Beyond Affinity Propagation: Message Passing Algorithms for Clustering

by

Inmar-Ella Givoni

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 2012 by Inmar-Ella Givoni

Abstract

Beyond Affinity Propagation:

Message Passing Algorithms for Clustering

Inmar-Ella Givoni

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2012

Affinity propagation is an exemplar-based clustering method that takes as input similarities between data points. It outputs a set of data points that best represent the data (*exemplars*), and assignments of each non-exemplar point to its most appropriate exemplar, thereby partitioning the data set into clusters. The objective of affinity propagation is to maximize the sum of similarities between the data points and their exemplars.

In this thesis, we develop several extensions of affinity propagation. The extensions provide clustering tools that go beyond the capabilities of the basic affinity propagation algorithm, and generalize it to various problems of interest in machine learning. We also investigate alternative approaches to the underlying mechanism of affinity propagation using recent inference techniques that are based on optimization theory.

Affinity propagation was first described using a particular graphical model for the exemplar-based clustering problem. We first provide an alternative graphical model and derivation of affinity propagation, which are more amenable to model manipulation. Building on this representation, we develop capacitated affinity propagation, semi-supervised affinity propagation, and the hierarchical affinity propagation algorithms. We also discuss the relationship of affinity propagation to some canonical problems in combinatorial optimization.

The underlying mechanism of affinity propagation is an approximate inference pro-

cedure known as max-product belief propagation. We provide a comparison of affinity propagation to alternative inference techniques such as max-product linear programming, and dual decomposition. We show that for a collection of benchmark data sets, affinity propagation outperforms these more theoretically justified approaches.

We conclude by discussing the contributions and findings of this thesis, and how they relate to current research themes in more general inference problems. We point to several interesting avenues for future research.

Dedication

לסבא - ישראל כהן,

מחנך, תומך, חבר, והגיבור שלי.

באהבה, אנמר.

To Grandpa - Israel Cohen

Teacher, supporter, friend, and my hero.

With love, Inmar.

Acknowledgements

And I ain't never called `malloc` without calling `free`,
there's so much drama in the PhD. (Monzy, 2005)

One of the first and most important things to realize while treading the academic waters is that success does not solely depend on the quality and originality of your research. It also depends on the social structure and unwritten rules and customs of the various communities you are part of, and on how successful you are at navigating this corpus. For some this is a curse, and for some a blessing, but at the end, almost no person gets through their doctoral studies (or any major endeavor) all by themselves[1]. Much of what has happened to me in these last few (many?) years – successes, failures, moments of joy and fulfilment, moments of gloom and despair – was in large parts due to the people around me, physically, or more abstractly, in my mind.

Since I do not foresee myself writing another thesis any time soon, I'd like to take this opportunity and attempt to properly thank and acknowledge some of the individuals and groups that were part of this process for me. Single-space or not, this may take a while.

First and foremost, thank you, mom and dad, Libit and Alon, for instilling in me the love for learning and knowledge from a very young age, for making me want to excel without demanding it of me, for giving me the freedom to explore and investigate my interests, for trusting me to do the right thing, and for supporting my sometimes unconventional choices. My sister and brother, Shir and Gil, I love you both very much and am extremely proud of what unique and wonderful individuals you two have grown to be. Though I wasn't able to be there throughout all the stages of this process, I am grateful for the times we do spend together, and for our cross-Atlantic relationship.

Brendan, it seems like it wasn't too long ago that I talked with you for the first time, quite nervous and considerably intimidated by the whole situation. I'm still not sure what you were thinking that day, but thank you, first of all, for taking the chance and inviting me to join the group. It has, in fact, been quite some time ago, and I dare say I've come a long way. I attribute considerable parts of that to you. Thank for you everything – for sharing your knowledge and often unique perspectives on machine learning, research,

[1]For a notable exception, see the Scsh Reference Manual (Shivers et al., 2003)

and life. For your patience in letting me develop my abilities and confidence. You never pushed me too hard, or left me to struggle all by myself. It was just right. Thank you for providing me with many opportunities to explore and experience what it means to be a researcher – in and outside the lab. Thank you for caring about me and my future. Thank you for always wanting and trying to make our research a fun process. It has, indeed, been great fun, and I know I'm going to miss it. I know I'm going to miss you.

