymbol{l}_i$ and l_i could change with i. In practice, we may see all variables taking labels from a common set $\boldsymbol{l}$ of size l. A particular realization (labelling) of X_i is denoted by x_i. A clique is a set of nodes such that every pair of nodes has an edge between them. In the language of graph theory, the nodes of a clique induces a complete subgraph. In this thesis, a clique could be a pairwise edge or a hyperedge depending on the context. The set of all hyperedges/cliques is denoted as C. Thus, each $c \in \mathcal{C}$ is a subset of variables. Note that even though small e is the standard notation for (hyper)edges in graphs, in the graphical models literature, we refer to them as cliques, with small c's. The set of variables corresponding to a clique c is indicated by bold capital letters, e.g., $\boldsymbol{X}_c$. Thus, $\boldsymbol{X}$ denotes the set of all variables in the graph. The nodes of a clique each take a label, thus together they lead to a realization (labelling) of the clique, which we refer to as $\boldsymbol{x}_c$, the bold notation shows that it is a vector of node labels. Thus the clique could be assigned a label from a set $\boldsymbol{l}_c$, whose size is equal to the product of the label set sizes of the clique's nodes.

2.2 MAP inference

A natural question to ask is, what does the notion of a graphical model on a set of random variables signify? Here we will discuss what properties about the variables is captured by an undirected graphical model. First, *conditional independence* relationships among the variables is readily represented by the graph. In figure 2.1, for the pairwise graph on the left, the conditional probability of X_1 given the other variables takes a simpler form as follows, $P(X_1|X_2, X_3, X_4, X_5) = P(X_1|X_2)$. In words, given X_2, X_1 is independent of the other variables. In general, the smallest set of nodes that renders a node conditionally independent of all the other nodes, is called its *Markov blanket*. When we remove all the nodes of the Markov blanket, there will be no path connecting that node with the rest of the graph. Another property represented by an MRF structure is that the joint probability distribution factorizes according to overlapping subsets of the variables. More precisely the joint probability distribution factorizes according to *potential functions* associated with *maximal* cliques of the graph. A maximal clique is a clique such that it is not possible to include one more node to the set such that it will still be a clique. We have to keep in mind that unlike directed graphs, the potential functions of undirected graphs do not have a clear probabilistic interpretation. They are generally designed to signify compatibility among the nodes, for a particular application. For the graph

on the left in figure 2.1, the joint probability distribution could factorize as follows,

$$P(X_1, X_2, X_3, X_4, X_5) = \Psi_1(X_1, X_2)\Psi_2(X_2, X_3)\Psi_3(X_2, X_4, X_5). \tag{2.1}$$

For the hypergraph in figure 2.1, the joint probability distribution could factorize as follows,

$$P(X_1, X_2, X_3, X_4, X_5) = \Psi_1(X_1, X_2, X_3)\Psi_2(X_3, X_4, X_5). \tag{2.2}$$

In both cases, $\Psi_i(\cdot)$ denote appropriate potential functions. In figure 2.1, we see a maximal clique made of three individual edges (left) and a clique that is completely contained inside another (right). We will see such grouping of variables in practice. In fact, it is possible to represent the joint probability distribution according to smaller cliques of the graph [Bishop 2006, §8.3.3]. Given that the potential functions do not have a clear probabilistic interpretation, it is possible to define potential functions with respect to cliques or even nodes, which are contained within maximal cliques. We have seen two properties encoded by an MRF: conditional independence according to Markov blankets and factorization of joint probability according to maximal cliques. The Hammersley-Clifford theorem [Koller 2009] states that if the potential functions are strictly positive, then these two properties are equivalent and one holds iff the other holds.

The restriction to strictly positive functions reveals a strong connection between Markov random fields and statistical physics. It is possible to write the strictly positive potential function of a clique c as,

$$\Psi_c(\boldsymbol{x}_c) = \exp(-\theta_c(\boldsymbol{x}_c)) \tag{2.3}$$

where $\theta_c(\boldsymbol{x}_c)$ is the *energy* of clique c for the labelling $\boldsymbol{x}_c$. Since, the joint probability distribution is a product of potential functions, we obtain a sum of energies in the exponent. Now, in graphical models literature, it is common to model problems based on potential functions defined on nodes and cliques. We will work with θ_i and θ_c, energies corresponding to the nodes and the cliques of the graphical model. $\theta_i(x_i)$ is the energy corresponding to the node i taking the label x_i and $\theta_c(\boldsymbol{x}_c)$ is the energy corresponding to the clique c for the labelling $\boldsymbol{x}_c$. It is better for modelling applications to have these two explicit types of potential functions. Thus the joint probability distribution takes the following form,

$$P(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i \in \mathcal{V}} \Psi_i(x_i) \prod_{c \in \mathcal{C}} \Psi_c(\boldsymbol{x}_c) = \exp\big(- \sum_{i \in \mathcal{V}} \theta_i(x_i) - \sum_{c \in \mathcal{C}} \theta_c(\boldsymbol{x}_c)\big). \tag{2.4}$$

An important computation that one would like to perform with MRFs is to find the labelling $\boldsymbol{x}$ of all the random variables, which will maximize the joint probability distribution, $P(\boldsymbol{X})$. This computation is called maximum a posteriori inference or MAP inference. However, except for some special graphs, this maximization over the set of possible labellings is NP-hard [Shimony 1994, Li 2016]. Thus, there has been considerable research effort to develop approximate inference algorithms for MRFs.

Maximizing the joint probability in (2.4) is equivalent to minimizing the following energy,

$$\sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{c\in\mathcal{C}} \theta_c(\boldsymbol{x}_c) \triangleq E(\boldsymbol{x};\boldsymbol{\theta}). \tag{2.5}$$

In the graphical models literature, θ_i and θ_c are referred to as *unary potentials* and *clique potentials*, respectively. This naming could be confused with the potential functions that we have seen earlier. In order to be compatible with the literature, we will also call these quantities as potentials. Talking about terminology, other terms that one will come across are *factors* and *regions*. We will briefly indicate what they mean. A graphical model can be equivalently represented by a bipartite graph called a factor graph [Kschischang 2001,Loeliger 2004], which is made of nodes corresponding to the random variables and nodes called factors. Each set of variables that define potential functions for the joint probability distribution(2.4) corresponds to a factor. Thus, we will see both nodes and cliques being referred to as factors. Also, we will see regions, which is a more general concept but they are also used to refer to the sets of variables that define the potential functions [Welling 2004].

Thus, the problem of MAP inference in Markov random fields is about finding the labels x_i of all nodes $i \in \mathcal{V}$ such that the energy in (2.5) is minimized. We would like to point out that if the clique potentials $\theta_c(\boldsymbol{x}_c)$ are not present, this energy could be trivially minimized over individual nodes. It is the presence of cliques, with overlapping sets of nodes that makes the problem intractably hard. Note that when the nodes of a clique c have been assigned labels, that automatically determines the labelling $\boldsymbol{x}_c$ of the clique. Generally, each node i could take one of l_i labels from the set $\boldsymbol{l}_i$. Suppose all label sets have same size l, then there are l^n possible label assignments for the entire graph. Thus, MAP inference has an exponential worst case complexity. Here, we will clearly state the discrete optimization problem that we are trying to solve,

$$\begin{aligned} \underset{x_1,x_2,\ldots,x_n}{\text{minimize}} \quad & \sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{c\in\mathcal{C}} \theta_c(\boldsymbol{x}_c) \\ \text{subject to} \quad & x_1 \in \boldsymbol{l}_1, x_2 \in \boldsymbol{l}_2, \ldots, x_n \in \boldsymbol{l}_n. \end{aligned} \tag{2.6}$$

Several approaches exist to minimize this energy efficiently and approximately: *graph cuts*, *belief propagation* and LP *relaxation* based methods. Refer [Kappes 2015] for a good survey about various methods. The LP relaxation approach has led to state-of-the-art algorithms and also, provides a theoretical foundation to the topic of MAP inference [Wainwright 2005a,Komodakis 2016]. An attractive property of this approach is that it readily lends itself to inference in higher-order MRFs [Komodakis 2009], [Sontag 2008]. Immediately, we will see MAP inference in trees by belief propagation, which will help us understand the LP relaxation based derivation better.

2.3 Exact inference in hypertrees

There are special cases depending on the graph topology and the clique potentials, where the exact MAP labelling could be inferred. Here, we will see one special case, which is useful as a subproblem while performing MAP inference in general graphs. This special case is that of trees, whether it is made of pairwise or higher-order cliques. The following description has been adapted from [Koller 2009, §10.2, 13.3].

Inference in graphical models could be understood through the process of message passing in a *factor graph* [Kschischang 2001,Loeliger 2004,Sudderth 2008]. In the case of a tree with higher-order cliques a useful data structure is a *clique tree*. Intuitively, it is the generalization of a tree with pairwise edges to a hypertree with hyperedges, i.e., edges with more than two nodes. See, [Koller 2009, §4.5.3] for a formal definition. A clique is a set of nodes and let us denote the ith clique as c_i. We define another set of nodes called the *sepset*. Given two cliques c_i and c_j, the sepset s_{ij} contains the nodes in $c_i \cap c_j$. In figure 2.2, we show the visualization using sepsets of a clique tree containing the cliques $\{\{X_0, X_1, X_2\}, \{X_1, X_2, X_3\}, \{X_2, X_3, X_4\}, \{X_3, X_4, X_5\}\}$. It is in fact a clique chain and we will see more about clique chains in section (3.7).

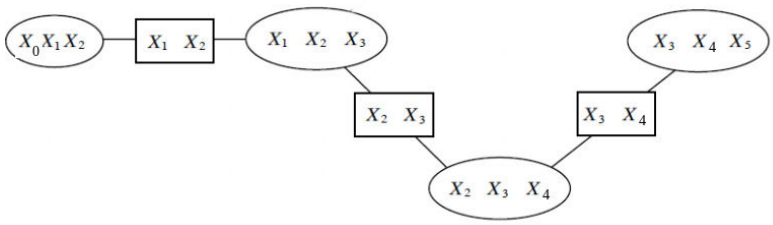

Figure 2.2: Sepset based visualization of a clique tree, where ovals indicate cliques and rectangles indicate sepsets.

In this clique tree, each message is between a clique and its neighbour and it is computed according to the following *max-product* or *min-sum* rule,

$$\boldsymbol{\delta}_{c_i \to c_j}(s_{ij}) \leftarrow \min_{\boldsymbol{x}_c \in c_i \sim s_{ij}}. \theta_{c_i}(\boldsymbol{x}_c) + \sum_{i:i \in c} \theta_i(x_i) + \sum_{c_k : c_k \in \mathcal{N}(c_i) \setminus c_j} \boldsymbol{\delta}_{c_k \to c_i}(s_{ki}). \tag{2.7}$$

To understand what is going on, let us first note that the size of the message that is sent is equal to the total number of labellings of the sepset between the two cliques. The total number of labellings of the source clique is more because it has more nodes than the sepset. Thus for a given labelling of the sepset $\boldsymbol{x}_{s_{ij}}$, there are several corresponding labellings $\boldsymbol{x}_{c_i}$ w.r.t. the source clique. We denote these labellings as $\boldsymbol{x}_c \in c_i \sim s_{ij}$. For each of these clique labellings, we can compute $\theta_c(\boldsymbol{x}_c) + \sum_{i:i \in c} \theta_i(x_i) + \sum_{c_k : c_k \in \mathcal{N}(c_i) \setminus c_j} \boldsymbol{\delta}_{c_k \to c_i}(s_{ki})$. Here, the last summation is over messages received from other neighbouring cliques $(\mathcal{N}(c_i) \setminus c_j)$. The minimum over $\boldsymbol{x}_c \in c_i \sim s_{ij}$ becomes the message corresponding to the sepset labelling $\boldsymbol{x}_{s_{ij}}$.

Intuitively, exact inference in a tree is possible because of how the message passing could be scheduled in a tree. Suppose, we denote some clique as the root clique and denote cliques connected with only one other clique as leaf cliques. Let us first pass messages from the leaf cliques towards the root clique, followed by messages from the root towards the leaves. Since, the graph has a tree topology, for a given message between a source clique and a target clique, it is possible for the source clique to receive messages from all other clique neighbours before passing a message to its target clique. Thisschedule is not possible for graphs with loops. In fact, in a tree, exact inference is achieved by one upward and one downward pass of messages. Thus, at the end of the procedure, a clique would have passed messages to all its neighbouring cliques and it would have received messages from all of them. Thus, we can compute *beliefs* for each of the clique labellings as follows,

$$\beta_{\boldsymbol{x}_c} \leftarrow \theta_{c_i}(\boldsymbol{x}_c) + \sum_{i:i\in c} \theta_i(x_i) + \sum_{k:k\in\mathcal{N}(c_i)} \boldsymbol{\delta}_{c_k\to c_i}(s_{ki}). \tag{2.8}$$

Such beliefs can be computed for each node-label pair, also. These are referred to as the *min-marginals* of the nodes. If the label of a node is held fixed and the energy is minimized, the resulting energy is the min-marginal for that node-label pair. Once this is computed for all nodes, the label corresponding to the minimum value for a given node is the MAP labelling of that node. Note, the original max-product algorithm works with the potential function representation (2.3). We have discussed the min-sum version that involves the energies. There is a related algorithm called *sum-product* message passing, which leads to the computation of the node marginals. We will see this algorithm in section (3.8). Both max-product and sum-product algorithms optimize subproblems within LP relaxation based algorithms for MAP inference.

2.4 LP relaxation based approach

Now, we will see about MAP inference in general graphs. First, we observe that the objective of the energy minimization problem has a linear form. Thus, it can be equivalently represented as an integer linear program (ILP) as follows,

$$\text{minimize} \quad \sum_i \sum_{x_i} \theta_i(x_i)\phi_i(x_i) + \sum_c \sum_{\boldsymbol{x}_c} \theta_c(\boldsymbol{x}_c)\phi_c(\boldsymbol{x}_c) \tag{2.9a}$$

$$\text{subject to} \quad \sum_{x_{c\setminus i}} \phi_c(\boldsymbol{x}_c) = \phi_i(x_i), \quad \forall (i,c) : i \in c \tag{2.9b}$$

$$\begin{gathered} \sum_{x_i} \phi_i(x_i) = 1, \quad \forall i \in \mathcal{V}; \quad \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c) = 1, \quad \forall c \in \mathcal{C} \\ \phi_i(x_i) \in \{0,1\}, \ \forall i \in \mathcal{V}, x_i \in \boldsymbol{l}_i; \quad \phi_c(\boldsymbol{x}_c) \in \{0,1\} \ \forall c \in \mathcal{C}, \boldsymbol{x}_c \in \boldsymbol{l}_c. \end{gathered} \tag{2.9c}$$

Here, $\phi_i(x_i)$ and $\phi_c(\boldsymbol{x}_c)$ are boolean indicator variables and are the decision variables of the discrete optimization problem. For each node $i \in \mathcal{V}$ and for each label $x_i \in \boldsymbol{l}_i$, there is a indicator variable $\phi_i(x_i) \in \{0,1\}$. Similarly, for each clique $c \in \mathcal{C}$ and

for each labelling $\boldsymbol{x}_c \in \boldsymbol{l}_c$, there is an indicator variable $\phi_c(\boldsymbol{x}_c) \in \{0, 1\}$. For a given labelling of the graph $\mathcal{G}$, only one of $\phi_i(x_i)$ will be set to 1 for each node i and for each clique c, only one of $\phi_c(\boldsymbol{x}_c)$ will be set to 1. This is expressed by the set of constraints (2.9c), called *normalization* constraints. When MAP inference is formulated as an energy minimization problem (2.6), we work with only the discrete node labels x_i as the optimization variables. The clique labellings $\boldsymbol{x}_c$ are dependent variables and are fixed automatically by the node labels. When the same MAP inference problem is formulated as an ILP (2.9), we work with two sets of discrete decision variables, $\phi_i(x_i)$ for the nodes and $\phi_c(\boldsymbol{x}_c)$ for the cliques. Nevertheless, the labelling of the nodes and the labelling of the cliques must agree and that is enforced by the constraint (2.9b). These constraints are called *marginalization* or *consistency* constraints. The summation $\sum_{x_{c\setminus i}} \phi_c(\boldsymbol{x}_c)$ is performed as follows: for a given clique c and node $i \in c$, there will be a set of $\boldsymbol{x}_c$'s which correspond to the node i taking label x_i, the summation is performed over the corresponding $\phi_c(\boldsymbol{x}_c)$'s and the constraint says that the sum should match with $\phi_i(x_i)$.

This representation with linear constraints between the indicator variables shows that the indicator variables are not independent, thus it is possible to have another set of potentials $\tilde{\theta}_i(x_i)$ and $\tilde{\theta}_c(\boldsymbol{x}_c)$, such that $P(\boldsymbol{X}; \theta) = P(\boldsymbol{X}; \tilde{\theta})$ or equivalently, $E(\boldsymbol{x}; \theta) = E(\boldsymbol{x}; \tilde{\theta})$. Thus the above representation based on indicator variables, is referred to as the *canonical overcomplete representation*. Also, θ is called a *reparameterization* of $\tilde{\theta}$ and we will revisit this concept, while discussing about optimization algorithms. Also, it is possible to define *minimal* representations using a smaller set of suitably defined discrete variables and still represent the same energy in (2.6) but we will go with the overcomplete representation, as it is well studied to develop optimization algorithms.

For ease of exposition, we will assume that there are n nodes, all taking labels from a common set of size l and that there are $|\mathcal{C}|$ cliques, each having s nodes. If not otherwised stated, the concepts that we will discuss hold in the general setting. For each node i, let $\boldsymbol{\phi}_i$ represent the vector obtained by stacking the indicator variables $\phi_i(x_i)$. Similarly, for each clique c, let $\boldsymbol{\phi}_c$ represent the vector obtained by stacking the indicator variables $\phi_c(\boldsymbol{x}_c)$. We will let $\boldsymbol{\phi}$ represent the vector, which is obtained by stacking all the vectors $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_c$. When the constraints (2.9b) and (2.9c) are satisfied, each instance of $\boldsymbol{\phi}$ represents a valid labelling of the entire graphical model. Each instance of $\boldsymbol{\phi}$ is a point in a high dimensional space, which is of size $n.l + |\mathcal{C}|l^s$. The convex hull of all possible instances of $\boldsymbol{\phi}$ is called the *marginal polytope* and is denoted by $\mathcal{M}(\mathcal{G})$. Now, each node i in the graphical model is associated with a marginal vector $\boldsymbol{\mu}_i$, which is the vector of probabilities $P(X_i = x_i)$. The marginal probabilities $P(X_i = x_i)$ are obtained by fixing the label of node i to x_i and summing $P(\boldsymbol{X})$ over the other variables. Similarly, each clique c is associated with a marginal vector $\boldsymbol{\mu}_c$. Each point $\boldsymbol{\mu}$ that belongs to $\mathcal{M}(\mathcal{G})$, is a concatenation of valid marginal vector for all nodes and cliques. Apart from maximum a posteriori inference, one could also seek marginal inference, which is to infer the marginal probabilities of all the variables in a graphical model. We will not focus on marginal inference in this thesis but point out that LP relaxation based techniques could be used for marginal

inference, also [Wainwright 2005b, Krishnan 2015].

In general, a polytope could be represented either as the convex hull of a set of points or as the intersection of *halfspaces* (refer (9.1) for definition). Intuitively, these halfspaces correspond to the facets of the polytope. More number of facets means more complexity in representation of the polytope. Ensuring that a vector of numbers is a valid marginal vector requires the vector to satisfy the *exhaustive* set of *marginalization* constraints, i.e., they involve all possible pairs of regions $(\alpha,\ \beta)$ where $\beta \subset \alpha$. This leads to a large number of facets for the marginal polytope. It is conjectured that the number of facets must be super-polynomial in the graph size [Wainwright 2008, section 4.1.4]. Suppose one tries to solve the energy minimization problem as a linear program, instead of an integer linear program, i.e., the decision variables could take continuous values. Suppose, we use the objective given in (2.9a) and define the constraints of our linear program through the marginal polytope. Then the solution of our linear program is one of the corners of the marginal polytope. By the definition of the marginal polytope, each corner is a valid labelling of the graphical model. Thus it looks like we have exactly found the labelling that corresponds to maximum a posteriori probability through a linear program. However, it will be computationally very hard to represent the marginal polytope. Thus we are still in square one.

Now, a powerful way to solve discrete optimization problem like the integer linear program in (2.9), is through convex relaxations. We will consider here, the linear programming (LP) relaxation of (2.9), which provably bounds the relaxations based on quadratic programs (QPs), semidefinite programs (SDPs) and second-order cone programs SOCPs [Kumar 2009]. Note, [Kumar 2009] still encourages research in SOCP relaxations and a useful SOC relaxation for higher-order MAP inference is an open research question.

The way to construct the linear programming relaxation can be thought of in terms of obtaining a simpler polytope to optimize over, thus leading to a computationally tractable approach. The linear programming relaxation takes the following form,

$$\text{minimize} \quad \sum_i \sum_{x_i} \theta_i(x_i)\phi_i(x_i) + \sum_c \sum_{\boldsymbol{x}_c} \theta_c(\boldsymbol{x}_c)\phi_c(\boldsymbol{x}_c) \tag{2.10a}$$

$$\text{subject to} \quad \sum_{x_{c\setminus i}} \phi_c(\boldsymbol{x}_c) = \phi_i(x_i), \quad \forall (i,c) : i \in c \tag{2.10b}$$

$$\begin{aligned} &\sum_{x_i} \phi_i(x_i) = 1, \quad \forall i \in \mathcal{V}; \quad \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c) = 1, \quad \forall c \in \mathcal{C} \\ &\phi_i(x_i) \geq 0,\ \forall i \in \mathcal{V}, x_i \in \boldsymbol{l}_i; \quad \phi_c(\boldsymbol{x}_c) \geq 0\ \ \forall c \in \mathcal{C}, \boldsymbol{x}_c \in \boldsymbol{l}_c. \end{aligned} \tag{2.10c}$$

This is referred to as the Schlesinger LP in the graphical models literature [Schlesinger 1976]. We first notice how similar the LP relaxation is to the ILP. All we have done is to relax the integer valued (also, called integrality) constraints in (2.9c). Relaxing the integrality constraints in ILP (2.9) could take the form, $0 \leq \phi_i(x_i) \leq 1$ and $0 \leq \phi_c(\boldsymbol{x}_c) \leq 1$ but the summation constraint in (2.10c) makes

non-negativity constraint sufficient. Since, we will be referring to this optimization problem (2.10) often, we will represent it in a succint form as follows,

$$\underset{\boldsymbol{\phi}}{\text{minimize}} \quad \langle \boldsymbol{\theta}, \boldsymbol{\phi} \rangle \tag{2.11a}$$

$$\boldsymbol{\phi} \in \mathcal{L}(\mathcal{G}) \tag{2.11b}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are vectors obtained by stacking the potentials and the overcomplete representation variables, respectively. For this linear program, the marginalization constraints (2.10b) and the normalization constraints (2.10c) define a polytope, which the graphical models literature calls as the *local polytope*. We will denote it as $\mathcal{L}(\mathcal{G})$. The local polytope has several interesting properties. The local polytope is an outer bound on the marginal polytope. It is defined by marginalization constraints (2.10b) between cliques and the nodes within each clique. This is only a subset of all the marginalization constraints that a valid marginal vector has to satisfy. Since, it enforces local marginalization constraints between a clique and its constituent nodes, it is called the local polytope. Note, we are able to define the marginal polytope using this same subset of marginalization constraints. This is because we also enforce the integrality constraint, which along with these marginalization constraints and normalization constraints are enough to define the corners of the marginal polytope. The local polytope will contain all the points within the marginal polytope and also, other fractional points that satisy constraints (2.10b) and (2.10c). The points contained within the local polytope are called *pseudomarginals*, while the marginal polytope contains the true marginals. In the rest of the thesis, we will be referring to the variables of the overcomplete representation, which are the decision variables of our optimization problem, as pseudomarginals.

The local polytope has fewer facets compared to the marginal polytope. In fact, we know it to be polynomial and upper bounded by $O(l.n+|\mathcal{C}|l^s)$. Thus, optimization of the LP relaxation given in (2.10) is tractable. When one is trying to solve an NP-hard problem, one can one only hope for a trade-off between computational complexity and approximation guarantee. Even though the local polytope has lesser number of facets compared to the marginal polytope, it has more corners. All the integer $\{0, 1\}$ valued corners of the marginal polytope are also corners of the local polytope. On top of these, there are several fractional valued corners in the local polytope. The total number of corners of the local polytope is not known in general [Wainwright 2008, section 4.1.4]. Thus, when the LP relaxation returns one of the corners as the solution, it could be a fractional valued corner. We still need to construct a valid labelling of the graphical model from this vector of fractional values. However, suppose the LP relaxation returns a integer valued corner, then we have the guarantee that we have recovered the exact MAP labelling of the graphical model. This is an important guarantee of the LP relaxation based approach [Wainwright 2005a]. Also, if the original graph is a tree or if it is a general graph with submodular clique potentials, then this LP relaxation is guaranteed to be tight. We will see in section (2.5), the derivation of the dual in terms of tractable subgraphs like trees.

It is difficult to visualize the marginal and local polytopes, especially with the overcomplete representation. Consider a simple graph, with only two nodes, each having a label set of size two. The polytope will lie in a space of dimension $2 + 2 + 2^2 = 8$. (2.3) represents these polytopes in an abstract way so that we keep in mind the fact that the local polytope contains both the integral corners of the marginal polytope and also, fractional corners.

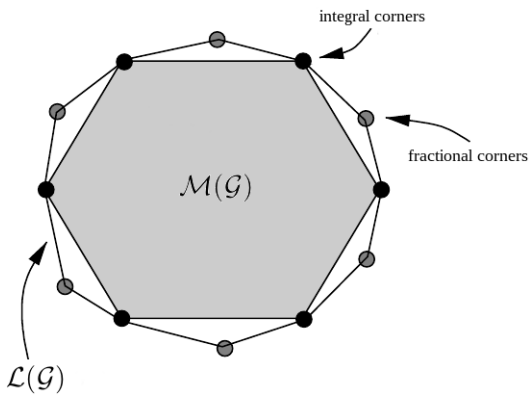

Figure 2.3: A cartoonish illustration of the relationship between the marginal polytope $\mathcal{M}(\mathcal{G})$ and the local polytope $\mathcal{L}(\mathcal{G})$. All the corners of $\mathcal{M}(\mathcal{G})$ are also corners of $\mathcal{L}(\mathcal{G})$. $\mathcal{L}(\mathcal{G})$ also has fractional corners. The fact that $\mathcal{L}(\mathcal{G})$ has lesser number of facets than $\mathcal{M}(\mathcal{G})$ is difficult to visualize. *Image courtesy [Wainwright 2008].*

In general, an LP relaxation minimizes a problem over a larger space of values for the decision variables compared to an ILP. A solution for the ILP will correspond to a feasible point within the domain of the LP relaxation. Suppose the LP relaxation has an optimal solution with objective value E_{lp}, this means that there is no feasible point within the domain of the LP relaxation with an objective value lesser than E_{lp}. Thus a solution for the ILP, will have an objective value $\geq E_{lp}$. Thus the solution returned by the LP relaxation (2.10) will be a *lower bound* to the energy minimization problem. In section (2.5), we will see a perspective where a lower bound is derived based on a tree based decomposition.

Given that the solution returned by the LP relaxation is fractional, we need ways to reconstruct integer valued variables that correspond to a valid labelling of the graphical models. It will be also desirable to study what is the quality of approximation that we achieve by first solving a LP relaxation, followed by reconstructing integer solutions. We will briefly discuss about these aspects in section (3.5). Immediately, we will see how the LP relaxation given in (2.10) could be solved efficiently.

2.5 Efficient dual based formulation

Over the past decades, there has been tremendous progress in the development of linear programming solvers like [CPLEX , MOSEK]. Will it be enough to apply one of these high performing solvers to our LP relaxation problem (2.10)? No, generic LP solvers are not good enough to solve this LP relaxation problem efficiently and scalably as the graph size increases. An empirical study of off-the-shelf LP solvers [Yanover 2006] observed their slower performance and also, their inability to go beyond a 50×50 image in a stereo disparity problem. We will continue to consider a graphical model $\mathcal{G}$ with n nodes, each taking labels from a common set of size l, with $|\mathcal{C}|$ cliques of size s each. The LP relaxation, will have $n.l + |\mathcal{C}|l^s$ variables. Generally, we will be working with graphs in which with more nodes, we will have more cliques. Thus, the problem size could increase tremendously with respect to all the model parameters: n, $|\mathcal{C}|$, l and s. Especially, the exponential growth with respect to clique size s is a major challenge to address while working with higher-order MRFs. Thus, there is a lot of benefit to explore LP solvers that exploit the graph structure of the MAP inference problem.

We have noticed that a graphical model leads to interaction between the variables according to local and overlapping cliques. This is evident in the objectives of the energy minimization problem (2.6) and the LP relaxation formulation (2.10). Also, one could notice this local interaction structure in the marginalization constraints (2.10b). Usually in optimization problems, we construct the *dual* problem, based on the constraints. Thus, by considering the dual of the LP relaxation, will it be possible to achieve a formulation that exploits the structure of the graphical model? Now, we will derive a linearly constrained dual, which is based on a subgraph based decomposition of the original graph.

Given a graph $\mathcal{G}$, it is possible to construct several hypertrees, which are subgraphs of $\mathcal{G}$. For convenience, we will call the hypertrees as trees and whenever we refer to subgraphs we are referring to trees, unless otherwise stated. We will consider a set $\mathcal{T}(\mathcal{G})$ of *overlapping* trees, wherein *each tree shares at least a node or a clique with one or more of the other trees* and together they *cover* the graph. For a node i, we will denote the trees containing it as $\mathcal{T}(i)$ and for a clique c, we will denote the trees containing it as $\mathcal{T}(c)$. Now, these trees are also graphical models and we could associate node potentials $(\theta_i^{\mathcal{T}}(x_i))$ and clique potentials $(\theta_c^{\mathcal{T}}(\boldsymbol{x}_c))$ with each $\mathcal{T} \in \mathcal{T}(\mathcal{G})$. We could define these potentials such that the following constraints are satisfied,

$$\sum_{\mathcal{T}\in\mathcal{T}(i)} \theta_i^{\mathcal{T}}(x_i) = \theta_i(x_i) \;\; \forall i \in \mathcal{V}, \qquad \sum_{\mathcal{T}\in\mathcal{T}(c)} \theta_c^{\mathcal{T}}(\boldsymbol{x}_c) = \theta_c(\boldsymbol{x}_c), \;\; \forall c \in \mathcal{C}. \tag{2.12}$$

A simple way to achieve this constraint is by setting the potentials as follows: $\theta_i^{\mathcal{T}}(x_i) = \frac{\theta_i(x_i)}{|\mathcal{T}(i)|}\ \forall \mathcal{T} \in \mathcal{T}(i)$ and $\theta_c^{\mathcal{T}}(\boldsymbol{x}_c) = \frac{\theta_c(\boldsymbol{x}_i)}{|\mathcal{T}(c)|}\ \forall \mathcal{T} \in \mathcal{T}(c)$. Just like we associate $\boldsymbol{x}$ with the labelling of all the nodes in the entire graph, we could associate $\boldsymbol{x}^{\mathcal{T}}$ with the labelling of the tree $\mathcal{T}$. Given a particular labeling $\boldsymbol{x}$ of the entire graph, $\boldsymbol{x}^{\mathcal{T}}$ could be constructed by picking the labels of the nodes which are in $\mathcal{T}$. We will

denote this operation as $\boldsymbol{x}^{\mathcal{T}} = \boldsymbol{x}_{|\mathcal{T}}$. Thus, the energy associated with the graphical model, i.e., $E(\boldsymbol{x};\boldsymbol{\theta})$ in (2.5), could be *decomposed* as follows,

$$E(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} E(\boldsymbol{x}^{\mathcal{T}};\boldsymbol{\theta}^{\mathcal{T}}) \tag{2.13}$$

by setting $\boldsymbol{x}^{\mathcal{T}} = \boldsymbol{x}_{|\mathcal{T}}$. We will repeat that the trees should be chosen such that they share at least one node or clique with at least one other tree. If this is not the case, that tree will not be connected with the rest of the trees. Since, our original graph is a connected graph, we are sure that subgraphs, which cover the original graph, are overlapping. Hence, our energy minimization problem could be equivalently formulated as follows,

$$\underset{\boldsymbol{x}^{\mathcal{T}},\boldsymbol{x}}{\text{minimize}} \sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} E(\boldsymbol{x}^{\mathcal{T}};\boldsymbol{\theta}^{\mathcal{T}}) \tag{2.14a}$$

$$\text{subject to} \quad \boldsymbol{x}^{\mathcal{T}} = \boldsymbol{x}_{|\mathcal{T}} \quad \forall\mathcal{T}\in\mathcal{T}(\mathcal{G}) \tag{2.14b}$$

where the variable $\boldsymbol{x}$ is defined over all the nodes and it ensures *consensus* between the labellings of the trees. Note, that we are not explicitly minimizing energy on the entire graph, i.e., we are not minimizing $E(\boldsymbol{x},\boldsymbol{\theta})$ explicitly. We are performing energy minimization on the trees and use $\boldsymbol{x}$ to make the labels agree across trees and in the process, minimize $E(\boldsymbol{x},\boldsymbol{\theta})$. Thus, as long the labels of the nodes agree between all the trees containing that node, then we could minimize the energy of the entire graph by minimizing energy over individual trees. However, ensuring agreement of the node labels across trees is a difficult task. This is enforced through the consensus constraint (2.14b). Probably, we could overcome the restriction imposed by this constraint through the dual formulation. In order to do that, let us first write the LP relaxation of (2.14) using pseudomarginals as follows,

$$\underset{\boldsymbol{\phi}_i^{\mathcal{T}},\boldsymbol{\phi}_c^{\mathcal{T}},\boldsymbol{\psi}_i,\boldsymbol{\psi}_c}{\text{minimize}} \sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} \Big(\sum_{i\in\mathcal{T}}\sum_{x_i} \theta_i^{\mathcal{T}}(x_i)\phi_i^{\mathcal{T}}(x_i) + \sum_{c\in\mathcal{T}}\sum_{\boldsymbol{x}_c} \theta_c^{\mathcal{T}}(\boldsymbol{x}_c)\phi_c^{\mathcal{T}}(\boldsymbol{x}_c) \Big) \tag{2.15a}$$

$$\left.\begin{aligned} \text{subject to} \quad & \sum_{x_{c\setminus i}} \phi_c^{\mathcal{T}}(\boldsymbol{x}_c) = \phi_i^{\mathcal{T}}(x_i), \quad \forall(i,c) : c\in\mathcal{T}, i\in c \\ & \sum_{x_i}\phi_i^{\mathcal{T}}(x_i) = 1, \quad \forall i\in\mathcal{T}; \quad \sum_{\boldsymbol{x}_c}\phi_c^{\mathcal{T}}(\boldsymbol{x}_c) = 1, \quad \forall c\in\mathcal{T} \\ & \phi_i^{\mathcal{T}}(x_i) \geq 0 \;\; \forall i\in\mathcal{T}; \quad \phi_c(\boldsymbol{x}_c) \geq 0 \;\; \forall c\in\mathcal{T} \end{aligned}\right\} \forall\mathcal{T}\in\mathcal{T}(\mathcal{G}) \tag{2.15b}$$

$$\forall\mathcal{T}\in\mathcal{T}(\mathcal{G}), \;\; \phi_i^{\mathcal{T}}(x_i) = \psi_i(x_i) \;\; \forall\mathcal{T} : i\in\mathcal{T}; \;\; \phi_c^{\mathcal{T}}(\boldsymbol{x}_c) = \psi_c(\boldsymbol{x}_c) \;\; \forall\mathcal{T} : c\in\mathcal{T}. \tag{2.15c}$$

The constraints (2.15b) are local to each tree and do not overlap. Suppose the original graph is itself a tree, then $\mathcal{T}(\mathcal{G})$ is composed of only one tree and (2.15) could be optimized exactly. Thus, the LP relaxation is tight for tree structured graphical models. In a general setting, a graphical model is decomposed into subgraphs like trees and the variables $\psi_i(x_i)$ and $\psi_c(\boldsymbol{x}_c)$ are global graph level variables that enforce consensus between the trees. The coupling constraint (2.15c) is the obstacle to efficient and scalable algorithms. One way to overcome problematic constraints is to

formulate the dual problem. The Lagrangian of (2.15), by considering the coupling constraint, is as follows,

$$\sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})}\Bigg(\sum_{i\in\mathcal{T}}\sum_{x_i}\theta_i^{\mathcal{T}}(x_i)\phi_i^{\mathcal{T}}(x_i)+\sum_{c\in\mathcal{T}}\sum_{\boldsymbol{x}_c}\theta_c^{\mathcal{T}}(\boldsymbol{x}_c)\phi_c^{\mathcal{T}}(\boldsymbol{x}_c)$$
$$+\sum_{i\in\mathcal{V}}\sum_{\mathcal{T}:i\in\mathcal{T}}\sum_{x_i}\delta_{\mathcal{T}i}(x_i)\Big(\phi_i^{\mathcal{T}}(x_i)-\psi_i(x_i)\Big)+\sum_{c\in\mathcal{C}}\sum_{\mathcal{T}:c\in\mathcal{T}}\sum_{\boldsymbol{x}_c}\delta_{\mathcal{T}c}(\boldsymbol{x}_c)\Big(\phi_c^{\mathcal{T}}(\boldsymbol{x}_c)-\psi_c(\boldsymbol{x}_c)\Big)\Bigg) \tag{2.16}$$

where $\delta_{\mathcal{T}i}(x_i)$ are the dual variables with respect to the nodes and for each tree $\mathcal{T}$, each node $i\in\mathcal{T}$ and for each label x_i, there is one such dual variable. $\delta_{\mathcal{T}c}(\boldsymbol{x}_c)$ are the dual variables with respect to the cliques and for each tree $\mathcal{T}$, each clique $c\in\mathcal{T}$ and for each possible clique labelling $\boldsymbol{x}_c$, there is one such dual variable. We will denote by $\boldsymbol{\delta}$ the vector of all these dual variables. Now, the dual optimization problem takes the following form,

$$\underset{\boldsymbol{\delta}}{\text{maximize}}\ \underset{\boldsymbol{\phi}_i^{\mathcal{T}},\boldsymbol{\phi}_c^{\mathcal{T}},\boldsymbol{\psi}_i,\boldsymbol{\psi}_c}{\text{minimize}}\sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})}\Bigg(\sum_{i\in\mathcal{T}}\sum_{x_i}\theta_i^{\mathcal{T}}(x_i)\phi_i^{\mathcal{T}}(x_i)+\sum_{c\in\mathcal{T}}\sum_{\boldsymbol{x}_c}\theta_c^{\mathcal{T}}(\boldsymbol{x}_c)\phi_c^{\mathcal{T}}(\boldsymbol{x}_c)$$
$$+\sum_{i\in\mathcal{V}}\sum_{\mathcal{T}:i\in\mathcal{T}}\sum_{x_i}\delta_{\mathcal{T}i}(x_i)\Big(\phi_i^{\mathcal{T}}(x_i)-\psi_i(x_i)\Big)+\sum_{c\in\mathcal{C}}\sum_{\mathcal{T}:c\in\mathcal{T}}\sum_{\boldsymbol{x}_c}\delta_{\mathcal{T}c}(\boldsymbol{x}_c)\Big(\phi_c^{\mathcal{T}}(\boldsymbol{x}_c)-\psi_c(\boldsymbol{x}_c)\Big)\Bigg) \tag{2.17a}$$

$$\left.\begin{aligned}\text{subject to}\ \sum_{x_{c\setminus i}}\phi_c^{\mathcal{T}}(\boldsymbol{x}_c)=\phi_i^{\mathcal{T}}(x_i),\quad\forall(i,c):c\in\mathcal{T},i\in c\\ \sum_{x_i}\phi_i^{\mathcal{T}}(x_i)=1,\quad\forall i\in\mathcal{T};\quad\sum_{\boldsymbol{x}_c}\phi_c^{\mathcal{T}}(\boldsymbol{x}_c)=1,\quad\forall c\in\mathcal{T}\\ \phi_i^{\mathcal{T}}(x_i)\geq 0\ \ \forall i\in\mathcal{T};\quad\phi_c(\boldsymbol{x}_c)\geq 0\ \ \forall c\in\mathcal{T}\end{aligned}\right\}\forall\mathcal{T}\in\mathcal{T}(\mathcal{G}). \tag{2.17b}$$

In the current form, the objective is linear with respect to the consensus variables $\psi_i(x_i)$ and $\psi_c(\boldsymbol{x}_c)$ and the objective will be unbounded below when minimized with respect to them. It is possible to eliminate these consensus variables if the dual variables satisfy the following constraint,

$$\begin{aligned}\sum_{\mathcal{T}:i\in\mathcal{T}}\delta_{\mathcal{T}i}(x_i)=0,\quad\forall i\in\mathcal{V}\\ \sum_{\mathcal{T}:c\in\mathcal{T}}\delta_{\mathcal{T}c}(\boldsymbol{x}_c)=0,\quad\forall c\in\mathcal{C}.\end{aligned} \tag{2.18}$$

Now, that the coupling consensus variables could be taken out of the objective, the minimization with respect to $\boldsymbol{\phi}_i^{\mathcal{T}}$ and $\boldsymbol{\phi}_c^{\mathcal{T}}$ could be taken inside the summation over the trees. Thus, we obtain the following optimization problem,

$$\underset{\boldsymbol{\delta}}{\max.}\sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})}\min_{\boldsymbol{\phi}_{\mathcal{T}i},\boldsymbol{\phi}_{\mathcal{T}c}.}\Bigg(\sum_{i\in\mathcal{T}}\sum_{x_i}(\theta_{\mathcal{T}i}(x_i)+\delta_{\mathcal{T}i}(x_i))\phi_{\mathcal{T}i}(x_i)+\sum_{c\in\mathcal{T}}\sum_{\boldsymbol{x}_c}(\theta_{\mathcal{T}c}(\boldsymbol{x}_c)+\delta_{\mathcal{T}c}(\boldsymbol{x}_c))\phi_{\mathcal{T}c}(\boldsymbol{x}_c)\Bigg) \tag{2.19a}$$

$$\left.\begin{aligned}\text{subject to } \sum_{x_{c\setminus i}} \phi_{\mathcal{T}c}(\boldsymbol{x}_c) = \phi_{\mathcal{T}i}(x_i), \quad \forall (i,c) : c \in \mathcal{T}, i \in c \\ \sum_{x_i} \phi_{\mathcal{T}i}(x_i) = 1, \quad \forall i \in \mathcal{T}; \quad \sum_{\boldsymbol{x}_c} \phi_{\mathcal{T}c}(\boldsymbol{x}_c) = 1, \quad \forall c \in \mathcal{T} \\ \phi_{\mathcal{T}i}(x_i) \geq 0 \;\; \forall i \in \mathcal{T}; \quad \phi_c(\boldsymbol{x}_c) \geq 0 \;\; \forall c \in \mathcal{T}\end{aligned}\right\} \forall \mathcal{T} \in \mathcal{T}(\mathcal{G}) \tag{2.19b}$$

$$\sum_{\mathcal{T}:i\in\mathcal{T}} \delta_{\mathcal{T}i}(x_i) = 0, \quad \forall i \in \mathcal{V}, \forall x_i; \quad \sum_{\mathcal{T}:c\in\mathcal{T}} \delta_{\mathcal{T}c}(\boldsymbol{x}_c) = 0, \quad \forall c \in \mathcal{C}, \forall \boldsymbol{x}_c. \tag{2.19c}$$

This dual formulation could be minimized with respect to $\boldsymbol{\phi}_{\mathcal{T}i}$ and $\boldsymbol{\phi}_{\mathcal{T}c}$ in a non-overlapping distributed manner. The dual problem could be compactly represented as follows,

$$\max_{\boldsymbol{\delta}}. \sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} \min_{\boldsymbol{\phi}^{\mathcal{T}}}. \left\langle (\boldsymbol{\theta}^{\mathcal{T}} + \boldsymbol{\delta}^{\mathcal{T}}), \boldsymbol{\phi}^{\mathcal{T}} \right\rangle \tag{2.20a}$$

$$\text{subject to } \boldsymbol{\phi}^{\mathcal{T}} \in \mathcal{L}(\mathcal{T}) \;\; \forall \mathcal{T} \in \mathcal{T}(\mathcal{G}) \tag{2.20b}$$

$$\sum_{\mathcal{T}:i\in\mathcal{T}} \delta_{\mathcal{T}i}(x_i) = 0, \quad \forall i \in \mathcal{V}, \forall x_i; \quad \sum_{\mathcal{T}:c\in\mathcal{T}} \delta_{\mathcal{T}c}(\boldsymbol{x}_c) = 0, \quad \forall c \in \mathcal{C}, \forall \boldsymbol{x}_c. \tag{2.20c}$$

We will see in chapter (3), how to maximize with respect to $\boldsymbol{\delta}$ and thus provably reach the global optimum of the problem. We will next see a way to derive a dual formulation, where the graph is decomposed with respect to individual cliques. This dual is important because it is an unconstrained formulation and also, has lesser number of dual variables, while retaining similar convergence rate. We will discuss in section (3.7) how to reduce the number of dual variables in the current formulation also, while ensuring good convergence rate. Note, the current formulation could also be made unconstrained because the dual variables are subject to simple linear constraints (2.18). This will lead to a subgraph that does not have dual variables of its own, instead its variables will be the negative sum of the dual variables from the overlapping subgraphs. From an algorithmic point of view, this is equivalent to imposing the linear constraint (2.18). Also, the resulting equations do not have the uniformity and symmetry that we will notice in the formulation given in the next section.

2.6 Unconstrained dual

We will now show a way to derive an unconstrained dual to the LP relaxation, which proceeds by introducing dual variables corresponding to the marginalization constraints. The following derivation is an extension to higher-order graphs of the one given for pairwise graphs by [Sontag 2010]. Note the following derivation could be obtained based on a clique based decomposition [Sontag 2011], also. As always, only for the sake of easier exposition, we will assume that all nodes take labels from a common label set of size l and that all cliques contain s nodes. Then for each clique the marginalization constraint,

$$\sum_{x_{c\setminus i}} \phi_c(\boldsymbol{x}_c) = \phi_i(x_i), \quad \forall i \in c \tag{2.21}$$

is made of s similar looking sets of constraints for each of the nodes within the clique. Each of these sets will contain l constraints, corresponding to the l possible labels that the node can take. Thus, totally we have $|\mathcal{C}|.s.l$ marginalization constraints. We will leave the normalization constraints (2.10c) untouched and only introduce dual variables for each of the marginalization constraints. We will denote them as $\delta_{ci}(x_i)$. It will be good to spare some time to understand this notation. Let us recall that some node i takes a label x_i, which corresponds to one of l labels. $\delta_{ci}(x_i)$ corresponds to the marginalization constraint for clique c with respect to node i and label x_i. Thus, the total number of dual variables is $|\mathcal{C}|.s.l$ and we will denote the vector of dual variables as $\boldsymbol{\delta}$.

Thus, the Lagrangian takes the form,

$$\sum_i \sum_{x_i} \theta_i(x_i)\phi_i(x_i) + \sum_c \sum_{\boldsymbol{x}_c} \theta_c(\boldsymbol{x}_c)\phi_c(\boldsymbol{x}_c) + \sum_{c\in\mathcal{C}} \sum_{i\in c} \sum_{x_i} \delta_{ci}(x_i)\Big(\phi_i(x_i) - \sum_{x_{c\setminus i}} \phi_c(\boldsymbol{x}_c)\Big). \tag{2.22}$$

Now, by expanding the summations and re-arranging terms, we can see that,

$$\begin{aligned} \sum_{c\in\mathcal{C}} \sum_{i\in c} \sum_{x_i} \delta_{ci}(x_i)\phi_i(x_i) &= \sum_{i\in\mathcal{V}} \sum_{x_i} \phi_i(x_i) \sum_{c:i\in c} \delta_{ci}(x_i) \\ \sum_{c\in\mathcal{C}} \sum_{i\in c} \sum_{x_i} \delta_{ci}(x_i) \sum_{x_{c\setminus i}} \phi_c(\boldsymbol{x}_c) &= \sum_{c\in\mathcal{C}} \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c) \sum_{i:i\in c} \delta_{ci}(x_i). \end{aligned} \tag{2.23}$$

Thus, the dual optimization problem takes the following form,

$$\max_{\boldsymbol{\delta}}. \min_{\boldsymbol{\phi}}. \sum_i \sum_{x_i} \phi_i(x_i)\Big(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\Big) + \sum_{c\in\mathcal{C}} \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c)\Big(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\Big) \tag{2.24a}$$

$$\begin{aligned} \text{subject to } \sum_{x_i} \phi_i(x_i) = 1, \quad \forall i \in \mathcal{V}; \quad \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c) = 1, \quad \forall c \in \mathcal{C} \\ \phi_i(x_i) \geq 0, \ \ \forall i \in \mathcal{V}; \quad \phi_c(\boldsymbol{x}_c) \geq 0 \ \ \forall c \in \mathcal{C}. \end{aligned} \tag{2.24b}$$

The minimization with respect to $\boldsymbol{\phi}$ subject to the constraints given in (2.24b), has an analytical solution. For each node i, $\phi_i(x_i)$ are subject to a simplex constraint. Thus, we have to set $\phi_i(x_i) = 1$ for the label x_i that minimizes the value of $(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i))$ and set the rest of the $\phi_i(x_i)$'s to zero. Similarly, we have to set $\phi_c(\boldsymbol{x}_c) = 1$ for the clique labelling $\boldsymbol{x}_c$ that minimizes the value of $\Big(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\Big)$ and set the rest of the $\phi_c(\boldsymbol{x}_c)$'s to zero.

Thus, we obtain the following dual optimization problem with respect to the dual variables $\boldsymbol{\delta}$,

$$\max_{\boldsymbol{\delta}}. \sum_{i\in\mathcal{V}} \min_{x_i}. \Big(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\Big) + \sum_{c\in\mathcal{C}} \min_{\boldsymbol{x}_c}. \Big(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\Big) \tag{2.25}$$

which is an unconstrained optimization problem. It is also possible to derive this dual based on a clique based decomposition, just like how we derived the dual based on tree based decomposition. We can interpret the terms in the dual as

a reparameterization of the node and clique potentials. In other words, for a given value of the dual variables, we are seeking optimal labels x_i's and $\boldsymbol{x}_c$'s for a graphical model with the following potentials: $\hat{\theta}_i(x_i) = \theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)$ and $\hat{\theta}_c(\boldsymbol{x}_c) = \theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i)$. Thus, this formulation goes by the name *reparameterization dual*. In the next chapter, we will see some existing continuous optimization algorithms for LP relaxation based MAP inference. We will especially see some algorithms which are applicable to higher-order MRFs.

Chapter 3

Optimization Algorithms for MAP Inference

3.1 Supergradient based optimization

The objective of either (2.20) or (2.25) is concave and non-smooth with respect to the dual variables $\boldsymbol{\delta}$. This is because the objective is obtained by pointwise minimization over a set of linear functions of the dual variables [Boyd 2004, §3.2.3]. Here, we will present an algorithm for the tree based decomposition. Since, a clique is a special case of a tree, adapting the algorithm for the clique based decomposition is easy. *Projected supergradient* methods offer a provably convergent method to reach the global optimum of the dual (2.20) [Komodakis 2011]. In fact, by lemma 1 of [Komodakis 2011], if we minimize the following problem with respect to a tree,

$$\min_{\boldsymbol{\phi}^{\mathcal{T}}} . \left\langle (\boldsymbol{\theta}^{\mathcal{T}} + \boldsymbol{\delta}^{\mathcal{T}}), \boldsymbol{\phi}^{\mathcal{T}} \right\rangle \tag{3.1a}$$

$$\text{subject to } \boldsymbol{\phi}^{\mathcal{T}} \in \mathcal{L}(\mathcal{T}) \tag{3.1b}$$

then the *minimizing label assignment* is a valid supergradient for the dual variables $\delta_{\mathcal{T}i}(x_i)$ and $\delta_{\mathcal{T}c}(\boldsymbol{x}_c)$. This minimization is exactly equivalent to performing energy minimization or MAP inference on that tree, which we can obtain exactly using the method shown in (2.3). This is an important aspect of LP relaxation based algorithms that exploit the graph structure. We solve the MAP inference problem in a general graph, by decomposing the problem over trees and we exploit algorithms that are suited for MAP inference over trees. Also, note that the MAP inference for all the trees could be computed independently. Once all the supergradients are obtained, we will project them into the valid set based on the linear constraints (2.18). Thus, a projected supergradient algorithm is obtained and it offers an approach that theoretically converges to the global optimum of the dual.

The algorithm for the clique based decomposition is an unconstrained supergradient algorithm, where the supergradients are obtained by minimizing $\sum_{i\in\mathcal{V}} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\right) + \sum_{c\in\mathcal{C}} \left(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\right)$ over each of the cliques. These operations are again MAP inferences within those individual cliques.

The supergradient based approach (constrained or unconstrained) has a rate of convergence of $O(\frac{1}{\varepsilon^2})$, i.e., to reach an ε-accurate solution it takes $O(\frac{1}{\varepsilon^2})$ iterations. Suppose the node i is shared by $n_i^{\mathcal{T}}$ trees and the clique c is shared by $n_c^{\mathcal{T}}$ trees, then the memory requirement for the tree based decomposition is $\sum_{i\in\mathcal{V}} n_i^{\mathcal{T}}.l + \sum_{c\in\mathcal{C}} n_c^{\mathcal{T}}.l^s$. This

is of the same order as the number of primal variables in the LP relaxation formulation given in (2.10). We could bring down the number of dual variables by enforcing consensus only between the pseudomarginals of the nodes. [Komodakis 2011, section 6.3] shows that as long as the subgraphs could be solved exactly, then it is enough to enforce consensus between the node variables alone. However, the convergence rate is affected by doing this. The clique based decomposition in (2.25) offers a formulation that has lesser number of variables ($|\mathcal{C}|.s.l$), which can also be optimized using supergradients, without affecting the convergence rate adversely. However, we will see in section (3.7) that solving clique based decomposition for large graphs leads to poor convergence for several algorithms, thus the tree based decomposition given in (2.20) is important in its own way. Next, we will see algorithms that have a convergence rate that is better than $O(\frac{1}{\varepsilon^2})$.

We note that it is the minimizing label assignment that leads to a valid supergradient. The max-product/min-sum algorithm that we discussed in (2.3) computes the min-marginals to obtain the minimizing label assignment. Depending on the clique potential, finding the minimizing labelling could be cheaper than computing min-marginals [Sontag 2011, Meshi 2015, Swoboda 2018].

3.2 Smooth accelerated gradient method

The poor rate of $O(\frac{1}{\varepsilon^2})$ that we saw in the previous section is because of the non-smooth nature of the problem. It is possible to perturb the non-smooth problem and obtain a smooth problem [Nesterov 2005]. It is achieved by exploiting the following theorem on the duality between strong convexity and strong smoothness. Please, refer to appendix (9.1) for the definition of terms mentioned in the theorem.

Theorem 1 (Strong/Smooth Duality)**.** *Assume that f is a closed and convex function. Then f is β-strongly convex w.r.t. a norm $\|\cdot\|$ if and only if its Fenchel conjugate f^* is $\frac{1}{\beta}$-strongly smooth w.r.t. the dual norm $\|\cdot\|_*$.*

Proof. Please, refer [Nesterov 2005]. □

This theorem gives a ready tool by which one could obtain a smooth dual by perturbing the primal by a strongly convex function. In fact, this is a useful result for MAP inference algorithms and we will see it again in smooth coordinate maximization methods in section (3.3). We will derive the smooth version of the unconstrained clique-based dual given in (2.25). It was [Jojic 2010] that showed an accelerated gradient descent scheme for the following *entropy based smoothing* approach. We would like to point out that [Meshi 2015] showed another way to achieve a smooth problem, using a quadratic perturbation term.

To start with we will perturb the primal objective of the LP relaxation (2.11) using entropy terms,

$$\langle \boldsymbol{\theta}, \boldsymbol{\phi} \rangle - \frac{1}{\tau} \sum_{i \in \mathcal{V}} H(\boldsymbol{\phi}_i) - \frac{1}{\tau} \sum_{c \in \mathcal{C}} H(\boldsymbol{\phi}_c) \tag{3.2}$$

where, $H(\boldsymbol{\phi}_i) = -\sum_{x_i} \phi_i(x_i)\log\phi_i(x_i)$ and $H(\boldsymbol{\phi}_c) = -\sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c)\log\phi_c(\boldsymbol{x}_c)$, are *strongly concave* entropy terms. Thus, we get a perturbed optimization problem as follows,

$$\begin{aligned} \underset{\boldsymbol{\phi}}{\text{minimize}} \;\; & \langle \boldsymbol{\theta}, \boldsymbol{\phi} \rangle - \frac{1}{\tau}\sum_{i\in\mathcal{V}} H(\boldsymbol{\phi}_i) - \frac{1}{\tau}\sum_{c\in\mathcal{C}} H(\boldsymbol{\phi}_c) \\ \text{subject to} \;\; & \boldsymbol{\phi} \in \mathcal{L}(\mathcal{G}). \end{aligned} \tag{3.3}$$

It can be precisely shown to what extent the optimum of this perturbed problem is away from the optimum of the original problem [Meshi 2014, lemma 1.1]. Suppose $\boldsymbol{\phi}^*$ is the optimum of the LP relaxation (2.10) and $\hat{\boldsymbol{\phi}}$ is the optimum obtained by optimizing the perturbed objective (3.2), then we could bound the primal energy as follows,

$$\langle \hat{\boldsymbol{\phi}}, \boldsymbol{\theta} \rangle \geq \langle \boldsymbol{\phi}^*, \boldsymbol{\theta} \rangle \geq \langle \hat{\boldsymbol{\phi}}, \boldsymbol{\theta} \rangle - \frac{1}{\tau} H_{max} \tag{3.4}$$

where $H_{max} = \sum_i \log l_i + \sum_c \log|\boldsymbol{x}_c|$. Thus, we obtain an $O(\frac{1}{\tau})$-optimal solution, where the bound will be affected by the number of nodes in the graph, the number of labels per node and the size of the cliques. We will see the effect of these parameters in practice and it is good to keep these bounds in mind.

Now, we need to derive the smooth dual corresponding to the perturbed primal objective (3.2). This is achieved by considering the Fenchel conjugate of the entropy function subject to simplex constraints, which is the log-sum-exp function [Boyd 2004, §3.3]. See [Luenberger 1997, Komodakis 2015] for an accessible explanation about deriving the dual problem based on Fenchel duality. Also, refer section (3.6) for more details. Thus, the smooth dual is of the following form,

$$\max_{\delta}. \sum_c \underset{\boldsymbol{x}_c}{\text{smin}}(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i); \tau) + \sum_i \underset{x_i}{\text{smin}}(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i); \tau). \tag{3.5}$$

We will denote the objective of the smooth dual as $h(\boldsymbol{\delta})$. Also, *smin* is the *negative soft-max* function. Given a set of numbers $\boldsymbol{s} = \{s_1, s_2, ..., s_n\}$, the negative soft-max over $\boldsymbol{s}$ is defined as follows,

$$\underset{s\in\boldsymbol{s}}{\text{smin}}(s) = -\frac{1}{\tau}\log\sum_{s\in\boldsymbol{s}} \exp(\tau s). \tag{3.6}$$

Thus, as $\tau \to \infty$, $\underset{s\in\boldsymbol{s}}{\text{smin}}(s) \;\; \to$ the minimum value in set $\boldsymbol{s}$. Also, the smin function *lower bounds* the minimum of the set, for finite values of τ.

We can readily notice how similar the smooth dual (3.5) is with respect to the non-smooth dual (2.25). This is facilitated by entropy based smoothing and it is very convenient from a graphical models point of view. Just like the non-smooth case, we could develop optimization algorithms that exploit the graph structure, in this case also. While using gradient based methods, one needs to compute the gradient, which takes the following form,

$$\frac{\partial h(\boldsymbol{\delta})}{\partial \delta_{ci}(x_i)} = \mu_{ci}(x_i) - \mu_i(x_i) \tag{3.7}$$

where $\mu_{ci}(x_i)$ and $\mu_i(x_i)$ are computed as follows,

$$\begin{aligned}\mu_{ci}(x_i) &= \frac{\sum\limits_{\boldsymbol{x_c}:\boldsymbol{x_c}(i)=x_i} \exp\big[\tau.(\theta_c(\boldsymbol{x_c}) - \sum\limits_{n:n\in c} \delta_{cn}(x_n))\big]}{\sum\limits_{\boldsymbol{x_c}} \exp\big[\tau.(\theta_c(\boldsymbol{x_c}) - \sum\limits_{n:n\in c} \delta_{cn}(x_n))\big]} \\ \mu_i(x_i) &= \frac{\exp\big[\tau.(\theta_i(x_i) + \sum\limits_{k:i\in k} \delta_{ki}(x_i))\big]}{\sum\limits_{x_l} \exp\big[\tau.(\theta_i(x_l) + \sum\limits_{k:i\in k} \delta_{ki}(x_l))\big]}.\end{aligned} \tag{3.8}$$

We will see in section (3.5) that these quantities are also used while computing optimal primal variables from optimal dual variables. Another interesting point is that the supergradient of the non-smooth dual (2.25) also takes a similar form [Schwing 2012]. In section (3.1), we noted that in the non-smooth case, computing supergradients for individual subgraphs (cliques in the case of (2.25)) is exactly equivalent to MAP inference in those subgraphs. Similarly, for the smooth dual (3.5), gradient computation is enabled by *marginal* inference in subgraphs (cliques in the case of (3.5)). [Jojic 2010] is one of the first papers to recognize this. As we will see in (3.8), the gradient computation could be made further efficient by exploiting the nature of the clique.

Algorithm 1 FISTA with backtracking line search

```
Input: L_0 > 0, β > 1, y = δ_0 ∈ R^N, t_0 = 1
while ‖∇h(δ_k)‖_∞ > ζ do
    L_k = L_{k-1}
    δ_k = prox(y − (1/L_k) ∇h(y))
    while h(δ_k) > h(y) + ∇h(y)^T (δ_k − y) + (L_k/2) ‖δ_k − y‖_2^2 do
        L_k = β L_k
        δ_k = prox(y − (1/L_k) ∇h(y))
    end while
    t_k = (1 + sqrt(1 + 4 t_{k-1}^2)) / 2
    y = δ_k + ((t_{k-1} − 1)/t_k) (δ_k − δ_{k-1})
end while
```

Thus, given a smooth optimization problem, one can readily apply an accelerated gradient ascent scheme (algorithm 1) [Beck 2009] and achieve a convergence rate of $O(\frac{1}{\varepsilon})$. In the case of MAP inference, with the smooth version of the constrainted formulation (2.20), the *proximal operator* (prox) in (1), is the projection onto the linear constraints (2.18). For the smooth unconstrained formulation (3.5), there is no proximal operation involved.

We would like to point out that [Schwing 2012] presents an alternative way to improve convergence of the non-smooth problem. That algorithm achieves better convergence rate by choosing the steepest descent direction among the ε-supergradients at a point. Thus, they don't work with a perturbed version of

the original problem. Another interesting work is by [Kappes 2012], who improve convergence by leveraging bundle methods from the field of discrete optimization.

3.3 Coordinate Maximization Methods

Coordinate maximization methods form a very important class of algorithms for MAP inference. They readily adapt to the graph structure of the problem and have led to very successful MAP inference algorithms. e.g., TRW-S [Kolmogorov 2006] for pairwise graphs. Coordinate maximization methods for MAP inference are generally simple to implement and give impressive results. For discussing about coordinate maximization methods, we will work with the dual formulation given in section 2.6. In general, coordinate optimization methods divide the decision variables into non-overlapping blocks. Then at each step a block of coordinates is chosen to be otimized, while holding the rest of the blocks fixed. We would like to point out that the dual of the LP relaxation given in 2.6 has the form of function h_2 in [Nutini 2015]. That could be one reason for the success of coordinate optimization methods in MAP inference. An interesting feature about block coordinate maximization methods for MAP inference is that the optimal value of the block that is being optimized is obtained in *closed form*. Hence, a full optimization is performed in each step, which is different from general coordinate ascent methods which take a gradient step with respect to the current block. Hence, we refer to these algorithms as coordinate maximization [Meshi 2012], rather than coordinate ascent algorithms for MAP inference.

To understand better how these algorithms work, consider the non-smooth dual (2.25). We will denote this objective function as $h(\boldsymbol{\delta})$. It is possible to divide the set of decision variables $\boldsymbol{\delta}$ into N *non-overlapping* sets of variables $\{S_1, S_2, ..., S_N\}$. Then, $\boldsymbol{\delta}_{S_i} = \boldsymbol{\delta}_{|S_i}$ denotes the decision variables restricted to the set S_i and $\overline{\boldsymbol{\delta}}_{S_i}$ is the set of all other variables. In each iteration, only a block of variables is updated, while the variables belonging to the other blocks are unchanged. For example, the block S_i will be updated as follows,

$$\boldsymbol{\delta}_{S_i}^{t+1} = \arg\max_{\boldsymbol{\delta}_{S_i}} h(\boldsymbol{\delta}_{S_i} \cup \overline{\boldsymbol{\delta}}_{S_i}^{t}). \tag{3.9}$$

Here, we have clearly indicated that a full optimization is performed with respect to the block. Also, the full optimization is obtained in a closed form expression. This presents interesting opportunities to perform convergence rate analysis of coordinate maximization methods with closed form update in each step [Meshi 2012,Meshi 2017].

Another coordinate optimization algorithm that we would like to highlight is the algorithm called MPLP [Globerson 2008], which is one of the first algorithms for inference in higher-order MRFs. That algorithm is based on a dual that is different from what we saw in sections (2.5) and (2.6). Instead of getting into the details of MPLP, we will describe here an algorithm called *convex max-product* [Weiss 2007,Hazan 2010] that optimizes the non-smooth dual (2.25). The following

update equations are repeated until convergence, for every node $i \in \mathcal{V}$,

$$\begin{aligned}\phi_c(x_i) &= \max_{\boldsymbol{x}_{c\setminus i}}. \left\{\theta_c(\boldsymbol{x}_c) + \sum_{j\in c\setminus i} \delta_{cj}(x_j)\right\} \;\; \forall x_i,\; \forall c : i \in c \\ \delta_{ci}(x_i) &= \frac{1}{1+N_i}\left(\theta_i(x_i) + \sum_{c:i\in c} \phi_c(x_i)\right) - \phi_c(x_i) \;\; \forall x_i,\; \forall c : i \in c\end{aligned} \tag{3.10}$$

where N_i is the number of cliques that contain the node i and $\phi_c(x_i) = \sum_{\boldsymbol{x}_{c\setminus i}} \phi_c(\boldsymbol{x}_c)$, as shown in section (2.4). One weakness about using coordinate maximization methods for MAP inference, is that they may get stuck in suboptimal points if the objective is not smooth [Bertsekas 1999, §6.3.4]. Even though, for some special structure, coordinate ascent converges to the global optimum for functions of the following form,

$$f(x) = g(x) + \sum_{i=1}^{n} h_i(x_i) \tag{3.11}$$

where g is convex and smooth and each h_i is convex and possibly non-smooth. The special structure is that the non-smooth part is *separable*. The dual objective in MAP inference is only *partially separable*. We will see in section (5.1.1), that this partial separability could be exploited for computing the Hessian efficiently. The intuition for the possibility of coordinate ascent getting stuck in a non-smooth problem could be seen in figure (3.1). At point P, we are not able to increase the objective by travelling along either axis.

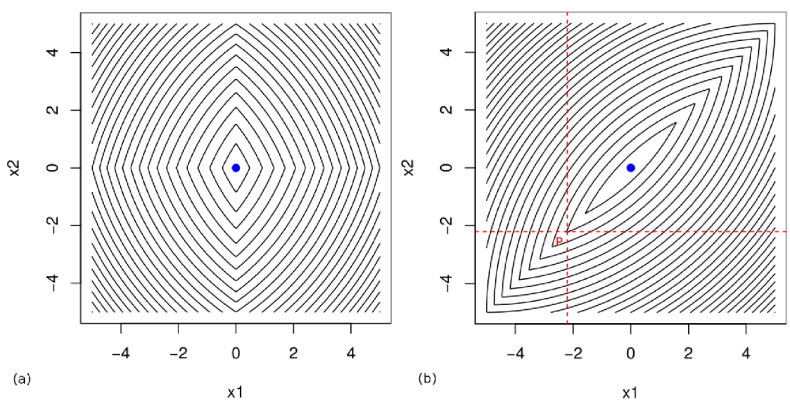

Figure 3.1: Superlevel sets of two dimensional non-smooth problems: (a) Separable non-smooth part, guaranteed convergence to global optimum (b) General non-smooth function, with a shape that is not favourable to coordinate ascent. *Image courtesy G. Gordon & R. Tibshirani, Optimization 10-725 slides, CMU.*

Thus, coordinate maximization for the non-smooth dual problem given in (2.19) could lead to sub-optimal solutions. This could be overcome by smoothing the dual, like we did in section (3.2). Though, in the case of (super)gradient based methods, it helped with improving convergence rate, for coordinate maximization it helps

with better convergence and guarantee to reach the global optimum. Thus, we will discuss algorithms to optimize the smooth, unconstrained dual given in (3.5).

A critical choice in coordinate maximization algorithms is the block size. One possible choice is all variables of the form $\delta_{ci}(\cdot)$, for a fixed clique c and node i within it. This is called the *min-sum-diffusion* (MSD) algorithm, for which both non-smooth [Werner 2007] and smooth [Werner 2009] versions exist. The update equation for the smooth MSD algorithm is closed form and it is as follows

$$\delta_{ci}^{t+1}(x_i) = \delta_{ci}^{t}(x_i) + \frac{1}{2\tau}\log\frac{\phi_c^t(x_i)}{\phi_i^t(x_i)} \quad \forall x_i \tag{3.12}$$

where $\phi_c^t(\boldsymbol{x}_c)$ and $\phi_i^t(x_i)$ are obtained through 3.25, which we present in a later section (3.5). Note, those equations are presented in the context of computing optimal primal variables from optimal dual variables. In the case of MSD, the update equations need intermediate variables that are computed exactly like optimal primal variables. We would like to point out that MSD based update has a role to play in an important algorithm for general higher-order inference called SRMP [Kolmogorov 2015]. SRMP is a non-smooth coordinate maximization algorithm that works very well in practice and is very efficient, too.

Another choice, which is a larger block, is all variables of the form $\delta_{\cdot i}(\cdot)$, i.e., we fix a node i and consider all labels of that node and all cliques that contain that node. This is called the *star* update in the literature [Meshi 2014]. Remarkably, star update also has a closed form expression as follows,

$$\delta_{ci}^{t+1}(x_i) = \delta_{ci}^{t}(x_i) + \frac{1}{\tau}\log\phi_c^t(x_i) - \frac{1}{N_i+1}\cdot\frac{1}{\tau}\log\Big(\phi_i^t(x_i)\cdot\prod_{c':i\in c'}\phi_{c'}^t(x_i)\Big). \tag{3.13}$$

The schedule by which the coordinate blocks are visited and updated affects the convergence rate and also, the techniques used for analysis. A straightforward schedule is to fix the order in which the blocks are visited and cycle through them in a deterministic manner. This turns out to be a hard schedule to analyse and [Saha 2013, Beck 2013] present some results, which are not applicable to our problem. It seems visiting the blocks in a non-cyclic manner leads to simpler analysis [Meshi 2012]. The straightforward non-cyclic schedule is to randomly choose the next coordinate block and it has been analysed for coordinate ascent in general [Lu 2015] and for coordinate maximization for MAP inference [Meshi 2012]. Also, the block to be updated could be chosen in a greedy manner. This leads to a deterministic non-cyclic approach which chooses at each step, the coordinate block that will give maximum improvement in the objective value. Of course, choosing the block that will lead to maximum gain could be computationally costly. However, [Meshi 2014] present an efficient priority queue based approach that is specific for MAP inference. We state here the convergence rates for the greedy and random schemes. Please, refer to [Meshi 2012] for proofs.

Theorem 2 (Greedy Block Maximization)**.** *Suppose* $|\delta^t - \delta^*|_1 \leq B_1$ *for all* t *for a suitable constant* B_1*. If there exists* $k > 0$ *so that coordinate maximization of each*

block S_i satisfies:

$$F(\delta^{t+1}) - F(\delta^t) \geq \frac{1}{k} \left|\nabla_{S_i} F(\delta^t)\right|_\infty^2 \tag{3.14}$$

for all t, then for any $\varepsilon > 0$ after $T = \frac{kB_1^2}{\varepsilon}$ iterations, we have $F(\delta^*) - F(\delta^t) \leq \varepsilon$.

Theorem 3 (Stochastic Block Maximization)**.** *Suppose $\left|\delta^t - \delta^*\right|_2 \leq B_2$ for all t for a suitable constant B_2. If there exists $k > 0$ so that coordinate maximization of each block S_i satisfies:*

$$F(\delta^{t+1}) - F(\delta^t) \geq \frac{1}{k} \left|\nabla_{S_i} F(\delta^t)\right|_2^2 \tag{3.15}$$

for all t, then for any $\varepsilon > 0$ after $T = \frac{k|\mathcal{S}|B_2^2}{\varepsilon}$ iterations, we have $\mathbb{E}[F(\delta^*) - F(\delta^t)] \leq \varepsilon$, where the expectation is taken with respect to the randomization of the blocks.

In our experiments, we obtained consistently good results with star update, where the blocks are chosen at random. We found MSD update to be always poorer to star update.

3.4 Augmented Lagrangian Methods

Alternating direction method of multipliers (ADMM) [Boyd 2011, Eckstein 1989] is an optimization approach that has led to considerable research effort both in theory and applications. It is a form of the augmented Lagrangian algorithm and one of its major advantages is its suitability for parallel and distributed computation. It can leverage the structure of the problem, for example, the objective being a sum of simpler functions. This is the case with LP relaxation for MAP inference. Thus ADMM based methods are yet another set of methods that leverage the graph structure of the MAP problem. Just like other methods based on dual decomposition, these algorithms involve local computation and message passing.

We briefly review the general ADMM approach for convex optimization. Consider the following optimization problem,

$$\text{minimize}\, f(x) + g(Mx). \tag{3.16}$$

Here, $M \in \mathcal{R}^{m \times n}$, f and g are convex functions and $x \in \mathcal{R}^n$ is the decision variable. Many problems in machine learning take the form 3.16, where the function f is a loss and g could be a regularizer or a constraint set. By introducing an additional decision variable ($z \in \mathcal{R}^m$), we could write equation 3.16 as

$$\text{minimize}\, f(x) + g(z) \quad \text{subject to} \;\; Mx = z. \tag{3.17}$$

In many problems, we will work with only one decision variable and the additional variable will lead to convenient optimization algorithms. Given the problem set-up (3.17), the *augmented Lagrangian* has the following form,

$$\mathcal{L}_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^\top (Mx - z) + \frac{\rho}{2} ||Mx - z||^2 \tag{3.18}$$

where $\lambda \in \mathcal{R}^n$ is a vector of Lagrange multipliers for the constraint $MX = z$ and ρ is a positive scalar parameter bounded away from 0. Thus optimization algorithms seek to find a saddle point of this augmented Lagrangian. The alternating direction method of multipliers (ADMM) for 3.17 (and hence, 3.16) has the following update equations,

$$\begin{aligned} x^{k+1} &= \arg\min_{x \in \mathcal{R}^n} \mathcal{L}_{\rho_k}(x, z^k, \lambda^k) \\ z^{k+1} &= \arg\min_{z \in \mathcal{R}^m} \mathcal{L}_{\rho_k}(x^{k+1}, z, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \rho_k(Mx^{k+1} - z^{k+1}) \end{aligned} \tag{3.19}$$

where (x^k, z^k) and λ^k are sequences of estimates of the solution (x^*, z^*) and the Lagrangian multipliers, respectively. [Boyd 2011, §3.4.1] describes a standard way to adjust ρ_k as the algorithm proceeds. Given the constant terms in each update equation, we can see that the update of x involves only a quadratic perturbation of f and the update of z involves only a quadratic perturbation of g. Thus the functions f and g are decoupled and the individual structure of f and g could be exploited for efficient updates and for parallelization.

We have earlier seen the advantage of smoothing based methods. The augmented Lagrangian achieves the same by the presence of the quadratic term. Even though several ADMM based approaches have been proposed so far for MAP inference [Martins 2015, Meshi 2011, Fu 2013, Miksik 2014], only two approaches explicitly present a convex optimization approach for higher-order MRFs: AD3 [Martins 2015] and ADLP [Meshi 2011]. Even though ADLP is guaranteed to converge for any type of clique potential, we will see AD3 in greater detail below, as it is more widely used in the community.

3.4.1 AD3

The algorithm called alternating directions dual decomposition (AD3) approaches the MAP inference problem by constructing the augmented Lagrangian of the primal problem, given in 2.10. We will write the constraints defined by the local polytope as follows,

$$\begin{aligned} \sum_{x_i} \phi_i(x_i) = 1 \;\text{ and }\; \phi_i(x_i) \geq 0 \;\; \forall i \in \mathcal{V}, \forall x_i \\ \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c) = 1 \;\text{ and }\; \phi_c(\boldsymbol{x}_c) \geq 0 \;\; \forall c \in \mathcal{C}, \forall \boldsymbol{x}_c \\ \boldsymbol{M}_{ic}\boldsymbol{\phi}_c = \boldsymbol{\phi}_i \quad \forall (i, c) : i \in c \end{aligned} \tag{3.20}$$

where for each clique c, $\boldsymbol{\phi}_c$ is the vector of $\phi_c(\boldsymbol{x}_c)$ values for all $\boldsymbol{x}_c$ and similarly, for each node i, $\boldsymbol{\phi}_i$ is the vector of $\phi_i(x_i)$ values for all x_i and the elements of matrix $\boldsymbol{M}_{ic}$ is 1 if the label of node i within the clique label $\boldsymbol{x}_c$ matches with its actual label x_i and 0, otherwise. Thus it is possible to see the parallels between the primal

LP formulation for MAP inference and the general form of ADMM problems, given in equation (3.17). Thus the augmented Lagrangian will take the following form,

$$\mathcal{L}_\rho(\Phi_c, \Phi_i, \lambda) = \sum_{c\in\mathcal{C}} \boldsymbol{\theta}_c^\top \boldsymbol{\phi}_c + \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{\phi}_i + \sum_{(i,c):i\in c} \boldsymbol{\lambda}_{ic}^\top (\boldsymbol{M}_{ic}\boldsymbol{\phi}_c - \boldsymbol{\phi}_i) + \frac{\rho}{2} \sum_{(i,c):i\in c} ||\boldsymbol{M}_{ic}\boldsymbol{\phi}_c - \boldsymbol{\phi}_i||^2. \tag{3.21}$$

Here, Φ_i is the set of all $\boldsymbol{\phi}_i$'s and Φ_c is the set of all $\boldsymbol{\phi}_c$'s. It can be re-written as follows based on the constraint $\boldsymbol{M}_{ic}\boldsymbol{\phi}_c = \boldsymbol{\phi}_i \;\; \forall (i,c) : i \in c$,

$$\sum_{c\in\mathcal{C}} \Big(\boldsymbol{\theta}_c + \sum_{i:i\in c} \boldsymbol{M}_{ic}^\top (\boldsymbol{\theta}_i + \boldsymbol{\lambda}_{ic})\Big)^\top \boldsymbol{\phi}_c - \sum_{(i,c):i\in c} \boldsymbol{\lambda}_{ic}^\top \boldsymbol{\phi}_i + \frac{\rho}{2} \sum_{(i,c):i\in c} ||\boldsymbol{M}_{ic}\boldsymbol{\phi}_c - \boldsymbol{\phi}_i||^2. \tag{3.22}$$

Thus, AD3 updates the variables Φ_i and Φ_c, along with the Lagrangian multipliers $\boldsymbol{\lambda}$ as follows,

$$\Phi_c^{k+1} = \underset{\Phi_c \in \mathcal{R}^{|\mathcal{C}l^s|}}{\arg\min}\, \mathcal{L}_{\rho_k}(\Phi_c, \Phi_i^k, \boldsymbol{\lambda}^k) \tag{3.23a}$$

$$\Phi_i^{k+1} = \underset{\Phi_i \in \mathcal{R}^{|\mathcal{V}|l}}{\arg\min}\, \mathcal{L}_{\rho_k}(\Phi_c^{k+1}, \Phi_i, \boldsymbol{\lambda}^k) \tag{3.23b}$$

$$\boldsymbol{\lambda}_{ic}^{k+1} = \lambda_{ic}^k + \rho_k \Big(\boldsymbol{M}_{ic}\boldsymbol{\phi}_c^{k+1} - \boldsymbol{\phi}_i^{k+1}\Big) \;\; \forall (i,c) : i \in c. \tag{3.23c}$$

Both the updates of the clique variables, $\boldsymbol{\phi}_c$'s (3.23a) and the node variables, $\boldsymbol{\phi}_i$'s (3.23b) could be parallelized. Notice that the optimization problem to update the clique variables, $\boldsymbol{\phi}_c$'s, is a quadratic program for each of the cliques. [Martins 2015] shows that these quadratic programs can be solved exactly and efficiently for particular types of models: Ising models, factor graphs imposing first-order logic (FOL) constraints and for Potts model (after binarization). For general clique potentials, they present an active-set approach based on the fact that the optimal $\boldsymbol{\phi}_c$'s are provably sparse. Given the clique variables, the primal node variables, $\boldsymbol{\phi}_i$'s, are updated with the following close form expression,

$$\boldsymbol{\phi}_i^{k+1}(x_i) = \frac{1}{|N(i)|} \sum_{c:i\in c} \boldsymbol{\phi}_{ic}^{k+1}, \;\; \text{where} \;\; \boldsymbol{\phi}_{ic}^{k+1} = \boldsymbol{M}_{ic}\boldsymbol{\phi}_c^{k+1}. \tag{3.24}$$

AD3 is able to achieve considerable speed-up by caching the results of the subproblems. If at iteration $(t+1)$, $\boldsymbol{\phi}_i^{(t)} = \boldsymbol{\phi}_{ic}^{(t+1)} \;\; \forall c : i \in c$, then according to (3.24) the primal variable $\boldsymbol{\phi}_i$ will not change. According to (3.23c), it will be the same case with $\boldsymbol{\lambda}_{ic}$. Thus if at some iteration t if all node variables and Lagrangean multipliers corresponding to clique c are idle, then $\boldsymbol{\phi}_c^{(t+1)} = \boldsymbol{\phi}_c^{(t)}$, hence, the quadratic program need not be solved. Also, the active-set apprach allows warm start, which leads to speed-up. AD3 has a convergence rate of $O(\frac{1}{\varepsilon})$ when the quadratic program is solved exactly. However, it is too costly to run the active-set method to optimum for general clique potentials. AD3 truncates the number of iterations according to a hard coded maximum number. In theory, AD3 would converge as long as sufficient decrease is obtained while solving the quadratic programs. However, determining how much is sufficient is a difficult question. Thus, it is difficult to analyse the convergence rate of AD3 with the active-set method. Also, there is no convergence guarantee for AD3 with the active-set method and we have observed this empirically.

3.5 Estimating primal variables

We have seen a broad range of optimization algorithms for MAP inference. In general, for optimization algorithms, it is possible to recover optimal primal variables from optimal dual variables. However, it is important to obtain feasible *intermediate* primal variables when the algorithm is in progress. It is useful for many purposes:

- To provide a non-heuristic stopping criterion for the optimization algorithm, based on primal-dual gap.
- To determine when to anneal the smoothing parameter (τ) for the smoothing based algorithms, again based on primal-dual gap.
- To perform convergence rate analysis of the primal problem, e.g., [Meshi 2014, 1.5.1].