I have had the luck and privilege to have several mentors who helped me and advised me throughout this journey. In particular, I would like to thank Nir, for graciously paving the road to the University of Toronto for me. Thank you for keeping an eye out for me afterwards, giving me advice, support and the occasional nudge along the way. Sheila, thank you for all the time spent in various forms of efforts on my behalf, and for always caring about the outcome. I would also like to thank my committee members for guiding me along and providing feedback on my work, and in particular to Rich, for collaborations and other interactions extending beyond my thesis. Sam, I so wish you were still here with us.

During my studies, Ultimate has been a great part of my life. Unquestionably, it has been a means of keeping sane and healthy, and an excuse for many adventures in Ontario, other parts of Canada, and really, the world. It gives me a community wherever I go. Ofer - thank you for dragging me to that very first game and getting me hooked. Thank you to everyone else I had fun cutting, handling, striking, and chugging with, and especially to the members of the Floppy Discs, Phud, Dunks, and Movi Disc.

In total, almost a year of my graduate studies was spent outside Canada, visiting other labs and meeting some extraordinary people. I'd like to thank my supervisors and hosts – Anitha, Zoubin, and Andrew – for giving me the opportunity to experience different approaches to research, and to try on problems and tools different from the ones I was familiar with. My visit to the CBL lab in Cambridge was particularly inspiring. Finale, working with you on things that are close to our hearts is always a great source of pleasure and an illuminating learning experience for me, be it organizing a workshop, or making vegan dishes. Simon, thank you so much for all the active, curious, and joyful listening and for all the advice, encouragement and perspective you've given me along the way.

For all those friends I had to leave behind, I have never stopped missing you. For those friends I met through my studies, I will miss you. Come and meet me when you get the chance. Christian, thank you for being my partner in commiserating during those rougher times, for avantgarde music and other questionable cultural adventures. Thank you for showing me the most enjoyable fringes of Toronto, while never ceasing to

complain about it. Lu, thank you for being so charming and liking us as much as we like you. Red Чебурашка, green Bob and blue Grover will hang out in our stead as we wait to meet again. Yoseph, I'm really really glad that you were around, for giving advice, music, movies, or just a cheer-up smile, a healthy smack on the shoulder, and a long coffee break whenever I needed one. I wish you and your family the best of luck in your new life. Yoni, it's such a treat to know you are my friend. I'm ready to stick around for eternity to continue debating the idiotic to the profound with you.

It's taken me a long while to understand it, but what I enjoy most about research is the chance to interact and collaborate with people. I'm happy that I eventually managed to do quite a bit of that and want to thank those who were a part of it. And then, in particular, Danny, thank you for all those hours of endless (un)puzzlement sessions, and for answering the same questions again and again and again. For some reason, it seems to have worked really well for us. And thanks for the not-so-research-related Irish carbombs and all that followed. These will continue to be some of my most pleasant conference memories. And Patrick. There's this poignant sage Jewish saying, about how you learn from your teachers, and you learn more from your friends, and you learn the most from your students. I find it very fitting, as you are all of these for me. Thank you for being my woods guide on the road to becoming more of who I want to be.

And finally, Gadi, my mifnetzet. You were there when it all began, the source and the cause. You were there with me, and for me, trusting I will succeed, helping me, and sharing this life, every day and every night, when it was great and when it was unbearable. I thank you.

And now, this adventure is over. New adventures await.