For primal-dual algorithms like AD3, feasible primal and dual variables are available in each iteration as the algorithm progresses. For some other algorithms feasible primal variables are only available at the optimum. For example, for the smoothing based algorithms, the perturbed primal problem (3.3) is convex with a differentiable objective and affine constraints for which *strong duality* obtains [Boyd 2004, section 5.2.3]. Therefore, a pair of optimal primal and dual variables must satisfy the KKT conditions. Given the *optimal* dual variables of the smooth dual given in (3.5), it is possible to estimate *optimal* primal variables as follows,

$$\begin{aligned}
\phi_c^*(\boldsymbol{x}_c) &= \frac{\exp\big(\tau\theta_c(\boldsymbol{x}_c) - \tau \sum\limits_{i:i\in c} \delta_{ci}(x_i)\big)}{\sum\limits_{\boldsymbol{x}_c} \exp\big(\tau\theta_c(\boldsymbol{x}_c) - \tau \sum\limits_{i:i\in c} \delta_{ci}(x_i)\big)} \\
\phi_i^*(x_i) &= \frac{\exp\big(\tau\theta_i(x_i) + \tau \sum\limits_{c:i\in c} \delta_{ci}(x_i)\big)}{\sum\limits_{x_i} \exp\big(\tau\theta_i(x_i) + \tau \sum\limits_{c:i\in c} \delta_{ci}(x_i)\big)}.
\end{aligned} \tag{3.25}$$

However, when the primal variables are estimated according to these equations at an intermediate stage, they need not be feasible. The same holds true for the primal variables estimated by supergradient ascent methods [Komodakis 2011], which obtains the primal variables as time averaged or step-size averaged supergradients. There are two main approaches in the literature that recover feasible intermediate primal variables [Savchynskyy 2014, Meshi 2014]. In both approaches first approximate intermediate primal variables are computed, say, according to (3.25). Unless at the optimum, these primal variables need not satisfy the marginalization constraints in (2.10b). The approach in [Savchynskyy 2014] solves small linear programs with respect to each clique to obtain feasible clique primal variables. These linear programs take the form of *transportation problems* and could be efficiently solved. The approach given in [Meshi 2014] is simpler and doesn't involve solving small optimization problems. This is the approach that we have adopted in our experiments and we describe the steps in algorithm (2).

First, estimates for the node and clique primal variables are obtained, which are most likely infeasible. (3.25) shows one such way to obtain these beliefs. At this stage, the vector of beliefs for each node and clique, will satisfy the simplex based normalization constraint. In order to make them satisfy the marginalization constraint, we adjust them next. However, after this step, they may no longer satisfy the simplex constraint. In the final step, they are adjusted again, such that they satisfy both the normalization and marginalization constraints.

Algorithm 2 Mapping to a feasible primal point

Step 0: Obtain infeasible primal estimates, $\phi_i(x_i)$ and $\phi_c(\boldsymbol{x}_c)$ according to (3.25).

Step 1: Obtain marginals satisfying marginalization constraint.

For all c obtain: $\phi_c(x_i) \triangleq \sum_{\boldsymbol{x}_{c\setminus i}} \phi_c(\boldsymbol{x}_c)$ according to (2.10b).

For all i obtain: $\hat{\phi}_i(x_i) = \frac{1}{1+\sum_{c:i\in c}\frac{1}{|\boldsymbol{X}_{c\setminus i}|}}\left(\phi_i(x_i) + \sum_{c:i\in c}\frac{1}{|\boldsymbol{X}_{c\setminus i}|}\phi_c(x_i)\right)$

For all c obtain: $\hat{\phi}_c(\boldsymbol{x}_c) = \phi_c(\boldsymbol{x}_c) - \sum_{i:i\in c}\frac{1}{|\boldsymbol{X}_{c\setminus i}|}\left(\phi_c(x_i) - \hat{\phi}_i(x_i)\right)$

Step 2: Obtain marginals satisfying simplex constraint.

$\lambda = 0$

for $c \in \mathcal{C}, \boldsymbol{x}_c$ **do**

  if $\hat{\phi}_c(\boldsymbol{x}_c) < 0$ **then**

  $\lambda = \max\left\{\lambda, \frac{-\hat{\phi}_c(\boldsymbol{x}_c)}{-\hat{\phi}_c(\boldsymbol{x}_c)+\frac{1}{|\boldsymbol{X}_c|}}\right\}$

  else if $\hat{\phi}_c(\boldsymbol{x}_c) > 1$ **then**

  $\lambda = \max\left\{\lambda, \frac{\hat{\phi}_c(\boldsymbol{x}_c)-1}{\hat{\phi}_c(\boldsymbol{x}_c)-\frac{1}{|\boldsymbol{X}_c|}}\right\}$

  end if

end for

for $l \in \mathcal{V}$ **do**

  $\tilde{\phi}_l(x_l) = (1-\lambda)\hat{\phi}_l(x_l) + \lambda\frac{1}{|X_l|}$

end for

for $l \in \mathcal{C}$ **do**

  $\tilde{\phi}_l(\boldsymbol{x}_l) = (1-\lambda)\hat{\phi}_l(\boldsymbol{x}_l) + \lambda\frac{1}{|\boldsymbol{X}_l|}$

end for

Even though an unconstrained optimization problem is being solved while optimizing the dual given in (3.5), first order methods do not reach low gradient norm values in practice (say, $\|\nabla g(\boldsymbol{\delta})\|_\infty \leq 10^{-3}$). Thus, exit condition based on primal-dual gap is important for first order methods. On the other hand, we have consistently observed $\|\nabla g(\boldsymbol{\delta})\|_\infty$ reaching low values, when the true Hessian is being used within Newton-type methods. This could be because of Newton-type methods having superior convergence behavior when they are close to the optimum.

Given optimal primal variables of the LP relaxation, we are still faced with the

task of recovering integral primal variables of the underlying ILP (2.9). Usually, one resorts to *rounding* the fractional values to integral values. [Ravikumar 2010] presents a thorough analysis of various rounding schemes. AD3 [Martins 2015] presents a branch-and-bound based post-processing that further improves the results towards the global optimum. We would also like to point out the fact that there will be ties in the optimum labelling obtained. [Meltzer 2005] has shown a way to resolve ties along monotonic chains and it has been used by other algorithms [Kolmogorov 2006, Komodakis 2011]. In our experiments, we have used a simple maximum based rounding scheme and we resolve ties randomly.

3.6 Lipschitz constant of the smooth dual

We have been discussing optimization algorithms and their convergence rates. We shown the rates in terms of the desired accuracy ε. However, we have to keep in mind that the convergence rate is also a function of the Lipschitz constant (L) of the function. We stated the convergence rate of accelerated gradient ascent to be $O(\frac{1}{\varepsilon})$, when in fact $O(\sqrt{\frac{L}{\varepsilon}})$ would be more accurate. In general, higher the Lipschitz constant, poorer is the rate of convergence of optimization algorithms. Thus, it is important to know what factors affect the Lipschitz constant. We will briefly see how to derive the Lipschitz constant of the smooth dual in (3.5). Our derivation is based on the technique proposed by [Nesterov 2005].

We could re-write the marginalization constraints in (2.10b) as follows,

$$A^\top \boldsymbol{\phi} = \mathbf{0} \tag{3.26}$$

where A is a matrix suitably constructed with only 0s, 1s and -1s. It is of dimension $|\boldsymbol{\phi}| \times |\boldsymbol{\delta}|$. Also, based on 9.1.7, we could define a proximal operator for the normalization constraints (2.10c) as follows,

$$d(\boldsymbol{\phi}) = \sum_{i \in \mathcal{V}} \left(\log l + \sum_{x_i} \phi(x_i) \log \phi_(x_i) \right) + \sum_{c \in \mathcal{C}} \left(s \log l + \sum_{\boldsymbol{x}_c} \phi(\boldsymbol{x}_c) \log \phi(\boldsymbol{x}_c) \right). \tag{3.27}$$

It has a strong convexity parameter of

$$\sigma = \frac{1}{|\mathcal{V}| + |\mathcal{C}|} \tag{3.28}$$

with respect to the l_1 norm. Thus, we could define the smooth dual objective as follows,

$$\begin{aligned} g_\tau(\boldsymbol{\delta}) = \underset{\boldsymbol{\phi}}{\text{maximize}} \quad & \boldsymbol{\delta}^\top A^\top \boldsymbol{\phi} - \langle \boldsymbol{\theta}, \boldsymbol{\phi} \rangle - \frac{1}{\tau} d(\boldsymbol{\phi}) \\ \text{subject to} \quad & \sum_{x_i} \phi_i(x_i) = 1, \quad \forall i \in \mathcal{V}; \quad \sum_{\boldsymbol{x}_c} \phi_c(\boldsymbol{x}_c) = 1, \quad \forall c \in \mathcal{C} \\ & \phi_i(x_i) \geq 0, \ \ \forall i \in \mathcal{V}; \quad \phi_c(\boldsymbol{x}_c) \geq 0 \ \ \forall c \in \mathcal{C}. \end{aligned} \tag{3.29}$$

After maximizing with respect to $\boldsymbol{\phi}$, we obtain the same smooth dual objective that we saw in (2.25). Note, that the derivation of the dual based on proximal operators is equivalent to obtaining the dual based on the perturbed primal objective in (3.2). According to [Nesterov 2005], the smooth dual objective function has a Lipschitz continuous gradient with respect to l_p norm as follows,

$$\begin{aligned} &\|\nabla g_\tau(\boldsymbol{x}) - \nabla g_\tau(\boldsymbol{y})\|_p \leq L_\tau \|\boldsymbol{x} - \boldsymbol{y}\|_p \;\; \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{R}^n \\ &\text{where} \;\; L_\tau = \frac{\tau \|A\|_{p\infty}^2}{\sigma} = \tau(|\mathcal{V}| + |\mathcal{C}|)\|A\|_{p\infty}^2. \end{aligned} \tag{3.30}$$

The $\|\cdot\|_{2\infty}$ norm of A is considered. In our experiments, we work with algorithms which assume that the objective function has a Lipschitz continuous gradient with respect to the l_2 norm. The range space is equipped with l_∞-norm because of the strong convexity of the entropy terms with respect to the l_1 norm and the dual of the l_1 norm is the l_∞ norm. See, [Nesterov 2005] for more details. Now, the following result is given in [Higham 1992],

$$\|A\|_{2\infty} = \max_{i \in [1,\ldots,|\phi|]} \|A_{i\cdot}\| = \sqrt{N_{max}} \tag{3.31}$$

where N_{max} is the maximum number of cliques sharing a node. Thus, we have the following expression for the Lipschitz constant,

$$L_\tau = \tau(|\mathcal{V}| + |\mathcal{C}|)N_{max}. \tag{3.32}$$

Thus, lesser smoothing (larger τ), more nodes, more cliques and more cliques sharing a node all affect the Lipschitz constant of the smooth dual. Thus, we have to expect convergence rate to be affected by all these factors. Note, a similar derivation can be made for the smooth version of the linearly constrained dual in (2.20). Given the expression of the Lipschitz constant, the parameter k in SCM (3.14, 3.15), for MSD update is $k = 4\tau$ and for star update is $k = 4\tau N_i$ [Meshi 2014, §1.4.3]. We recommend a detailed table in [Meshi 2015] for a ready reference regarding the convergence rates of MAP inference algorithms.

3.7 Dual decomposition for large graphs

In order to obtain practical convergence, there is an interplay between the sizes of the graph and the subgraphs over which the dual decomposition is derived [Komodakis 2009, Wang 2013, Miksik 2014]. This is especially true for gradient based and augmented Lagrangian based algorithms. For medium sized graphs, decomposing into individual cliques like (3.5), leads to excellent practical convergence. However, for large graphs, only bigger subgraphs like chains of cliques lead to a practical convergence rate and there is considerable empirical evidence regarding larger subgraphs leading to better convergence [Komodakis 2009, Schmidt 2011, Wang 2013, Miksik 2014]. Now, we will consider some possible explanations for observing poorer convergence with smaller subgraphs. When working with the non-smooth dual (2.20)

using a supergradient algorithm, we have seen that MAP inference over individual subgraphs leads to a valid supergradient. Too many subgraphs, leads to less overlap among the subgraphs, thus it could lead to noisier supergradients. When working with the smooth dual (3.5), we have seen in section (3.6) that the Lipschitz constant increases with the number of subgraphs. If the graph is decomposed using large subgraphs, then the total number of subgraphs comes down, leading to a lower Lipschitz constant. Note, if one has densely connected graphs, i.e., a node shared by many cliques leading to very large N_{max}, then convergence is going to be affected irrespective of graph decomposition. Also, there will be larger memory requirement because of more number of dual variables. In this thesis, we focus on sparse graphs.

Now, we will show how to derive a dual when the graph is large like a stereo problem. Also, show a way to reduce problem size when a node is shared by only two subgraphs. The dual in (2.25) is derived according to clique based decomposition and it is not the right way to go for gradient based algorithms for large graphs. We have observed this empirically, also. Consider the dual optimization problem in (2.19). We observed that if we have dual variables corresponding to pseudomarginals of both nodes and cliques, then we have $\sum_{i\in\mathcal{V}} n_{\mathcal{T}i}.l + \sum_{c\in\mathcal{C}} n_{\mathcal{T}c}.l^s$ dual variables. Here, each node i is shared by $n_{\mathcal{T}i}$ trees and takes l labels; each clique c is shared by $n_{\mathcal{T}c}$ and contains s nodes. Having so many dual variables is prohibitive for large problems. We noted in section (3.1) that the number of dual variables could be brought down by considering consistency constraints only between node pseudomarginals. This is valid as long as the subgraphs could be optimized exactly, i.e., they satisfy the integrality condition. However, enforcing consistency between only the node pseudomarginals reduces the rate of convergence of the algorithm. According to (3.32), we could improve the convergence rate by increasing the size of the subgraphs because in return this reduces the number of subgraphs. Thus, similar to section (2.5) we could derive the non-smooth dual as follows,

$$\max_{\boldsymbol{\delta}}. \sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} \min_{\boldsymbol{\phi}_i^{\mathcal{T}},\boldsymbol{\phi}_c^{\mathcal{T}}}. \left(\sum_{i\in\mathcal{T}} \sum_{x_i} (\theta_i^{\mathcal{T}}(x_i) + \delta_{\mathcal{T}i}(x_i))\phi_i^{\mathcal{T}}(x_i) + \sum_{c\in\mathcal{T}} \sum_{\boldsymbol{x}_c} \theta_c^{\mathcal{T}}(\boldsymbol{x}_c)\phi_c^{\mathcal{T}}(\boldsymbol{x}_c) \right) \tag{3.33a}$$

$$\text{subject to } \boldsymbol{\phi}^{\mathcal{T}} \in \mathcal{L}(\mathcal{T}) \;\; \forall \mathcal{T} \in \mathcal{T}(\mathcal{G}) \tag{3.33b}$$

$$\sum_{\mathcal{T}:i\in\mathcal{T}} \delta_{\mathcal{T}i}(x_i) = 0, \quad \forall i \in \mathcal{V}. \tag{3.33c}$$

Here, the dual variables are $\delta_{\mathcal{T}i}(x_i)$, one for each node i and each node label x_i, for all trees $\mathcal{T}$ that contain i. We can notice the reduction in number of dual variables compared to (2.19). Now, suppose, each node is shared by only two subgraphs. For example, if the graphical model corresponds to an image grid, the rows could form one subgraph and the columns could form another subgraph. Note, a subgraph need

not be connected. Then, we will have the following dual problem,

$$\max_{\boldsymbol{\delta}}. \min_{\boldsymbol{\phi}_i^{\mathcal{T}_1}, \boldsymbol{\phi}_c^{\mathcal{T}_1}} \left(\sum_{i \in \mathcal{T}_1} \sum_{x_i} (\theta_i^{\mathcal{T}_1}(x_i) + \delta_i(x_i)) \phi_i^{\mathcal{T}_1}(x_i) + \sum_{c \in \mathcal{T}_1} \sum_{\boldsymbol{x}_c} \theta_c^{\mathcal{T}_1}(\boldsymbol{x}_c) \phi_c^{\mathcal{T}_1}(\boldsymbol{x}_c) \right)$$
$$+ \min_{\boldsymbol{\phi}_i^{\mathcal{T}_2}, \boldsymbol{\phi}_c^{\mathcal{T}_2}} \left(\sum_{i \in \mathcal{T}_2} \sum_{x_i} (\theta_i^{\mathcal{T}_2}(x_i) - \delta_i(x_i)) \phi_i^{\mathcal{T}_2}(x_i) + \sum_{c \in \mathcal{T}_2} \sum_{\boldsymbol{x}_c} \theta_c^{\mathcal{T}_2}(\boldsymbol{x}_c) \phi_c^{\mathcal{T}_2}(\boldsymbol{x}_c) \right) \tag{3.34a}$$

$$\text{subject to } \boldsymbol{\phi}^{\mathcal{T}_1} \in \mathcal{L}(\mathcal{T}_1), \;\; \boldsymbol{\phi}^{\mathcal{T}_2} \in \mathcal{L}(\mathcal{T}_2) \tag{3.34b}$$

where, we have used the following simplification of the constraint on the dual variables,

$$\delta_{\mathcal{T}_1 i}(x_i) + \delta_{\mathcal{T}_2 i}(x_i) = 0 \implies \delta_{\mathcal{T}_1 i}(x_i) = \delta_i(x_i), \; \delta_{\mathcal{T}_2 i}(x_i) = -\delta_i(x_i). \tag{3.35}$$

Hence, the dual variables are $\delta_i(x_i)$, one for each node i and each node label x_i. Thus, we can see the reduction in number of dual variables for this special case. Note that in these large scale problems, when we refer to subgraphs, we refer to trees, unless otherwise stated. Note, we have shown all the results with non-smooth duals but they can readily be smoothed, just like (3.5).

3.8 Efficient belief propagation in trees

In section (3.2), we mentioned how the terms in the gradient of the smooth dual could be computed by the sum-product algorithm. This also holds true for terms in update equations for smooth coordinate maximization. Also, the active set method used by AD3 has an intermediate step that is equivalent to max-product computation. We have been looking at subgraphs in the form of trees, where we are guaranteed to obtain exact inference. As number of labels (l) increases or as the clique size (s) increases the complexity of inference which is $O(l^s)$ increases. Efficient inference by exploiting the specific nature of the clique potentials has received considerable research attention: [Potetz 2008] presents an $O(l^2)$ approach for both MAP and marginal inference for linear constraint nodes; [Duchi 2007] shows an approach for matchings; [Tarlow 2010] discusses efficient MAP inference for cardinality based and order based potentials; [Komodakis 2009] deals with efficient MAP inference in sparse pattern-based potentials. Another interesting approach is to use a randomized algorithm [Noorshams 2013] that achieves $O(l)$ complexity for pairwise graphs. Here, we will discuss our approach to marginal inference in higher-order pattern-based potentials. We have implemented this approach for all algorithms optimizing the smooth dual (3.5).

Sparse pattern-based clique potentials are very useful in computer vision [Komodakis 2009, Rother 2009]. In these potentials, a big majority of the labellings take a constant (usually high) value and a small subset (of size k) take other significant values. Many clique potentials fit this description. [Komodakis 2009] showed an

efficient way to perform max-product computation in chains of such cliques. It is not clear how to extend their work to sum-product computation. [Coughlan 2010] showed a sum-product approach for pattern-based potentials in pairwise MRFs. They achieved a complexity of $O(2l + k)$, instead of $O(l^2)$. They mention that their idea can be used for higher order cliques with a factor graph representation but don't go into details. We have found that with a factor graph representation, their approach has a complexity of $O(l^{s-1} + l + k)$. Instead of a factor graph, if a clique tree representation (2.3) is used, it is possible to achieve much better complexity of $O(l^{s-n} + l^n + k)$, where n is the size of a subset of clique nodes. Suppose we want to compute marginals within a single clique, these subsets will be (nearly) equal sized partitions of the clique. For clique chains, this is the size of the sepset between two overlapping cliques. We will recall that the sepset contains the nodes shared by the two cliques. For example, for cliques of size 4, with either subset or sepset of size 2, one can quickly see the efficiency gain.

The graphical model for which we perform sum-product inference has potentials, $\hat{\theta}_i(x_i)$ and $\hat{\theta}_c(\boldsymbol{x}_c)$, obtained according to the reparameterization implied by the dual. Refer to section (2.6) for this reparameterization interpretation. In order to discuss about sum-product algorithm, it is more convenient to work with potential functions than potentials (2.3). Let $\psi_i(x_i) = \exp(-\hat{\theta}_i(x_i))$ and $\psi_c(\boldsymbol{x}_c) = \exp(-\hat{\theta}_c(\boldsymbol{x}_c))$ denote the node and clique potential functions, respectively. We will let $p_i^c(x_i)$ denote the marginal probability of node i taking label x_i in clique c of this graphical model. We saw in equation (3.7), that the gradient is made of two terms $\mu_{ci}(x_i)$ and $\mu_i(x_i)$. Now, $\mu_i(x_i)$ could be computed with complexity $O(l)$ and it is not an impediment. Interestingly, $\mu_{ci}(x_i)$ is equal to $p_i^c(x_i)$ and could be computed by leveraging efficient sum-product algorithms.

In the following, we will consider clique chains. This is for simpler notation and these results are applicable to trees, in general. Also, these results can be reduced to the case of individual cliques. As shown in figure (3.2), let us work with three neighbouring cliques in a chain, a, b and c. m is the sepset between a and b and n is the sepset between b and c. Sometimes, the sepsets can be complementary to each other, i.e., $m = b \setminus n$.

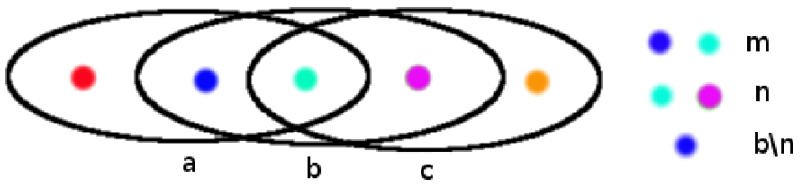

Figure 3.2: A clique chain.

A sum-product message is passed from b to c, as follows,

$$\begin{aligned} m_{b\to c}(\boldsymbol{x}_n) &= \sum_{\boldsymbol{x}_{b\setminus n}} \psi_b(\boldsymbol{x}_b)\nu(\boldsymbol{x}_{b\setminus n}) \\ \text{where, } \nu(\boldsymbol{x}_{b\setminus n}) &= \Big(\prod_{i\in b\setminus n} \psi_i(x_i)\Big) m_{a\to b}(\boldsymbol{x}_{b\setminus n}) \end{aligned} \tag{3.36}$$

$m_{a\to b}(\boldsymbol{x}_{b\setminus n})$ is derived from the message sent by clique a to b. That message was for the sepset m and $m_{a\to b}(\boldsymbol{x}_{b\setminus n})$ is obtained by marginalization. Let $\bar{\psi}$ denote the large constant value taken by the clique potential function. Let $Pat(\boldsymbol{x}_n)$ denote the pattern-based clique labellings corresponding to the sepset labelling $\boldsymbol{x}_n$, then the first equation of (3.36) can be written as,

$$\begin{aligned} m_{b\to c}(\boldsymbol{x}_n) &= \sum_{\substack{\boldsymbol{x}_{b\setminus n} \\ \in Pat(\boldsymbol{x}_n)}} \psi_b(\boldsymbol{x}_b)\nu(\boldsymbol{x}_{b\setminus n}) + \sum_{\substack{\boldsymbol{x}_{b\setminus n} \\ \notin Pat(\boldsymbol{x}_n)}} \bar{\psi}\nu(\boldsymbol{x}_{b\setminus n}) \\ &= \sum_{\substack{\boldsymbol{x}_{b\setminus n} \\ \in Pat(\boldsymbol{x}_n)}} (\psi_b(\boldsymbol{x}_b) - \bar{\psi})\nu(\boldsymbol{x}_{b\setminus n}) + \bar{\psi}\sum_{\boldsymbol{x}_{b\setminus n}} \nu(\boldsymbol{x}_{b\setminus n}) \end{aligned} \tag{3.37}$$

In (3.37), the second summation could be computed only once with complexity $O(l^{s-n})$ and used for all elements of $\boldsymbol{x}_n$. For each labelling of $\boldsymbol{x}_n$, the first summation could be computed with complexity $O(k)$. Finally, by iterating once through all labellings of $\boldsymbol{x}_n$, we obtain the message with $O(l^n)$ complexity. Within this last loop to compute the message, the probabilities $p_i^b(x_i)$ of the corresponding nodes within the clique b could be computed.

Refer (9.2), for a numerically stable implementation of this scheme. In the coming chapters we will be discussing about Newton-type methods for MAP inference. We would like to point out that having cliques which allow efficient sum-product computation makes Newton-type methods more suitable for that problem.

Chapter 4

Newton Methods and Linear System Solvers

Newton methods are a family of optimization methods that have played a long and important role in optimization literature. In general, optimization algorithms make progress towards the optimum by computing a search direction in each iteration. The central concept in Newton methods is the use of curvature information to compute the search direction. The gradient of a function holds first order information based on which the search direction could be obtained. The Hessian holds second order information (i.e., curvature), which could lead to a better search direction. In each iteration, a local quadratic approximation based on the Hessian is constructed and the desired Newton direction points towards the minimum of this quadratic approximation. In figure (4.1), this concept is illustrated for a two dimensional function f, which is a function of (x, y). At iteration t, the optimization algorithm will be at (x^t, y^t). A quadratic q is constructed based on the Hessian matrix at this point. We can see that q is tangent to f at (x^t, y^t). The algorithm moves towards the minimizer of this quadratic and that point becomes x^{t+1}, y^{t+1}. Thus, the Newton method progresses towards the minimizer of f.

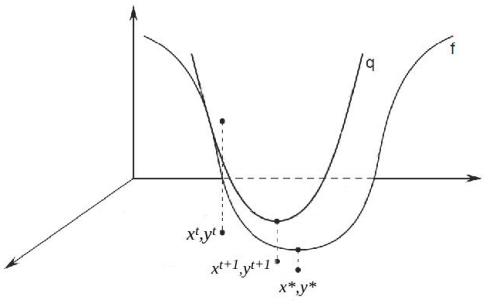

Figure 4.1: Computing the Newton direction by forming the quadratic approximation. *Image adapted from https://suzyahyah.github.io/calculus/optimization/2018/04/06/Taylor-Series-Newtons-Method.html.*

Many concepts in the literature of Newton methods and Linear system solvers are in terms of a minimization problem. Hence, we will consider the negative of the smooth dual in (3.5) as our model problem as we discuss about Newton methods

and linear systems. We will state our model problem here for ready reference,

$$\min_{\delta}. \sum_{c} \operatorname*{smax}_{\boldsymbol{x}_c}(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i); \tau) + \sum_{i} \operatorname*{smax}_{x_i}(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i); \tau). \quad (4.1)$$

We will denote the objective of the optimization problem as $h(\boldsymbol{\delta})$. We already came across the negative soft-max function smin in (3.6) and *smax* is the *soft-max* function. Given a set of numbers $\boldsymbol{s} = \{s_1, s_2, ..., s_n\}$, the soft-max over $\boldsymbol{s}$ is defined as follows,

$$\operatorname*{smax}_{s\in\boldsymbol{s}}(s) = \frac{1}{\tau}\log\sum_{s\in\boldsymbol{s}}\exp(\tau s). \quad (4.2)$$

Thus, as $\tau \to \infty$, $\operatorname*{smax}_{s\in\boldsymbol{s}}(s) \quad \to$ the maximum value in set $\boldsymbol{s}$. Thus, the smin function *upper bounds* the maximum of the set, for finite values of τ.

Suppose at some iteration, we are at $\boldsymbol{\delta}$ and we are seeking the search direction $\boldsymbol{p}$. The Taylor series expansion of h at $\boldsymbol{\delta}$ takes the following form,

$$h(\boldsymbol{\delta} + \boldsymbol{p}) = h(\boldsymbol{\delta}) + \nabla h(\boldsymbol{\delta})^T\boldsymbol{p} + \boldsymbol{p}^T\nabla^2 h(\boldsymbol{\delta})\boldsymbol{p} + ... \quad (4.3)$$

Thus, a quadratic approximation to h at $\boldsymbol{\delta}$ takes the following form,

$$q(\boldsymbol{p}) = h(\boldsymbol{\delta}) + \nabla h(\boldsymbol{\delta})^T\boldsymbol{p} + \boldsymbol{p}^T B(\boldsymbol{\delta})\boldsymbol{p} \quad (4.4)$$

where, $B(\boldsymbol{\delta})$ could be the true Hessian ($\nabla^2 h(\boldsymbol{\delta})$) or a modified Hessian (we will see about this in section (4.8)) or a Hessian approximation (e.g., obtained through a quasi-Newton method). Thus, $B(\boldsymbol{\delta}) \in \mathcal{R}^{N\times N}$ carries the curvature information that the Newton method exploits. We denote $|\boldsymbol{\delta}|$ as N. Given a quadratic approximation at a point, Newton methods seek a $\boldsymbol{p}$ that points in the direction of the minimum of the *quadratic form*. Geometrically, a quadratic form is a bowl whose shape is determined by B. Thus, the Newton direction, $\boldsymbol{p}^*$, could be obtained as the minimizer of the quadratic form,

$$\boldsymbol{p}^* = \operatorname*{argmin}_{\boldsymbol{p}} q(\boldsymbol{p}) = \operatorname*{argmin}_{\boldsymbol{p}} \nabla h(\boldsymbol{\delta})^T\boldsymbol{p} + \boldsymbol{p}^\top B(\boldsymbol{\delta})\boldsymbol{p}. \quad (4.5)$$

For an unconstrained convex problem, the Newton direction could be found by minimizing the quadratic form in an unconstrained manner. $q(\boldsymbol{p})$ is also a smooth function and it is being optimized in a unconstrained manner. In such a setting, the minimum, $\boldsymbol{p}^*$ satisfies $\nabla q(\boldsymbol{p}^*) = 0$ [Boyd 2004, §9.1]. Thus, we get $\boldsymbol{p}^*$ to be the solution of the following linear system,

$$B(\boldsymbol{\delta})\boldsymbol{p} = -\nabla h(\boldsymbol{\delta}). \quad (4.6)$$

If $B(\boldsymbol{\delta})$ is positive definite, $\boldsymbol{p}^*$ is guaranteed to be a descent direction. If the linear system is solved exactly, we will obtain $\boldsymbol{p}^*$ with the following form,

$$\boldsymbol{p}^* = -B(\boldsymbol{\delta})^{-1}\nabla h(\boldsymbol{\delta}). \quad (4.7)$$

Later, we will see that we need not solve this linear system exactly to obtain the Newton direction. That will lead to saving of computational cost. Given a descent

direction, the next question is determining the length of the step along this direction. We will denote this step length as α. Therefore, the iterations in Newton's method will be of the following form,

$$\boldsymbol{\delta}^{t+1} = \boldsymbol{\delta}^t - \alpha^t \overline{B}^{-1} \nabla \bar{h} \tag{4.8}$$

where for notational convenience, we make the following abbreviations,

$$B(\boldsymbol{\delta}^t) = \overline{B} \qquad h(\boldsymbol{\delta}^t) = \bar{h}. \tag{4.9}$$

Before we see how to determine α^t, we present a strong result regarding convergence of Newton methods.

Theorem 4. *(Quadratic local convergence) If $\overline{B}$ is the exact Hessian, $\nabla^2 \bar{h}$ and if $\alpha^t = 1$ in the Newton iteration (4.8), then quadratic convergence is achieved, when the iterations are started sufficiently close to the optimum [Bertsekas 1999, §1.4].*

We have to note that this strong rate of convergence holds only when the iterations start *sufficiently close* to the optimum. In fact, Newton methods may not converge if the iterations start from a far away point and α^t is always 1. Thus, Newton methods require something called a *globalization* strategy, in order to ensure convergence when the iterations start from a far away point. The globalization strategy sets the value of the step length, α^t, in a suitable manner to ensure convergence. It could either follow a *line search* or a *trust-region* procedure. We will discuss about these procedures in sections (4.5) and (4.6). Immediately, we will look more closely at linear system solvers. The reason is that Newton methods do not treat the linear system solver as a black box that returns the exact solution of the linear system in each iteration. Instead, the algorithmic steps within the linear system solver become part of the algorithmic steps of Newton methods. Especially, there are two situations that are relevant to us: solving the linear system only upto an approximate solution and enforcing a trust-region while minimizing the quadratic form.

4.1 Linear System Solvers

While choosing a linear system solver, we have to keep in mind the size of the problem. MRFs for computer vision problems result in large problem sizes. Typically, linear system solvers are grouped as either *direct* or *iterative* methods. As the names suggest, iterative methods perform certain steps repeatedly in a loop, getting closer to the solution as the algorithm progresses. On the other hand, direct methods obtain the solution by an algorithm which is a collection of steps. One aspect about the linear system that can be exploited by an algorithm is the sparsity of the matrix. We will encounter a sparse linear system in chapter 5 while applying a Newton method to perform MAP inference. Even though there has been considerable progress for solving sparse linear systems by direct methods [Davis 2006, Davis 2016], there are advantages to using iterative methods within a Newton method. Before we see what they are, we will clarify that we are interested in linear systems of the form

(4.6), where $\overline{B} \in \mathcal{R}^{N \times N}$ is a *square* matrix and it is *sparse, symmetric* and *positive definite*, $-\nabla \bar{h} \in \mathcal{R}^N$ and $\boldsymbol{p} \in \mathcal{R}^N$ is the solution of the linear system that we are after. In general, the complexity of solving a such a linear system is $O(N^3)$, for both direct and iterative methods. Apart from sparsity, both families of methods could devise other algorithmic steps to achieve better efficiency.

There are two advantages to iterative methods that are suitable for their use within Newton methods. First, we will see in section 4.4 that it is not necessary to obtain the exact solution of the linear system in order to ensure that a Newton method converges to the solution. In fact, such *inexact* Newton methods achieve considerable computational efficiency, while maintaining convergence guarantee. Compared to direct methods, iterative methods like the conjugate gradient method (CG) offer a simple mechanism to achieve this approximate solution. Second, we will see in section 4.2 that iterative methods like conjugate gradients only require Hessian-vector products at each iteration and not the Hessian explicitly. Depending on the problem, it is possible to leverage underlying structure to compute Hessian-vector products efficiently. In fact, there are problems where computing Hessian-vector products is efficient, while computation of the Hessian is still costly. Such problem instances lead to efficient *Hessian-free* Newton methods, where the Hessian is never computed explicitly and only efficient Hessian-vector products are computed. We will see more about efficient Hessian-vector products in section 4.11. In our Newton method for MAP inference, we do not take a Hessian-free approach. Instead, we exploit problem structure to compute the sparse Hessian efficiently and to achieve efficient Hessian-vector products, as we will see in section 5.1.1.

In summary, iterative methods allow truncation of the iterations through which an approximate solution could be computed and these methods readily exploit efficient Hessian-vector products. Thus we will focus on iterative linear system solvers in this thesis. Among various iterative methods, three classic ones are Jacobi, Gauss-Seidel and successive over-relaxation (SOR). However, they have poor convergence properties and restricted applicability. This is in contrast to a class of iterative methods called *Krylov subspace* methods. In particular, the conjugate gradient method (CG) will be of interest to us. For more information on iterative methods, one could refer to [Barrett 1994, Saad 2003].

4.2 The Conjugate Gradient Method

The conjugate gradient method is an iterative method for solving linear systems of the form shown in equation (4.6). Here, we will focus on matrix $\overline{B} \in \mathcal{R}^{N \times N}$ being sparse, symmetric and positive definite. The linear system (4.6) was obtained by setting the gradient of (4.5) to zero. Note that for a symmetric positive definite $\overline{B}$, (4.6) and (4.5) have an unique solution. The objective function in (4.5) is called a *quadratic form*. We would like to point out that the interpretation as minimization of a quadratic form gives an intuitive understanding of how the conjugate gradient method works. For the sake of visualization, let us imagine a two dimensional system.

Then, for A being symmetric positive definite, the quadratic form is a convex bowl, with elliptic level-sets. Keeping this picture in mind is useful, when we discuss about preconditioned CG.

Now, we will briefly describe some concepts, before presenting the algorithmic steps of CG. Apart from the books on iterative linear solvers mentioned earlier, [Demmel 1997, Shewchuk 1994] are excellent resources to understand more about the CG method. Within CG, an important quantity called the *residual* is computed and used both algorithmically and for the truncation of the iterations that we mentioned in the previous section. The residual is defined as follows and it is equal to the gradient of the quadratic form,

$$\boldsymbol{r}(\boldsymbol{p}_k) = \overline{B}\boldsymbol{p}_k + \nabla\overline{h}. \tag{4.10}$$

For better notation, at iteration k of CG, we will refer to the residual as $\boldsymbol{r}_k$ and the intermediate solution as $\boldsymbol{p}_k$. Also, the initial value $\boldsymbol{x}_0$ is specified by the user. Note, we will denote Newton iterations with a superscript t and CG iterations with a subscript k.

For a set of nonzero vectors $\boldsymbol{d}_0, \boldsymbol{d}_1, ..., \boldsymbol{d}_l$ and a symmetric positive definite matrix $\overline{B}$, if the following property holds

$$\boldsymbol{d}_i^\top \overline{B}\boldsymbol{d}_j = 0 \quad \forall i \neq j \tag{4.11}$$

then those vectors are said to be *conjugate* with respect to $\overline{B}$. Such a set of vectors is also linearly independent. For N variables, it will be desirable to have algorithms that can converge to the solution in atmost N iterations. It can be shown that it is indeed possible, if the steps are conjugate with respect to $\overline{B}$, thus leading to *conjugate directions* algorithms. The step taken by a conjugate directions algorithm is of the following form,

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \alpha_k \boldsymbol{d}_k \quad \text{where} \quad \alpha_k = -\frac{\boldsymbol{r}_k^\top \boldsymbol{d}_k}{\boldsymbol{d}_k^\top \overline{B}\boldsymbol{d}_k}. \tag{4.12}$$

Note that α_k is chosen to be the step length that minimizes the quadratic form in (4.5) along the direction $\boldsymbol{d}_k$. The residual $\boldsymbol{r}_k$ has the following properties,

$$\begin{aligned} \boldsymbol{r}_k^\top \boldsymbol{d}_i &= 0 \quad \forall\, i = 0, 1, ..., k-1 \\ \boldsymbol{r}_{k+1} &= \boldsymbol{r}_k + \alpha_k \overline{B}\boldsymbol{d}_k. \end{aligned} \tag{4.13}$$

In words, the residual at the current iteration is orthogonal to all previous steps of the algorithm. The second relationship could be derived from (4.12) and (4.10), the definition of the residual. Now, comes the important question of how to choose the conjugate directions. The conjugate gradient method presents an approach that is efficient in terms of memory and computation. The current direction $\boldsymbol{d}_k$ is chosen to be a linear combination of the current residual $\boldsymbol{r}_k$ and the direction $\boldsymbol{d}_{k-1}$ of the previous iteration, as follows,

$$\boldsymbol{d}_k = -\boldsymbol{r}_k + \beta_k \boldsymbol{d}_{k-1} \quad \text{where} \quad \beta_k = \frac{\boldsymbol{r}_k^\top \overline{B}\boldsymbol{d}_{k-1}}{\boldsymbol{d}_{k-1}^\top \overline{B}\boldsymbol{d}_{k-1}}. \tag{4.14}$$

CG chooses $\boldsymbol{d}_0$ to be the steepest descent direction of the quadratic form at $\boldsymbol{p}_0$. This is essential for proving that the algorithm converges. Since, the residual at a point is the gradient of the quadratic form, we have $\boldsymbol{d}_0 = -\boldsymbol{r}_0$. Thus the conjugate gradient method takes the form shown in algorithm (3). The update equations for α_k and β_k could be derived from (4.12), (4.13) and (4.14). This is done so that we need to perform only one matrix-vector product ($\overline{B}\boldsymbol{d}_k$) in each iteration. This matrix-vector product is the costliest step within CG and leveraging the sparsity of $\overline{B}$ is one possible way to achieve more efficiency. Apart from this matrix-vector product, algorithm (3) involves a few dot products of vectors.

Algorithm 3 The conjugate gradient method

1: Set $\boldsymbol{r}_0 \leftarrow \overline{B}\boldsymbol{p}_0 + \nabla\overline{h}$, $\boldsymbol{d}_0 \leftarrow -\boldsymbol{r}_0$, $k \leftarrow 0$, $\varepsilon > 0$
2: **while** $\boldsymbol{r}_k > \varepsilon$ **do**
3: $\quad \alpha_k \leftarrow \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{d}_k^T \overline{B} \boldsymbol{d}_k}$
4: $\quad \boldsymbol{p}_{k+1} \leftarrow \boldsymbol{p}_k + \alpha_k \boldsymbol{d}_k$
5: $\quad \boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k + \alpha_k \overline{B}\boldsymbol{d}_k$
6: $\quad \beta_{k+1} \leftarrow \frac{\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^\top \boldsymbol{r}_k}$
7: $\quad \boldsymbol{d}_{k+1} \leftarrow -\boldsymbol{r}_{k+1} + \beta_{k+1}\boldsymbol{d}_k$
8: $\quad k \leftarrow k+1$
9: **end while**

One important property of CG is that all the residuals and the search directions till iteration k are contained within the *Krylov subspace* of degree k with respect to $\boldsymbol{r}_0$, which is defined as,

$$\mathcal{K}(\boldsymbol{r}_0; k) \triangleq \text{span}(\boldsymbol{r}_0, \overline{B}\boldsymbol{r}_0, ..., \overline{B}^k \boldsymbol{r}_0). \tag{4.15}$$

The Krylov subspace representation is critical to analyse the convergence rate of CG. Now, the directions $\{\boldsymbol{d}_0, \boldsymbol{d}_1, ..., \boldsymbol{d}_k\}$ are independent by definition and span a subspace. Also, it can be shown that $\boldsymbol{r}_k^\top \boldsymbol{r}_i = 0$, $\forall$i = 0, 1,..., k-1, i.e., the residuals $\{\boldsymbol{r}_0, \boldsymbol{r}_1, ..., \boldsymbol{r}_k\}$ are orthogonal to each other and hence, span a subspace. Note, choosing $\boldsymbol{d}_0 = -\boldsymbol{r}_0$ is necessary to show that the residuals are orthogonal. It can be shown that these subspaces are the same,

$$\mathcal{K}(\boldsymbol{r}_0; k) = \text{span}(\boldsymbol{d}_0, \boldsymbol{d}_1, ..., \boldsymbol{d}_k) = \text{span}(\boldsymbol{r}_0, \boldsymbol{r}_1, ..., \boldsymbol{r}_k). \tag{4.16}$$

Note, this is a property of Krylov subspace methods in general. Among Krylov subspace methods, the advantage of CG is that it minimizes the distance to the solution after each step. The distance measure is the weighted norm, $\|\boldsymbol{x}\|_{\overline{B}}^2 = \boldsymbol{x}^\top \overline{B}\boldsymbol{x}$. So far, the only guarantee regarding convergence rate is that CG will converge in at most N steps. This is not a very useful result when N is large, which is the case for us. We will see in the next section that the CG method could be made to converge in much lesser number of iterations.

Before we end this section, here are some interesting historical notes about CG. It was first presented as a direct method in 1952 [Hestenes 1952]. The method

did not gain acceptance due to loss of conjugacy with finite precision arithmetic computers, leading to lack of convergence within n iterations. Later, its usefulness was recognized as an iterative method [Reid 1971]. Even now, for small problems this sensitivity to numerical precision is an issue and direct methods are preferred.

4.3 Convergence of CG and Preconditioning

In this section, we will see what factors influence the convergence rate of the conjugate gradient method and what can be done to improve it. We will present the *preconditioned* conjugate gradient method that considerably improves convergence rate by using a preconditioner.

A bound on the convergence rate of CG could be derived in terms of the condition number of the matrix $\overline{B}$. Let's recollect that the condition number is defined as, $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ where λ_{min} and λ_{max} are the minimum and maximum eigenvalues of $\overline{B}$, respectively. The convergence rate of CG satisfies the following bound,

$$\frac{|\boldsymbol{p}_k - \boldsymbol{p}^*|_{\overline{B}}}{|\boldsymbol{p}_0 - \boldsymbol{p}^*|_{\overline{B}}} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k. \tag{4.17}$$

Visually, we will imagine a two dimensional system. If $\kappa \gg 1$, that means the quadratic form is a long and narrow bowl. The closer the bowl is to a hemisphere, the closer κ is to 1. This bound based on the condition number is still not strong enough. More than the condition number, the distribution of the eigenvalues of $\overline{B}$ has a provably stronger influence on the convergence rate of CG. The following theorem [Nocedal 2006, section 5.1] illustrates this influence.

Theorem 5 (Eigenvalue distribution of $\overline{B}$)**.** *If $\overline{B}$ has only r distinct eigenvalues, then* CG *will reach the solution in at most r iterations. In fact, if the eigenvalues occur in r distinct clusters,* CG *in general will achieve an approximate solution in around r steps.*

We point out that the design of CG such that the directions and residuals till iteration k are restricted to the Krylov subspace $\mathcal{K}(\boldsymbol{r}_0; k)$ is necessary to prove these bounds. How does one use this understanding about the influence of the condition number and/or the eigenvalue distribution to speed-up the convergence of CG? We somehow need a new linear system which has favourable properties. Say, we could modify $\overline{B}$ to $\hat{B} = M^{-1}\overline{B}$, where M by construction is symmetric positive definite. Then we could work with the following linear system,

$$M^{-1}\overline{B}\boldsymbol{p} = -M^{-1}\nabla\overline{h}. \tag{4.18}$$

For example, choosing $M = \overline{B}$, leads to $\hat{B} = I$, for which all eigenvalues are equal to 1 but it involves the computation of A^{-1}. However, avoiding the explicit computation of $\overline{B}^{-1}$ is a fundamental motivation of linear system solvers. We need an easy to invert M, such that $M^{-1}\overline{B}$ has low condition number and/or well clustered eigenvalues. There is one important issue that we need to address: is

$M^{-1}\overline{B}$ symmetric positive definite? It is not necessary. To circumvent this problem, instead of thinking in terms of transforming $\overline{B}$, we will think in terms of transforming $\boldsymbol{p}$ as follows,

$$\hat{\boldsymbol{p}} = C\boldsymbol{p} \quad \Longrightarrow \quad \boldsymbol{p} = C^{-1}\hat{\boldsymbol{p}}. \tag{4.19}$$

Substituting this in (4.5), gives us the quadratic form: $\frac{1}{2}\hat{\boldsymbol{p}}^\top(C^{-\top}\overline{B}C^{-1})\hat{\boldsymbol{p}}+(C^{-\top}b)^\top\hat{\boldsymbol{p}}$. Minimizing this quadratic form is equivalent to solving the following linear system,

$$(C^{-\top}\overline{B}C^{-1})\hat{\boldsymbol{p}} = -C^{-\top}\nabla\overline{h}. \tag{4.20}$$

Note that $C^{-\top}\overline{B}C^{-1}$ is symmetric positive definite for any C, as long as $\overline{B}$ is symmetric positive definite. Suppose, we define $M \triangleq C^\top C$, then M is symmetric positive definite for any C. Remarkably, for this choice of M, the eigenvalues of $M^{-1}\overline{B}$ and $C^{-\top}\overline{B}C^{-1}$ are the same. However, the eigenvectors differ by a linear transformation: if v is the eigenvector of $M^{-1}\overline{B}$ corresponding to the eigenvalue λ, then Cv is the eigenvector of $C^{-\top}\overline{B}C^{-1}$ for the same λ. Thus, we will work with the transformed linear system (4.20), rather than (4.18). Apart from leading to a symmetric positive definite linear system, this approach also leads to an algorithm, that is a simple modification of algorithm (3). The *preconditioned conjugate gradient* algorithm is presented in algorithm (4).

Algorithm 4 The preconditioned conjugate gradient method

Set $\boldsymbol{r}_0 \leftarrow \overline{B}\boldsymbol{p}_0 + \nabla\overline{h}$
Solve $M\boldsymbol{y}_0 = \boldsymbol{r}_0$ for $\boldsymbol{y}_0$
Set $\boldsymbol{d}_0 \leftarrow -\boldsymbol{y}_0$, $k \leftarrow 0$, $\varepsilon > 0$
while $\|\boldsymbol{r}_k\| \geq \varepsilon$ **do**
    $\alpha_k \leftarrow \frac{\boldsymbol{r}_k^T\boldsymbol{y}_k}{\boldsymbol{d}_k^T\overline{B}\boldsymbol{d}_k}$
    $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{p}_k + \alpha_k\boldsymbol{d}_k$
    $\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k + \alpha_k\overline{B}\boldsymbol{d}_k$
    Solve $M\boldsymbol{y}_{k+1} = \boldsymbol{r}_{k+1}$
    $\beta_{k+1} \leftarrow \frac{\boldsymbol{r}_{k+1}^\top\boldsymbol{y}_{k+1}}{\boldsymbol{r}_k^\top\boldsymbol{y}_k}$
    $\boldsymbol{d}_{k+1} \leftarrow -\boldsymbol{y}_{k+1} + \beta_{k+1}\boldsymbol{d}_k$
    $k \leftarrow k+1$
end while

Just like algorithm (3), algorithm (4) also computes the same matrix-vector product $\overline{B}\boldsymbol{p}_k$ and similar number of dot products of vectors in each iteration. On top of these computations, algorithm (4) also involves the solution of a linear system $M\boldsymbol{y}_{k+1} = \boldsymbol{r}_{k+1}$ in each iteration. It is essential to solve this linear system efficiently. We should keep in mind that solving $M\boldsymbol{y}_{k+1} = \boldsymbol{r}_{k+1}$ in each iteration should not outweigh the reduction in number of iterations. Usually, this means an easy to invert M. We refer to this matrix M as the *preconditioner*.

Depending on the problem and the preconditioning technique, it may be easier to reduce the condition number of the linear system or it may be easier to get well

clustered eigenvalues. Also, we have already seen that the other computationally heavy step in CG is the matrix-vector product. We will see in section (4.11) ways to achieve efficient matrix-vector products. We note that preconditioning and Hessian-vector products are two algorithmic steps within Newton methods, where one can exploit problem structure and propose new ideas.

We would like to recall that the original dual formulation (2.25) is not smooth. We will see in the next chapter, that even though we are optimizing a smooth approximation (3.5), we have to reduce the smoothing as the algorithm proceeds. This leads to a better approximation of the original non-smooth problem and more accurate results. [Rockafellar 2009, §9] shows that any locally Lipschitz non-smooth function can be viewed as a limit of increasingly ill-conditioned differentiable functions. Thus, even though we will get closer to the non-smooth problem by increasing τ in (3.5), we have to optimize ill-conditioned problems. Thus, preconditioning is an important problem to address for MAP inference through LP relaxation. We will see about preconditioners that we experimented with in section (4.10). Immediately, we will see more about how CG fits into Newton methods.

4.4 Truncated Newton methods

While solving the linear sytem in order to obtain the Newton direction, it is not necessary to obtain the exact solution of the linear system in order to ensure the proper working of the Newton method. Let us recall that Newton methods approximate the landscape at a point by a quadratic form and the solution to the linear system points to the minimizer of that quadratic form. A step towards this minimizer is taken in Newton methods while ensuring convergence through globalization strategies like line-search or trust-region. What is necessary and sufficient for Newton methods to converge is to choose an inexact direction that achieves *sufficient decrease* in each iteration. We emphasize that it is necessary to achieve enough decrease in each iteration for the Newton method to converge, when it starts from a faraway point. Otherwise, it will not converge. We have observed this empirically and describe our experience in section 5.1.3. At the same time, inexact computation of Newton direction is essential to make Newton methods practical and iterative methods like conjugate gradients are readily suited for such *truncated* Newton methods [Dembo 1982, Nash 2001]. Generally, the terms "inexact" and "truncated" are used exchangeably. We prefer "truncated" because the approximate solution of the linear system is achieved by truncating the CG iterations.

4.4.1 Truncating CG iterations

While optimizing the smooth dual (3.5) by a Newton method, the linear system that is being solved is given in equation (4.6). Suppose, we are at t^{th} of the Newton method and CG is at k^{th} iteration. Generally, CG iterations are terminated based on the residual,

$$\boldsymbol{r}_k = \overline{B}\boldsymbol{p}_k + \nabla\bar{h}. \tag{4.21}$$

One possible termination rule to stop CG iterations could be based on the following condition,

$$||\boldsymbol{r}_k|| \leq \eta^t ||\nabla \bar{h}|| \tag{4.22}$$

and the sequence $\{\eta^t\}$ is called the *forcing sequence*. The p_k corresponding to CG termination is returned as the Newton direction. The forcing sequence has to be chosen properly to reduce computational cost and to ensure convergence. If η^t is very loose, the truncated Newton method may not converge. Truncated Newton methods are notorious for their sensitivity to the forcing sequence. Thus, there is a delicate trade-off between the quality of the step and the cost of computing it [Schlick 1987, Eisenstat 1996, Morales 2002]. A popular choice for the forcing sequence is as follows [Nocedal 2006, §7.1],

$$\eta^t = \min\Big(0.5, \sqrt{||\nabla \bar{h}||}\Big). \tag{4.23}$$

We would like to point out that the conjugate gradient method does not monotonically reduce its residuals. Thus, a truncation rule based on the residual may not be the right choice. If we are particular about reducing the residual, then we may choose the MINRES solver which is designed to monotonically reduce the solver [Fong 2011]. However, we did not go further with this path and used CG in our experiments. We would like to point out that page 116 of [Fong 2011] has a nice flowchart about choosing linear system solvers under various conditions. [Martens 2010] presents a stopping criterion based on the relative decrease of the quadratic objective $q(\boldsymbol{p})$ given in (4.4). This is because CG decreases the quadratic form monotonically. We will see more about the truncation rule in our experiments with MAP inference in section (5.1.3).

4.5 Line search Newton

Now, we will start addressing the important question of how to set the step length α in Newton iterations (4.8). Usually, it is set by a globalization strategy and line search is one such strategy. In fact, as long as the optimization problem is well conditioned, a truncated Newton method with line search is an effective approach and we need not consider other globalization strategies. We will see what to do when the problem is ill-conditioned in the next section. For now, we will understand one efficient approach to do line search: *polynomial interpolation based backtracking*.

In every Newton iteration, CG will return a Newton direction p. A line search procedure seeks to fix the step length α along this direction. Generally, two conditions are desired from the step obtained by a line search procedure: decrease in function value and step length which is not too short. The first condition is called the *sufficient decrease* condition and is enforced by seeking a step length α that satisfies the following condition,

$$h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t) \leq h(\boldsymbol{\delta}^t) + c_1 \alpha \nabla h(\boldsymbol{\delta}^t)^\top \boldsymbol{p}^t. \tag{4.24}$$

The above condition is also called the *Armijo condition*. Achieving sufficient decrease is not enough. We also need to avoid short step lengths and it could be achieved by the following *curvature condition*,

$$\nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t)^\top \boldsymbol{p}^t \geq c_2 \nabla h(\boldsymbol{\delta}^t)^\top \boldsymbol{p}^t. \tag{4.25}$$

This condition ensures that the *magnitude* of the slope at $\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t$ is sufficiently less than the magnitude at $\boldsymbol{\delta}^t$, i.e., the function will reach a "flatter" place. Together these conditions are called the *Wolfe conditions*. An efficient algorithm to satisfy the Wolfe conditions is *backtracking* line search (5).

Algorithm 5 Backtracking line search

1: Choose: $c \in (0,1)$, $\beta \in (0,1)$, $\alpha_{init} > 0$. Set $\alpha \leftarrow \alpha_{init}$.
2: **while** $h(\boldsymbol{\delta}^t + \alpha \boldsymbol{\delta}^t) > h(\boldsymbol{\delta}^t) + c\alpha \nabla h(\boldsymbol{\delta}^t)$ **do**
3: $\quad \alpha \leftarrow \beta \alpha$
4: **end while**

Notice that only the sufficient decrease condition is explicitly enforced in the backtracking algorithm. The curvature condition is enforced implicitly by backtracking the step length from a sufficiently large initial value α_{init}. Generally, c is set to a low value, so that the line search algorithm exits in lesser number of iterations. An initial step length of 1 is what is usually tested for and it is critical for convergence analysis of Newton methods to set $\alpha_{init} = 1$. Larger values of β leads to finer search of the step length, while a smaller value will agressively reduce the step length. We set $c = 10^{-4}$, $\alpha_{init} = 1$ and $\beta = 0.8$ in our experiments with Newton methods.

The computational complexity of each interation of algorithm (5) is composed of a function evaluation at the new point and a vector dot product. Generally, this algorithm leads to too many iterations if β is large and very small steps if β is small. A more efficient and effective way to adjust the step length is through *polynomial interpolation*. Here, we will describe a cubic polynomial based approach, which needs both the function value and the gradient at the next candidate point. After fitting a cubic polynomial between the current point and the candidate point, the point corresponding to the minimum of the cubic polynomial is chosen as the next candidate point. Given $h(\boldsymbol{\delta}^t)$, $h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t)$, $\nabla h(\boldsymbol{\delta}^t)$, $\nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{\delta}^t)$, if $h(\boldsymbol{\delta}^t + \alpha \boldsymbol{\delta}^t) > h(\boldsymbol{\delta}^t) + c\alpha \nabla h(\boldsymbol{\delta}^t)$, then the next candidate α is estimated through algorithm (6). We observed substantial reduction in the number of line search iterations by using the polynomial interpolation approach.

Thus, we see that line search Newton moves in the direction returned by CG according to a suitable step length. The underlying assumption is that the quadratic function (4.4) is a good approximation of the landscape in each iteration. Generally, this is true only for well conditioned problems. We will see what to do in the case of ill-conditioned problems in the next section.

Algorithm 6 Polynomial interpolation based step length

1: $\alpha_{prev} \leftarrow \alpha$
2: compute: $d_1 = \nabla h(\boldsymbol{\delta}^t) + \nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t) - 3\left(\frac{h(\boldsymbol{\delta}^t) - h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t)}{\alpha}\right)$
3: **if** $d_1 > \sqrt{\nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t) \nabla h(\boldsymbol{\delta}^t)}$ **then**
4: compute: $d_2 = \sqrt{d_1^2 - \nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t) \nabla h(\boldsymbol{\delta}^t)}$
5: set: $\alpha \leftarrow \alpha - \alpha \left[\frac{\nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t) + d_2 - d_1}{\nabla h(\boldsymbol{\delta}^t + \alpha \boldsymbol{p}^t) - \nabla h(\boldsymbol{\delta}^t) + 2d_2}\right]$
6: **else**
7: set: $\alpha \leftarrow \frac{\alpha}{2}$
8: **end if**
9: Bracket α between $[0, 1] : \alpha \leftarrow \min.(\max.(\alpha, 0), \alpha_{prev})$

4.6 Trust-region Newton

When a problem is ill-conditioned, we need a globalization strategy based on a trust-region. Intuitively, we will consider the two dimensional problem shown in figure (4.2). At an ill-conditioned point, the quadratic approximation is an elongated bowl. This bowl matches the landscape around the current iterate only upto a small extent, beyond which it is not at all a surrogate for the function. Line search Newton would lead to a step that points to the bottom of the bowl. But the bottom of the bowl is at a place where the quadratic approximation is a poor surrogate of the function. Thus, this is a poor Newton direction. Performing line search along this direction, will, of course, lead to a step length that has a reduced function value. But the decrease will not be sufficient and too many Newton iterations will occur. We have observed this empirically in our MAP inference problem, when we implemented a truncated CG based line search Newton algorithm.

Another condition, which will lead to large steps is when the Hessian is not strictly positive definite. When the Hessian is strictly positive definite, we say the problem is *strongly convex*. The smooth dual (3.5) is only strictly convex [Savchynskyy 2012] and not strongly convex. A strictly convex function is guaranteed to have a unique global optimum but the Hessian will be strictly positive definite only in the neighbourhood of the optimum. Thus, at faraway points, the linear system could be rank deficient, leading to a poor approximation of the function landscape.

Thus, in an ill-conditioned problem, we impose a *trust-region* on the quadratic approximation and minimize it within this trust-region.The broad framework of trust-region methods in shown in (7). There are two major computations that we need to address. In each Newton iteration, a constrained quadratic program is solved as follows,

$$\underset{p}{\text{minimize}}\, \nabla h(\boldsymbol{\delta}^t)^T \boldsymbol{p} + \boldsymbol{p}^\top B(\boldsymbol{\delta}^t) \boldsymbol{p} \quad \text{subject to} \quad ||\boldsymbol{p}|| \leq \Delta^t. \tag{4.26}$$

Among the early algorithms to solve the trust-region subproblem (4.26) are those that are based on the Cauchy point like the dog-leg method. However, these

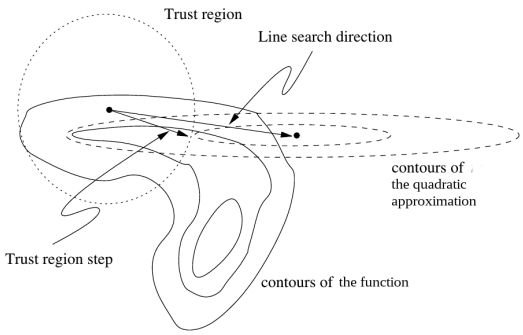

Figure 4.2: For an ill-conditioned problem, the quadratic approximation is a long bowl. *Image adapted from [Nocedal 2006, §4].*

algorithms are not scalable and we will not discuss them here. More details can be found in [Nocedal 2006, §4.1]. The second important computation is how to set the trust-region radius Δ^t at each iteration. We will discuss about this in section (4.8). Immediately, we will see one popular scalable method to solve the trust-region subproblem.

Algorithm 7 Framework for Trust-region Newton Methods

1: Initialization: initial point $\boldsymbol{\delta}^0$; initial trust-region radius.
2: Step calculation: compute a step $\boldsymbol{p}^t$ that minimizes the quadratic model within the trust-region.
3: Update trust-region radius.

4.7 The Steihaug Method

The *Steihaug* method is a scalable approach to solve the trust-region subproblem (4.26) that is widely used [Nocedal 2006]. The Steihaug method truncates the conjugate gradient iterations according to a trust-region. Thus, within a given outer iteration t of the trust-region Newton method, the CG iterates step-by-step move towards the solution of the linear system. According to the Steihaug method, apart from the forcing sequence η^t, CG can be truncated based on whether the intermediate solution computed by CG crosses a trust-region. An important advantage of the Steihaug method is that it allows the trust-region to be shaped into an ellipsoid according to a preconditioner M. This is because of considering the matrix weighted norm of the potential Newton step, while comparing it with the trust-region radius. Among machine learning researchers there is a misunderstanding that the Steihaug method cannot be preconditioned [Martens 2012, §8.6]. We would like to clarify that this is not the case. This aspect about the Steihaug method is not mentioned in standard numerical optimization textbooks like [Nocedal 2006]. You can find it in [Conn 2000, §7.5.1]. Intuitively, the trust-region subproblem given in (4.26), is

solved with a weighted norm based constraint, i.e., $||\boldsymbol{p}||_M \leq \Delta^t$, where M is the preconditioning matrix.

In algorithm 8, you can notice that the preconditioner based weighted norm (9.1.8) of the intermediate solution is compared with the trust-region radius Δ^t. Thus, the trust-region is shaped by the preconditioner and the performance of the Steihaug method critically depends on the preconditioner. For our MAP inference problem, we observed that CG reached the trust-region based exit condition in a very few iterations. Thus, we were not getting sufficient decrease in each Newton iteration. The preconditioners, Jacobi and block Jacobi, were tried with Steihaug and we did not observe improvement in performance.

Algorithm 8 Steihaug truncated CG method with preconditioning

Given tolerance $\varepsilon = \eta^t$; trust-region radius $= \Delta^t$.
Set $\boldsymbol{p}_0 = 0$, $\boldsymbol{r}_0 = \nabla \bar{h}$;
Solve $M\boldsymbol{y}_0 = \boldsymbol{r}_0$; Set $\boldsymbol{d}_0 = -\boldsymbol{r}_0$. ▷ Init. same as PCG (algorithm 4).
for j = 0, 1, 2, ... **do**
  Set $\alpha_j = \frac{\boldsymbol{r}_j^\top \boldsymbol{y}_j}{\boldsymbol{d}_j^\top \overline{B} \boldsymbol{d}_j}$;
  Set $\boldsymbol{p}_{j+1} = \boldsymbol{p}_j + \alpha_j \boldsymbol{d}_j$;
  if $||\boldsymbol{p}_{j+1}||_M \geq \Delta$ **then**
    Compute σ_j as the positive root of $||\boldsymbol{p}_j + \sigma \boldsymbol{d}_j||_M = \Delta^t$.
    Return $\boldsymbol{p}_j + \sigma_j \boldsymbol{d}_j$.
  end if
  Set $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j + \alpha_j \overline{B} \boldsymbol{d}_j$;
  if $||\boldsymbol{r}_{j+1}|| \leq \varepsilon$ **then**
    Return $\boldsymbol{p}_{j+1}$.
  end if
  Solve $M\boldsymbol{y}_{j+1} = \boldsymbol{r}_{j+1}$; ▷ Same as PCG (algorithm 4).
  Set $\beta_j = \frac{\boldsymbol{r}_{j+1}^\top \boldsymbol{y}_{j+1}}{\boldsymbol{r}_j^\top \boldsymbol{y}_j}$;
  Set $\boldsymbol{d}_{j+1} = -\boldsymbol{y}_{j+1} + \beta_j \boldsymbol{d}_j$;
end for

4.8 Damping matrix based approach

The Levenberg-Marquardt algorithm, which is generally used in nonlinear least squares problems, offers another approach to impose a trust-region. In fact, that algorithm is seen as a progenitor of trust-region methods [Nocedal 2006, §10.3]. The idea that we borrow is the damping matrix. This matrix is added to the Hessian and the resulting matrix is called the *modified* Hessian. For our problem, this damping matrix acts as a regularizer to address the lack of strong convexity and the ill-conditioning. We modify the Hessian as follows, $B(\boldsymbol{\delta}) = \nabla^2 h(\boldsymbol{\delta}) + \lambda I$, where $\lambda > 0$. While Steihaug works explicitly with a trust radius Δ, we implicitly impose

it through λ. In other words, the damping matrix restricts the Newton direction returned by CG to a trust-region. This can be seen through the following theorem that ties the damping parameter λ to a trust radius Δ.

Theorem 6 (Damping matrix based trust-region)**.** *The vector $\boldsymbol{p}^*$ is a global solution of the trust-region subproblem described in equation (4.26), if and only if $\boldsymbol{p}^*$ is feasible and there is a scalar $\lambda \geq 0$ such that the following conditions are satisfied:*

$$\begin{aligned} (\nabla^2 h(\boldsymbol{\delta}) + \lambda I)\boldsymbol{p}^* &= -\nabla h(\boldsymbol{\delta}) \\ \lambda(\Delta - \|\boldsymbol{p}^*\|) &= 0 \end{aligned} \tag{4.27}$$

Proof. Refer [Nocedal 2006, §4]. □

Thus, by suitably setting λ^t after every iteration, we implicitly enforce a trust-region on the Newton step. We found the following rule to adapt λ^t to be suitable for us [Moré 1983],

$$\begin{gathered} \rho < 0.25 : \lambda^{t+1} \leftarrow 2\lambda^t; \;\; 0.25 < \rho < 0.5 : \lambda^{t+1} \leftarrow \lambda^t \\ 0.5 < \rho < 0.9 : \lambda^{t+1} \leftarrow 0.5\lambda^t; \;\; 0.9 < \rho : \lambda^{t+1} \leftarrow 0.25\lambda^t \\ \text{where,} \quad \rho = \frac{h(\boldsymbol{\delta}^t + \boldsymbol{p}^t) - h(\boldsymbol{\delta}^t)}{q(\boldsymbol{p}^t) - q(\mathbf{0})}. \end{gathered} \tag{4.28}$$

ρ signifies how well the quadratic form (4.4) approximates the dual (3.5). As the algorithm approaches the optimum, we are reaching a sufficiently positive definite region. The quadratic form will approximate the function landscape better and λ^t will be reduced. In fact, λ^t becomes vanishingly small and the algorithm reaches the true optimum without any perturbation. According to (4.27), whenever λ^t is close to zero, the Newton direction $\boldsymbol{p}^*$ is computed with the true Hessian. Also, if $\lambda^t > 0$ then $\|\boldsymbol{p}^*\| = \Delta$, i.e., the direction is restricted by the trust-region. In ill-conditioned regions, λ will be large and the enforced trust radius will be small.

A similar approach works for adapting Δ^t in the Steihaug method. We have observed excellent empirical performance with the damping matrix based approach and our results are based on this approach to enforce a trust-region.

4.9 Quasi-Newton methods

A Newton method approximates the landscape of a problem through a quadratic form $q(\boldsymbol{p})$ given in (4.4). Now, $\overline{B}$ need not be the true Hessian $\nabla^2 g(\boldsymbol{\delta}^t)$ at $\boldsymbol{\delta}^t$. Quasi-Newton methods estimate $B(\boldsymbol{\delta}^t)$ ($\overline{B}$ in short) in a principled manner. This can be advantageous in problems where computing the true Hessian is expensive, which is one of the primary impediments to applying Newton methods to problems.

In order to estimate $\overline{B}$, we could impose a first order condition on $q(\boldsymbol{p})$ that the gradient of $q(\boldsymbol{p})$ at the current iterate and previous iterate should match the gradients of h at these points. The current iterate, corresponds to $\boldsymbol{p} = 0$ and $\nabla q(0) = \nabla \overline{h}$ by construction. Suppose, the Newton direction was $\boldsymbol{p}^{t-1}$ in the previous iteration and

we took a Newton step of $\alpha^{t-1}\boldsymbol{p}^{t-1}$, where α^{t-1} is determined by a globalisation procedure like line search. Matching the gradient at the previous iteration means we want,

$$\nabla q(-\alpha^{t-1}\boldsymbol{p}^{t-1}) = \nabla h(\boldsymbol{\delta}^t) - \alpha^{t-1}\overline{B}\boldsymbol{p}^{t-1} = \nabla h(\boldsymbol{\delta}^{t-1}). \tag{4.29}$$

Re-arranging the terms, we obtain the *secant* condition,

$$\overline{B}\alpha^{t-1}\boldsymbol{p}^{t-1} = \nabla h(\boldsymbol{\delta}^t) - \nabla h(\boldsymbol{\delta}^{t-1}). \tag{4.30}$$

The difference of iterates ($\boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1} = \alpha^{t-1}\boldsymbol{p}^{t-1}$) and the difference of gradients that we observe in the secant equation are important quantities in quasi-Newton methods and we will define separate variables to denote them,

$$\boldsymbol{s}^t = \boldsymbol{\delta}^{t+1} - \boldsymbol{\delta}^t = \alpha^t\boldsymbol{p}^t, \quad \boldsymbol{y}^k = \nabla h(\boldsymbol{\delta}^{t+1}) - \nabla h(\boldsymbol{\delta}^t). \tag{4.31}$$

We need to impose further conditions on $\overline{B}$ in order to get a unique estimate. Typically, what is done is the explicit imposition of the condition that it should be symmetric and also its difference with respect to the previous iterate $B(\boldsymbol{\delta}^{t-1})$ is minimized according to some matrix norm. The choice of this norm determines the form of the update equation for $\overline{B}$. Among quasi-Newton methods, the BFGS method is considered to be the most effective and the BFGS update equation is of the following form,

$$B(\boldsymbol{\delta}^{t+1}) = B(\boldsymbol{\delta}^t) - \frac{B(\boldsymbol{\delta}^t)\boldsymbol{s}^t\boldsymbol{s}^{t\top}B(\boldsymbol{\delta}^t)}{\boldsymbol{s}^{t\top}B(\boldsymbol{\delta}^t)\boldsymbol{s}^t} + \frac{\boldsymbol{y}^t\boldsymbol{y}^{t\top}}{\boldsymbol{y}^{t\top}\boldsymbol{s}^t}. \tag{4.32}$$

Suppose, the BFGS updates start with a symmetric and positive definite matrix (B_0), an important aspect of the BFGS update formula is that it preserves positive definiteness if the following *curvature* condition is satisfied [Nocedal 2006, §6.1],

$$\boldsymbol{s}^{t\top}\boldsymbol{y}^t > 0 \;\; \forall\, t. \tag{4.33}$$

Whenever, the objective function of the optimization problem is strongly convex, this condition is satisfied for any two points $\boldsymbol{\delta}^t$ and $\boldsymbol{\delta}^{t+1}$. Remarkably, as long as we enforce a line search procedure satisfying the Wolfe conditions, (4.24) and (4.25), we can ensure (4.33).

We should note that quasi-Newton methods are not limited to approximating the Hessian. In fact, the BFGS method was derived by aiming for an approximation of the Hessian inverse, which we will denote as H^t. By using the Sherman-Morrison-Woodbury formula, the BFGS update equation for the Hessian (4.32) could be derived from the following update equation for the Hessian inverse,

$$H^{t+1} = (I - \rho^t\boldsymbol{s}^t\boldsymbol{y}^{t\top})H^t(I - \rho^t\boldsymbol{s}^t\boldsymbol{y}^{t\top}) + \rho^t\boldsymbol{s}^t\boldsymbol{s}^{t\top} \tag{4.34}$$

where, $\rho^t = \frac{1}{\boldsymbol{y}^{t\top}\boldsymbol{s}^t}$. We will be working with the Hessian approximation based method because the smooth dual is ill-conditioned in nature. Thus, working with the Hessian approximation within a trust-region framework is what is suitable for us.

The Hessian approximation $\overline{B}$ is generally dense and it is not suitable to work with for large scale problems. When we look at the BFGS update equation in (4.32) carefully, we notice that it carries the effect of all $(\boldsymbol{s}^t, \boldsymbol{y}^t)$ pairs starting from the initial $(\boldsymbol{s}^0, \boldsymbol{y}^0)$ pair. Intuitively, to estimate the curvature information at a given iterate, only the recent $(\boldsymbol{s}^t, \boldsymbol{y}^t)$ pairs will be most informative. The older iterates are far away and less informative. It is this intuition that led to L-BFGS, the limited memory BFGS method. In this method, only the m recent $(\boldsymbol{s}^t, \boldsymbol{y}^t)$ pairs are retained as follows,

$$S^t = [\boldsymbol{s}^{t-m}, ..., \boldsymbol{s}^{t-1}], \quad Y^t = [\boldsymbol{y}^{t-m}, ..., \boldsymbol{y}^{t-1}] \tag{4.35}$$

where $S^t \in \mathcal{R}^{N\times m}$, $Y^t \in \mathcal{R}^{N\times m}$ and $L^t \in \mathcal{R}^{m\times m}$. Given this memory of $(\boldsymbol{s}^t, \boldsymbol{y}^t)$ vectors, the L-BFGS update takes the following form,

$$\begin{aligned} B(\boldsymbol{\delta}^t) &= \sigma^t I - [\sigma^t S^t Y^t] \begin{bmatrix} \sigma^t S^{t\top} S^t & L^t \\ L^{t\top} & -D^t \end{bmatrix}^{-1} \begin{bmatrix} \sigma^t S^{t\top} \\ Y^{t\top} \end{bmatrix} \\ \text{where } \; D^t &= \text{diag}[\boldsymbol{s}^{t-m\top}\boldsymbol{y}^{t-m}, ..., \boldsymbol{s}^{t-1\top}\boldsymbol{y}^{t-1}] \\ L^t_{i,j} &= \begin{cases} \boldsymbol{s}^{t-m-1+i\top}\boldsymbol{y}^{t-m-1+j} & \text{if } \; i > j \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.36}$$

Note that the above update equation is from [Byrd 1994] and it is a compact way of representing the Hessian approximation. The important aspect about this representation is that it leads to efficient matrix-vector products.

Now, the Hessian approximation that we are considering is of the form of an identity + rank-r matrix. An interesting property of the conjugate gradient method is that it converges in $O(r)$ iterations for such a linear system [Golub 2012, §11.3.4]. The time taken by CG will reduce significantly when it converges within such limited number of iterations. In fact, we need not seek preconditioners to reduce the number of iterations.

Thus, with these two aspects: 1. ability to compute efficient matrix-vector products, 2. convergence of CG within a small number of iterations, it will be interesting to see whether the L-BFGS method will be competitive with Nesterov's accelerated gradient method. We will see experimental results regarding the same in section (5.3.1). In the quasi-Newton method that we have described so far, we have not exploited problem structure in any way. In chapter (6), we describe a particular structure called *partial separability* present in the LP relaxation for MAP inference and how quasi-Newton methods could exploit it.

4.10 Preconditioning

Now, we will discuss preconditioning methods for improving the convergence of CG iterations. We will present the methods that were studied for Newton methods for MAP inference.

4.10.1 Diagonal preconditioner

The simplest idea that one could try is to take the main diagonal of $B(\boldsymbol{\delta})$ as the preconditioner. In this case, M^{-1} is another diagonal matrix with just the inverse of the diagonal elements of $B(\boldsymbol{\delta})$. Thus, we only need to scale the elements of $\boldsymbol{r}_k$ in each iteration of algorithm (4). This preconditioner goes by the name, *Jacobi preconditioner*. This is because just like the iterative Jacobi method for solving a linear system, this preconditioner scales each element parallelly. The diagonal preconditioner is especially useful if the linear system matrix has widely varying elements along its diagonal. This is not the case with the true Hessian of the smooth dual. Nevertheless, we observed better performance with the diagonal preconditioner than solving CG without any preconditioning.

4.10.2 Block diagonal preconditioner

In many real-world problems, the Hessian will be composed of meaningful blocks of values. In other words, it will be a block matrix. In such a case, it is a good idea to consider the blocks along the main diagonal as the preconditioner. Thus, we have a block diagonal preconditioner, which is called *block Jacobi*. For small enough blocks computing M^{-1} is not computationally intensive. Individual blocks need to be inverted and could be done so in parallel. Block Jacobi is guaranteed to minimize the condition number to a greater extent than Jacobi [Demmel 1997, §6.6.2]. We found considerable speed-up with a block diagonal preconditioner, where each block corresponds to a clique.

4.10.3 Incomplete Cholesky Preconditioner

Any symmetric positive definite matrix (including $B(\boldsymbol{\delta})$ could be factorized as $LL^\top$, where L is a lower triangular matrix. Even if $B(\boldsymbol{\delta})$ is sparse, L will not be sparse. It will have *fill-in*. Suppose, we obtain a factor $\widetilde{L}$ that has a sparsity pattern that resembles $B(\boldsymbol{\delta})$, then $\widetilde{L}\widetilde{L}^\top \approx B(\boldsymbol{\delta})$. Thus, we choose the preconditioner $M = \widetilde{L}\widetilde{L}^\top$, as our incomplete Cholesky preconditioner. Solving the linear system $Mr = y$ is through the solution of two triangular systems. Since, $\widetilde{L}$ is sparse like $B(\boldsymbol{\delta})$, the computational cost is similar to a single matrix-vector multiplication involving $B(\boldsymbol{\delta})$. This implies that $C = \widetilde{L}$ in equation (4.20). However, one needs to choose the elements that are in $\widetilde{L}$ carefully, otherwise, $\widetilde{L}^{-\top}A\widetilde{L}^{-1}$ may not be sufficiently positive definite. [Jones 1995] proposed some ideas. They compute the exact Cholesky factor L and then they decide the sparsity pattern of $\widetilde{L}$ based on the entries of $B(\boldsymbol{\delta})$. The kth column of $\widetilde{L}$ only has the n_k largest elements in the strict lower triangular part of the kth column of L, where n_k is the number of elements in the kth column of the strict lower triangular part of A. [Lin 1999] has improved their approach further and we found the implementation at https://xueyuhanlang.github.io/software/HLBFGS/. Incomplete Cholesky is a generic preconditioner, which may or may not benefit a particular problem. We found it to perform poorer than our block diagonal preconditioner.

4.10.4 Multigrid Preconditioner

Multigrid linear system solvers are especially suitable for problems that have a geometric structure to them. Many problems in physics and image processing have benefited from them. The essential idea of these solvers is to create a heirarchy of fine to coarse problems. The solver goes through multiple V-cycles, each of which involves a fine to coarse and a coarse to fine computation. In fine to coarse phase, each level receives a coarser version of the residual from the finer level above and a simple solver (called a smoother) like Gauss-Seidel is run for a few iterations. At the coarsest level, a small linear system is solved. In the coarse to fine phase, each level receives an interpolated version of the residual from the coarser level below and again a smoother is run for a few iterations. Geometric multigrid builds the coarser levels directly based on the geometry of the problem. For example, a grid graph could be coarsened to a smaller grid graph by merging neighbours together. Algebraic multigrid builds the coarser levels based on the values present in the system matrix. These methods are well studied and we found [Stüben 1999] to be a well written introduction to this topic.

Within the preconditioned conjugate gradient method, we would like to compute $M^{-1}r_k$ at the preconditioning step. Instead, a single V-cycle of a multigrid method could be applied on the residual r_k. This is how a multigrid method is used as a preconditioner within CG. We experimented with the BoomerAMG libary [Henson 2002]. We observed poorer performance with respect to the block diagonal preconditioner. One possible reason could be as given in [Stüben 1999, §10]: ... *if the given matrix is symmetric positive definite, contains relatively large positive off-diagonal entries, and is far from being weakly diagonally dominant. In such cases, the performance of classical AMG may be only suboptimal.* The matrix $B(\boldsymbol{\delta})$ in MAP inference is not even weakly diagonally dominant and it has several positive off-diagonal entries.

The multilevel ILU preconditioner [Saad 2005] will work with more general matrices. Whether multilevel ILU could lead to faster convergence with our Hessian needs to be investigated.

4.10.5 Quasi-Newton preconditioner

In a Newton method, the linear systems from adjacent outer iterations are somewhat similar. We call such linear systems as slowly varying. In such a scenario, one could assemble the vectors s_k and y_k that are needed for a quasi-Newton method (refer section (4.9)) from the conjugate gradient method of the current iteration and use them to approximate the Hessian inverse for the next iteration. This approximate Hessian inverse could be used as the desired preconditioner within CG in the next outer iteration. This is described in [Morales 2000] and the code is available at: http://users.iems.northwestern.edu/ nocedal/preqn.html. We are not able to fully understand the poor performance of this preconditioner for our problem.

4.10.6 Combinatorial preconditioner

Combinatorial preconditioners (also, called tree based preconditioners) have led to state-of-the-art results for laplacian matrices and diagonally dominant matrices, in general [Koutis 2011, Bern 2006]. It may be possible to extend these techniques to general symmetric positive definite matrices using a concept called *support theory* [Boman 2003]. It will be an interesting future exercise for our problem.

4.11 Hessian-vector products

We saw in section (4.2), that one of the computationally costly steps in the conjugate gradient method is the Hessian-vector product. In the worst case, it is of complexity $O(n^2)$, where n is the number of decision variables. Since, CG only requires Hessian-vector products, it is possible to design *Hessian-free* Newton methods. They are especially successful in training neural networks [Pearlmutter 1994, Martens 2010]. [Knoll 2004] is an excellent introductory resource on Hessian-free methods. Suppose, we have some function f and we represent its Hessian at some point $\boldsymbol{x}$ as B. Then, the product of b with respect to some vector $\boldsymbol{y}$ could be approximated through numerical differentiation based on a small step length μ as follows,

$$B\boldsymbol{y} \approx \frac{\nabla f(\boldsymbol{x} + \mu \boldsymbol{y}) - \nabla f(\boldsymbol{x})}{\mu}. \tag{4.37}$$

However, this simple procedure is affected by cancellation errors. One general way to obtain very accurate Hessian-vector products is through *complex step differentiation* [CSD]. The Taylor series expansion for $\nabla f(\boldsymbol{x})$ for a small complex step is as follows,

$$\nabla f(\boldsymbol{x} + i\mu \boldsymbol{y}) = \nabla f(\boldsymbol{x}) + i\mu B\boldsymbol{y} + O(\mu^2) \tag{4.38}$$

from which one could obtain $B\boldsymbol{y} \approx Im(\nabla f(\boldsymbol{x} + i\mu \boldsymbol{y}))/\mu$. Thus, the Hessian-vector product could be computed at the cost of one extra gradient computation. We implemented complex step differentiation for obtaining Hessian-vector products. We found it to be slower than computing the Hessian and computing Hessian-vector products directly. This is because the Hessian is *sparse* for the smooth dual (3.5). Exploiting the sparsity of the Hessian is another way to obtain efficient Hessian-vector products. We will see more about this in section (5.1.1).

Chapter 5

Newton Methods for Inference in Higher-order Markov Random Fields

5.1 Ill-conditioning and affine invariance

We saw in chapter (3) that the optimum of the smooth dual (3.5) is at a bounded distance away from the optimum of the non-smooth dual (2.25). Thus, in order to reach closer to the optimum of the non-smooth dual, optimization over a series of smooth problems could be carried out, where we reduce smoothing as we go closer to the optimum. As τ increases, (3.5) is a closer approximation of (2.25) and we get closer to the optimum of the non-smooth dual. However, we saw in section (4.10) that a smooth approximation becomes increasingly ill-conditioned as it approaches the shape of the non-smooth problem. Hence, it is costlier to optimize for larger values of τ. It is better to start with a very smooth version and switch to less smoothness, with warm start from the previous smoother version [Savchynskyy 2012]. This is the approach that we take and we will show how Newton methods could be used for optimization.