Contents

Table of Contents **xi**

List of Figures **xiii**

List of Notation and Abbreviations **xiv**

1 Introduction **1**

1.1 Clustering 1

1.2 Affinity Propagation 5

1.3 Beyond Affinity Propagation 7

2 Background **11**

2.1 Exemplar-based Clustering 12

2.1.1 Affinity Propagation 15

2.1.2 Facility Location 16

2.2 Clustering Objectives and Algorithms 17

2.2.1 Different Approaches to Clustering 17

2.2.2 Constrained Forms of Clustering 21

2.2.3 Semi-Supervised Clustering 21

2.2.4 Hierarchical Clustering 24

2.3 Algorithms for MAP Inference 25

2.3.1 The MAP Inference Problem 26

2.3.2 Max Product Belief Propagation 28

2.3.3 MAP Inference with Convex Relaxation Methods 34

2.4 Discussion and Thesis Outline 46

3 A Binary Variable Model for Affinity Propagation **49**

3.1 Motivation and Background 49

3.2 A Binary Model for Affinity Propagation 51

3.3 Message Updates for the Binary AP Model 54
3.4 Explicit Exemplar Variable Formulation 59
3.5 Facility Location 61
3.6 Bipartite Maximum Weighted b-Matching 64
3.7 Discussion 68

4 Capacitated Affinity Propagation 70
4.1 Model and Message Updates for Capacitated AP 71
4.2 Experimental Evaluation 73
4.3 Discussion 75

5 Semi-Supervised Affinity Propagation 78
5.1 Motivation and Background 78
5.2 A Model for Semi-Supervised AP 80
5.3 Message Updates for Semi-Supervised AP 86
5.4 Experimental Evaluation 87
5.4.1 Evaluation Criterion 89
5.4.2 Results 91
5.5 Discussion 93

6 Hierarchical Affinity Propagation 96
6.1 Motivation and Background 97
6.2 Models for HAP and HFL 98
6.2.1 Hierarchical Affinity Propagation (HAP) Model 99
6.2.2 Hierarchical Facility Location (HFL) 101
6.2.3 Alternative Objective Function (HAP-B) 102
6.3 Message Updates for Hierarchical AP 103
6.3.1 Message Updates for HFL and HAP-B 104
6.4 Experimental Evaluation 105
6.4.1 Synthetic 2D Data 105
6.4.2 Analysis of Synthetic HIV Sequences 107
6.4.3 Analysis of Real HIV Sequences 110
6.4.4 Protein Prediction from Mass Spectrometry Data 115
6.5 Additional Experimental Setup and Results 117
6.6 Discussion 118

7 Exemplar-based Clustering Using Convex Relaxation Methods **121**
7.1 Exemplar-based Clustering Using Max-Product Linear Programming . . 122
7.1.1 MPLP - Max-Product Linear Programming for Exemplar-based Clustering . . . 122
7.1.2 Dual Objective and Decoding for MPLP . . . 124
7.2 Exemplar-based Clustering Using Dual Decomposition . . . 127
7.2.1 DD - Dual Decomposition for Exemplar-based Clustering . . . 127
7.2.2 Decoding for DD . . . 129
7.2.3 Discussion . . . 131

8 Comparison of Exemplar-based Clustering Algorithms **132**
8.1 Experimental Setup . . . 133
8.2 Data Sets . . . 134
8.3 Results . . . 136
8.3.1 Toy Data . . . 136
8.3.2 Sentences Data . . . 137
8.3.3 Cities Data . . . 138
8.3.4 Faces-400 Data . . . 139
8.3.5 Faces-900 Data . . . 140
8.3.6 Large Scale Data - Faces-Video and Exons . . . 140
8.4 Discussion . . . 141

9 Conclusions **149**

A The Linear Programming Relaxation **156**

B Full Derivation of HAP **159**
B.1 Functions . . . 159
B.2 Update Equations . . . 161
B.3 Full Derivation . . . 162
B.3.1 Message $\hat{p}_j^l$ . . . 162
B.3.2 Message from E_j^l to e_j^l . . . 163
B.3.3 Message from Level l to $l+1$ (τ_j^{l+1}): . . . 164
B.3.4 Message from Level l to $l-1$ (ϕ_j^{l-1}): . . . 164
B.3.5 Message $\alpha_j^{l<L}$: . . . 166
B.3.6 Message $\rho^{l>1}$ . . . 168

C Step-size Selection in Dual Decomposition **173**
C.1 Using static update Eq. C.1 176
C.2 Using static update Eq. C.2 177
C.3 Using static update Eq. C.3 179
C.4 Using dynamic update Eq. C.4 with Eq. C.5 180
C.5 Using dynamic update Eq. C.4 with Eq. C.6 181
C.6 Step-size choice 181

Bibliography **182**

List of Figures

1.1 Different types of data points to cluster. 2
1.2 Data points to cluster and an AP clustering solution. 6

2.1 The factor graph model. 27

3.1 N-ary variable model for affinity propagation. 50
3.2 A binary variable model and messages for affinity propagation. 52
3.3 A graphical model and messages for explicit exemplar variable AP. 59

4.1 A comparison of CAP and capacitated k-medoids. 75

5.1 SSAP on the half-moons data with instance-level constraints. 79
5.2 A graphical model and half-moons data results for directly adding constraints. 81
5.3 Incorporating instance-level constraints by similarity modifications. . . . 82
5.4 A visualization of clustering with meta-points. 84
5.5 A graphical model for semi-supervised affinity propagation (SSAP). . . . 85
5.6 An example of image segmentation using SSAP. 89
5.7 Clustering accuracy for image segmentation. 92
5.8 Some examples of image segmentation based on instance-level constraints. 94

6.1 A graphical model and messages for hierarchical affinity propagation (HAP). 100
6.2 Two alternative objectives for hierarchical exemplar-based clustering. . . 103
6.3 2D synthetic data: comparison of net similarity achieved by HAP and its greedy counterpart. 105
6.4 Synthetic HIV data: precision-recall for HAP, Greedy, HKMC and HK-Means. 109
6.5 Synthetic HIV data: distribution of Rand index for different experiments using HAP and Greedy 110

6.6 HIV data: comparison of the net similarity achieved by HAP, Greedy, and HKMC. 111
6.7 HIV data: distribution of cluster sizes for HAP and Greedy. 113
6.8 HIV data: cluster continent and subtype composition. 114
6.9 HIV data: second layer. 115
6.10 Protein identification using HAP and other methods. 117
6.11 2D synthetic data: runtimes 118

8.1 Toy data, shown with optimal clustering. Clusters are color coded. A black box marks the cluster exemplar. 136
8.2 Faces-900 data comprised of 10 individuals, with varying light, gaze, expression and rotation. 143
8.3 Comparison of methods for the toy data. 144
8.4 Comparison of methods for the sentences data. 145
8.5 Comparison of methods for the cities data. 146
8.6 Comparison of methods for the faces-400 data. 146
8.7 Comparison of methods for the faces-900 data. 147
8.8 Comparison of AP and MPLP for large scale data sets. 148

C.1 2D toy data results for step-size (Eq. C.1). 177
C.2 2D toy data results for step-size (Eq. C.1), non-converged. 178
C.3 2D toy data results for step-size (Eq. C.2). 179
C.4 2D toy data results for step-size (Eq. C.3). 180
C.5 2D toy data results for step-size (Eq. C.3), non-converged. 183
C.6 2D toy data results for step-size (Eq. C.4 with Eq. C.5). 184
C.7 2D toy data results for step-size (Eq. C.4 with Eq. C.6). 185
C.8 Faces data results for step-size (Eq. C.2). 186