We repeat that smoothing is reduced as the algorithm proceeds and this results in ill-conditioning. Newton methods have some robustness against ill-conditioning due their affine invariance [Boyd 2004, §9.5.1]. This concept could be understood by considering two functions $f(\boldsymbol{x})$ and $\bar{f}(\bar{\boldsymbol{x}}) = f(A\boldsymbol{x})$, where A is an invertible square matrix. The iterates of a Newton method will be related as $\bar{\boldsymbol{p}}^t = A\boldsymbol{p}^t$. Hence, in theory, these methods are not affected by ill-conditioning and in finite precision computers, the range of condition numbers in which Newton methods exibit robust behavior is more than that of first order methods. Thus, there is one more motivation to investigate the suitability of Newton methods for MAP inference.

5.1.1 Hessian related computations

We will recall that we are interested in solving the following linear system in each iteration of the Newton method,

$$\boldsymbol{B}(\boldsymbol{\delta})\boldsymbol{p}^* = -\nabla g(\boldsymbol{\delta}). \tag{5.1}$$

Even though Newton methods have advantages, it may be expensive to compute the Hessian and/or solve the linear system (5.1). We recall, that the computationally

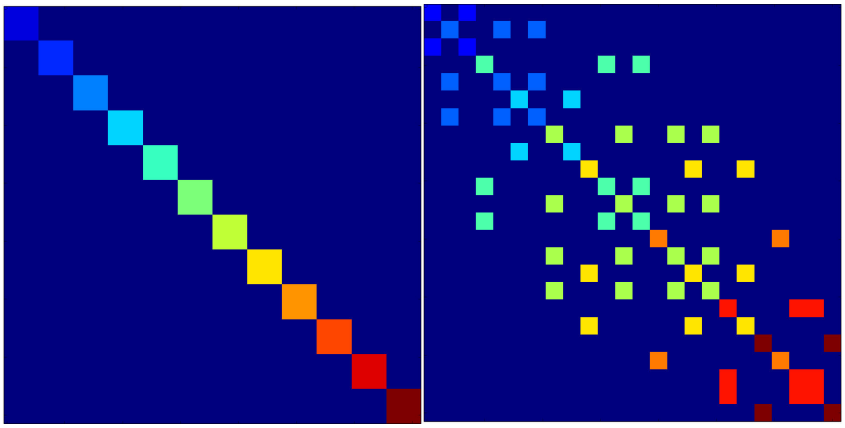

Figure 5.1: The two components of the Hessian. Within each component, blocks of the same color have the same values. In component one, there are as many unique blocks as cliques. In component two, each row/column of blocks has the same block, corresponding to the shared node.

heavy steps for the conjugate gradient (CG) method are: computing Hessian-vector products and regularly constructing and applying a preconditioning matrix. As mentioned in section (3.7), for medium sized problems it is sufficient to decompose according to cliques and achieve practical convergence. We will now show that for decompositions according to small subgraphs like individual cliques, the Hessian can be computed very efficiently. In fact, it only takes time of about twice the gradient computation time to compute both the gradient and Hessian together.

If we take a closer look at the Hessian, we observe that it can be written as the sum of two components, as shown in figure 5.1. Both components have a block structure. In section (3.2), we saw that the gradient is made of two components $\mu_{ci}(x_i)$ and $\mu_i(x_i)$ (3.8). Now, consider the following quantity,

$$\mu_{cij}(x_i, x_j) = \frac{\sum\limits_{\boldsymbol{x_c}:\boldsymbol{x_c}(i)=x_i,\boldsymbol{x_c}(j)=x_j} \exp\big[\tau.(\theta_c(\boldsymbol{x_c}) - \sum\limits_{n:n\in c} \delta_{cn}(x_n))\big]}{\sum\limits_{\boldsymbol{x_c}} \exp\big[\tau.(\theta_c(\boldsymbol{x_c}) - \sum\limits_{n:n\in c} \delta_{cn}(x_n))\big]}. \tag{5.2}$$

It is computed like $\mu_{ci}(x_i)$ but by keeping the labels of two nodes fixed. It can be seen that component one of the Hessian is a block diagonal matrix, with as many unique symmetric blocks as there are cliques. For some clique c, the elements of the block can be written as $B_{c,ij}(x_i, x_j) = \tau(\mu_{cij}(x_i, x_j) - \mu_{ci}(x_i)\mu_{cj}(x_j))$. Component two of the Hessian has non-zero elements only when two cliques a and b share a node i. One such $(l \times l)$ block, has the following elements along its diagonal: $B_{ab,i}(x_i, x_i) = \tau\mu_i(x_i)(1 - \mu_i(x_i))$ and the following off-diagonal elements: $B_{ab,i}(x_i, x_j) = -\tau\mu_i(x_i)\mu_i(x_j)$.Component two has as many unique symmetric blocks as there are nodes and a given row or column has repeated copies of the same block corresponding to the shared node. Also, it is component two, which contains off-diagonal blocks, which is caused by overlapping cliques.

Hence, the elements of the Hessian can be computed by only iterating once through all cliques and nodes. Iterations corresponding to pairs of overlapping cliques is avoided because two overlapping cliques only result in blocks corresponding to the shared nodes. Now, for computing the gradient, we need to loop once through all cliques and nodes. Within those same loops the symmetric blocks needed by the Hessian could be computed. Practically, due to cache re-use, this is achieved with the overhead time of one gradient computation only. These blocks can be readily used for parallelizing Hessian-vector multiplication and we do it with simple OpenMP code. Thus, efficient Hessian-vector products lead to an efficient CG routine in our case.

In our problem, the Hessian is sparse because a clique overlaps only with a few other cliques. Nevertheless, because of the special structure of components one and two, there is no need to exploit this sparsity for computing and storing the unique blocks of the Hessian. Also, if there are many overlapping cliques, computing Hessian-vector products is not adversely affected because we will only have more of the same block along each row of the component two matrix, corresponding to the shared node. Moreover, we will see in section (5.1.4) that these components could be readily used for constructing an effective preconditioner.

5.1.2 Damping matrix approach

We have noted that the smooth dual (3.5) is only strictly and not strongly convex, i.e., away from the optimum, the Hessian is only positive semidefinite. A trust-region approach is necessary to obtain meaningful Newton steps. While developing our trust-region Newton method for MRF inference (TRN-MRF), we addressed several issues: how to enforce the trust-region, how to improve speed of convergence by coping with the ill-conditioning and how to design a suitable preconditioner. It is a combination of these choices that lead to a usable algorithm and we note that, what works for MAP inference, may not work for other tasks and vice-versa.

The Steihaug method (4.7) seems like the first method to try for large problems. As mentioned earlier, it has the nice property of shaping the trust-region into an ellipsoid, according to the landscape. However, in the absense of a good preconditioner, in a given outer iteration, the inner iterations quickly reach the trust-region radius, before computing a good direction. This leads to several outer iterations. In our experiments with Steihaug, we tried the diagonal and block-diagonal preconditioner and they did not result in performance improvement. Experiment with the incomplete Cholesky preconditioner could be still attempted. However, based on our experience with incomplete Cholesky for our damping matrix based approach, we strongly feel incomplete Cholesky may not lead to substantial improvement in the performance of Steihaug.

We adopt the damping matrix based approach described in section (4.8) and we have obtained excellent convergence rate with this approach.

5.1.3 Forcing sequence for CG truncation

Having found out the suitability of damping matrix based approach, it is still necessary to properly address the ill-conditioning caused by annealing. We saw in section (4.4.1) that we proceed by taking approximate Newton steps: within each outer iteration the run of CG iterations is truncated by a suitable criterion. However, as the algorithm approaches optimality, it is critical to solve the linear systems to greater accuracy and get better Newton steps [Schlick 1987]. Otherwise, the algorithm will take too long to converge or will not converge at all. Generally, at an outer iteration t, CG can be truncated at iteration j according to the condition (4.22). Through η^t we could reduce the residual and achieve more accurate Newton direction. For MAP inference, we found textbook choices of the forcing sequence like (4.23) leading to non-convergence of the Newton iterations. This is because the residuals never become low enough to achieve the more accurate directions required for further progress. Hence, for TRN-MRF, the following criterion has been designed,

$$\eta^t = min\left(\frac{\varepsilon_\tau}{t}, \sqrt{\|\nabla g(\boldsymbol{\delta}^t)\|}\right). \tag{5.3}$$

This condition naturally imposes stronger condition on the residual for later iterations. Also, the term (ε_τ), which takes smaller values as the annealing progresses, ensures that sufficiently accurate Newton steps are obtained, as smoothing reduces.

5.1.4 Clique based Preconditioner

The computational efficiency of the conjugate gradient method could lead to substantial speed-up of our approach. We experimented with several preconditioning approaches starting with the simple diagonal preconditioner to incomplete Cholesky, quasi-Newton and multigrid. We obtained the best performance with a block diagonal preconditioner, which has an interesting interpretation from an MRF point of view. Each block, $\hat{B}^c$, corresponds to a clique c. Therefore, $\hat{B}^c_{i,x_i,j,x_j} = \frac{\partial^2 h(\boldsymbol{\delta})}{\partial\delta_{ci}(x_i)\partial\delta_{cj}(x_j)}$, where i, j are nodes belonging to the clique c and x_i, x_j are their labels. Since its structure is closely related to the MAP inference problem, it performs quite well. The computational cost is of computing and inverting these blocks individually. These clique specific blocks could be readily assembled from component one and component two of the Hessian. Hence, no additional computation is required to construct the preconditioner. The individual blocks are of size $s.l$ and inverting them is not computationally heavy. The inversion of the blocks could be parallelized, if need be. We have not yet parallelized this step.

Note that for sufficiently large values of the damping parameter λ^t, the modified Hessian is automatically well-conditioned and CG will converge fast. It is towards the optimum, when λ^t reduces to vanishing values, that CG runs for large number of iterations. We observed considerable improvement in CG performance with this preconditioner at this stage. Also, in general, we need to stop running CG after a maximum allowed number of iterations. We have set it as 250 for all our experiments.

5.1.5 Backtracking search

The last aspect concerning efficiency involves backtracking line search. When ρ in equation (4.28) is less than a small value (say, ε_ρ), it means a step of either very less decrease or an increase in the function value. So, we cannot directly take a step along this direction. However, it is desirable to take a step of sufficient decrease in every outer iteration of TRN-MRF and hence, we perform a backtracking line search in these cases. Such a combination of line search with trust-region has already led to improved results in other problems [Nocedal 1998]. Performing a backtracking line search along the direction computed by CG, gave a huge speed-up for TRN-MRF. We have implemented the cubic interpolation based line search described in section (4.5), which is very efficient.

Note according to some literature on Newton methods [Martens 2010], the backtracking can be performed along a curved path. Cache CG iterations regularly and backtrack with respect to the path taken by CG. Although backtracking along a curved path gives good results in these other problems, we observed poor results for MAP inference. We observed that the final direction after backtracking was very close to the steepest descent direction.

5.1.6 Annealing schedule and stopping condition

[Savchynskyy 2012] suggests annealing τ by periodically computing the primal-dual (PD) gap of the current smooth problem. We noted in section (3.5) that they recover feasible primal variables by solving a small transportation problem for each clique. Since, their approach is adapted to pairwise graphs, their entire algorithm runs with less than thousand oracle calls (an oracle call is either an iteration or computation of the PD gap). However, for higher-order MRFs, computing feasible primal variables and the primal objective (2.10a) is costly. Hence, computing PD gap of the smooth problem, every few iterations greatly affects computational efficiency.

We have used a simple but intuitive criterion for judging when to anneal. Since, we are working with an unconstrained concave objective, regions with lesser values of gradient euclidean norm are guaranteed to be closer to the optimum than regions with much greater values. Hence, if $\nabla g(\boldsymbol{\delta}^t)$ signifies the gradient after running t iterations with a given τ, we update $\tau \leftarrow \alpha\tau, 1 < \alpha$, if $\|\nabla g(\boldsymbol{\delta}^t)\|_2 < \gamma_\tau$. If we impose a strong enough threshold γ_τ, we are guaranteed to achieve sufficient improvement for that particular τ. Thus, annealing the τ value is justified. We set $\gamma_\tau = \beta\|\nabla g(\boldsymbol{\delta})\|_2$, when iterations for a given τ begins. Similar to the proof in [Savchynskyy 2012], this annealing approach will work with any optimization algorithm that will converge to the global optimum for a fixed value of τ. All the algorithms tested by us, have this guarantee for our unconstrained smooth and concave dual. Also, we ensure τ has reached a large enough value τ_{max} in order to obtain accurate results with respect to the original non-smooth problem.

In order to exit the least smooth problem, [Savchynskyy 2012] use the non-smooth PD gap. We have observed that TRN-MRF, due to its quadratic convergence

rate, can exit based on classical gradient based condition itself. More precisely, with the condition $\|\nabla g(\boldsymbol{\delta})\|_\infty \leq \zeta$, where $\zeta = 10^{-3}$ (§8, [Gill 1981]), TRN-MRF achieves good exit behaviour. However, first order methods take too long a time to achieve this gradient based exit condition and several times never do so. Hence, we have implemented a PD gap based approach, so that the first order algorithms can exit gracefully. As mentioned earlier, we implemented the method presented in [Meshi 2014]. While implementing this method, small valued variables should be handled carefully. In order to get numerically correct implementation, it is necessary to round variable values down to zero based on a small threshold.

5.1.7 TRN-MRF algorithm and parameter settings

Our complete trust-region Newton method, within an annealing framework, is described in algorithm (9). In our experiments, we set, $\lambda_0 = 1$, $\alpha = 2$, $\beta = \frac{1}{6}$, $\varepsilon_\rho = 10^{-4}$, $\zeta = 10^{-3}$, $\tau_{max} = 2^{13}$ and $\varepsilon_\tau = 0.1$ if $\tau < \frac{\tau_{max}}{4}$, 0.01 if $\frac{\tau_{max}}{4} < \tau < \frac{\tau_{max}}{2}$, 0.001 if $\frac{\tau_{max}}{2} < \tau$.

Algorithm 9 TRN-MRF: Trust-region Newton for MAP inference

1: Input: $\lambda_0, \tau, \tau_{max} > 0; \boldsymbol{\delta}_0 \in \mathbb{R}^{\mathcal{N}}; \alpha > 1.$
2: $\lambda \leftarrow \lambda_0, \gamma_\tau = \beta\|\nabla g(\boldsymbol{\delta}_0)\|_2$
3: **while** $\|\nabla g(\boldsymbol{\delta}_k)\|_\infty > \zeta$ or $\tau < \tau_{max}$ **do**
4: **if** $\|\nabla g(\boldsymbol{\delta}_k)\|_2 \leq \gamma_\tau$ and $\tau < \tau_{max}$ **then**
5: $\tau \leftarrow \alpha\tau$; $\gamma_\tau = \beta\|\nabla g(\boldsymbol{\delta}_k)\|_2$; adjust ε_τ
6: **end if**
7: $\boldsymbol{B}(\boldsymbol{\delta}_k) = \nabla^2 g(\boldsymbol{\delta}_k) + \lambda \boldsymbol{I}$
8: set $\eta_k = min(\frac{\varepsilon_\tau}{k}, \sqrt{\|\nabla g(\boldsymbol{\delta}_k)\|})$
9: Run CG while $\|\boldsymbol{r}^j\| > \eta_k\|\nabla g(\boldsymbol{\delta}_k)\|$
10: obtain Newton direction $\boldsymbol{p}$ and calculate ρ
11: update λ according to equation (4.28)
12: if $\rho < \varepsilon_\rho$ then backtracking line search along $\boldsymbol{p}$ to obtain Newton step $\boldsymbol{p}_k$
13: $\boldsymbol{\delta}_{k+1} = \boldsymbol{\delta}_k + \boldsymbol{p}_k$
14: **end while**

5.2 Quasi-Newton approach for large graphs

In section (3.7), we pointed out the trade-off between the sizes of the graph and the subgraphs to obtain practical convergence. Also, we saw that for problems decomposed according to large subgraphs, it is sufficient to enforce consistency between node pseudomarginals. Also, if a node is shared by only two subgraphs then the number of dual variables reduces considerably. In our experiments, we have worked with large graphs where a node is shared by only two subgraphs. For problems where nodes are shared by more than two subgraphs, the problem size

will be larger and it is a challenge that is shared by all algorithms solving the dual of the LP relaxation.

Even though smoothing based approach has been scaled for large pairwise graphs [Savchynskyy 2012], higher order MRFs with large graphs are less studied. We saw in section (5.1.1), the Hessian for clique based decomposition requires the marginals for pairs of nodes within that clique. This is true for tree based decomposition of large graphs, also. We have observed empirically that for faraway nodes within a chain the corresponding block of values in the Hessian goes to zero. This is because the two nodes are independent of each other for all practical purposes. Nevertheless, it is a computationally heavy task to compute the Hessian blocks for nearby nodes within a chain. Maybe, by using automatic differentiation [Domke 2011], we could achieve some speed-up. Even then, storing the non-zero blocks of the Hessian is going to be memory intensive. Thus, instead of computing the true Hessian, we work with a Hessian approximation obtained through the L-BFGS method. Note that quasi-Newton methods have only super-linear convergence rate when sufficiently close to the optimum.

We saw in section (4.9) that for ill-conditioned problems, the Hessian inverse approximation will lead to large steps. We have also observed that in our experiments, where we implemented the Hessian inverse approximation of L-BFGS and also, used the implementation in the ceres solver by google [Agarwal]. Downscaling the large step to a trust region radius does not help, as the direction itself is poor to start with. A trust-region based approach using the Hessian approximation is the better choice. Also, for large problems a limited memory variant is required, based on the most recent m iterations. Hence, we take the approach described in algorithm (9) with an L-BFGS based Hessian approximation in the place of $\nabla^2 g(\boldsymbol{\delta}_k)$. We also implemented the SR1 method but the results were poorer than Hessian approximation based L-BFGS.

We noted in section (4.9) that the conjugate gradient method converges in $O(r)$ iterations for a linear system in the form an identity + rank-r matrix. Morever, matrix-vector products in quasi-Newton can be obtained through a compact representation (4.36) [Byrd 1994]. Thus the cost for CG is a very small fraction of the cost for computing the gradient (0.5% in our experiments). Thus, given the iteration cost of quasi-Newton being comparable to first order methods, it is worthwhile to check whether quasi-Newton converges faster to the optimum.

5.3 Experiments

We present results based on higher-order MRF models. As our baseline, we used FISTA with backtracking line search (section (3.2)), Smooth Coordinate Minimization (SCM) based on star update (section (3.3)) and AD3 implementation from openGM (section (3.4.1)). Note that Star update based SCM, consistently performed better than min-sum diffusion based update in our experiments.

First, we tested on medium sized problems and compared TRN-MRF, FISTA, SCM

	$\sigma = 0.5$				$\sigma = 0.8$			
Algorithm	TRN	SCM	FISTA	AD3	TRN	SCM	FISTA	AD3
time (seconds)	948.9	1742.7	4368.1	369.35 (738.7)	2513.8	6884.2	9037.1	No convergence
Non-smooth dual	-279.69	-279.69	-279.69	-279.69	-259.28	-258.27	-258.7	-260.94
Non-smooth primal	-279.67	-279.69	-279.66	N/A	-261.04	-258.36	-258.86	N/A
Integer primal	-279.69	-279.69	-279.69	-279.69	-247.86	-248.24	-250.3	-249.82

Table 5.1: Results for synthetic problem.

	Frame 70				Frame 90				Frame 110			
Algorithm	TRN	SCM	FISTA	AD3	TRN	SCM	FISTA	AD3	TRN	SCM	FISTA	AD3
time (seconds)	2374.5	3702.9	11964.5	1428.7 (2857.4)	4731.6	4206.4	12561.2	2303.05 (4606.1)	4451.8	10498.8	21171.1	No convergence
Non-smooth dual	-368.59	-368.59	-368.59	-368.59	-337.81	-337.81	-337.81	-337.81	-333.03	-331.51	-331.31	-335.5
Non-smooth primal	-368.56	-368.57	-368.37	N/A	-337.77	-337.78	-337.36	N/A	-336.16	-331.95	-330.65	N/A
Integer primal	-368.59	-368.59	-368.59	-368.59	-337.81	-337.81	337.81	-337.81	-315.93	-317.69	-317.78	314.16

Table 5.2: Results for House dataset.

and AD3. We formulate the problem of matching two sets of interest points as MAP inference, based on [Duchenne 2011]. For n points, each point in the source can be matched to any of the points in the target, i.e., n labels. The MRF is constructed by generating $4n$ triangles in the source and each triangle in the source and target are characterized by tuples of side length. For each source tuple, we find the top 30K nearest neighbours among the target tuples. 30K is much lesser than all possible triangles in the target and this is a sparse, pattern-based clique potential. The higher-order cliques potentials are defined as $\exp(\frac{-1}{\gamma}((s_a - t_a)^2 + (s_b - t_b)^2 + (s_c - t_c)^2))$, where s and t refer to source and target, respectively and γ is the average squared differences between source tuple and its 30K nearest neighbours in the target. To get state-of-the-art results additional terms to disallow many-to-one mapping will be needed. For the sake of simplicity we ignore this issue.

We first tested with a synthetic problem, with $n = 81$ on the 2D plane. We added Gaussian noise to these points to create the target image. The unary potentials are defined by assigning the value i to node i in both the images and taking the absolute differences. The results are presented in table 5.1, where two levels of added noise were tested.

We next tested with matching points on the House dataset [hou]. $n = 74$ points are marked in all the frames and the points in the first frame are matched with points in later frames (110 being the last frame). The unaries are set to zero. The camera rotates considerably around the object and we show results for three frames in table 5.3.

The main reason for being able to tackle such problems is the efficient computation of the gradient using sparse, pattern based potentials (section 3.8). TRN-MRF, FISTA and SCM, all benefit from this. AD3 can also exploit sparse, pattern-based potentials, since, within every iteration of the active-set method, max-product computation takes place. Hence, the routine from [Komodakis 2009] could be used by AD3. Since, these modifications cannot be made to the AD3 version in openGM, we show within brackets the actual time taken by openGM's AD3. Since, our current implementation of sparse sum-product gives two times speed-up, the outside figure for AD3 is half

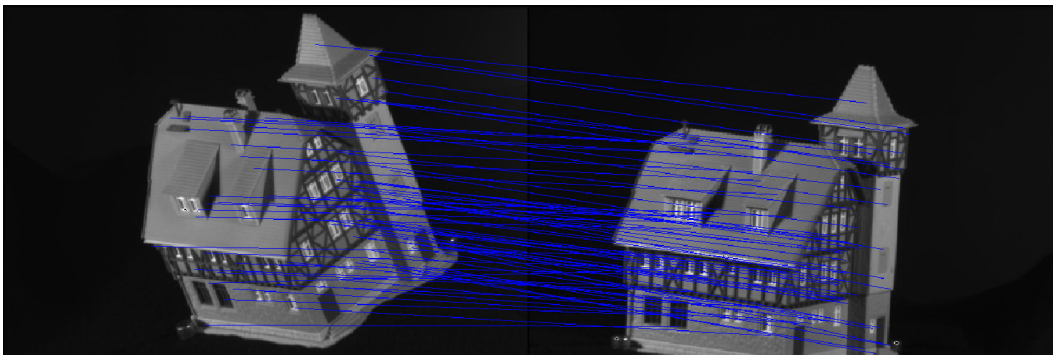

Figure 5.2: Matching 1st frame to 90th frame.

the actual time taken. We note that the actual time saved by AD3 because of sparse potentials could be lesser because of other operations in each iteration.

We would like to make a note about parallel computation at this point. The Hessian requires marginals of pairs of nodes within each clique, i.e., $\mu_{c,ij}(x_i, x_j)$ described in section (5.1.1). While computing them using sparse pattern based potentials, there is scope to parallelize some steps and we have done it using OpenMP code. Apart from parallelizing a part of Hessian-vector product using OpenMP, mentioned in (5.1.1), this is the only other use of parallel computing in our approach. For both TRN-MRF and FISTA there is considerable scope to parallelize the independent clique based computations in each iteration. We have not yet done the same. Once, we do that both these algorithms will show similar gains in speed. Also, for our method, the preconditioner has a block-diagonal form and could be applied in a parallel manner. This could further improve our method's speed. Note, recently [Meshi 2017] has demonstrated considerable speed-up for SCM through parallel asynchronous updates. In a parallel asynchronous setting, SCM could very well be the best performing algorithm for the problems that we have considered. One has to try and see. There are more details about parallel algorithms for LP relaxation based techniques in the next chapter.

An important observation is that TRN-MRF, FISTA and SCM have reliable convergence compared to AD3 (an ADMM based approach). Among these three, TRN-MRF is competitive and is the fastest in many cases. We observed the quadratic convergence rate of TRN-MRF in practice because we always observed it reach the stricter gradient based exit condition around the same time as the PD gap based exit condition. On the other hand, the first order methods depended on the weaker PD gap based exit condition in order to exit in a principled manner.

In section (3.5), we mentioned that we do simple rounding of the primal variables to recover the labels. This rounding leads to energies that can be slightly better or worse between algorithms. We have not focused our efforts on better rounding schemes, since, the crux of our work is about getting closer to the global optimum of the non-smooth dual. The results of AD3 are based on its own rounding scheme.

	Frame 70				Frame 90				Frame 110			
Algorithm	TRN	SCM	FISTA	AD3	TRN	SCM	FISTA	AD3	TRN	SCM	FISTA	AD3
time (seconds)	2374.5	3702.9	11964.5	1428.7 (2857.4)	4731.6	4206.4	12561.2	2303.05 (4606.1)	4451.8	10498.8	21171.1	No convergence
Non-smooth dual	-368.59	-368.59	-368.59	-368.59	-337.81	-337.81	-337.81	-337.81	-333.03	-331.51	-331.31	-335.5
Non-smooth primal	-368.56	-368.57	-368.37	N/A	-337.77	-337.78	-337.36	N/A	-336.16	-331.95	-330.65	N/A
Integer primal	-368.59	-368.59	-368.59	-368.59	-337.81	-337.81	337.81	-337.81	-315.93	-317.69	-317.78	314.16

Table 5.3: Results for House dataset.

5.3.1 Higher-order Stereo

Next, we present results for stereo with curvature prior on 1×3 and 3×1 cliques. The clique energy is truncated, i.e., it is pattern-based. The unary potentials are based on absolute difference. We present results for Tsukuba, image size 144×192, 16 depth levels. This is a large problem and pattern-based sum-product was computed on clique chains. We compare quasi-Newton with FISTA and AD3. AD3 showed poor convergence behavior for this large problem.

In our experiments, we have used a stopping criterion that is simultaneously based on function value difference, variable difference and the norm of the gradient (adapted from [Gill 1981, §8.2.3.2]). Given adjacent iterations t and $t+1$, for the dual function $h(\boldsymbol{\delta})$, function value difference is $\Delta h = h(\boldsymbol{\delta}^{t+1}) - h(\boldsymbol{\delta}^{t})$ and l_∞ norm of the step taken is $\Delta\boldsymbol{\delta} = \|\boldsymbol{\delta}^{t+1} - \boldsymbol{\delta}^{t}\|_\infty$. Thus, the conditions enforced are $\Delta h < 10^{-3}, \Delta\boldsymbol{\delta} < 10^{-3}, \|\nabla h(\boldsymbol{\delta}^{t+1})\|_\infty < 0.5$ and we exit if all these conditions are satisfied. We recommend this principled exit condition as an alternative to the computationally heavy PD gap based approach.

	Tsukuba		Venus	
Algorithm	Quasi-Newton	FISTA	Quasi-Newton	FISTA
Iterations	357	594	102	> 180
Non-smooth dual	29105.9	29105.5	39153	≈ 39153
Integer primal	29347	29282.5	39230.5	≈ 39230

Table 5.4: Stereo estimation: Tsukuba & Venus.

5.4 Discussion

We presented Newton methods that offer convergence guarantee and very good convergence rates, through appropriate choices made concerning their algorithmic components. Specifically, for problems in which sum-product computation is efficient, these Newton-type methods are very suitable. We showed promising results with higher-order MRFs of medium and large sizes. We hope this work spurs further research on exploiting curvature information within optimization algorithms for MAP inference. We make some observations regarding the same in the next chapter.

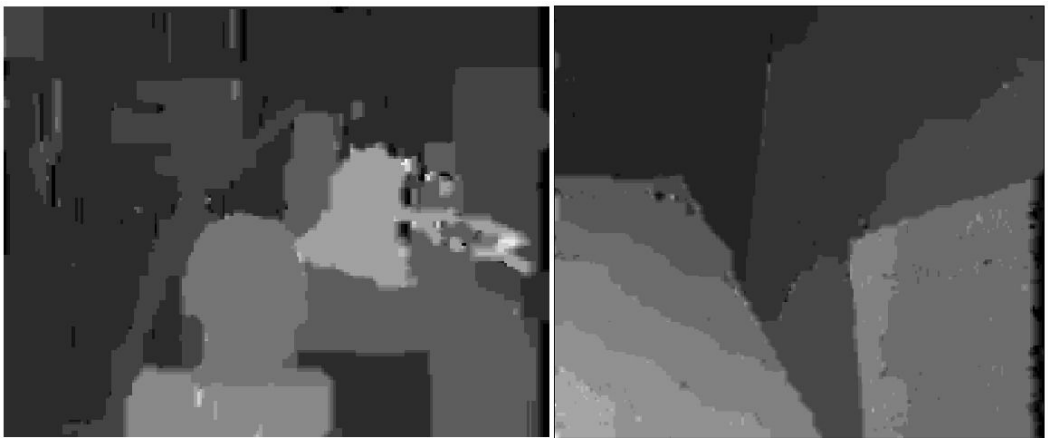

Figure 5.3: Tsukuba and Venus results for quasi-Newton.

Chapter 6

Some thoughts on continuous relaxation based MAP inference

LP relaxation based inference in graphical models is a research topic that continues to get the attention of the research community [Ajanthan 2017, Swoboda 2018]. In this chapter we would like to discuss some aspects.

6.1 Notes about Newton methods

6.1.1 Stochastic Newton

There have been many works in recent years regarding Newton methods in machine learning [Pilanci 2015, Erdogdu 2015, Byrd 2016, Mutnỳ 2017]. One aspect that stands out is the stochastic nature of the updates among these modern methods. Another frequent aspect is approximation of curvature information by quasi-Newton methods. Given the stochastic nature of the updates, analysing this quasi-Newton approximation leads to new results. Stochastic approximation is a less explored topic in MAP inference. [Meshi 2015] shows a strongly convex and smooth primal problem, which is suitable for optimization by SDCA [Shalev-Shwartz 2014]. In this formulation, the strong-smooth duality is driven by quadratic perturbation terms. Please, refer [Meshi 2015] for more details. It will be interesting to try stochastic Newton methods with this formulation. The more a Newton method exploits problem structure, the better it is for making it efficient and scalable. Maybe it will be a straightforward application of stochastic Newton in this case. It will be good to clarify this point.

6.1.2 Partial separability and quasi-Newton methods

A function is partially separable if it has the following form,

$$f(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x}_i) \tag{6.1}$$

where $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_s)$ are s overlapping subsets of the set of variables. Some of the dual formulations for MAP inference have this structure (3.5, 3.34). There are quasi-Newton methods that exploit this partial separability structure [Nocedal 2006, §7.4]. They may not turn out to be better than the approach described in the previous chapter using the true Hessian for (3.5). It may be more suitable for (3.34). We are currently facing an implementation error while pursuing this path.

6.1.3 Projected Newton for small and medium graphs

We have observed that for small and medium sized graphs, gradient based methods converge well with clique based decomposition. If we consider the smooth version of the constrained formulation (2.20), the Hessian has a block diagonal structure. For a clique of size s, where each node takes l labels each, the blocks are of size $s.l \times s.l$. It is computationally feasible to invert these blocks. In fact, the block diagonal preconditioner that we considered in section (5.1.4), is made of blocks of the same size. In such a scenario, it may be a good idea to try a projected Newton method to solve the smooth version of (2.20). Especially, we could consider the spectral projected gradient method to compute the Newton step like it is done in [Schmidt 2009]. Since, the Hessian has block diagonal structure, where the blocks are easily invertible, a preconditioned spectral projected gradient method could be readily tried and could lead to fast convergence.

However, for small and medium sized graphs there are two strong baselines. For small and medium sized graphs, AD3 [Martins 2015] shows excellent convergence speed. However, it does not come with convergence guarantee and it has been observed to not converge in some cases. Recently, [Meshi 2017] has a shown an asynchronous parallel block coordinate maximization algorithm that can lead to considerable speed-up. The scope for parallelization will be similar for AD3 and the projected Newton method that we have outlined. In all cases, the parallelization is over clique based computations. Maybe, parallel projected Newton could beat parallel AD3 but whether it could beat asynchronous BCM is an open question.

6.2 Preconditioning first order optimization methods

We would like to point out that preconditioning of first order optimization methods has seen significant research effort in recent years [Koutis 2011, Pock 2011, Giselsson 2014, Dauphin 2015, Raguet 2015, Fougner 2018, Möllenhoff 2018]. Among various works, we can more easily relate [Pock 2011, Möllenhoff 2018] to a primal-dual algorithm for MAP inference [Schmidt 2011]. We note that [Schmidt 2011] experimentally studied the primal-dual algorithm of [Chambolle 2011] without preconditioning. Now, [Pock 2011] presented a diagonal preconditioner in the context of other computer vision problems. We noted in section (4.10) that a block diagonal preconditioner gave substantial improvement with respect to a diagonal preconditioner. Interestingly, [Möllenhoff 2018] also points out the benefit of diagonal preconditioning and presents better results compared to [Pock 2011] for computer vision problems. It will be interesting to understand whether it is possible to accelerate the primal-dual algorithm with a tree based preconditioner for MAP inference, also. As such, it is worth investigating the suitability of preconditioning with respect to various first order methods for MAP inference.

6.3 Numerical linear algebra and Graphical models

Apart from preconditioning optimization problems, there are other interesting connections between these two topics. For example, inference in Gaussian graphical models involves the solution of a linear system [Shental 2008]. Also, the tree based preconditioner that we came across in section (4.10.6) has been explored for various contexts within graphical models [Sudderth 2004, Lafferty 2006, Chandrasekaran 2008, Malioutov 2008]. It will be interesting to understand these works better. Finally, it will be interesting to know more about connection between the fast fourier transform and message passing in factor graphs [Kschischang 2001].

6.4 Tightening the relaxation

We saw in section (2.5), the LP relaxation is guaranteed to be exact for any tree-structured graphical model. However, for general graphs a tree based decomposition is not tight. This is because the local polytope ($\mathcal{L}(\mathcal{G})$) is a outer bound to the marginal polytope ($\mathcal{M}(\mathcal{G})$), in general. Thus, with the local polytope we may get a fractional corner as the optimal solution and depending on the problem, we may be arbitrarily faraway from the optimum of the discrete problem. It is possible to strengthen the relaxation based on additional constraints. It is possible to consider bigger regions of the graph, which will contain as a subset the cliques used to define the graphical model. It is possible to define pseudomarginals on those bigger regions, also and impose consistency constraints between them and the cliques that they contain. This will tighten the polytope. [Ravikumar 2010] also point out that this particular progression of tighter relaxations underlies belief propagation algorithms also, i.e., the Bethe (sum-product) to Kikuchi (generalized sum-product) hierarchy [Yedidia 2005]. [Wainwright 2008, Sontag 2010] have more information about such LP hierarchies. A related perspective is that of *frustrated cycles*. When one goes along a cyclic path of the graph, when the optimal label for the nodes and the cliques agree, then that cycle is said to consistent. Otherwise, it is a frustrated cycle. A fractional optimal solution to the LP relaxation means there are frustrated cycles in the graph. There has been work to identify and "repair" such frustrated cycles [Komodakis 2008, Sontag 2012]. [Komodakis 2011, §6.2] has shown that if we decompose the original graph according to subgraphs that are not trees then we obtain a tighter relaxation. Another takeaway from that paper is that all tree based decompositions lead to the same local polytope. Suppose we work with a decomposition where the subgraphs are not trees then we have the possibility of a tighter relaxation. However, we are faced with the challenge of performing exact MAP or marginal inference on such subgraphs. This was the reason for us to base all our discussions in terms of tree shaped subgraphs in previous chapters. There have been works that address this challenge. For example, [Wang 2013] shows a fast and exact method for MAP inference in cycles. They obtain dual decomposition by considering cycles of three nodes and show how it leads to better energy because of tighter relaxation. Further, the semidefinite programming (SDP) based relaxation

and the LP relaxation are not comparable [Wainwright 2008, §9.3]. Computational cost has been a major barrier for SDP relaxation and there have been works to address this issue [Huang 2014, Wang 2016]. In general, [Werner 2007, Sontag 2007, Werner 2008, Komodakis 2008, Sontag 2008, Batra 2011, Sontag 2012, Mezuman 2013] are excellent references to learn more about tighter relaxations.

6.5 Parallel algorithms to solve the LP relaxation

Generally, inference in graphical models becomes more challenging with the increase of one or more of the following: graph size, clique size and the number of labels per node. So far, insight in convex optimization has led to design of suitable algorithms. For example, the Frank-Wolfe algorithm in [Meshi 2015] for large cliques like global cardinality. We believe similar insight from optimization literature has led to and will lead to several parallel algorithms. Belief propagation based techniques readily lend themselves to parallelization [Felzenszwalb 2006, Gonzalez 2009, Yang 2010]. Even though there has been considerable practical success with these techniques, they come with poor guarantees. Parallel graph-cut algorithms is also an active field of research [Delong 2008, Liu 2010, Shekhovtsov 2013].

As far as LP relaxation based methods are concerned, dual decomposition based MAP inference [Komodakis 2011] and the research efforts that it has inspired, is readily suited for parallel computation. They have the added advantage of theoretical guarantees. Moreover, dual decomposition has been helpful in parallelization of other methods like graph cuts [Strandmark 2011]. We saw in section (3.2) that smoothing is not the only way to improve convergence with respect to the non-smooth formulation. [Schwing 2012] improves convergence by choosing the steepest descent direction among the ε-subgradients and [Schwing 2014] shows how that approach could be executed in parallel based on a Frank-Wolfe algorithm. We have seen in section (3.4), the ability of ADMM to decouple the update equations, which enables us to exploit the seperable structure in the objective function. To our knowledge, all the ADMM based algorithms for MAP inference [Martins 2015, Meshi 2011, Fu 2013, Miksik 2014] can be decomposed according to subgraphs. [Jin 2013] implemented the approach in [Fu 2013] on a cluster and presented results with large scale climate data.

We have noted that coordinate maximization methods are an important class of methods for MAP inference. For example, TRW-S [Kolmogorov 2006], a method meant for pairwise graphs, gives good output in practice and usually it is the fastest serial method based on LP relaxation for many problem instances. However, its update equations have a particular serial schedule. [Choi 2012] recognized that for a grid graph, a different schedule based on updating nodes one diagonal after the other, leads to parallelization among the nodes of a diagonal. [Hurkat 2015] has extended their work based on multiscale representation [Felzenszwalb 2006] and has shown good empirical performance. [Shekhovtsov 2016, Knobelreiter 2017] present a majorization-minimization [Lange 2000] based block coordinate optimization algorithm that has the same guarantees as TRW-S and is considerably faster due

to a formulation that allows parallel execution. They divide the graph into two blocks, where each block is made of independent parts. The most common example, is a grid graph like an image, where the two blocks are rows and columns. Within each block, the chains are independent of each other. Thus, when each block is updated, the individual parts within the block could be updated parallelly. In block coordinate optimization algorithms, it is possible to identify blocks made of parallelizable parts using graph coloring. It is an idea that originated in parallel numerical methods [Bertsekas 1989, §1.2.4]. We can see this idea in MAP inference, also [Felzenszwalb 2006, Schwing 2011, Chen 2014a].

[Schwing 2011] presents an approach to parallelize the coordinate maximization approach based on a smooth version of *convex BP* (3.10). They partition the graph in a distributed memory setting, where each partition executes parallelly, occassionally passing messages to enforce consistency at the boundary of the partitions. Within each partition they propose to perform coordinate maximization in a parallel manner, where the blocks are chosen according to graph coloring. They prove that their approach converges and leads to speed-up. They also show empirically the effect of the frequency with which messages are passed between partitions.

We should point out that a sizeable research effort in parallel algorithms for MAP inference has been from the point of view implementation for a specific hardware architecture. Usually, these efforts also explore appropriate heuristics like multiscale representation [Felzenszwalb 2006]. Recently, [Meshi 2017] has presented theoretical analysis of asynchronous block coordinate maximization. It is the asynchronous extension of the MSD and star updates that we saw in section (3.3). What is interesting about that work is the convergence rate analysis. To our knowledge, this is the first work in MAP inference to present a detailed theoretical perspective in a parallel processing setting. We foresee an increase in more theoretically grounded work in the coming days. In recent years, there has been considerable progress in numerical analysis and machine learning with respect to algorithmic aspects of parallel computing [Bekkerman 2011, Boyd 2011, Recht 2011, Ghysels 2013, Demmel 2014, Li 2014, Cevher 2014, Jordan 2015, Liu 2015, Hsieh 2015, Smith 2016]. There has been renewed interest in the analysis of parallel and distributed numerical methods [Bertsekas 1989], which was initiated several years ago.

A major thrust behind recent innovation has been the consideration of *synchronization* and *communication* costs in parallel algorithms. Generally, when we think about parallel algorithms, it is mainly in terms of parallel *computation*. For example, most, if not all, parallel algorithms for MAP inference have focused on parallelizing computation. These parallel computations have to happen in a *synchronous* manner, though. This is because the smaller parts performing parallel computation depend on the output of other parts from the previous iteration. This synchronization cost could be significant. [Meshi 2017] has addressed exactly this synchronization problem in a shared memory setting that guarantees bounded delay.