List of Notation and Abbreviations

h_{ij}, e_j	Binary indicator hidden variable	14
$\mu_{i\to c}(x_i)$	BP message from variable to function	29
$\mu_{c\to i}(x_i)$	BP message from function to variable	29
s_{ij}	Similarity between points i and j	14
p_j	Exemplar preference for point j	14
S_{ij}	AP similarity function	52
I_i	AP 1-of-N function	52
E_j	AP exemplar consistency function	52
ρ_{ij}	AP messages from h_{ij} to E_j	15
α_{ij}	AP messages from E_j to h_{ij}	15
β_{ij}	AP messages from h_{ij} to I_j	52
η_{ij}	AP messages from I_j to h_{ij}	52
BP	Belief propagation	29
FL	The uncapacitated facility location problem	11
AP	Affinity propagation	5
CAP	Capacitated affinity propagation	47
HAP	Hierarchical affinity propagation	47
SSAP	Semi-supervised affinity propagation	47
MPBP	Max-product belief propagation	8
SPBP	Sum-product belief propagation	29
DD	Dual decomposition	37
MPLP	Max-product linear programming	37
TRW(-S)	Tree-reweighted belief propagation (-Sequential)	141
HKMC	Hierarchical k-medians clustering	108
HKMeans	Hierarchical k-means clustering	108

Chapter 1

Introduction

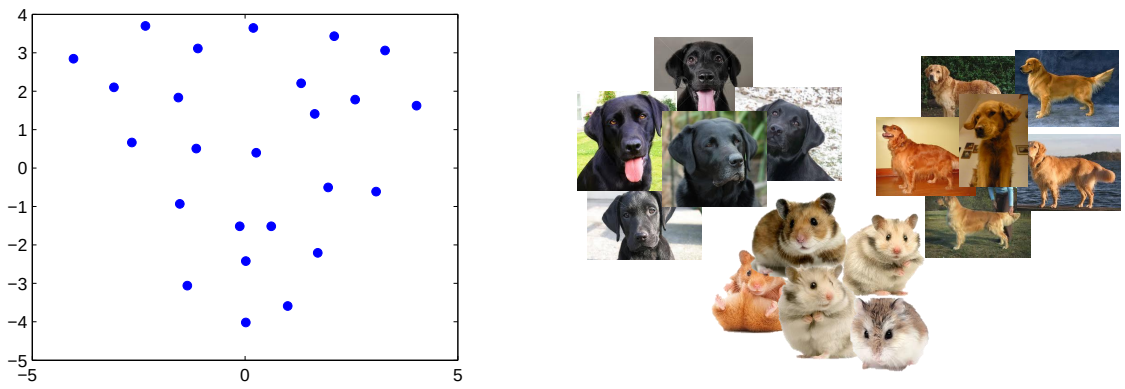
In this chapter, we give a high-level motivation and summary of the research described in this thesis. It is meant to be accessible to readers of all backgrounds. We therefore aim to avoid technical terminology. When necessary, we provide brief explanations of basic computer-science related terms.

1.1 Clustering

Clustering, or cluster analysis, is a fundamental task in computerized data analysis. It concerns the problem of partitioning a collection of data points into groups, often referred to as *clusters*, such that the members of each cluster are 'similar'. For now, let us assume we are able to determine the similarity between data points. For example, via a numerical table describing the similarity between every pair of points; the higher the number, the more similar the points are.

Fig. 1.1 gives more intuition as to what we mean by data points. These can literally be points drawn on a regular piece of paper, as in Fig. 1.1(a). Data points can also refer to more complicated entities such as a collection of photographs, where we consider each photograph to be a data point, as in Fig. 1.1(b). In the former case, we can easily describe the similarity between any two data points by measuring the distance between

them with a ruler, and negating this number (since we assumed that a bigger value means two points are more similar). In the latter case, it may be more complicated to come up with an automated way of measuring similarities.



(a) (b)

Figure 1.1: Different types of data points to cluster.