Usually, the updated decision variables that are passed between the various parts of the graph are in the form of messages. The associated communication cost is also very significant. Even though there are already communication avoiding

framework type of algorithms that address a broad range of machine learning problems [Smith 2016], there has been progress in recent years, where problem specific communication avoiding algorithms have been developed [Devarakonda 2016, Soori 2017, Devarakonda 2017, Koanantakool 2018]. It will be interesting to see similar developments in MAP inference. To our knowledge, [Schwing 2011] is one work in MAP inference that refers to the need for avoiding communication and it sends consensus messages between computers in lesser frequency compared to consensus messages within each computer. It will be interesting to study the effect of skipping messages between computers in a theoretical manner like [Meshi 2017]. It will be also interesting to test and analyse the coordinate maximization algorithm in [Schwing 2011] in an asynchronous setting. We would like to point out one possible way to improve the communication cost in [Schwing 2011] in the following subsection.

6.5.1 Enforcing node consensus in distributed setting

[Schwing 2011] partitions the graph into $\mathcal{P}$ graphs according to some boundaries. Thus, these graphs share *cliques* and *nodes* at the boundaries. For example, a rectangular grid graph could be divided into four smallers rectangular grids. They optimize the following optimization problem,

$$\begin{aligned} \underset{\boldsymbol{\psi},\boldsymbol{\phi}^p}{\text{minimize}} \quad & \sum_{p\in\mathcal{P}} \boldsymbol{\theta}^p \cdot \boldsymbol{\phi}^p - \frac{1}{\tau}\sum_{i\in\mathcal{V}^p} H(\boldsymbol{\phi}_i^p) - \frac{1}{\tau}\sum_{c\in\mathcal{C}^p} H(\boldsymbol{\phi}_c^p) \\ \text{subject to} \quad & \boldsymbol{\phi}^p \in \mathcal{L}(\mathcal{G}^p) \\ & \boldsymbol{\phi}_c^p = \boldsymbol{\psi}_c \;\; \forall c \in \mathcal{P}_{\mathcal{B}} \end{aligned} \tag{6.2}$$

which is a distributed version of the perturbed LP relaxation that we saw in section (3.2). Here, $\mathcal{P}_{\mathcal{B}}$ is the set of shared cliques at the boundaries of the partitions. Note the clique based consensus constraints, given in the last line. Similar to that earlier formulation, a dual based block coordinate optimization scheme is possible for this formulation. Apart from the dual variables corresponding to cliques within in each partition, there will be dual variables corresponding to the shared cliques. For each shared clique there will be dual variables of the size of total number of clique labellings and these dual variables are exchanged between computers at a frequency lesser than the updates within each computer. Suppose the graph is made of higher-order terms, then the number of overlapping cliques at the boundary of the partitions is much more compared to a pairwise graph. Thus, for a higher-order graph there is substantial communication cost. However, it is not necessary to impose consensus between the shared cliques, it is enough to impose consensus between the shared nodes. It is enough to have constraints of the following form,

$$\boldsymbol{\phi}_i^p = \boldsymbol{\psi}_i \;\; \forall i \in \mathcal{P}_{\mathcal{B}} \tag{6.3}$$

which will lead to dual variables for each of the shared nodes. Suppose a node has l labels, it will lead to l dual variables. Thus for a shared clique of size s,

with each node taking l labels, [Schwing 2011] exchanges l^s dual variables between computers, while it is enough to pass $l \cdot s$ dual variables. Note, enforcing consensus only between nodes may lead to slower overall convergence. However, slow down of [Schwing 2011] due to high communication cost with higher-order graphical models has been observed by others [Zhang 2014]. Thus, the time saved by smaller messages exchanged could lead to overall speed-up.

6.5.2 A note on asynchronous projected gradient for MAP inference

We have been observing that the two major forms of the dual are as shown in (2.19), which is more suitable for gradient based methods and (2.25), which is more suitable for coordinate maximization methods. Given the work of [Meshi 2017], the natural question to ask is whether an asynchronous projected gradient ascent algorithm is helpful for MAP inference. This is especially relevant for problems where gradient based methods have performed better than coordinate maximization [Koo 2010].

We will consider the smooth version of the formulation in (2.19). The asynchronous update equations for this formulation has been presented and studied in [Bertsekas 1989, §7.5]. The main point of concern is the step length parameter. It is possible to set it based on the Lipschitz constant (3.32) and the maximum allowed delay between updates. It has been observed that setting the Lipschitz constant this way is too conservative and it is better to perform backtracking line search [Savchynskyy 2011]. Due to the presence of the projection constraint, these line searches could slow-down the overall algorithm. It is an open question, whether asynchronous projected gradient with a fixed step length parameter could beat asynchronous BCM, which has no step length parameter.

Chapter 7

Unsupervised learning: a perspective based on optimal transport and sparse regularization

In this chapter, we will see some applications of numerical methods to unsupervised learning problems. Unsupervised learning is a fundamental problem in machine learning. In the previous chapters of this thesis, we came to appreciate that graphical models present very high dimensional problems. It will be desirable to develop unsupervised learning techniques that reduce the complexity of high dimensional problems. In this chapter, we will see a framework based on optimal transport and sparse regularization that allows us to view various unsupervised learning problems as special cases. Specifically, we will see how continuous optimization based formulations of k-medoids clustering (also, called exemplar clustering), center based clustering and kernel PCA could be formulated with suitable regularizers within this framework. One way to classify unsupervised learning problems could be as either *clustering* problems or *dimensionality reduction* problems. In this chapter, we will look at both types of approaches. We present an optimal transport based perspective to a loss that has already been used for exemplar clustering [Elhamifar 2012]. We show how this loss can be used for other problems through the concept of archetypal analysis and also, we show that this loss penalizes variance of the output points. We study existing sparse regularizers for k-medoids and center based clustering and present a novel low rank regularizer for the pre-image problem in kernel PCA. We study optimization algorithms for all the problems. Especially, in the case of kernel PCA, we present a majorization-minimization algorithm that is novel in the context of kernel PCA.

One may observe that there is a geometric flavor to unsupervised learning problems and approaches [Roweis 2000, Belkin 2006, Amari 2007, Von Luxburg 2007, Absil 2009]. In recent years, the subject of optimal transport has successfully addressed machine learning problems with a geometric flavour [Kolouri 2017]. Optimal transport has also helped to theoretically understand unsupervised learning algorithms [Canas 2012, Weed 2017]. The books by Cedric Villani [Villani 2003, Villani 2008] are excellent references on theoretical aspects of this subject and [Santambrogio 2015, Peyré 2017] are useful when working with applications. Given two probability distributions, optimal transport seeks the most efficient way of

transforming one to the other, given a ground cost for moving the mass. The original formulation by Gaspard Monge did not allow the splitting of mass and the solution to that optimal transport problem is referred to as the *transport map*. Disallowing mass splitting leads to no guarantee of the existence of a transport map. For example, if the source is a dirac and the target is not or if the source and target are supported on different numbers of diracs. In machine learning problems, data is available as point clouds, even though they may represent underlying smooth distributions. Leonid Kantorovich formulated a problem that allowed mass splitting. The solution to this formulation is called the *transport plan*.

The precise way to discuss about optimal transport is in terms of measure theoretic notions. We will take a more informal style here. We will work with two random variables X and Y, each with discrete probability distributions. Let the mass functions be $\boldsymbol{\mu}$ with n states and $\boldsymbol{\nu}$ with m states, respectively. Optimal transport finds a transport plan $\Gamma \in \mathcal{R}^{n\times m}$ that solves the following linear program,

$$\underset{\Gamma}{\text{minimize}} \quad \sum_{i,j} D_{i,j}\Gamma_{i,j} \tag{7.1a}$$

$$\text{subject to} \quad \Gamma^T \mathbb{1} = \nu \quad \text{and} \quad \Gamma\mathbb{1} = \mu \tag{7.1b}$$

$$\Gamma_{i,j} \geq 0 \quad \forall (i,j). \tag{7.1c}$$

Here $D \in \mathcal{R}^{n\times m}$ consists of the *ground* distances between all possible pairs of the two distributions. In many applications, the ground distance would be the Euclidean distance or the squared Euclidean distance. Constraints (7.1b) are marginalization constraints on Γ with respect to the two distributions. The constraint (7.1c) ensures that non-negative mass is transported between the distributions. Thus, Γ is a joint probability distribution between the two random variables and the rows and columns of Γ satisfy *simplex* constraints [Condat 2016]. The optimal objective (7.1a) is the total cost of transporting mass to transform one distribution to the other. Note, we could rewrite $\sum_{i,j} D_{i,j}\Gamma_{i,j}$ as $\operatorname{tr}(\Gamma^\top D)$.

Solving the linear program (7.1) for large problem sizes is hampered by the *coupling* constraints between the rows and columns of Γ. It is possible to define an optimization problem, where we retain constraints on either the rows or columns and we enforce structure on Γ through a regularizer. Such an optimization problem could take the following form,

$$\underset{\Gamma}{\text{minimize}} \quad \operatorname{tr}(\Gamma^\top D) \;+\; \lambda \operatorname{pen}(\Gamma) \tag{7.2a}$$

$$\text{subject to} \quad \begin{aligned} &\Gamma\mathbb{1} = \mu \\ &\Gamma_{i,j} \geq 0 \quad \forall (i,j) \end{aligned} \tag{7.2b}$$

where we have retained simplex constraints only on the rows. The function $\operatorname{pen}(\Gamma)$ is a suitably defined regularizer. Note, in an algorithm point of view, these simplex constraints could now be enforced in a parallel fashion.

In this optimization problem also, the matrix Γ continues to be meaningful as a joint probability matrix. Since, Γ is a joint probability matrix, it is possible to define a related conditional probability matrix P as follows,

$$\Gamma_{i,j} = P(A = x_i, B = x_j) = P(B = x_j | A = x_i) P(A = x_i) = P_{i,j}\mu_i \tag{7.3}$$

where the random variable corresponding to the input distribution is A and the random variable corresponding to the output distribution is B and both take values from $\{x_1, x_2, ..., x_n\}$. For a giveninput point x_i, the constraint (7.2b) ensures, $\sum_j \Gamma_{i,j} = \sum_j P_{i,j}\mu_i = \mu_i$. Thus, we could define an optimization problem with the conditional probability matrix as the decision variable as follows,

$$\underset{\Gamma}{\text{minimize}} \quad \text{tr}\left(P^\top \text{Diag}(\mu) D\right) \; + \; \lambda \, \text{pen}\left(P\right) \tag{7.4a}$$

$$\begin{aligned} \text{subject to} \quad & P\mathbb{1} = \mathbb{1} \\ & P_{i,j} \geq 0 \quad \forall (i,j). \end{aligned} \tag{7.4b}$$

We will next see how this conditional probability matrix P could be used to represent the output of unsupervised learning problems. For example, we could obtain latent variables like cluster centers through P.

7.1 Archetypal analysis

Archetypal analysis [Cutler 1994] is an unsupervised learning technique that seeks latent *archetypes*, which are constrained to be convex combinations of the data points. It leads to interpretable archetypes, which is useful for some applications like genomics. Apart from the archetypes being constrained to be convex combinations of the data points, the data points are approximated as convex combinations of the archetypes. The error in representing the data points this way is minimized, leading to the following non-convex optimization problem,

$$\underset{P \in \mathcal{R}^{p \times n}, O \in \mathcal{R}^{n \times p}}{\text{minimize}} ||X - OPX||_F^2 \tag{7.5a}$$

$$\begin{aligned} \text{subject to} \quad & P\mathbb{1}_n = \mathbb{1}_p \\ & O\mathbb{1}_p = \mathbb{1}_n \\ & P_{i,j} \geq 0 \quad O_{i,j} \geq 0 \quad \forall (i,j). \end{aligned} \tag{7.5b}$$

Here, $X \in \mathcal{R}^{n \times d}$ represents the input data points and the output archetypes are obtained by the product PX. One could guess an interesting connection between optimal transport and archetypal analysis, since both the problems work with simplex constraints on the elements of the output matrices. In 7.4, we presented an optimization problem, where the decision variable was a conditional probability matrix P. It is possible to interpret this matrix based on archetypal analysis and represent output points as the product PX. Depending on how P is regularized

in (7.4), PX could represent medoids or cluster centers or projection into a low dimensional subspace etc.

We would like to point out that the weighted average representation, PX, implicitly assumes that the distance between two data points is measured in terms of the Euclidean distance. Other metric distances between points could be considered in the distance matrix D in (7.4). In general, if a non-Euclidean metric is being considered then the output point is represented as the weighted *Fréchet mean* of the input points, where the weights are according to the rows of P. For example, optimal transport distance itself is a metric if the ground distance between the masses of the two distributions is a metric [Santambrogio 2015]. As a concrete example, the data points could be images from two distributions. We could consider each image itself as a probability distribution and use the optimal transport cost between two images i and j as the distance D_{ij} between them in (7.4). Finding Fréchet mean based on optimal tranport metric (in other words, barycenter in a *Wasserstein* space) is an ongoing research topic [Cuturi 2014, Bonneel 2015].

Theorem 7 (Expected output given input)**.** *In an Euclidean space, the archetypal representation, PX, leads to a probabilistic interpretation of the output points. The ith row of PX represents the expected value of the output point corresponding to a given input point x_i, where the expectation is taken over all points, i.e., $(PX)_{i\cdot} = \mathbb{E}(C_i)$. Here, C_i is a random variable denoting the output point corresponding to the input point x_i and it takes values from $\{x_1, x_2, ..., x_n\}$.*

Proof. We recall that the random variable corresponding to the input distribution is A and that corresponding to the output distribution is B. Both take values from $\{x_1, x_2, ..., x_n\}$. The probability distribution of C_i is the same as $P(B|A = x_i)$.

Taking expectation over all data points,

$$\mathbb{E}(C_i) = \sum_j p(B = x_j|A = x_i)x_j = \sum_j \frac{p(x_i, x_j)x_j}{\sum_j p(x_i, x_j)}. \tag{7.6}$$

Now,

$$\sum_j p(x_i, x_j)x_j = \sum_j P_{i,j}\mu_i x_j = \mu_i \sum_j P_{i,j}x_j \tag{7.7}$$

and,

$$\sum_j p(x_i, x_j) = \sum_j P_{i,j}\mu_i = \mu_i \sum_j P_{i,j} = \mu_i \quad \because P\mathbb{1} = \mathbb{1}. \tag{7.8}$$

Hence,

$$\begin{aligned} \mathbb{E}(C_i) &= \sum_j \frac{p(x_i, x_j)x_j}{\sum_j p(x_i, x_j)} = \frac{\mu_i \sum_j P_{i,j}x_j}{\mu_i} \\ &= \sum_j P_{i,j}x_j = (PX)_{i\cdot} \end{aligned} \tag{7.9}$$

□

If one observes continuous optimization based formulations of unsupervised learning problems involving data points $X \in \mathcal{R}^{n\times d}$, we will notice several formulations where the decision variables would also be points Y, usually of the same number and dimension, i.e., $Y \in \mathcal{R}^{n\times d}$. These may represent cluster centers [Hocking 2011] to which the input points are mapped to or projection of the input points into a low dimensional subspace [Candès 2011] etc. Instead of directly optimizing with respect to Y, we propose to optimize with respect to P, where the output points are represented as PX. This leads to a situation where we seek to represent the output points within the convex hull of the input points. We would like to point out that this happens naturally in several unsupervised learning problems like clustering, principal component analysis (PCA) etc. Also, this representation of the output points as a convex combination of input points could be achieved with a *sparse* set of input points. How to achieve this sparse representation in practice? We note that various applications in sparse modeling literature show that enforcing non-negativity constraint leads to sparse solutions [Chen 2014b]. Also, the optimal transport based cost, $\operatorname{tr}\left(P^\top \operatorname{Diag}(\mu) D\right)$, encourages mass to be transported between near data points. Thus, our formulation based on optimal transport and simplex constraints, promotes sparsity in the output matrix P.

7.2 A loss encouraging reduced output variance

In general, an optimization based formulation for machine learning problems like clustering and dimensionality reduction has an objective which is made of two terms, a loss term and a regularization term. Generally, the loss term signifies data similarity and the regularizer signifies prior knowledge on the decision variables. In unsupervised learning, these decision variables could represent cluster centers or data points after projection into a low dimensional subspace or problem specific notions like self expressiveness in subspace clustering. Usually, the data similarity term is formulated using the squared Frobenius norm. For example, in section (7.4) we discuss about a continuous optimization based approach to clustering [Hocking 2011]. Given a matrix $X \in \mathcal{R}^{n\times d}$ representing n data points in a space of dimension d, the output, $Y \in \mathcal{R}^{n\times d}$, represents cluster centers and note that it is a matrix of the same size. They encourage several rows of Y to be the same through suitable regularization.

Thus, the similarity of the output to the input is expressed through the squared Frobenius norm, $\|X - Y\|_F^2$. Based on archetypal representation, we could represent the output points as PX. Thus, the output loss could take the form $\|X - PX\|_F^2$. Now, comes the question of whether PX will represent a different set of points compared to Y. Since, the Frobenius norm based loss is based on Euclidean distance between the input X and output points Y, the optimal output point is bound to be in the convex hull of the input points. Intuitively, we can see this to be true, irrespective of any reasonable regularizer. Thus, representing the output points as convex combinations of data points, results in the same set of output points.

We noted earlier that for several applications, the ground distance matrix D in optimal transport is the squared Euclidean distance. In unsupervised learning, this Euclidean distance could be defined in the space where the points X belong to, thus $D_{ij} = ||X_{i\cdot} - X_{j\cdot}||^2$. With such a ground cost, we make the following observation about optimizing the optimal transport inspired loss.

Theorem 8 (Penalizing output variance)**.** *Suppose, the ground cost D is defined as the squared Euclidean distance between points, i.e., $D_{ij} = ||X_{i\cdot} - X_{j\cdot}||^2$. Then the optimal transport inspired loss $P^\top \operatorname{Diag}(\mu)D$ leads to lesser variance of the coordinates of the output points, compared to the Frobenius norm based loss, $\|X - PX\|_F^2$. I.e., var$(C_{i,j})$ is penalized for all j, where $C_{i,j}$ is one coordinate of C_i.*

Proof.

$$\begin{aligned}
\operatorname{tr}\left(P^T \operatorname{Diag}(\mu)D\right) =& \sum_i \mu_i \sum_j P_{ij}\|x_i - x_j\|^2 \\
=& \sum_i \mu_i \left(\|x_i\|^2 - 2x_i^T \sum_j P_{ij} x_j + \sum_j P_{ij}\|x_j\|^2 \right) \\
=& \sum_i \mu_i \left(\|x_i\|^2 - 2x_i^T \mathbb{E}(C_i) + \mathbb{E}((C_i)^2) \right) \\
=& \sum_i \mu_i \Big(\|x_i\|^2 - 2x_i^T \mathbb{E}(C_i) + \|\mathbb{E}(C_i)\|^2 + \\
& \qquad \mathbb{E}(C_i^\top C_i) - \|\mathbb{E}(C_i)\|^2 \Big) \\
=& \sum_i \mu_i \left(\|x_i - \mathbb{E}(C_i)\|^2 + \sum_j \operatorname{var}(C_{i,j}) \right) \\
=& \sum_i \mu_i \left(\|X_{i\cdot} - (PX)_{i\cdot}\|^2 + \sum_j \operatorname{var}(C_{i,j}) \right).
\end{aligned} \tag{7.10}$$

□

Therefore, the loss $\operatorname{tr}\left(P^\top \operatorname{Diag}(\mu)D\right)$ penalizes the variance of the output points, when compared to squared Frobenius norm.This is true, irrespective of the form of the regularizer. In all the unsupervised learning problems that we discuss here, we have used this loss. In all these problems, this perspective is useful. We point out that this result is of independent interest and could be used in other problems, also.

7.3 K-medoids clustering

The k-medoids problem seeks to partition the data with respect to cluster centers such that one of the data points is itself the cluster center. This is unlike the k-means problem, where the cluster center is the centroid of the data points in that cluster. This problem is NP-hard and we show here a convex approximation to the k-medoids problem.

Given a data set of size n, a convex relaxation of the k-medoids clustering problem could be formulated as follows,

$$\underset{P}{\text{minimize}} \quad \text{tr}\left(P^T \,\text{Diag}(\mu) D\right) \;+\; \lambda \sum_j \|P_{\cdot j}\|_\infty \tag{7.11a}$$

$$\text{subject to} \quad P\mathbb{1} = \mathbb{1} \tag{7.11b}$$

$$P_{i,j} \geq 0 \quad \forall (i,j). \tag{7.11c}$$

Note, if the input data points have uniform probability mass (i.e., $\mu_i = \frac{1}{n}$), this formulation has been already proposed in the name of exemplar clustering by [Elhamifar 2012]. However, the link between optimal transport and this formulation for exemplar clustering has not been recognized in the research community. [Carli 2013] present a convex clustering approach based on optimal transport which is in the form of (7.11) but the l_2 norm of the columns are penalized, instead of l_∞. They do not cite any of the exemplar clustering papers. Also, l_∞ norm is more effective than l_2 norm for this problem [Elhamifar 2012]. Moreover, interpreting the matrix P as a conditional probability matrix, readily leads to the ability to assign non-uniform probability mass to the input points. This could be useful for some applications and it has not been tried earlier, to the best of our knowledge. Also, by representing the output points as convex combinations of the input points, along with the optimal transport inspired loss, makes this formulation as one that penalizes the variance of the output points 7.2. This perspective is new to the exemplar clustering problem.

Note, the above formulation solves the k-medoids problem exactly only when each row of P has exactly one element set to 1 and the other elements being zero. In practice, by minimizing the l_∞ norm of the columns of P and by imposing simplex constraints on the rows of P, we get several columns of P being zeroed out and also, the non-zero values that appear tend to take uniform values. [Elhamifar 2012, §3] shows conditions on the dataset under which the k-medoids could be recovered. In general conditions, we could do one of the following: we could set the cluster centers as the product PX (thus, the output points need not be exactly one of the input points) or for a row i in P, we could select the column j that has the maximum conditional probability P_{ij} as its cluster center. We would recommend PX because we can reason with it in a more principled manner (7).

Among various algorithms to solve the formulation given in [Elhamifar 2012], [Yen 2016] has shown a fast and scalable approach. They first construct the augmented Lagrangian for the problem (7.11) as follows,

$$\mathcal{L}(P,\lambda) = \text{tr}\left(P^T \,\text{Diag}(\mu)\boldsymbol{D}\right) + \lambda \sum_j \|P_{\cdot j}\|_\infty + \boldsymbol{\lambda}^\top (P\mathbb{1} - \mathbb{1}) + \frac{\rho}{2}||P\mathbb{1} - \mathbb{1}||^2 \tag{7.12}$$

where the non-negativity constraint on P is retained. The variables P and $\boldsymbol{\lambda}$ are updated alternatingly as follows,

$$P^{k+1} = \underset{P \geq 0}{\text{argmin}} \mathcal{L}(P, \lambda^k) \tag{7.13a}$$

$$\lambda^{k+1} = \eta(P\mathbb{1} - \mathbb{1}) + \lambda^k \tag{7.13b}$$

using a suitably small step length η. In (7.11) the simplex constraints are enforced on the rows of P, while the l_∞ norm of the columns are minimized. By taking the simplex constraints into the augmented Lagrangian, $\mathcal{L}$ is separable according to the columns of $\mathcal{L}$. Thus, (7.13a) could be optimized in a block coordinate fashion. Since, $\mathcal{L}$ is strongly convex when restricted to a subspace, [Yen 2016] optimize (7.13a) for only one sweep over the blocks in each iteration and achieve a provably convergent and fast algorithm. They also show a greedy method to choose the blocks and achieve further speed-up. We note that according to their formulation input data points are not weighted, which is equivalent to having $\mu_i = \frac{1}{n}$ in (7.11). Since, we have made the observation that this formulation is related to optimal transport, it leads to the possibility of weighting the input points non-uniformly. We will discuss our experiments with optimization algorithms for solving (7.11) in section (7.6).

7.4 Center based clustering

[Hocking 2011] presented a continuous optimization based approach to clustering as follows,

$$\underset{Y \in \mathcal{R}^{n\times d}}{\text{minimize}} \ \frac{1}{2}||X - Y||_F^2 + \lambda \sum_{i<j} w_{ij}||Y_{i\cdot} - Y_{j\cdot}||_q \tag{7.14}$$

where the l_q norm, $q \in 1, 2, \infty$, promotes sparsity in the row differences of Y. Also, in practice weighting the norm, with $w_{ij} = e^{-\gamma\|x_i - x_j\|^2}$ gives better results. Thus, we expect Y to have many rows of the same value and the unique row values of Y represent the cluster centers. Thus, points in X get mapped to cluster centers represented by Y. In our experiments, we first verified that the representation PX could replace Y and recovers the same output cluster centers. Thus, we present the following formulation for clustering,

$$\underset{P \in \mathcal{R}^{n\times n}}{\text{minimize}} \ \text{tr}\left(P^T \,\text{Diag}(\mu) D\right) + \lambda \sum_{i<j} w_{ij}||(PX)_{i\cdot} - (PX)_{j\cdot}||_q \tag{7.15a}$$

$$\text{subject to} \quad P\mathbb{1} = \mathbb{1} \tag{7.15b}$$

$$P_{i,j} \geq 0 \quad \forall (i,j). \tag{7.15c}$$

We immediately notice how similar (7.15) is to (7.11) and how both fit into the framework shown in (7.4). Thus, a different way to regularize the conditional probability matrix P helps us address a different problem in unsupervised learning.

The formulation by [Hocking 2011] has been improved with non-convex regularization terms [Pan 2013, Shah 2017]. For example, [Shah 2017] imposes a robust penalty on the Euclidean distance between rows of Y as follows,

$$\rho(||Y_{i\cdot} - Y_{j\cdot}||_2) \ \text{ where } \ \rho(y) = \frac{\mu y^2}{\mu + y^2}. \tag{7.16}$$

This robust penalty is called the Geman-McClure estimator. They show results on datasets with difficult to separate clusters. We showed in (7.2) that the optimal

transport based loss $\operatorname{tr}\left(P^T \operatorname{Diag}(\mu) D\right)$ leads to reduced variance in the output points. Thus, the following formulation could improve the results of [Shah 2017],

$$\underset{P \in \mathcal{R}^{n \times n}}{\text{minimize}} \operatorname{tr}\left(P^T \operatorname{Diag}(\mu) D\right) + \lambda \sum_{i<j} w_{ij} \rho(||(PX)_{i\cdot} - (PX)_{j\cdot}||_2) \tag{7.17a}$$

$$\text{subject to} \quad P\mathbb{1} = \mathbb{1} \tag{7.17b}$$

$$P_{i,j} \geq 0 \quad \forall (i,j) \tag{7.17c}$$

where ρ is defined in (7.16). In (7.6), we present some experimental results using the optimal transport loss for continuous optimization based clustering. In the next section, we will see one more topic in unsupervised learning that could be addressed through the framework shown in (7.4).

7.5 Kernel PCA and the pre-image problem

Kernel principal component analysis (KPCA) is an effective approach to model data lying on non-linear manifolds [Schölkopf 1999]. The pre-image problem is a central challenge in kernel PCA. For example, KPCA models the data as lying in a linear subspace in a high (possibly, infinite) dimensional feature space, $\mathcal{H}$, which is the RKHS associated with the kernel $k(x,y) = \phi(x)^\top \phi(y)$, where $\phi(x) : \mathcal{X} \to \mathcal{H}$ is usually a nonlinear mapping. Data points are represented and processed in $\mathcal{H}$ and mapping some point $\Psi \in \mathcal{H}$ to the input space $\mathcal{X}$ is the pre-image problem. This inverse mapping may be computationally difficult to evaluate and usually the exact pre-image may not exist. Thus, a suitable pre-image needs to be found for points like Ψ.

Let $X \in \mathcal{R}^{n \times d}$ represent n input points in a d dimensional space. Traditional KPCA performs eigendecomposition in the feature space. In other words, traditional KPCA is PCA performed on the centered kernel matrix as follows,

$$HKH = U \Lambda U^\top \tag{7.18}$$

where, $H = I - \frac{1}{n}\mathbb{1}\mathbb{1}^\top$ is the centering matrix and K is the kernel matrix, i.e., $K_{ij} = \phi(x_i)^\top \phi(x_j)$. U contains the eigenvectors, $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_n$ and Λ is a diagonal matrix containing the eigenvalues, $\lambda_1, \lambda_2, ..., \lambda_n$.

KPCA models the data in a linear subspace of dimension M in feature space. We will now see how this model is maintained in feature space and how a new data point is projected onto this linear subspace. Let us consider the centered feature vectors, $\widetilde{\boldsymbol{\phi}} = [\widetilde{\phi}(x_1), \widetilde{\phi}(x_2), ..., \widetilde{\phi}(x_n)]$, where $\widetilde{\phi}(x_i) = \phi(x_i) - \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$. We will denote $\frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$ as $\bar{\phi}$. The mth component has the form,

$$\boldsymbol{V}_m = \frac{1}{\sqrt{\lambda_m}} \widetilde{\boldsymbol{\phi}} \boldsymbol{\alpha}_m. \tag{7.19}$$

Then, for a new data point x, the projection of $\phi(x)$ onto the mth component has the coefficient,

$$\beta_m = \frac{1}{\sqrt{\lambda_m}} \sum_{i=1}^{n} \alpha_{mi} \widetilde{k}(x, x_i) \tag{7.20}$$

where $\widetilde{k}(x,y) = k(x,y) - \frac{1}{n}\mathbb{1}^\top k_x - \frac{1}{n}\mathbb{1}^\top k_y + \frac{1}{n^2}\mathbb{1}^\top K\mathbb{1}$ and $k_x = [k(x,x_1), k(x,x_2), ..., k(x,x_n)]^\top$. We will also need $\widetilde{k}_x = [\widetilde{k}(x,x_1), \widetilde{k}(x,x_2), ..., \widetilde{k}(x,x_n)]^\top = H\Big(k_x - \frac{1}{n}K\mathbb{1}\Big)$. Thus, the projection of $\phi(x)$ on the M dimensional subspace is of the following form,

$$P_M\phi(x) = \sum_{k=1}^{N} \frac{1}{\lambda_k}(\boldsymbol{\alpha}_k^\top \widetilde{k}_x)(\widetilde{\boldsymbol{\phi}}\boldsymbol{\alpha}_k) + \bar{\phi} = \sum_{i=1}^{n} \gamma_i \phi(x_i) \tag{7.21}$$

for suitably defined γ_i. The pre-image problem in KPCA seeks a z such that $\phi(z)$ is close to $P_M\phi(x)$.

Several approaches have been proposed to tackle the pre-image problem. [Mika 1999] presents an approach for radial basis function (RBF) kernels. They seek to minimize,

$$\rho(z) = ||\phi(z) - P_M\phi(x)||^2. \tag{7.22}$$

For RBF kernels they show a fixed-point iteration to obtain the desired pre-image z as follows,

$$z^{k+1} = \frac{\sum_{i=1}^{n} \gamma_i \exp(-c^{-1}||z^k - x_i||^2) x_i}{\sum_{j=1}^{n} \gamma_i \exp(-c^{-1}||z^k - x_j||^2)}. \tag{7.23}$$

[Bakir 2004] exploits the fact that given a data set, the pairs $(x, \phi(x))$ are available. A map is learnt based on kernel ridge regression, whose performance depends on another kernel, different from that used by the original problem. Their learnt map is of the form $\Gamma(P_M\phi(x)) = [\Gamma_1(P_M\phi(x)), \Gamma_2(P_M\phi(x)), ..., \Gamma_d(P_M\phi(x))]$, where for each dimension in the input space a map is learnt and it is of the following form,

$$z_i = \Gamma_i(P_N\phi(x)) = \sum_{j=1}^{n} \beta_j^i \kappa(P_M\phi(x), P_M\phi(x_j)) \tag{7.24}$$

where κ is a kernel function different from that used by KPCA. Suppose, $B = [\boldsymbol{\beta}^1, ..., \boldsymbol{\beta}^d]$ and $\boldsymbol{\beta}^i = [\beta_1^i, ..., \beta_n^i]^\top$. B is obtained by ridge regression and it is of the following form,

$$B = (\widetilde{K}^\top \widetilde{K} + \gamma I_n)^{-1} \widetilde{K}^\top X \tag{7.25}$$

where $\widetilde{K}_{ij} = \kappa(x_i, x_j)$ and γ is a regularization parameter for kernel ridge regression. Even though this method could work for any kernel function k, it has too many hyperparameters to tune. [Kwok 2004] uses the fact that for isotropic and dot product kernels, if we can find m nearest neighours in feature space of $P_M(\phi(x))$, then it is possible to estimate the distances to these m points in input space. If x_i

is one of the nearest neighbours in feature space, then the pre-image z would be located at a distance d_i from x_i in input space. For isotropic kernels of the form $k(||x_i - x_j||^2)$, this distance d_i between z and x_i could be estimated as follows,

$$k(d_i^2) = \frac{1}{2}\left(\phi(x)^\top \phi(x) + \phi(x_i)^\top \phi(x_i) - \widetilde{d}_i^2\right) \tag{7.26}$$

where $\widetilde{d}_i$ is the distance in feature space and [Kwok 2004] shows that it can be computed for any kernel. As long as the kernel function k is invertible, d_i could be computed. For dot product kernels of the form $k(x_i^\top x_j)$, they exploit the fact that the distance d_i in input space is of the form,

$$d_i^2 = ||z - x_i||^2 = z^\top z + x_i^\top x_i - 2z^\top x_i \tag{7.27}$$

which is in fact just a general relationship and we don't know the pre-image z yet. However, the dot products in input space could be computed from dot products in kernel space for dot product kernels, as long as the kernel function k is invertible. Thus the distances $d^2 = [d_1^2, d_2^2, ..., d_m^2]^\top$ are obtained and z is estimated from $X_m = [x_1, x_2, ..., x_m]$ based on multi-dimensional scaling MDS. Given the singular value decomposition (SVD) of the centered nearest neighbours, $X_m H_m = U\Lambda V^\top = UZ$, the pre-image is obtained as follows,

$$z = U\hat{z} + \bar{x}. \tag{7.28}$$

Here, H_m is an $m \times m$ centering matrix, $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$, $\hat{z} = -\frac{1}{2}\Lambda^{-1}V^\top(d^2 - d_0^2)$ and $d_0^2 = [||Z_{\cdot 1}||^2, ||Z_{\cdot 2}||^2, ..., ||Z_{\cdot n}||^2]^\top$. This approach gives good performance, once the number of nearest neighbours m is selected based on validation data. However, this approach will not be applicable to kernels other than isotropic and dot product kernels and also, working in high input dimensions, with m nearest neighbours from feature space, could lead to poorer results.

Given clean data, all these approaches build a linear subspace model in feature space based on traditional KPCA and denoise a test sample using that model. However, the noise in the test sample will affect how the similarity is measured in feature space. [Nguyen 2009] considers an additional robust similarity measure between the test sample and the denoised output in input space and presents a fixed-point iteration that works with RBF kernels. Other pre-image methods generally tend to improve one of the above methods in some way [Arias 2007], [Rathi 2006], thus retaining the nature of the underlying method. For example, [Zheng 2010] is inspired by both [Mika 1999] and [Kwok 2004] and minimizes (7.23) by constraining the pre-image to be a convex combination of the k nearest neighbours in feature space and by regularizing using labeled samples, if available.

A common feature among pre-image algorithms is the representation of the pre-image as a linear combination of the data points. For [Mika 1999] we can see from (7.23) that z^{k+1} is a linear combination of x_i's, the input points. For [Bakir 2004], we can see from (7.24) and (7.25) that representation of the pre-image as a linear

combination of the data points is again true. For [Kwok 2004], we could define a matrix $B_m = H_m V \Lambda^{-1} \hat{z}$ and see that the pre-image is a linear combination of the points that are nearest neighbours in feature space, $z = X_m B_m + \bar{x}$. It can be easily verified for other algorithms, also [Nguyen 2009], [Abrahamsen 2011], [Honeine 2011] that the pre-image is a linear combination of the data points. Imposing further constraints on the linear combination, like non-negative weights [Kallas 2013] and convex combination [Zheng 2010] have also been explored.

We model the pre-image of a data point according to archetypal analysis, i.e., as a convex combination of the data points. By imposing simplex constraints on the combination weights, we can model the pre-image without a priori fixing the number of data points used in the convex combination. This is an advantage compared to algorithms like [Kwok 2004, Zheng 2010] that have to tune the number of nearest neighbours based on a validation set. Also, in practice the non-negative constraint of the simplex, encourages sparse weights, i.e., a small subset of data points are automatically chosen to represent the pre-image.

Estimating the pre-image for a broad range of kernels is still an open problem. Even though there are many kernels (e.g., string kernels [Giguere 2015]), which may require specialized solutions, addressing the pre-image problem beyond isotropic and dot-product kernels may be useful. For example, finding pre-images for visualization during dictionary learning [Shrivastava 2015] and manifold learning [Arif 2010] could be useful. We saw earlier that [Bakir 2004] presents a general approach by learning a map based on kernel ridge regression. However, there is room for understanding how the various parameters interact (KPCA's kernel, number of components in KPCA, kernel ridge regression's kernel and regularization parameter). Some of the other pre-image algorithms exploit the RBF kernel to design optimization algorithms [Mika 1999], [Nguyen 2009], [Rathi 2006], [Abrahamsen 2011] and hence, are restricted to work with that kernel. Some others exploit some relationship between input space and feature space [Kwok 2004], [Zheng 2010], [Honeine 2011] and hence, are restricted to kernels (isotropic and dot-product) where this relationship is meaningful. To find a way that estimates the pre-image without depending on the kernel, we first observe that the archetypal representation in the input space, carries over to the feature space, as well. Thus, in section (7.5.1) we propose a regularization that again involves the conditional probability matrix P and also, the kernel matrix K, leading to a method that is independent of the kernel function k.

Another desirable property of a pre-image algorithm is a loss function that considers both input space and feature space similarities of the pre-image with the rest of the points. Having both terms has been shown to stabilize the output [Abrahamsen 2011]. [Mika 1999], [Rathi 2006], [Zheng 2010] exclusively models similarity in the feature space. [Kwok 2004] discovers nearest neighbours of the target point based on similarity in feature space alone. The regression coefficients in [Bakir 2004] are dependent on the feature space representation of the target point only. [Nguyen 2009] has a robust similarity term in the input space along with a projection error term in feature space but do not motivate their loss as these two terms stabilizing each other. In our work we propose two similarity terms, one in

input space and one in feature space. The one in the input space is naturally suited for geometric similarity between the data points and the one in the feature space models the desire of discovering a low dimensional linear subspace in feature space.

In the next section, we present our formulation for KPCA and the pre-image problem. We describe interesting properties like flexible choice of the kernel and working with robust geometric distance between points. Later, we also describe an efficient optimization approach to solve the formulation and present results.

7.5.1 Our Approach

We observe that there is a strong geometric flavor in the pre-image problem and several existing methods are based on explicit geometric reasoning in their formulation [Kwok 2004], [Nguyen 2009], [Honeine 2011]. We bring in geometric notions through the optimal transport based loss. Thus, our approach to nonlinear dimensionality reduction takes the following form,

$$\underset{P}{\text{minimize}} \quad \text{tr}\left(P^\top \text{Diag}(\mu) D\right) \; + \; \lambda \| H P K_c^{\frac{1}{2}} \|_p^p \tag{7.29a}$$

$$\begin{aligned} \text{subject to} \quad & P\mathbb{1} = \mathbb{1} \\ & P_{i,j} \geq 0 \quad \forall (i,j). \end{aligned} \tag{7.29b}$$

Again, we rightaway notice the similarity to the form given in (7.4). Also, $H = I - \frac{1}{n}\mathbb{1}\mathbb{1}^\top$ is the centering matrix, $K_c = HKH$ is the centered Kernel matrix and $\|\cdot\|_p$ is the Schatten p-norm. As before, the pre-images of the data points after projection onto the linear subspace in feature space, are modeled according to archetypal analysis, as follows,

$$Z = PX. \tag{7.30}$$

According to (7), $(PX)_{i\cdot}$ is the expected output point for input point x_i. This is a property expected from the pre-image of a point x_i, in at least some applications.

Now, we will motivate our regularization term, $\|HPK_c^{\frac{1}{2}}\|_p^p$. In traditional KPCA, the data is modeled by the top eigenvectors of the centered kernel matrix $K_c = HKH$. Even though $\mathcal{H}$ is an infinite dimensional space, any finite dataset spans a subspace of finite dimension [Bakir 2004]. Hence, if the dataset exibits a low-dimensional structure in feature space, traditional KPCA is able to model it with a small set of eigenvectors. Similarly, we want the matrix, $HPK_c^{\frac{1}{2}}$, to be of low rank. We will see how this leads to a low-dimensional representation in feature space. We have the data points $X = [x_1^\top, x_2^\top, ..., x_n^\top]^\top$ and their representation in feature space $\mathcal{H}$, $\Phi(X) = [\phi(x_1)^\top, \phi(x_2)^\top, ..., \phi(x_n)^\top]^\top$. We would like the archetypes in feature space $(P\Phi(X))$ to lie in a low dimensional subspace. In other words, the centered vectors $HP\Phi(X)$ could be made to lie in a low dimensional linear subspace by penalizing the rank of the matrix, $HP\Phi(X)$. Matrix rank is combinatorial in nature and the nuclear norm $(\|\cdot\|_*)$ and in general the Schatten p-norm $(\|\cdot\|_p)$ is a continuous valued surrogate for the rank. In fact, the nuclear norm is the Schatten p-norm with $p = 1$. However, we cannot directly work with $\Phi(X)$ which could be infinite

dimensional. Instead, we exploit the kernel trick and obtain a finite dimensional matrix as follows,

$$\|HP\Phi(X)\|_p^p = \operatorname{tr}\left(HP\Phi(X)\Phi(X)^\top P^\top H\right)^{\frac{p}{2}} = \operatorname{tr}\left(HPKP^\top H\right)^{\frac{p}{2}} = \|HPK^{\frac{1}{2}}\|_p^p. \tag{7.31}$$

Hence, minimizing the rank of $HPK^{\frac{1}{2}}$ could lead to the desired low-dimensional representation in $\mathcal{H}$. We work with $HPK_c^{\frac{1}{2}}$ which is better from a numerical implementation point of view because in practice the dynamic range of non-zero eigenvalues is lesser for K_c. Generally, K_c is of size $(n \times r)$, with $r < n$. Hence, $HPK_c^{\frac{1}{2}}$ is of size $(n \times r)$. This will be useful for designing efficient optimization algorithms.

In recent years, seeking a low rank representation in feature space has been explored by other works [Garg 2016, Fan 2018]. However, these works are not scalable because one of the decision variables in their formulation is the entire kernel matrix itself. Thus, the per iteration cost of their algorithms is $O(n^3)$. We explore archetypal representation in the feature space and achieve a per iteration complexity of $O(r^3)$. Also, we have an objective function that simultaneously considers similarity in input space ($\operatorname{tr}\left(P^\top \operatorname{Diag}(\mu)D\right)$) and feature space ($\|HPK^{\frac{1}{2}}\|_p^p$), which could lead to better results.

We observe that the optimization problem with the objective 7.29a and constraints 7.29b is independent of what kernel is used to define the kernel matrix K. Thus, our approach may work in any application where the pre-image is reasonably represented as a convex combination of data points. We have already seen that general pre-image algorithms represent the pre-image as a linear combination of data points and the advantage of seeking a convex combination of data points has also been studied [Zheng 2010].

Our formulation allows flexibility to suitably define the geometry of the input space. The optimal transport based loss, $\operatorname{tr}\left(P^\top \operatorname{Diag}(\mu)D\right)$, allows freedom in the definition of the ground distance matrix D. To the best of our knowledge, apart from [Nguyen 2009], which considers the robust Geman-McClure distance in the input space, our work is one of the first to allow a flexible geometry in the input space.

We would like to point out that our algorithm involves tuning of the regularization parameter λ. A regularization path of this parameter needs to be considered and a final value should be chosen. The criterion for choosing the appropriate value of λ depends on the application. For example, a supervised denoising application could use the reconstruction error on a validation data set. We note that the same procedure is needed for traditional KPCA for choosing the number of components. In fact in that approach, a simultaneous search for number of components along with the parameters of the pre-image algorithm is required [Alam 2014]. In all approaches, the parameters of the kernel has to be chosen by some procedure, which we will describe in the experiments section.

7.5.2 Optimization

Like in other problems described in this chapter, P, the conditional probability matrix, is the decision variable. In the objective, the Schatten p-norm is not smooth. Together with the simplex constraints on the rows of P, this presents a challenge to design an optimization algorithm. We take an approach where we smooth the Schatten p-norm and thus, have a smooth version of the objective. This enables the use of accelerated projected gradient algorithms like FISTA for the convex case and a general majorization-minimization approach for any value of p.

Given a $P \in \mathcal{R}^{n\times n}$, $P\mathbb{1} = \mathbb{1}$ and $P \geq 0$, for easier notation we will work with a matrix M, where $M = HPK_c^{\frac{1}{2}}$, where $H = I - \frac{1}{n}\mathbb{1}\mathbb{1}^\top$ is the centering matrix and $K_c^{\frac{1}{2}} \in \mathcal{R}^{n\times p}$ is the square root of the centered kernel matrix. The Schatten p-norm of M raised to the power p is $\sum_k \sigma_k(M)^p = \operatorname{tr}(M^\top M)^{\frac{p}{2}}$, which is not smooth and is only convex for $p \geq 1$. We consider the following smoothing of the Schatten p-norm [Zhang 2010],

$$\sum_k \sigma_k(M)^p = \operatorname{tr}(M^\top M)^{\frac{p}{2}} = \operatorname{tr}(\varepsilon^2 I + M^\top M)^{\frac{p}{2}}. \tag{7.32}$$

This leads to the following potentially non-convex smooth objective for the optimization problem,

$$\operatorname{tr}(P^\top \operatorname{Diag}(\mu)D) + \lambda \operatorname{tr}(\varepsilon^2 I + M^\top M)^{\frac{p}{2}}. \tag{7.33}$$

Given the potential non-convexity of the optimization problem, we seek a majorization-minimization algorithm [Lange 2000]. Specifically, we derive a suitable variational majorant surrogate and use it as described in section 2.2 of [Mairal 2013]. In brief, for a real-valued function h which is to be minimized over a convex set $\mathcal{X}$, let f be a real-valued function defined on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y}$ is another convex set. Under reasonable smoothness assumptions and μ-strong convexity of f, $\forall x \in \mathcal{X}$, given any $y \in \mathcal{Y}$, the following function is a majorant surrogate for f at some (κ_1, κ_2), $\hat{f} : x \to f(x, \kappa_2)$, where $\kappa_2 = \arg\min_y f(\kappa_1, y)$. Now, we will derive a variational majorant surrogate along these lines for the smooth objective function (7.33). First, we make the following observation,

$$\operatorname{tr}(\varepsilon^2 I + M^\top M)^{\frac{p}{2}} = \inf_B p \operatorname{tr}\left(B(\varepsilon^2 I + M^\top M)^{\frac{1}{2}}\right) + (1-p)\operatorname{tr} B^{\frac{-p}{1-p}}. \tag{7.34}$$

This leads to the folllowing form for the smooth objective: $\operatorname{tr}(P^\top \operatorname{Diag} \mu D) + \lambda \inf_B p \operatorname{tr}\left(B(\varepsilon^2 I + M^\top M)^{\frac{1}{2}}\right) + (1-p)\operatorname{tr} B^{\frac{-p}{1-p}}$. Hence, it is possible to define a function $f(P, B)$ as follows,

$$f(P, B) = \operatorname{tr}(P^\top \operatorname{Diag} \mu D) + \lambda p \operatorname{tr}\left(B(\varepsilon^2 I + M^\top M)^{\frac{1}{2}}\right) + (1-p)\operatorname{tr} B^{\frac{-p}{1-p}}. \tag{7.35}$$

For a given P_t and $M_t = HP_tK_c^{\frac{1}{2}}$, the following function is a majorant at (P_t, B_t)

for f: $f(P, B_t)$, where B_t is obtained as follows,

$$\begin{aligned} B_t =& \arg\min_B f(P_t, B) = \arg\min_B \; p \operatorname{tr}\left(B(\varepsilon^2 I + M_t^\top M_t)^{\frac{1}{2}}\right) + (1-p) \operatorname{tr} B^{\frac{-p}{1-p}} \\ =& (\varepsilon^2 I + M_t^\top M_t)^{-\frac{p}{2}}. \end{aligned} \tag{7.36}$$

Hence, our optimization approach is a block coordinate descent procedure with two blocks (P, B) as described in Algorithm (10). In each iteration, B_t is obtained in closed form as shown in (7.36) and P_t is obtained by minimizing the function in (7.35) for a fixed $B = B_t$, subject to simplex constraints. Currently, this P_t is obtained using FISTA. Suppose (P_t, B_t) converge to (P^*, B^*), then P^* is a local optimum for our optimization problem with the smooth objective (7.33).

Finding P_t for a given B_t requires a projected gradient descent procedure. The projection onto the simplex constraints is carried out using an efficient off-the-shelf method [Condat 2016]. In order to solve the optimization sub-problem to find P_t for a given B_t, the gradient of $f(P, B_t)$ with respect to P is required. The derivative with respect to P could be obtained from the derivative with respect to M through the chain rule. Thus, what we need is the derivative of $\operatorname{tr}\left(B_t(\varepsilon^2 I + M^\top M)^{\frac{1}{2}}\right)$ with respect to M, which is as follows,

$$2MU\bar{B}U^\top \tag{7.37}$$

where U is a matrix whose columns are the eigenvectors $\boldsymbol{u}_i$ of the eigendecomposition of $M^\top M$. $\bar{B}_{i,j} = \frac{\tilde{B}_{i,j}}{(\varepsilon^2+\sigma_i^2)^{\frac{1}{2}}+(\varepsilon^2+\sigma_j^2)^{\frac{1}{2}}}$, where $\tilde{B} = U^\top BU$. Thus the per iteration complexity of computing the gradient is dominated by the eigendecomposition of $M^\top M$, which is $O(r^3)$. Thus by working with the square root of the kernel matrix ($\in \mathcal{R}^{n\times r}$ with $r < n$), our approach has a more desirable computational complexity. The other computationally demanding step in our approach is the projection into the simplex, to ensure the simplex constraints on the rows of P. This operation is of complexity $O(n \log n)$ for each of the n rows and it is trivially parallelizable over the rows. We generate P's over the regularization path of λ values. The computational load of generating a regularization path over λ values, could be considerably reduced by warm starting with the P obtained with previous lesser λ value.

We present in the appendix, the derivation of equation (7.37). In the appendix, we also show how to obtain the derivative for weighted Schatten p-norm where the singular values are non-uniformly weighted [Gu 2017] or where only the lower singular values are penalized [Oh 2016]. These are novel results and it can be readily applied to other problems, involving the Schatten p-norm.

The best performing P (say, P^*) could be obtained based on performance on a validation data set. Let us denote the corresponding regularization parameter as λ^*. Given the optimal P^*, the pre-image of the data points is obtained as $Z = P^*X$.

7.5.3 Out-of-sample extension

So far, we have shown how a set of data points can be projected onto a low dimensional linear subspace in feature space and how to obtain the pre-images as

Algorithm 10 Block coordinate descent scheme

1: input $P_0 = I$, $\lambda = 10^{-3}$, $K_c \in \mathcal{R}^{n \times p}$
2: **while** regularization path left to be explored **do**
3: choose λ_c to explore based on rank of $HP_\lambda K_c^{\frac{1}{2}}$ and performance on validation set of previous λ's.
4: $\lambda \leftarrow \lambda_c$
5: warm start P_0 with P_λ corresponding to largest earlier λ
6: **for** $n = 1, 2, 3, ...$ **do**
7: compute B_n according to equation (7.36) using P_{n-1}.
8: compute P_n by minimizing (7.35) subject to simplex constraints (7.29b), using B_n.
9: **end for**
10: save optimal P as P_{λ_c}.
11: **end while**

a convex combination of the data points. In many applications, we seek the pre-image of a new test point, say $x_{oos} \in \mathcal{R}^d$. It can be readily done with the following approach. We will work with a conditional probability matrix $P_{oos} \in \mathcal{R}^{(n+1)\times n}$ as the optimization variable. P_{oos} is warm started as follows: The first n rows are assigned P^*, obtained in the previous section. The last row ($\in \mathcal{R}^{n\times 1}$) is initialized by zero except for k nearest neighbours to the test point, each having uniform value of $\frac{1}{k}$. We choose $k = 10$ but this is to only give a warm start and it is not critical. A centered kernel square root $K_{c,oos} \in \mathcal{R}^{(n+1)\times p}$ is computed in an incremental manner with respect to $K_c^{\frac{1}{2}}$. The pairwise cost matrix is also extended as $D_{oos} \in \mathcal{R}^{(n+1)\times n}$. Then we optimize for a fixed value of $\lambda = \lambda_*$ using the optimization scheme mentioned in the previous section is run. Based on the optimal P^*_{oos}, one could get the pre-image of the test point as $\left(P^*_{oos}X\right)_{(n+1)\cdot}$.

7.5.4 Robust pairwise cost

In the previous section, we showed our approach for out-of-sample extension, where the pre-image of a test point is a convex combination of the training points. There are applications where we want to consider the test point also, as part of the convex combination. For example, when the training points are clean samples and the test point is a noisy one, to be denoised. Among pre-image algorithms, [Nguyen 2009] is one work that has considered this possibility in a more principled manner. They consider the robust Geman-McClure distance between the noisy test point and the pre-image of the denoised point in feature space. In our approach, we will work with a conditional probability matrix $P_\rho \in \mathcal{R}^{(n+1)\times(n+1)}$ as the optimization variable. In other words, we let the test point also to define the linear subspace that we are seeking in feature space. P_ρ is warm started similar to the previous section. Also, the centered kernel matrix takes the same form as in the previous section. It is in the design of the pairwise cost matrix $D_\rho \in \mathcal{R}^{(n+1)\times(n+1)}$, where we exploit the ability

to suitably define the geometry in the input space, which is afforded by optimal transport. We could define the pairwise cost between the points based on a robust metric (say, Euclidean distance which corresponds to 1-Wasserstein distance). By this approach, the linear subspace that we seek is not adversely affected by the noisy test point and at the same time, we get a pre-image which considers the test point in the convex combination. Let $X_{oos} = [X^\top, x_{oos}^\top]^\top$, then the desired pre-image is $(P_\rho^* X_{oos})_{(n+1)}$.

7.6 Experiments

For the convex relaxation of the k-medoids problem (7.3), we tried several optimization algorithms: FISTA with a smooth objective (obtained by smoothing the l_∞ norm) with simplex constraints; block coordinate descent with the simplex constraints encoded as a soft quadratic penalty with randomized and greedy selection of blocks; an ADMM based approach according to [Parikh 2014, §3.1]. We found the ADMM based approach to be fastest, with a suitable choice of the augmented Lagrangian parameter. We have observed that the block coordinate descent formulation with quadratic penalty makes quick initial progress but is ill-conditioned as the algorithm progresses. Ill-conditioning of the quadratic penalty has been recognized by the research community [MS&E318] and expressing the constraints through the augmented Lagrangian is a solution. Thus, the augmented Lagrangian with truncated block coordinate descent in [Yen 2016] is indeed an elegant algorithm for this problem.

For the continuous clustering formulation, we experimented with both the Frobenius norm based loss and the optimal transport based loss, along with l_1 and l_2 regularization. l_2 norm is rotation invariant and suitable in a general setting [Hocking 2011]. We show results on two dimensional toy problems in figure (7.1). We observe that in both cases points are grouped together into the same clusters. However, in the case of the optimal transport loss, the cluster center is closer to one or more of the existing data points. It will be an interesting exercise to test the optimal transport loss along with the robust regularization given in [Shah 2017]. Since, they showed promizing results with difficult to separate clusters, our variance penalizing loss could further improve the results.

A set of good baselines for the KPCA pre-image problem are: fixed point iteration [Mika 1999], MDS based approach [Kwok 2004], learning pre-image map using kernel ridge regression [Bakir 2004] and the approach based on a robust loss [Nguyen 2009]. It is not clear whether the code shared by [Nguyen 2009] implements the algorithm described in the paper and we have not used it in our comparisons. In our formulation, we have shown results with nuclear norm, i.e., $p = 1$ for the Schatten p-norm.

First, we test on a two-dimensional problem, 250 points generated along the circumference of a circle of radius 5, perturbed by Gaussian noise of standard deviation 1. The polynomial kernel is very suitable for this problem and we have tested our approach with both polynomial and RBF kernels. Among the baseline

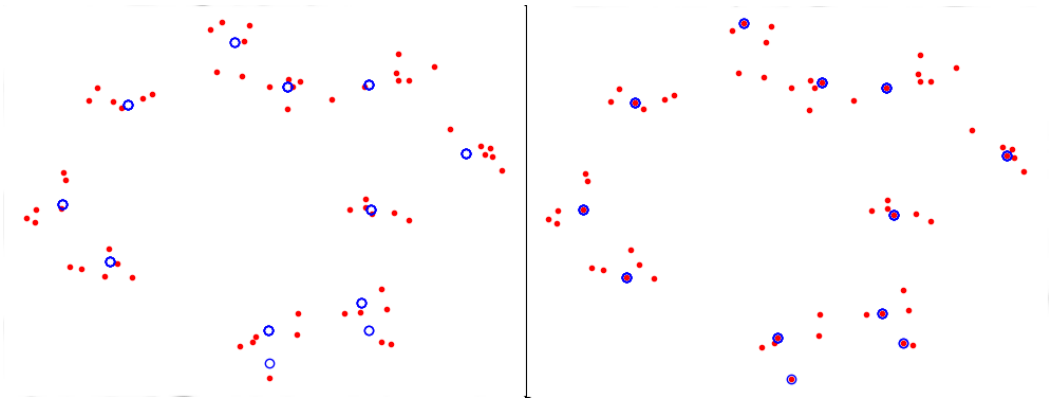

Figure 7.1: Two dimensional data with 60 points. (left) frobenius norm based loss, (right) optimal transport based loss. Optimal transport based loss penalizes variance of the output points and those points are close to or coincide with input points.

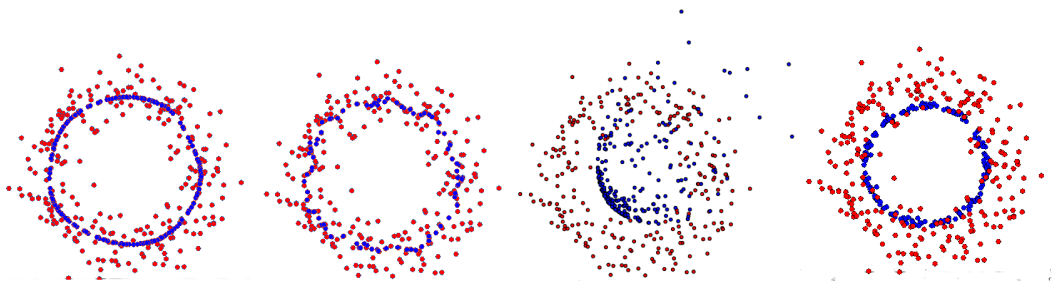

Figure 7.2: From left to right: [Mika 1999], [Kwok 2004], [Bakir 2004], our approach. [Mika 1999] and [Kwok 2004] use RBF kernel. [Bakir 2004] and ourselves use polynomial kernel.

algorithms, [Kwok 2004] and [Bakir 2004] can tackle the polynomial kernel. Currently, the implementation of [Kwok 2004] that we have only supports RBF kernel [stprtool]. The implementation of [Bakir 2004] is from scikit-learn [Pedregosa 2011] and it supports both RBF and polynomial kernels. Results are shown in figure 7.2. The best parameter setting for each algorithm is used. We found [Mika 1999] and [Kwok 2004] to be very sensitive to the bandwidth of the RBF kernel. We had to test 10th percentile, median and 90th percentile of the pairwise distances and take the best result. Compared to this our algorithm is very robust to choice of the bandwidth parameter. In all our experiments it has worked with bandwidth set to the median pairwise distance. For different choices of the bandwidth parameter, our algorithm gives acceptable results with a different λ. The implementation of the polynomial kernel based KPCA does not allow us to tune any parameter to improve the result of [Bakir 2004].

We also show results with the USPS dataset, which consists of 1100 instances

Digit	Our approach	MDS approach
0	11.998	12.1319
1	12.797	13.291
2	11.645	11.7458
3	11.7013	11.8094
4	11.8597	12.0211
5	11.7803	11.881
6	11.8273	11.9465
7	12.3235	12.461
8	11.6172	11.6834
9	11.9246	12.0224

Table 7.1: USPS Dataset: PSNR of denoised image for Gaussian noise.

Digit	Our approach	MDS approach
0	12.3546	12.5631
1	13.723	14.1372
2	11.7791	11.9145
3	11.8962	12.03
4	12.1903	12.3596
5	11.9898	12.135
6	12.1054	12.2716
7	12.7595	13.0382
8	11.6714	11.8569
9	12.2911	12.3471

Table 7.2: USPS Dataset: PSNR of denoised image for salt & pepper noise.

for each of the ten digits (0...9). The digit images are in binary format, with 0 for background and 1 for foreground. We add noise to all the images and we test the ability of the algorithms to denoise the images by finding a linear subspace in feature space. We have tested with two types of noise: Gaussian ($\mu = 0$ and $\sigma = 0.4$) and salt-&-pepper ($p = 0.1$). In our experiments, the MDS based approach consistently performed better than the other baselines. Hence, we only compare with [Kwok 2004]. In the tables (7.1) and (7.2) we show PSNR of the denoised images, for Gaussian and salt-&-pepper noises, respectively. For both algorithms, we show the results corresponding to the best parameter setting. For our approach, we have to set the regularization parameter λ and for the MDS approach, we have to choose the number of components in KPCA and the number of nearest neighbours. The bandwidth parameter of the RBF kernel is set as the median of all the pairwise Euclidean distances.

As future work regarding our pre-image work we have the following thoughts.

Computing the pre-image as a weighted average in Euclidean input space seems to be a prevalent feature in pre-image literature. Our formulation readily allows

non-Euclidean distances in input space. We would like to leverage the advancement in computing Fréchet mean in different non-Euclidean spaces and see the effectiveness of our approach.

Surprisingly, many kernel PCA applications involve the RBF kernel. In this setting, the MDS based approach [Kwok 2004] performs reasonably well. It has been observed that this approach suffers in high-dimensional input spaces [Zheng 2010]. It would be good idea to compare our approach with it in such a setting. For KPCA to be meaningful it is enough if the kernel is a *universal* kernel [Micchelli 2006] and we would like to explore such kernels. Some possible kernels are those which consider the non-differentiable histogram-of-gradient (HOG) feature inside the RBF kernel [Huang 2011] and kernels based on multiple kernel learning [Lin 2011, Shrivastava 2015].

Several manifold learning techniques could be formulated in the form of a kernel PCA problem [Mohri 2012, §12.3]. The kernel function for these formulations is not in isotropic or dot-product form. There is no flexible pre-image algorithm that will work for all of these manifold learning techniques. We would like to try our approach with these manifold learning techniques and see how we perform against specialized algorithms like [Cloninger 2017].

7.7 Concluding remarks

We have shown how the optimal transport inspired loss, $\operatorname{tr}\left(P^\top \operatorname{Diag}(\mu) D\right)$ along with archetypal representation of the output points could be used fruitfully in unsupervised learning problems. We have shown that when the pairwise distances are set as squared Euclidean distances, this loss penalizes the variance of the output points. This is a general result that could be useful for other problems, also. We have shown how several unsupervised learning algorithms could be all seen as performing different sparse regularizations on the conditional probability matrix P. For the problem of kernel PCA and associated pre-image estimation, we have proposed a novel low rank regularizer. Some of the advantages of our approach is that it has only one parameter to tune and it is not tied to a particular kernel. From an optimization algorithm point of view, we have shown a majorization-minimization scheme that works with a smooth version of the Schatten p-norm. This smooth Schatten p-norm based scheme is of independent interest and it could be used for other problems involving low rank regularization.

Chapter 8

Conclusion

We have discussed our numerical experience with Newton methods for MAP inference and how it compares with state-of-the-art first order methods. With an efficient parallel implementation the Newton methods that we presented could prove to be very useful. That is because these methods come with a convergence guarantee and other properties regarding the optimum of the LP relaxation. We have indicated various avenues for future research. We especially find research into tighter relaxations for more accurate MAP inference to be very relevant.

Continuous optimization based formulations for unsupervised learning has shown promising results. We have added further understanding and new models to that literature. The result that optimal transport based loss leads to reduced variance in the output points holds independent interest and could help other algorithms. The formula for the gradient of the smooth Schatten p-norm holds independent interest, also and could be used within other optimization schemes.

At this juncture, we would like to highlight the importance of numerical linear algebra for various problems in machine learning and computer vision. Usually, these problems involve numerical optimization. There is scope to exploit application specific structure to design preconditioners and to improve other aspects of solving linear systems. There are strong links between numerical linear algebra and graphical models that could be further explored. For example, inference in Gaussian graphical models. Also, eigenvalue problems present interesting avenues for research. We could appreciate this topic while discussing about low-rank regularization in chapter (7).

Chapter 9

Appendix

9.1 Convex Optimization

9.1.1 Halfspace

A *hyperplane* $\{x|a^\top x = b,\ a \in \mathcal{R}^n,\ a \neq 0,\ b \in \mathcal{R}\}$ divides $\mathcal{R}^n$ into two *halfspaces*. Given, $a \in \mathcal{R}^n,\ a \neq 0$, the following set is one of the two closed halfspaces,

$$\{x|a^\top x \leq b\} \tag{9.1}$$

9.1.2 Dual norm

Let $||\cdot||$ be a norm. The associated dual norm, $||\cdot||_*$, is defined as,

$$||z||_* = \sup\left\{z^\top x \ \ \| \ \ ||x|| \leq 1\right\} \tag{9.2}$$

9.1.3 Fenchel conjugate

For a function $f : \mathcal{R}^n \to \mathcal{R}$, its Fenchel conjugate is defines as,

$$f^*(y) = \sup_{x \in \mathcal{R}^n} \left\{y^\top x - f(x)\right\} \tag{9.3}$$

9.1.4 α strong convexity

A function $f : \mathcal{R}^n \to \mathcal{R}$ is α-strongly convex with respect to the norm $||\cdot||_p$ if it satisfies the following condition $\forall x, y$,

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\alpha}{2}||x - y||^2 \tag{9.4}$$

For twice differentiable functions, this is equivalent to,

$$\nabla^2 f(x) \succeq \alpha I \quad \forall x \tag{9.5}$$

9.1.5 L smoothness

A function $f : \mathcal{R}^n \to \mathcal{R}$ is L-smooth with respect to the norm $||\cdot||_p$ if it satisfies the following condition $\forall x, y$,

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}||x - y||^2 \tag{9.6}$$

9.1.6 Strict convexity

9.1.7 Prox function for the simplex

Given a set satisfying the simplex constraint,

$$\left\{\mathbf{w} = (w_1, ..., w_n) \in \mathcal{R}^n \| \mathbf{w} \geq 0, \mathbf{w}^\top \mathbb{1} = 1\right\} \tag{9.7}$$

then, the function,

$$d(\mathbf{w}) = \log n + \sum_{i=1}^{n} w_i \log w_i \tag{9.8}$$

is a prox function of the set with respect to the l_1 norm [Nesterov 2005].

9.1.8 M norm of a vector

Given a preconditioner M, algorithms like the Steihaug method measure distances using the *M norm*,

$$||x||_M = \sqrt{x^\top M x}. \tag{9.9}$$

This vector norm weighted by a matrix also goes by the name of scaled norm.

9.2 Numerically stable sum-product computation

When the damping parameter is increased to high values (low smoothness), the terms present in equations (3.36) and (3.37) will all take zero value and the summations in these equations will be of value zero, which is erroneous. This is because these terms are in fact exponentials of terms corresponding to the nodes and cliques. So, $\psi_c(\boldsymbol{x}_c)$ and $\psi_i(x_i)$ are of the form $\exp(\theta_c(\boldsymbol{x}_c))$ and $\exp(\theta_i(x_i))$, respectively, in a suitably defined graphical model. This numerical issue is addressed by storing offsets $(M_{bc}(\boldsymbol{x}_n))$ for each term of the message $m_{b\to c}(\boldsymbol{x}_n)$. Hence, $m_{b\to c}(\boldsymbol{x}_n)$ in equation (3.36) is computed as follows,

$$m_{b\to c}(\boldsymbol{x}_n) = \sum_{\boldsymbol{x}_{b\setminus n}} \exp\big[\theta_b(\boldsymbol{x}_b) + \sum_{i\in b\setminus n} \theta_i(x_i) + \log(m_{a\to b}(\boldsymbol{x}_{b\setminus n})) - M_{bc}(\boldsymbol{x}_n)\big] \tag{9.10}$$

$$\text{where} \quad M_{bc}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{b\setminus n}} \big[\theta_b(\boldsymbol{x}_b) + \sum_{i\in b\setminus n} \theta_i(x_i) + \log(m_{a\to b}(\boldsymbol{x}_{b\setminus n}))\big] \tag{9.11}$$

This ensures that at least one term in the summation is of value 1, hence, the summation is assured to be non-zero. Later, the required marginals are computed as $p_i^c(x_i) = \frac{\psi_i(x_i)}{Z} \sum_{\boldsymbol{x}_n} \big[(\prod_{j\in n\setminus i} \psi_j(x_j)) m_{b\to c}(\boldsymbol{x}_n)\big]$, where Z ensures that the marginals sum to one. In this expression each summand is computed as $\big[\sum_{j\in n\setminus i} \theta_j(x_j) + \log(m_{b\to c}(\boldsymbol{x}_n)) + M_{bc}(\boldsymbol{x}_n)\big]$. This approach can be carefully extended to equation (3.37), also. We will need to maintain two offsets, $M_{bc}(\boldsymbol{x}_n)$ as before and also, $\overline{M}$ corresponding to the second sum in equation (3.37).

9.3 Gradient of Variational Majorant

We have for any A which is symmetric definite with eigenvectors u_i, and eigenvalues λ_i, and any H which is symmetric:

$$\begin{aligned}&(A+H)^{1/2} - A^{1/2}\\ &= \sum_{i,j} \frac{u_i^\top H u_j}{\lambda_i^{1/2} + \lambda_j^{1/2}} u_i u_j^\top + O(\|H\|^2)\end{aligned} \tag{9.12}$$

Thus for our matrix M, which is a rectangular matrix, we compute all the eigenvalues σ_i^2 and the corresponding eigenvectors u_i of $M^\top M$. Note, this includes the zero eigen values, also. Hence, equation (9.12) could be written for $\varepsilon^2 I + M^\top M$ as follows,

$$\begin{aligned}&(\varepsilon^2 I + (M+\Delta)^\top (M+\Delta))^{\frac{1}{2}} - (\varepsilon^2 I + M^\top M)^{\frac{1}{2}}\\ &\qquad = \sum_{i,j} \frac{u_i^\top (M^\top \Delta + \Delta^\top M) u_j}{(\varepsilon^2+\sigma_i^2)^{\frac{1}{2}} + (\varepsilon^2+\sigma_j^2)^{1/2}} u_i u_j^\top + O(\|\Delta\|^2)\end{aligned} \tag{9.13}$$

In order to obtain the gradient of $\operatorname{tr}[B_t(\varepsilon^2 I + M^\top M)^{1/2}]$, we have,

$$\begin{aligned}&\operatorname{tr}[B_t(\varepsilon^2 I + (M+\Delta)^\top (M+\Delta))^{1/2}] - \operatorname{tr}[B_t(\varepsilon^2 I + M^\top M)^{1/2}]\\ &= \sum_{i,j} \frac{u_i^\top (M^\top \Delta + \Delta^\top M) u_j}{(\varepsilon^2+\sigma_i^2)^{1/2} + (\varepsilon^2+\sigma_j^2)^{1/2}} u_j^\top B_t u_i + O(\|\Delta\|^2)\\ &= \operatorname{tr} \Delta^\top \Big(\sum_{i,j} \frac{u_j^\top B_t u_i}{(\varepsilon^2+\sigma_i^2)^{1/2} + (\varepsilon^2+\sigma_j^2)^{1/2}} [M u_j u_i^\top + M u_i u_j^\top] \Big) + O(\|\Delta\|^2)\\ &= 2 \operatorname{tr} \Delta^\top \Big(\sum_{i,j} \frac{u_j^\top B_t u_i}{(\varepsilon^2+\sigma_i^2)^{1/2} + (\varepsilon^2+\sigma_j^2)^{1/2}} M u_j u_i^\top \Big) + O(\|\Delta\|^2)\end{aligned} \tag{9.14}$$

Taking the limit $\Delta \to 0$, the gradient of $\operatorname{tr}[B_t(\varepsilon^2 I + M^\top M)^{1/2}]$ at M is

$$\begin{aligned}&2 \sum_{i,j} \frac{u_j^\top B u_i}{(\varepsilon^2+\sigma_i^2)^{1/2} + (\varepsilon^2+\sigma_j^2)^{1/2}} M u_j u_i^\top\\ &= 2 M U \bar{B} U^\top\end{aligned} \tag{9.15}$$

The last expression is obtained by grouping terms and by evaluating the sums as matrix products. Here U is the matrix whose columns are the eigenvectors u_i, $\bar{B}_{i,j} = \frac{\tilde{B}_{i,j}}{(\varepsilon^2+\sigma_i^2)^{1/2} + (\varepsilon^2+\sigma_j^2)^{1/2}}$ and $\tilde{B} = U^\top B U$.

Bibliography

[Abrahamsen 2011] T. J. Abrahamsen and L. K. Hansen. *Regularized pre-image estimation for kernel PCA de-noising*. Journal of Signal Processing Systems, vol. 65, no. 3, pages 403–412, 2011. (Cited on page 90.)

[Absil 2009] P-A Absil, R. Mahony and R. Sepulchre. Optimization algorithms on matrix manifolds. Princeton University Press, 2009. (Cited on page 79.)

[Agarwal] S. Agarwal, K. Mierle and Others. *Ceres Solver*. http://ceres-solver.org. (Cited on page 65.)

[Ajanthan 2017] T. Ajanthan, A. Desmaison, R. Bunel, M. Salzmann, P. H. S. Torr and M. P. Kumar. *Efficient linear programming for dense CRFs*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2934–2942, 2017. (Cited on page 71.)

[Alam 2014] M. A. Alam and K. Fukumizu. *Hyperparameter selection in kernel principal component analysis*. Journal of Computer Science, vol. 10, pages 1139–1150, 2014. (Cited on page 92.)

[Amari 2007] S-i. Amari and H. Nagaoka. Methods of information geometry, volume 191. American Mathematical Soc., 2007. (Cited on page 79.)

[Arias 2007] P. Arias, G. Randall and G. Sapiro. *Connecting the out-of-sample and pre-image problems in kernel methods*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2007. (Cited on page 89.)

[Arif 2010] O. Arif, P. Vela and W. Daley. *Pre-image problem in manifold learning and dimensional reduction methods*. In International Conference on Machine Learning and Applications (ICMLA), pages 921–924, 2010. (Cited on page 90.)

[Bakir 2004] G. H. Bakir, J. Weston and B. Schölkopf. *Learning to find pre-images*. In Advances in Neural Information Processing Systems (NIPS), 2004. (Cited on pages 1, 88, 89, 90, 91, 96 and 97.)

[Barrett 1994] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst. Templates for the solution of linear systems: building blocks for iterative methods, volume 43. SIAM, 1994. (Cited on page 42.)

[Batra 2011] D. Batra, S. Nowozin and P. Kohli. *Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm*. In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 146–154, 2011. (Cited on page 74.)

[Beck 2009] A. Beck and M. Teboulle. *A fast iterative shrinkage-thresholding algorithm for linear inverse problems.* SIAM journal on imaging sciences, vol. 2, no. 1, pages 183–202, 2009. (Cited on page 24.)

[Beck 2013] A. Beck and L. Tetruashvili. *On the convergence of block coordinate descent type methods.* SIAM journal on Optimization, vol. 23, no. 4, pages 2037–2060, 2013. (Cited on page 27.)

[Bekkerman 2011] R. Bekkerman, M. Bilenko and J. Langford. Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press, 2011. (Cited on page 75.)

[Belkin 2006] M. Belkin, P. Niyogi and V. Sindhwani. *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples.* Journal of machine learning research, vol. 7, no. Nov, pages 2399–2434, 2006. (Cited on page 79.)

[Bern 2006] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen and S. Toledo. *Support-graph preconditioners.* SIAM Journal on Matrix Analysis and Applications, vol. 27, no. 4, pages 930–951, 2006. (Cited on page 58.)

[Bertsekas 1989] D. P. Bertsekas and J. N. Tsitsiklis. Parallel and distributed computation: numerical methods, volume 23. Prentice hall Englewood Cliffs, NJ, 1989. (Cited on pages 75 and 77.)

[Bertsekas 1999] D. P. Bertsekas. Nonlinear programming. Athena scientific Belmont, 1999. (Cited on pages 26 and 41.)

[Bishop 2006] C. M. Bishop. Pattern recognition and machine learning. Springer, 2006. (Cited on page 7.)

[Blake 2011] A. Blake, P. Kohli and C. Rother. Markov random fields for vision and image processing. Mit Press, 2011. (Cited on page 5.)

[Boman 2003] E. G. Boman and B. Hendrickson. *Support theory for preconditioning.* SIAM Journal on Matrix Analysis and Applications, vol. 25, no. 3, pages 694–717, 2003. (Cited on page 58.)

[Bonneel 2015] N. Bonneel, J. Rabin, G. Peyré and H. Pfister. *Sliced and radon wasserstein barycenters of measures.* Journal of Mathematical Imaging and Vision, vol. 51, no. 1, pages 22–45, 2015. (Cited on page 82.)

[Boyd 2004] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge university press, 2004. (Cited on pages 21, 23, 31, 40 and 59.)

[Boyd 2011] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers.* Foundations and Trends® in Machine learning, vol. 3, no. 1, pages 1–122, 2011. (Cited on pages 28, 29 and 75.)

[Byrd 1994] R. H. Byrd, J. Nocedal and R. B. Schnabel. *Representations of quasi-Newton matrices and their use in limited memory methods.* Mathematical Programming, vol. 63, no. 1-3, pages 129–156, 1994. (Cited on pages 55 and 65.)

[Byrd 2016] R. H. Byrd, S. L. Hansen, J. Nocedal and Y. Singer. *A stochastic quasi-Newton method for large-scale optimization.* SIAM Journal on Optimization, vol. 26, no. 2, pages 1008–1031, 2016. (Cited on page 71.)

[Canas 2012] G. Canas and L. Rosasco. *Learning probability measures with respect to optimal transport metrics.* In Advances in Neural Information Processing Systems (NIPS), pages 2492–2500, 2012. (Cited on page 79.)

[Candès 2011] E. J. Candès, X. Li, Y. Ma and J. Wright. *Robust principal component analysis?* Journal of the ACM (JACM), vol. 58, no. 3, page 11, 2011. (Cited on page 83.)

[Carli 2013] F. P. Carli, L. Ning and T. T. Georgiou. *Convex Clustering via Optimal Mass Transport.* arXiv preprint arXiv:1307.5459, 2013. (Cited on page 85.)

[Cevher 2014] V. Cevher, S. Becker and M. Schmidt. *Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics.* IEEE Signal Processing Magazine, vol. 31, no. 5, pages 32–43, 2014. (Cited on page 75.)

[Chambolle 2011] A. Chambolle and T. Pock. *A first-order primal-dual algorithm for convex problems with applications to imaging.* Journal of mathematical imaging and vision, vol. 40, no. 1, pages 120–145, 2011. (Cited on page 72.)

[Chandrasekaran 2008] V. Chandrasekaran, J. K. Johnson and A. S. Willsky. *Estimation in Gaussian graphical models using tractable subgraphs: A walk-sum analysis.* IEEE Transactions on Signal Processing, vol. 56, no. 5, pages 1916–1930, 2008. (Cited on page 73.)

[Chen 2006] L. Chen, M. J. Wainwright, M. Cetin and A. S. Willsky. *Data association based on optimization in graphical models with application to sensor networks.* Mathematical and computer modelling, vol. 43, no. 9-10, pages 1114–1135, 2006. (Cited on page 5.)

[Chen 2014a] Q. Chen and V. Koltun. *Fast MRF optimization with application to depth reconstruction.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3914–3921, 2014. (Cited on page 75.)

[Chen 2014b] Y. Chen, J. Mairal and Z. Harchaoui. *Fast and robust archetypal analysis for representation learning.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. (Cited on page 83.)

[Choi 2012] J. Choi and R. A. Rutenbar. *Hardware implementation of MRF map inference on an FPGA platform.* In International Conference on Field Programmable Logic and Applications (FPL), pages 209–216. IEEE, 2012. (Cited on page 74.)

[Cloninger 2017] A. Cloninger, W. Czaja and T. Doster. *The pre-image problem for Laplacian Eigenmaps utilizing L 1 regularization with applications to data fusion.* Inverse Problems, vol. 33, no. 7, page 074006, 2017. (Cited on page 99.)

[Condat 2016] L. Condat. *Fast projection onto the simplex and the l_1 ball.* Mathematical Programming, vol. 158, no. 1-2, pages 575–585, 2016. (Cited on pages 80 and 94.)

[Conn 2000] A. R. Conn, N. I. M. Gould and P. L. Toint. Trust region methods. SIAM, 2000. (Cited on page 51.)

[Coughlan 2010] J. M. Coughlan and H. Shen. *An embarrassingly simple speed-up of belief propagation with robust potentials.* arXiv preprint arXiv:1010.0012, 2010. (Cited on page 37.)

[CPLEX] CPLEX. *V12.8.0.* International Business Machines Corporation. (Cited on page 15.)

[CSD] CSD. *Complex Step Differentiation.* `http://blogs.mathworks.com/cleve/2013/10/14/complex-step-differentiation/`. Accessed: 2016-10-30. (Cited on page 58.)

[Cutler 1994] A. Cutler and L. Breiman. *Archetypal analysis.* Technometrics, vol. 36, no. 4, pages 338–347, 1994. (Cited on page 81.)

[Cuturi 2014] M. Cuturi and A. Doucet. *Fast computation of Wasserstein barycenters.* In International Conference on Machine Learning (ICML), pages 685–693, 2014. (Cited on page 82.)

[Dauphin 2015] Y. Dauphin, H. de Vries and Y. Bengio. *Equilibrated adaptive learning rates for non-convex optimization.* In Advances in Neural Information Processing Systems (NIPS), pages 1504–1512, 2015. (Cited on page 72.)

[Davis 2006] T. A. Davis. Direct methods for sparse linear systems, volume 2. Siam, 2006. (Cited on page 41.)

[Davis 2016] T. A. Davis, S. Rajamanickam and W. M. Sid-Lakhdar. *A survey of direct methods for sparse linear systems.* Acta Numerica, vol. 25, pages 383–566, 2016. (Cited on page 41.)

[Delong 2008] Andrew Delong and Yuri Boykov. *A scalable graph-cut algorithm for ND grids.* In IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2008. (Cited on page 74.)

[Dembo 1982] R. S. Dembo, S. C. Eisenstat and T. Steihaug. *Inexact newton methods.* SIAM Journal on Numerical analysis, vol. 19, no. 2, pages 400–408, 1982. (Cited on page 47.)

[Demmel 1997] J. W. Demmel. Applied numerical linear algebra. SIAM, 1997. (Cited on pages 43 and 56.)

[Demmel 2014] J. Demmel and K. Yelick. *Communication avoiding and other innovative algorithms.* The Berkeley Par Lab: Progress in the Parallel Computing Landscape, pages 243–250, 2014. (Cited on page 75.)

[Devarakonda 2016] A. Devarakonda, K. Fountoulakis, J. Demmel and M. W. Mahoney. *Avoiding communication in primal and dual block coordinate descent methods.* arXiv preprint arXiv:1612.04003, 2016. (Cited on page 76.)

[Devarakonda 2017] A. Devarakonda, K. Fountoulakis, J. Demmel and M. W. Mahoney. *Avoiding Synchronization in First-Order Methods for Sparse Convex Optimization.* arXiv preprint arXiv:1712.06047, 2017. (Cited on page 76.)

[Domke 2011] J. Domke. *Parameter learning with truncated message-passing.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2937–2943, 2011. (Cited on page 65.)

[Duchenne 2011] O. Duchenne, F. Bach, I-S. Kweon and J. Ponce. *A tensor-based algorithm for high-order graph matching.* IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 12, pages 2383–2395, 2011. (Cited on page 66.)

[Duchi 2007] J. C. Duchi, D. Tarlow, G. Elidan and D. Koller. *Using combinatorial optimization within max-product belief propagation.* In Advances in Neural Information Processing Systems (NIPS), pages 369–376, 2007. (Cited on page 36.)