A naive approach for measuring image similarities would be to begin by resizing all images to be of the same dimension. Then, for every pair of images, and every location in the image, we can compute the difference in pixel values. We can then add all of these values together to get a sum of differences. More precisely, we would add the values of the **squared** differences, (or the absolute differences), so that negative and positive differences will not cancel out. Finally, we can negate this sum of squared differences to obtain a number that indicates how similar the two images are. It is not too difficult to see that for two identical images, we will get a value of zero. For two images showing almost the same content, but having a small amount of pixels of different colors, we will get a negative value that is not too far from zero. For two very different images, we will get a very large negative value. Thus, this procedure outputs numbers that align well with our definition of similarity.

The specific procedure we just described is, in fact, not a good way of generating similarities between images. Regardless, we hope that with some additional contemplation, the reader will become convinced that any collection of objects, be it images, DNA

sequences, sentences in a book, and so forth, can be thought of as data points. Furthermore, one can typically come up with some numerical representation of how similar those objects are.

Now, we can move forward to the real task at hand: clustering. We wish to devise a list of instructions, written in a computer understandable language, that describes how to find a good partitioning of the data points into groups of similar points. Such a list of instructions is known as an *algorithm*.

As it turns out, this is not an easy task. The human brain is exceptionally well-equipped to deal with this problem, to the extent that we may even find it trivial. However, the challenges of how to automatically perform the clustering task, as well as determining how many clusters to find, and how to decide which of several solutions is best, are considerable. In computer-science, the research into this problem is typically studied within the sub-field of machine-learning.

To appreciate the difficulty of the problem, let us assume for a moment that we already know how many clusters we wish to find (let us call this number k). Furthermore, we will assume that given any two clustering solutions, we can automatically assess which one is better. Then a simple procedure for finding the best clustering is the following: enumerate all possible ways of dividing the data points into k groups, and then asses all such solutions against each other so as to find the best one.

The problem with that approach is that for any reasonable amount of data points (say 200 or more), and for any non-trivial number k (if k=1 we know the solution, of course: put all data points in one cluster), the total number of possible solutions is so huge that it is typically bigger than the number of atoms in the universe . Comparing all these solutions requires more time than the expected remaining lifespan of the universe. Therefore, we cannot hope to succeed using this procedure.

Many computer-science problems have that flavor. It is easy to enumerate all possible solutions, and come up with a simple procedure that looks at all the solutions to determine

which one is best. However, there are just too many possible solutions, and it is not feasible to actually carry out the procedure in a reasonable amount of time. Many such problems turn out to belong to one of the most difficult families of problems in computer-science – the problems for which we do not know how to find good exact solutions. We therefore must resort to finding solutions that are only approximations of the best solution. Members of this family of problems are known as NP-hard problems. All of the problems discussed in this work are NP-hard, and all the algorithms we develop for them here are approximation algorithms.

Now that we are convinced that the clustering problem is not trivial, let us ignore that fact for a moment. Let us consider what it is we actually want the algorithm to do. When we devise a clustering algorithm, we often begin by deciding what set of properties we want the clustering solution to have. For example, we may want to have the property that the sum of all pairwise similarities of points within every cluster is maximal. We can write this desired property down in terms of a mathematical equation that we want to maximize. This equation is often known as the *objective function*. An alternative property we may be interested in, is that across all clusters, the worst pairwise similarity within a cluster is as high as possible. There are many other properties we may wish to focus on. Different flavors of these properties can often be expressed by different objective functions. The goal of the clustering algorithm is to optimize that objective function, and thus, clustering is essentially a hard optimization problem.

Finally, let us note that what we have described so far is the most fundamental type of clustering. There are additional requirements we may want the solution to have, depending on the particular task we wish to solve. We may wish to only allow for solutions that have similar numbers of points in every cluster. We may have information regarding the data points, stating that regardless of their similarity, some points cannot both be in the same cluster. We wish to develop clustering algorithms that can account for such additional constraints and information.

1.2 Affinity Propagation

Affinity propagation (AP, (Frey & Dueck, 2007)) is a clustering algorithm that, given a set of similarities (also denoted by *affinities*) between pairs of data points, partitions the data set into clusters such that each partition is associated with a prototypical point that best describes that cluster. AP associates each data point with one such prototype, or *exemplar*. Thus, the objective of AP is to maximize the overall sum of similarities between data points and their exemplars[1].

Fig. 1.2 shows an example of data to cluster on the left, and the solution found by AP on the right. Every point is either an exemplar (as denoted by a black border around the point) or is assigned to a particular exemplar (as denoted by the color and an edge to the exemplar). Overall, this gives a clustering of the shown data points into three distinct clusters.

Affinity propagation generated much interest in the machine learning community, as well as in other fields of computer science where machine learning techniques are routinely applied (*e.g.* computer vision and computational biology). Though the AP algorithm has no theoretical guarantees (the algorithm may not even find a solution, or may find a very poor one), in practice it clusters data points remarkably well in a short amount of time, and it is simple to use and to interpret. It has therefore become a highly popular method for clustering, with nearly 900 citations to date[2].

Since its publication, affinity propagation has been used as an analysis tool to tackle a wide array of problems in different research areas. In computational biology, it was applied to HIV vaccine design, and to the detection of DNA coding parts (exons) in DNA

[1]For the sake of accuracy, we point out that while every point can be an exemplar, there is also an input cost associated with making a point into an exemplar, and different points may have different costs. The cost is necessary in order to avoid trivial solutions whereby each point chooses itself as an exemplar. The cost is, in fact, a way of controlling the number of clusters found by the algorithm. Costs are an alternative strategy to that of setting the number of clusters to find to some fixed, and given k. Thus, the AP objective is actually to maximize the overall sum of similarities between data points and their exemplars while minimizing the total exemplar costs. This point is discussed in detail in Chapter 2.

[2]Data obtained from Google Scholar, November 2011.

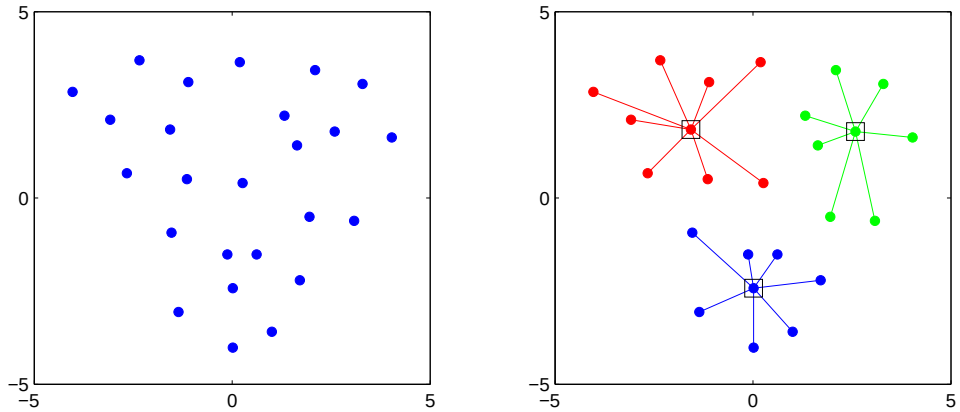


Figure 1.2: Data points to cluster (left). An exemplar-based clustering solution found by AP (right).

sequences (Dueck, 2009). Recent works utilizing AP include protein function prediction based on assigning proteins to the functional family of their chosen exemplar (Yeats et al., 2008), analysis of chemical communication patterns in the fruit fly (Kent et al., 2007), transcription factor DNA binding site analysis (Wei et al., 2010), and binding site analysis of RNA binding proteins (Ray et al., 2009).

In neuroscience, AP was recently used for the analysis of spike synchronization in rat brains (Takahashia et al., 2010), in learning the role of sleep slow wave activity in visuomotor learning (Landsness et al., 2009), and in the discovery of synaptic connectivity organizational principles (Perin et al., 2011).