[Eckstein 1989] J. Eckstein. *Splitting methods for monotone operators with applications to parallel optimization.* PhD thesis, Massachusetts Institute of Technology, 1989. (Cited on page 28.)

[Eisenstat 1996] S. C. Eisenstat and H. F. Walker. *Choosing the forcing terms in an inexact Newton method.* SIAM Journal on Scientific Computing, vol. 17, no. 1, pages 16–32, 1996. (Cited on page 48.)

[Elhamifar 2012] E. Elhamifar, G. Sapiro and R. Vidal. *Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery.* In Advances in Neural Information Processing Systems, pages 19–27, 2012. (Cited on pages 79 and 85.)

[Erdogdu 2015] M. A. Erdogdu and A. Montanari. *Convergence rates of sub-sampled newton methods.* In Neural Information Processing Systems, pages 3052–3060, 2015. (Cited on page 71.)

[Fan 2018] J. Fan and T. WS. Chow. *Exactly Robust Kernel Principal Component Analysis.* arXiv preprint arXiv:1802.10558, 2018. (Cited on page 92.)

[Felzenszwalb 2006] P. F. Felzenszwalb and D. P. Huttenlocher. *Efficient belief propagation for early vision.* International journal of computer vision, vol. 70, no. 1, pages 41–54, 2006. (Cited on pages 74 and 75.)

[Fix 2014] A. Fix, C. Wang and R. Zabih. *A Primal-Dual Algorithm for Higher-Order Multilabel Markov Random Fields.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1138–1145, 2014. (Cited on page 5.)

[Fong 2011] C. L. Fong. *Minimum-Residual Methods for Sparse Least-Squares Using Golub-Kahan Bidiagonalization.* PhD thesis, Citeseer, 2011. (Cited on page 48.)

[Fougner 2018] C. Fougner and S. Boyd. *Parameter selection and preconditioning for a graph form solver.* In Emerging Applications of Control and Systems Theory, pages 41–61. Springer, 2018. (Cited on page 72.)

[Fountoulakis 2015] K. Fountoulakis and J. Gondzio. *Performance of First- and Second-Order Methods for Big Data Optimization.* arXiv preprint arXiv:1503.03520, 2015. (Cited on page 3.)

[Frey 1998] B. J. Frey. Graphical models for machine learning and digital communication. MIT press, 1998. (Cited on page 5.)

[Fu 2013] Q. Fu, H. Wang and A. Banerjee. *Bethe-ADMM for Tree Decomposition based Parallel MAP Inference.* In Uncertainty in Artificial Intelligence (UAI), page 222, 2013. (Cited on pages 29 and 74.)

[Garg 2016] R. Garg, A. Eriksson and I. Reid. *Non-linear dimensionality regularizer for solving inverse problems.* arXiv preprint arXiv:1603.05015, 2016. (Cited on page 92.)

[Ghysels 2013] P. Ghysels, T. J. Ashby, K. Meerbergen and W. Vanroose. *Hiding global communication latency in the GMRES algorithm on massively parallel machines.* SIAM Journal on Scientific Computing, vol. 35, no. 1, pages C48–C71, 2013. (Cited on page 75.)

[Giguere 2015] S. Giguere, A. Rolland, F. Laviolette and M. Marchand. *Algorithms for the hard pre-image problem of string kernels and the general problem of string prediction.* In International Conference on Machine Learning (ICML), 2015. (Cited on page 90.)

[Gill 1981] P. E. Gill, W. Murray and M. H. Wright. *Practical optimization.* 1981. (Cited on pages 64 and 68.)

[Giselsson 2014] P. Giselsson and S. Boyd. *Diagonal scaling in Douglas-Rachford splitting and ADMM.* In IEEE Conference on Decision and Control (CDC), pages 5033–5039, 2014. (Cited on page 72.)

[Globerson 2008] A. Globerson and T. S. Jaakkola. *Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations.* In Advances in neural information processing systems (NIPS), pages 553–560, 2008. (Cited on page 25.)

[Golub 2012] G. H. Golub and C. F. Van Loan. Matrix computations, volume 3. JHU Press, 2012. (Cited on page 55.)

[Gonzalez 2009] J. Gonzalez, Y. Low and C. Guestrin. *Residual splash for optimally parallelizing belief propagation.* In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 177–184, 2009. (Cited on page 74.)

[Gu 2017] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng and L. Zhang. *Weighted nuclear norm minimization and its applications to low level vision.* International Journal of Computer Vision, vol. 121, no. 2, pages 183–208, 2017. (Cited on page 94.)

[Hazan 2010] T. Hazan and A. Shashua. *Norm-product belief propagation: Primal-dual message-passing for approximate inference.* IEEE Transactions on Information Theory, vol. 56, no. 12, pages 6294–6316, 2010. (Cited on page 25.)

[Henson 2002] V. E. Henson and U. M. Yang. *BoomerAMG: a parallel algebraic multigrid solver and preconditioner.* Applied Numerical Mathematics, vol. 41, no. 1, pages 155–177, 2002. (Cited on page 57.)

[Hestenes 1952] M. R. Hestenes and E. Stiefel. *Methods of Conjugate Gradients for Solving Linear Systems.* Journal of Research of the National Bureau of Standards, vol. 49, no. 6, 1952. (Cited on page 44.)

[Higham 1992] N. J. Higham. *Estimating the matrixp-norm.* Numerische Mathematik, vol. 62, no. 1, pages 539–555, 1992. (Cited on page 34.)

[Hocking 2011] T. D. Hocking, A. Joulin, F. Bach and J.-P. Vert. *Clusterpath an algorithm for clustering using convex fusion penalties.* In International Conference on Machine Learning (ICML), 2011. (Cited on pages 83, 86 and 96.)

[Honeine 2011] P. Honeine and C. Richard. *A closed-form solution for the pre-image problem in kernel-based machines.* Journal of Signal Processing Systems, vol. 65, no. 3, pages 289–299, 2011. (Cited on pages 90 and 91.)

[hou] *House dataset.* `http://vasc.ri.cmu.edu//idb/html/motion/house/index.html`. Accessed: 2016-10-30. (Cited on page 66.)

[Hsieh 2015] C-J. Hsieh, H-F. Yu and I. S. Dhillon. *PASSCoDe: Parallel ASynchronous Stochastic dual Co-ordinate Descent.* In International Conference on Machine Learning (ICML), pages 2370–2379, 2015. (Cited on page 75.)

[Huang 2011] D. Huang, Y. Tian and F. De la Torre. *Local isomorphism to solve the pre-image problem in kernel methods.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2761–2768, 2011. (Cited on page 99.)

[Huang 2014] Q. Huang, Y. Chen and L. Guibas. *Scalable semidefinite relaxation for maximum a posterior estimation.* In International Conference on Machine Learning (ICML), pages 64–72, 2014. (Cited on page 74.)

[Hurkat 2015] S. Hurkat, J. Choi, E. Nurvitadhi, J. F. Martínez and R. A. Rutenbar. *Fast hierarchical implementation of sequential tree-reweighted belief propagation for probabilistic inference.* In International Conference on Field Programmable Logic and Applications (FPL), pages 1–8. IEEE, 2015. (Cited on page 74.)

[Ishikawa 2011] H. Ishikawa. *Transformation of general binary MRF minimization to the first-order case.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 6, pages 1234–1249, 2011. (Cited on page 5.)

[Jin 2013] C. Jin, Q. Fu, H. Wang, A. Agrawal, W. Hendrix, W-k. Liao, M. M. A. Patwary, A. Banerjee and A. Choudhary. *Solving combinatorial optimization problems using relaxed linear programming: a high performance computing perspective.* In Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, pages 39–46. ACM, 2013. (Cited on page 74.)

[Jojic 2010] V. Jojic, S. Gould and D. Koller. *Accelerated dual decomposition for MAP inference.* In International Conference on Machine Learning (ICML), pages 503–510, 2010. (Cited on pages 22 and 24.)

[Jones 1995] M. T. Jones and P. E. Plassmann. *An improved incomplete Cholesky factorization.* ACM Transactions on Mathematical Software (TOMS), vol. 21, no. 1, pages 5–17, 1995. (Cited on page 56.)

[Jordan 2015] M. I. Jordan and T. M. Mitchell. *Machine learning: Trends, perspectives, and prospects.* Science, vol. 349, no. 6245, pages 255–260, 2015. (Cited on page 75.)

[Kallas 2013] M. Kallas, P. Honeine, C. Richard, C. Francis and H Amoud. *Non-negativity constraints on the pre-image for pattern recognition with kernel machines.* Pattern Recognition, vol. 46, no. 11, pages 3066–3080, 2013. (Cited on page 90.)

[Kappes 2012] J. H. Kappes, B. Savchynskyy and C. Schnörr. *A bundle approach to efficient MAP-inference by Lagrangian relaxation.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1688–1695, 2012. (Cited on page 25.)

[Kappes 2015] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy and C. Rother. *A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems.* International Journal of Computer Vision, vol. 115, no. 2, pages 155–184, 2015. (Cited on page 8.)

[Knobelreiter 2017] P. Knobelreiter, C. Reinbacher, A. Shekhovtsov and T. Pock. *End-to-End Training of Hybrid CNN-CRF Models for Stereo.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1456–1465, 2017. (Cited on page 74.)

[Knoll 2004] D. A. Knoll and D. E. Keyes. *Jacobian-free Newton–Krylov methods: a survey of approaches and applications.* Journal of Computational Physics, vol. 193, no. 2, pages 357–397, 2004. (Cited on page 58.)

[Koanantakool 2018] P. Koanantakool, A. Ali, A. Azad, A. Buluc, D. Morozov, L. Oliker, K. Yelick and S-Y. Oh. *Communication-Avoiding Optimization Methods for Distributed Massive-Scale Sparse Inverse Covariance Estimation.* In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 1376–1386, 2018. (Cited on page 76.)

[Kohli 2009] P. Kohli, M. P. Kumar and P. H. S. Torr. *P^3 & Beyond: Move Making Algorithms for Solving Higher Order Functions.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 9, pages 1645–1656, 2009. (Cited on page 5.)

[Koller 2007] D. Koller, N. Friedman, L. Getoor and B. Taskar. *Graphical models in a nutshell.* Introduction to statistical relational learning, pages 13–55, 2007. (Cited on page 5.)

[Koller 2009] D. Koller and N. Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009. (Cited on pages 5, 7 and 9.)

[Kolmogorov 2006] V. Kolmogorov. *Convergent tree-reweighted message passing for energy minimization.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 10, pages 1568–1583, 2006. (Cited on pages 25, 33 and 74.)

[Kolmogorov 2015] V. Kolmogorov. *A new look at reweighted message passing.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 37, no. 5, pages 919–930, 2015. (Cited on pages 5 and 27.)

[Kolouri 2017] S. Kolouri, S. R. Park, M. Thorpe, D. Slepcev and G. K. Rohde. *Optimal Mass Transport: Signal processing and machine-learning applications.* IEEE Signal Processing Magazine, vol. 34, no. 4, pages 43–59, 2017. (Cited on pages 4 and 79.)

[Komodakis 2008] N. Komodakis and N. Paragios. *Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles.* In European conference on computer vision, pages 806–820. Springer, 2008. (Cited on pages 73 and 74.)

[Komodakis 2009] N. Komodakis and N. Paragios. *Beyond pairwise energies: Efficient optimization for higher-order MRFs.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2985–2992, 2009. (Cited on pages 5, 8, 34, 36 and 66.)

[Komodakis 2011] N. Komodakis, N. Paragios and G. Tziritas. *MRF energy minimization and beyond via dual decomposition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 3, pages 531–552, 2011. (Cited on pages 21, 22, 31, 33, 73 and 74.)

[Komodakis 2015] N. Komodakis and J-C. Pesquet. *Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems.* IEEE Signal Processing Magazine, vol. 32, no. 6, pages 31–54, 2015. (Cited on page 23.)

[Komodakis 2016] N. Komodakis, M. P. Kumar and N. Paragios. *(hyper)-graphs inference through convex relaxations and move making algorithms: Contributions and applications in artificial vision.* Foundations and Trends® in Computer Graphics and Vision, vol. 10, no. 1, pages 1–102, 2016. (Cited on page 8.)

[Koo 2010] T. Koo, A. M. Rush, M. Collins, T. Jaakkola and D. Sontag. *Dual decomposition for parsing with non-projective head automata.* In Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1288–1298, 2010. (Cited on page 77.)

[Koutis 2011] I. Koutis, G. L. Miller and D. Tolliver. *Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing.* Computer Vision and Image Understanding, vol. 115, no. 12, pages 1638–1646, 2011. (Cited on pages 58 and 72.)

[Krishnan 2015] R. G. Krishnan, S. Lacoste-Julien and D. Sontag. *Barrier Frank-Wolfe for marginal inference.* In Advances in Neural Information Processing Systems (NIPS), pages 532–540, 2015. (Cited on page 12.)

[Kschischang 2001] F. R. Kschischang, B. J. Frey and H-A. Loeliger. *Factor graphs and the sum-product algorithm.* IEEE Transactions on information theory, vol. 47, no. 2, pages 498–519, 2001. (Cited on pages 8, 9 and 73.)

[Kumar 2009] M. P. Kumar, V. Kolmogorov and P. H. S. Torr. *An analysis of convex relaxations for MAP estimation of discrete MRFs.* Journal of Machine Learning Research, vol. 10, no. Jan, pages 71–106, 2009. (Cited on page 12.)

[Kwok 2004] J. T-Y. Kwok and I. W-H. Tsang. *The pre-image problem in kernel methods.* IEEE transactions on neural networks, vol. 15, no. 6, pages 1517–1525, 2004. (Cited on pages 1, 88, 89, 90, 91, 96, 97, 98 and 99.)

[Lafferty 2006] J. D. Lafferty and P. K. Ravikumar. *Preconditioner approximations for probabilistic graphical models.* In Advances in Neural Information Processing Systems (NIPS), pages 1113–1120, 2006. (Cited on page 73.)

[Lange 2000] K. Lange, D. R. Hunter and I. Yang. *Optimization transfer using surrogate objective functions.* Journal of computational and graphical statistics, vol. 9, no. 1, pages 1–20, 2000. (Cited on pages 74 and 93.)

[Lee 2012] J. Lee, Y. Sun and M. Saunders. *Proximal Newton-type methods for convex optimization.* In Advances in Neural Information Processing Systems (NIPS), pages 836–844, 2012. (Cited on page 3.)

[Li 2014] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita and B-Y. Su. *Scaling Distributed Machine Learning with the Parameter Server.* In OSDI, volume 14, pages 583–598, 2014. (Cited on page 75.)

[Li 2016] M. Li, A. Shekhovtsov and D. Huber. *Complexity of discrete energy minimization problems.* In European Conference on Computer Vision (ECCV), pages 834–852. Springer, 2016. (Cited on page 7.)

[Lin 1999] C-J. Lin and J. J. Moré. *Incomplete Cholesky factorizations with limited memory.* SIAM Journal on Scientific Computing, vol. 21, no. 1, pages 24–45, 1999. (Cited on page 56.)

[Lin 2011] Y-Y. Lin, T-L. Liu and C-S. Fuh. *Multiple kernel learning for dimensionality reduction.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 6, pages 1147–1160, 2011. (Cited on page 99.)

[Liu 2010] Jiangyu Liu and Jian Sun. *Parallel graph-cuts by adaptive bottom-up merging.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2181–2188, 2010. (Cited on page 74.)

[Liu 2015] J. Liu and S. J. Wright. *Asynchronous stochastic coordinate descent: Parallelism and convergence properties.* SIAM Journal on Optimization, vol. 25, no. 1, pages 351–376, 2015. (Cited on page 75.)

[Loeliger 2004] H-A. Loeliger. *An introduction to factor graphs.* IEEE Signal Processing Magazine, vol. 21, no. 1, pages 28–41, 2004. (Cited on pages 8 and 9.)

[Lu 2015] Z. Lu and L. Xiao. *On the complexity analysis of randomized block-coordinate descent methods.* Mathematical Programming, vol. 152, no. 1-2, pages 615–642, 2015. (Cited on page 27.)

[Luenberger 1997] D. G. Luenberger. Optimization by vector space methods. John Wiley & Sons, 1997. (Cited on page 23.)

[Ma 2014] J. Ma, S. Wang, Z. Wang and J. Xu. *MRFalign: protein homology detection through alignment of Markov random fields.* PLoS computational biology, vol. 10, no. 3, page e1003500, 2014. (Cited on page 5.)

[Mairal 2013] J. Mairal. *Optimization with first-order surrogate functions.* In International Conference on Machine Learning (ICML), 2013. (Cited on page 93.)

[Mairal 2014] J. Mairal, F. Bach and J. Ponce. *Sparse modeling for image and vision processing.* Foundations and Trends® in Computer Graphics and Vision, vol. 8, no. 2-3, pages 85–283, 2014. (Cited on page 4.)

[Malioutov 2008] Dmitry M Malioutov, Jason K Johnson, Myung Jin Choi and Alan S Willsky. *Low-rank variance approximation in GMRF models: Single and multiscale approaches.* IEEE Transactions on Signal Processing, vol. 56, no. 10, pages 4621–4634, 2008. (Cited on page 73.)

[Martens 2010] J. Martens. *Deep learning via Hessian-free optimization.* In International Conference on Machine Learning (ICML), pages 735–742, 2010. (Cited on pages 3, 48, 58 and 63.)

[Martens 2012] J. Martens and I. Sutskever. *Training deep and recurrent networks with hessian-free optimization.* In Neural networks: Tricks of the trade, pages 479–535. Springer, 2012. (Cited on page 51.)

[Martins 2015] A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith and E. P. Xing. *Ad3: Alternating directions dual decomposition for map inference in graphical models.* Journal of Machine Learning Research, vol. 16, pages 495–545, 2015. (Cited on pages 29, 30, 33, 72 and 74.)

[Meltzer 2005] T. Meltzer, C. Yanover and Y. Weiss. *Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation.* In IEEE International Conference on Computer Vision (ICCV), volume 1, pages 428–435, 2005. (Cited on page 33.)

[Meshi 2011] O. Meshi and A. Globerson. *An alternating direction method for dual MAP LP relaxation.* In Machine Learning and Knowledge Discovery in Databases, pages 470–483. Springer, 2011. (Cited on pages 29 and 74.)

[Meshi 2012] O. Meshi, A. Globerson and T. S. Jaakkola. *Convergence rate analysis of MAP coordinate minimization algorithms.* In Advances in Neural Information Processing Systems (NIPS), pages 3014–3022, 2012. (Cited on pages 25 and 27.)

[Meshi 2014] O. Meshi, T. Jaakkola and A. Globerson. *Smoothed Coordinate Descent for MAP Inference.* Advanced Structured Prediction, pages 103–131, 2014. (Cited on pages 23, 27, 31, 34 and 64.)

[Meshi 2015] O. Meshi, M. Mahdavi and A. Schwing. *Smooth and strong: Map inference with linear convergence.* In Advances in Neural Information Processing Systems (NIPS), pages 298–306, 2015. (Cited on pages 22, 34, 71 and 74.)

[Meshi 2017] O. Meshi and A. Schwing. *Asynchronous Parallel Coordinate Minimization for MAP Inference.* In Advances in Neural Information Processing Systems (NIPS), pages 5738–5748, 2017. (Cited on pages 25, 67, 72, 75, 76 and 77.)

[Mezuman 2013] E. Mezuman, D. Tarlow, A. Globerson and Y. Weiss. *Tighter linear program relaxations for high order graphical models.* In Uncertainty in Artificial Intelligence (UAI), pages 421–430, 2013. (Cited on page 74.)

[Micchelli 2006] C. A. Micchelli, Y. Xu and H. Zhang. *Universal kernels.* Journal of Machine Learning Research, vol. 7, no. Dec, pages 2651–2667, 2006. (Cited on page 99.)

[Mika 1999] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz and G. Rätsch. *Kernel PCA and de-noising in feature spaces.* In Advances in Neural Information Processing Systems (NIPS), 1999. (Cited on pages 1, 88, 89, 90, 96 and 97.)

[Miksik 2014] O. Miksik, V. Vineet, P. Pérez, P. H. S. Torr and F. C. Sévigné. *Distributed non-convex admm-inference in large-scale random fields.* In British Machine Vision Conference (BMVC), 2014. (Cited on pages 29, 34 and 74.)

[Mohri 2012] M. Mohri, A. Rostamizadeh and A. Talwalkar. Foundations of machine learning. MIT press, 2012. (Cited on page 99.)

[Möllenhoff 2018] T. Möllenhoff, Z. Ye, T. Wu and D. Cremers. *Combinatorial Preconditioners for Proximal Algorithms on Graphs.* In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 38–47, 2018. (Cited on page 72.)

[Morales 2000] J. L. Morales and J. Nocedal. *Automatic preconditioning by limited memory quasi-Newton updating.* SIAM Journal on Optimization, vol. 10, no. 4, pages 1079–1096, 2000. (Cited on page 57.)

[Morales 2002] J. L. Morales and J. Nocedal. *Enriched methods for large-scale unconstrained optimization.* Computational Optimization and Applications, vol. 21, no. 2, pages 143–154, 2002. (Cited on page 48.)

[Moré 1983] J. J. Moré. *Recent developments in algorithms and software for trust region methods.* In Mathematical programming The state of the art, pages 258–287. Springer, 1983. (Cited on page 53.)

[MOSEK] MOSEK. *V8.1.0.24.* MOSEK ApS. (Cited on page 15.)

[MS&E318] MS&E318. *Notes by Prof. M. Saunders.* `http://stanford.edu/class/msande318/notes/notes11-BCL.pdf`. Accessed: 2017-08-10. (Cited on page 96.)

[Mutnỳ 2017] M. Mutnỳ and P. Richtárik. *Parallel Stochastic Newton Method.* arXiv preprint arXiv:1705.02005, 2017. (Cited on page 71.)

[Nash 2001] S. G. Nash. *A survey of truncated-Newton methods.* In Numerical Analysis: Historical Developments in the 20th Century, pages 265–279. Elsevier, 2001. (Cited on page 47.)

[Nesterov 2005] Y. Nesterov. *Smooth minimization of non-smooth functions.* Mathematical programming, vol. 103, no. 1, pages 127–152, 2005. (Cited on pages 22, 33, 34 and 104.)

[Nguyen 2009] M. H. Nguyen and F. De la Torre. *Robust kernel principal component analysis.* In Advances in Neural Information Processing Systems (NIPS), 2009. (Cited on pages 89, 90, 91, 92, 95 and 96.)

[Nocedal 1998] J. Nocedal and Y-X. Yuan. Combining trust region and line search techniques, volume 14 of *Applied Optimization*, pages 153–175. Springer, 1998. (Cited on page 63.)

[Nocedal 2006] J. Nocedal and S. Wright. Numerical optimization. Springer, 2006. (Cited on pages 1, 45, 48, 51, 52, 53, 54 and 71.)

[Noorshams 2013] N. Noorshams and M. J. Wainwright. *Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm.* IEEE Transactions on Information Theory, vol. 59, no. 4, pages 1981–2000, 2013. (Cited on page 36.)

[Nutini 2015] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander and H. Koepke. *Coordinate descent converges faster with the Gauss-Southwell rule than random selection.* In International Conference on Machine Learning (ICML), pages 1632–1641, 2015. (Cited on page 25.)

[Oh 2016] T-H. Oh, Y-W. Tai, J-C. Bazin, H. Kim and I. S. Kweon. *Partial sum minimization of singular values in robust PCA: Algorithm and applications.*

IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 4, pages 744–758, 2016. (Cited on page 94.)

[Pan 2013] W. Pan, X. Shen and B. Liu. *Cluster analysis: unsupervised learning via supervised learning with a non-convex penalty.* The Journal of Machine Learning Research, vol. 14, no. 1, pages 1865–1889, 2013. (Cited on page 86.)

[Parikh 2014] N. Parikh and S. Boyd. *Block splitting for distributed optimization.* Mathematical Programming Computation, vol. 6, no. 1, pages 77–102, 2014. (Cited on page 96.)

[Pearlmutter 1994] B. A. Pearlmutter. *Fast exact multiplication by the Hessian.* Neural computation, vol. 6, no. 1, pages 147–160, 1994. (Cited on page 58.)

[Pedregosa 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay. *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, vol. 12, pages 2825–2830, 2011. (Cited on page 97.)

[Peyré 2017] G. Peyré and M. Cuturi. *Computational Optimal Transport.* Rapport technique, 2017. (Cited on page 79.)

[Pilanci 2015] M. Pilanci and M. J. Wainwright. *Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence.* arXiv preprint arXiv:1505.02250, 2015. (Cited on page 71.)

[Pock 2011] T. Pock and A. Chambolle. *Diagonal preconditioning for first order primal-dual algorithms in convex optimization.* In IEEE International Conference on Computer Vision (ICCV), pages 1762–1769, 2011. (Cited on page 72.)

[Potetz 2008] B. Potetz and T. S. Lee. *Efficient belief propagation for higher-order cliques using linear constraint nodes.* Computer Vision and Image Understanding, vol. 112, no. 1, pages 39–54, 2008. (Cited on page 36.)

[Raguet 2015] H. Raguet and L. Landrieu. *Preconditioning of a generalized forward-backward splitting and application to optimization on graphs.* SIAM Journal on Imaging Sciences, vol. 8, no. 4, pages 2706–2739, 2015. (Cited on page 72.)

[Rathi 2006] Y. Rathi, S. Dambreville and A. Tannenbaum. *Statistical shape analysis using kernel PCA.* In Image processing: algorithms and systems, neural networks, and machine learning, volume 6064, 2006. (Cited on pages 89 and 90.)

[Ravikumar 2010] P. Ravikumar, A. Agarwal and M. J. Wainwright. *Message-passing for graph-structured linear programs: Proximal methods and rounding schemes.* Journal of Machine Learning Research, vol. 11, no. Mar, pages 1043–1080, 2010. (Cited on pages 33 and 73.)

[Recht 2011] B. Recht, C. Re, S. Wright and F. Niu. *Hogwild: A lock-free approach to parallelizing stochastic gradient descent.* In Advances in neural information processing systems (NIPS), pages 693–701, 2011. (Cited on page 75.)

[Reid 1971] J. K. Reid. *On the method of conjugate gradients for the solution of large sparse systems of linear equations.* In Oxford conference of institute of mathematics and its applications, pages 231–254, 1971. (Cited on page 45.)

[Rockafellar 2009] R. T. Rockafellar and R. J-B. Wets. Variational analysis, volume 317. Springer Science & Business Media, 2009. (Cited on page 47.)

[Rother 2009] C. Rother, P. Kohli, W. Feng and J. Jia. *Minimizing sparse higher order energy functions of discrete variables.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1382–1389, 2009. (Cited on page 36.)

[Roweis 2000] S. T. Roweis and L. K. Saul. *Nonlinear dimensionality reduction by locally linear embedding.* science, vol. 290, no. 5500, pages 2323–2326, 2000. (Cited on page 79.)

[Saad 2003] Y. Saad. Iterative methods for sparse linear systems, volume 82. SIAM, 2003. (Cited on page 42.)

[Saad 2005] Y. Saad. *Multilevel ILU with reorderings for diagonal dominance.* SIAM Journal on Scientific Computing, vol. 27, no. 3, pages 1032–1057, 2005. (Cited on page 57.)

[Saha 2013] A. Saha and A. Tewari. *On the nonasymptotic convergence of cyclic coordinate descent methods.* SIAM Journal on Optimization, vol. 23, no. 1, pages 576–601, 2013. (Cited on page 27.)

[Santambrogio 2015] F. Santambrogio. *Optimal transport for applied mathematicians.* Birkäuser, NY, pages 99–102, 2015. (Cited on pages 79 and 82.)

[Savchynskyy 2011] B. Savchynskyy, S. Schmidt, J. Kappes and C. Schnörr. *A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1817–1823, 2011. (Cited on page 77.)

[Savchynskyy 2012] B. Savchynskyy, S. Schmidt, J. Kappes and C. Schnörr. *Efficient MRF energy minimization via adaptive diminishing smoothing.* Uncertainty in Artificial Intelligence (UAI), pages 746–755, 2012. (Cited on pages 50, 59, 63 and 65.)

[Savchynskyy 2014] B. Savchynskyy and S. Schmidt. *Getting Feasible Variable Estimates from Infeasible Ones: MRF Local Polytope Study.* Advanced Structured Prediction, pages 133–158, 2014. (Cited on page 31.)

[Scheinberg 2016] K. Scheinberg and X. Tang. *Practical inexact proximal quasi-Newton method with global complexity analysis.* Mathematical Programming, vol. 160, no. 1-2, pages 495–529, 2016. (Cited on page 3.)

[Schlesinger 1976] M. I. Schlesinger. *Syntactic analysis of two-dimensional visual signals in the presence of noise.* Cybernetics and systems analysis, vol. 12, no. 4, pages 612–628, 1976. (Cited on page 12.)

[Schlick 1987] T. Schlick and M. Overton. *A powerful truncated Newton method for potential energy minimization.* Journal of Computational Chemistry, vol. 8, no. 7, pages 1025–1039, 1987. (Cited on pages 48 and 62.)

[Schmidt 2009] M. W. Schmidt, E. Van Den Berg, M. P. Friedlander and K. P. Murphy. *Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm.* In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 456–463, 2009. (Cited on page 72.)

[Schmidt 2010] M. Schmidt. *Graphical model structure learning using L1-regularization.* PhD thesis, 2010. (Cited on page 3.)

[Schmidt 2011] S. Schmidt, B. Savchynskyy, J. H. Kappes and C. Schnörr. *Evaluation of a first-order primal-dual algorithm for MRF energy minimization.* In International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 89–103. Springer, 2011. (Cited on pages 34 and 72.)

[Schölkopf 1999] B. Schölkopf, A. J. Smola and K-R. Müller. *Kernel principal component analysis.* Advances in kernel methods, pages 327–352, 1999. (Cited on page 87.)

[Schwing 2011] A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun. *Distributed message passing for large scale graphical models.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1833–1840, 2011. (Cited on pages 75, 76 and 77.)

[Schwing 2012] A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun. *Globally convergent dual MAP LP relaxation solvers using Fenchel-Young margins.* In Advances in Neural Information Processing Systems (NIPS), pages 2384–2392, 2012. (Cited on pages 24 and 74.)

[Schwing 2014] A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun. *Globally convergent parallel MAP LP relaxation solver using the Frank-Wolfe algorithm.* In International Conference on Machine Learning (ICML), pages 487–495, 2014. (Cited on page 74.)

[Shah 2017] S. A. Shah and V. Koltun. *Robust continuous clustering.* Proceedings of the National Academy of Sciences, vol. 114, no. 37, pages 9814–9819, 2017. (Cited on pages 86, 87 and 96.)

[Shalev-Shwartz 2014] S. Shalev-Shwartz and T. Zhang. *Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization.* In International Conference on Machine Learning (ICML), pages 64–72, 2014. (Cited on page 71.)

[Shekhovtsov 2013] A. Shekhovtsov and V. Hlaváč. *A distributed mincut/maxflow algorithm combining path augmentation and push-relabel.* International journal of computer vision, vol. 104, no. 3, pages 315–342, 2013. (Cited on page 74.)

[Shekhovtsov 2016] A. Shekhovtsov, C. Reinbacher, G. Graber and T. Pock. *Solving dense image matching in real-time using discrete-continuous optimization.* In Computer Vision Winter Workshop, 2016. (Cited on page 74.)

[Shental 2008] O. Shental, P. H. Siegel, J. K. Wolf, D. Bickson and D. Dolev. *Gaussian belief propagation solver for systems of linear equations.* In IEEE International Symposium on Information Theory (ISIT), pages 1863–1867, 2008. (Cited on page 73.)

[Shewchuk 1994] J. R. Shewchuk. *An introduction to the conjugate gradient method without the agonizing pain*, 1994. (Cited on page 43.)

[Shimony 1994] S. E. Shimony. *Finding MAPs for belief networks is NP-hard.* Artificial Intelligence, vol. 68, no. 2, pages 399–410, 1994. (Cited on page 7.)

[Shrivastava 2015] A. Shrivastava, J. K. Pillai and V. M. Patel. *Multiple kernel-based dictionary learning for weakly supervised classification.* Pattern Recognition, vol. 48, no. 8, pages 2667–2675, 2015. (Cited on pages 90 and 99.)

[Smith 2016] V. Smith, S. Forte, C. Ma, M. Takác, M. I. Jordan and M. Jaggi. *CoCoA: A general framework for communication-efficient distributed optimization.* arXiv preprint arXiv:1611.02189, 2016. (Cited on pages 75 and 76.)

[Sontag 2007] D. Sontag and T. S. Jaakkola. *New outer bounds on the marginal polytope.* In Advances in Neural Information Processing Systems (NIPS), pages 1393–1400, 2007. (Cited on page 74.)

[Sontag 2008] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola and Y. Weiss. *Tightening LP Relaxations for MAP using Message Passing.* In Uncertainty in Artificial Intelligence (UAI), pages 503–510, 2008. (Cited on pages 8 and 74.)

[Sontag 2010] D. Sontag. *Approximate inference in graphical models using LP relaxations*. PhD thesis, Massachusetts Institute of Technology, 2010. (Cited on pages 18 and 73.)

[Sontag 2011] D. Sontag, A. Globerson and T. Jaakkola. *Introduction to dual decomposition for inference*. Optimization for Machine Learning, vol. 1, pages 219–254, 2011. (Cited on pages 18 and 22.)

[Sontag 2012] D. Sontag, D. K. Choe and Y. Li. *Efficiently searching for frustrated cycles in MAP inference*. In Uncertainty in Artificial Intelligence (UAI), pages 795–804, 2012. (Cited on pages 73 and 74.)

[Soori 2017] S. Soori, A. Devarakonda, J. Demmel, M. Gurbuzbalaban and M. M. Dehnavi. *Avoiding communication in proximal methods for convex optimization problems*. arXiv preprint arXiv:1710.08883, 2017. (Cited on page 76.)

[stprtool] stprtool. *Statistical Pattern Recognition Toolbox*. `https://cmp.felk.cvut.cz/cmp/software/stprtool/index.html`. Accessed: 2018-03-15. (Cited on page 97.)

[Strandmark 2011] P. Strandmark, F. Kahl and T. Schoenemann. *Parallel and distributed vision algorithms using dual decomposition*. Computer Vision and Image Understanding, vol. 115, no. 12, pages 1721–1732, 2011. (Cited on page 74.)

[Stüben 1999] K. Stüben. *Algebraic Multigrid (AMG): An introduction with applications*. GMD-Report 70, 1999. (Cited on page 57.)

[Sudderth 2004] E. B. Sudderth, M. J. Wainwright and A. S. Willsky. *Embedded trees: Estimation of Gaussian processes on graphs with cycles*. IEEE Transactions on Signal Processing, vol. 52, no. 11, pages 3136–3150, 2004. (Cited on page 73.)

[Sudderth 2008] E. Sudderth and W. Freeman. *Signal and image processing with belief propagation [DSP applications]*. IEEE Signal Processing Magazine, vol. 25, no. 2, pages 114–141, 2008. (Cited on page 9.)

[Sutton 2012] C. Sutton and A. McCallum. *An introduction to conditional random fields*. Foundations and Trends® in Machine Learning, vol. 4, no. 4, pages 267–373, 2012. (Cited on page 5.)

[Swoboda 2018] P. Swoboda and V. Kolmogorov. *MAP inference via Block-Coordinate Frank-Wolfe Algorithm*. arXiv preprint arXiv:1806.05049, 2018. (Cited on pages 22 and 71.)

[Tarlow 2010] D. Tarlow, I. Givoni and R. Zemel. *Hop-map: Efficient message passing with high order potentials*. In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 812–819, 2010. (Cited on page 36.)

[Villani 2003] C. Villani. Topics in optimal transportation. Numeéro 58. American Mathematical Soc., 2003. (Cited on page 79.)

[Villani 2008] C. Villani. Optimal transport: old and new, volume 338. Springer Science & Business Media, 2008. (Cited on page 79.)

[Vineet 2014] V. Vineet, J. Warrell and P. H. S. Torr. *Filter-based mean-field inference for random fields with higher-order terms and product label-spaces.* International Journal of Computer Vision, vol. 110, no. 3, pages 290–307, 2014. (Cited on page 5.)

[Von Luxburg 2007] U. Von Luxburg. *A tutorial on spectral clustering.* Statistics and computing, vol. 17, no. 4, pages 395–416, 2007. (Cited on page 79.)

[Wainwright 2005a] M. J. Wainwright, T. S. Jaakkola and A. S. Willsky. *MAP estimation via agreement on trees: message-passing and linear programming.* IEEE Transactions on Information Theory, vol. 51, no. 11, pages 3697–3717, 2005. (Cited on pages 8 and 13.)

[Wainwright 2005b] M. J. Wainwright, T. S. Jaakkola and A. S. Willsky. *A new class of upper bounds on the log partition function.* IEEE Transactions on Information Theory, vol. 51, no. 7, pages 2313–2335, 2005. (Cited on page 12.)

[Wainwright 2008] M. J. Wainwright and M. I. Jordan. *Graphical models, exponential families, and variational inference.* Foundations and Trends® in Machine Learning, vol. 1, no. 1–2, pages 1–305, 2008. (Cited on pages 1, 5, 12, 13, 14, 73 and 74.)

[Wang 2013] H. Wang and D. Koller. *A fast and exact energy minimization algorithm for cycle MRFs.* In International Conference on Machine Learning (ICML), volume 1, page 1, 2013. (Cited on pages 34 and 73.)

[Wang 2016] P. Wang, C. Shen, A. van den Hengel and P. H. S. Torr. *Efficient semidefinite branch-and-cut for MAP-MRF inference.* International Journal of Computer Vision (IJCV), vol. 117, no. 3, pages 269–289, 2016. (Cited on page 74.)

[Weed 2017] J. Weed and F. Bach. *Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance.* arXiv preprint arXiv:1707.00087, 2017. (Cited on page 79.)

[Weiss 2007] Y. Weiss, C. Yanover and T. Meltzer. *MAP estimation, linear programming and belief propagation with convex free energies.* In Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI), pages 416–425, 2007. (Cited on page 25.)

[Welling 2004] Max Welling. *On the choice of regions for generalized belief propagation.* In The 20th conference on Uncertainty in artificial intelligence (UAI), pages 585–592, 2004. (Cited on page 8.)

[Werner 2007] T. Werner. *A linear programming approach to max-sum problem: A review.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 7, pages 1165–1179, 2007. (Cited on pages 27 and 74.)

[Werner 2008] T. Werner. *High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF).* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2008. (Cited on page 74.)

[Werner 2009] T. Werner. *Revisiting the decomposition approach to inference in exponential families and graphical models.* Research Reports of CMP. 2009, No. 6, 2009. (Cited on page 27.)

[Yang 2010] Q. Yang, L. Wang and N. Ahuja. *A constant-space belief propagation algorithm for stereo matching.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1458–1465, 2010. (Cited on page 74.)

[Yanover 2006] C. Yanover, T. Meltzer and Y. Weiss. *Linear Programming Relaxations and Belief Propagation–An Empirical Study.* Journal of Machine Learning Research, vol. 7, no. Sep, pages 1887–1907, 2006. (Cited on page 15.)

[Yedidia 2005] J. S. Yedidia, W. T. Freeman and Y. Weiss. *Constructing free-energy approximations and generalized belief propagation algorithms.* IEEE Transactions on information theory, vol. 51, no. 7, pages 2282–2312, 2005. (Cited on page 73.)

[Yen 2016] I. E-H. Yen, D. Malioutov and A. Kumar. *Scalable exemplar clustering and facility location via augmented block coordinate descent with column generation.* In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 1260–1269, 2016. (Cited on pages 85, 86 and 96.)

[Zhang 2010] T. Zhang. *Analysis of multi-stage convex relaxation for sparse regularization.* Journal of Machine Learning Research, vol. 11, no. Mar, pages 1081–1107, 2010. (Cited on page 93.)

[Zhang 2014] J. Zhang, A. Schwing and R. Urtasun. *Message passing inference for large scale graphical models with high order potentials.* In Advances in Neural Information Processing Systems (NIPS), pages 1134–1142, 2014. (Cited on page 77.)

[Zheng 2010] W-S. Zheng, J. Lai and P. C. Yuen. *Penalized preimage learning in kernel principal component analysis.* IEEE Transactions on Neural Networks, vol. 21, no. 4, pages 551–570, 2010. (Cited on pages 89, 90, 92 and 99.)

ÉCOLE DOCTORALE

Sciences et technologies de l'information et de la communication (STIC)

Titre: Quelques applications de l'optimisation numérique aux problèmes d'inférence et d'apprentissage

Mots clés: Vision par ordinateur, Apprentissage automatique, Modèles graphiques, Inférence MAP, Apprentissage non-supervisé, Optimisation numérique

Résumé: Les relaxations en problème d'optimisation linéaire jouent un rôle central en inférence du maximum a posteriori (MAP) dans les champs aléatoires de Markov discrets. Nous étudions ici les avantages offerts par les méthodes de Newton pour résoudre efficacement le problème dual (au sens de Lagrange) d'une reformulation lisse du problème. Nous comparons ces dernières aux méthodes de premier ordre, à la fois en terme de vitesse de convergence et de robustesse au mauvais conditionnement du problème.

Nous exposons donc un cadre général pour l'apprentissage non-supervisé basé sur le transport optimal et les régularisations parcimonieuses. Nous exhibons notamment une approche prometteuse pour résoudre le problème de la préimage dans l'ACP à noyau. Du point de vue de l'optimisation, nous décrivons le calcul du gradient d'une version lisse de la norme p de Schatten et comment cette dernière peut être utilisée dans un schéma de majoration-minimisation.

Title: Few Applications of Numerical Optimization in Inference and Learning

Keywords: Computer vision, Machine learning, Graphical models, MAP inference, Unsupervised learning, Numerical optimization

Abstract: Linear programming relaxations are central to maximum a posteriori (MAP) inference in discrete Markov Random Fields (MRFs). In this thesis, we study the benefit of using Newton methods to efficiently optimize the Lagrangian dual of a smooth version of the problem. We investigate their ability to achieve superior convergence behavior and to better handle the ill-conditioned nature of the formulation, as compared to first order methods.

We show a general framework for unsupervised learning based on optimal transport and sparse regularization. We especially demonstrate a promising approach to address the pre-image problem in kernel PCA. From an optimization point of view, we show how to compute the gradient of a smooth version of the Schatten p-norm and how it can be used within a majorization-minimization scheme.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France