In the emerging field of online social network analysis, AP has been repeatedly cited as a useful tool for community detection and community structure learning (Kang & Choi, 2009; Jia et al., 2010; Ji Liu, 2011; Lai et al., 2011). More generally, in the world of Web 2.0 applications, AP has been used for text analysis of online media sources such as Twitter (Kang et al., 2010), and for visual data processing such as automated creation of a media rich online encyclopedia (Hong et al., 2010), photo album summarization (Sinha et al., 2009), and image-tag alignment (Zhou et al., 2011).

In core computer vision research, AP has been used for multiple view image segmentation (Xiao et al., 2007), human action and gesture recognition (Abdelkader et al., 2011), saliency-based image segmentation (Donoser et al., 2009), sub-space segmentation (Lazic et al., 2009), and vehicle detection and traffic surveillance (Yang et al., 2011), among other applications.

1.3 Beyond Affinity Propagation

The fact that affinity propagation is a highly usable and useful algorithm results in the desire to further extend the applicability of the algorithm. The first and main focus of this thesis is the generalization of the basic AP algorithm so that it can address more complicated variants of clustering in a principled manner.

More generalized clustering problems can typically be described in terms of integrating additional structural constraints into the original clustering problem formulation. One such constraint is introducing semi-supervised information. In the semi-supervised clustering setting, side information about data points is conveyed in terms of constraints on subsets of data points. Typically, these include pairwise constraints indicating which data points must be in the same cluster, and which data points must not be in the same cluster.

Another example of a clustering variant is capacity constraints on the number of points in each cluster. The constraints indicate some exemplars can only support a bounded number of points, a situation which may arise if the exemplars are thought of as providing some limited resource to every point in the cluster, and different potential exemplars are able to supply different quantities of that resource.

Finding a hierarchical structure in the data to cluster constitutes another generalization of the clustering problem. In this scenario, at a high granularity the algorithm finds many different clusters, and at a low granularity only a few clusters are used to explain

the data. The clusterings at different granularities are expected to be related. Typically, points clustered together at a high granularity should also be clustered together at the low granularity. In other words, the algorithm should find a set of clustering solutions with a decreasing number of clusters in each solution, such that the clusters are nested across the levels of the hierarchy.

In this thesis we develop a notational framework that allows us to derive variants of AP. We then describe the mathematical derivation and experimental performance of these AP-based algorithms for the generalized clustering problems discussed above.

The second focus of this thesis is also related to the overarching question of how to improve AP. We investigate whether we can improve the underlying optimization mechanism of the AP algorithm. At heart, affinity propagation is a heuristic procedure for solving an NP-hard optimization problem (Kariv & Hakimi, 1979; Dueck, 2009). AP finds an approximate solution using an underlying optimization strategy by the name of max-product belief propagation (Pearl, 1988; Kschischang et al., 2001), which we denote by MPBP. We describe MPBP in detail in Chapter 2.

For a small set of well-understood scenarios, using MPBP results in an optimal solution. However, in most cases, such as the case of AP, very little is known about the theoretical guarantees of using MPBP. In fact, algorithms based on MPBP are not even guaranteed to finish executing (or *converge*). When an MPBP based algorithm does converge, the solution it provides is not guaranteed to be interpretable. Even when it converges to an interpretable solution, that solution is not guaranteed to be the solution for the optimization problem. It is therefore interesting to investigate whether we can use AP in conjunction with a different underlying optimization strategy, which has more satisfying theoretical guarantees, while preserving, or even improving upon, AP's empirical performance.

Considerable research effort has been exerted to better understand MPBP (and the highly related sum-product belief propagation algorithm, also discussed in Chapter 2),

mostly in the first half of the previous decade, *e.g.* (Weiss, 2000; Weiss & Freeman, 2001; Wainwright et al., 2004; Yedidia et al., 2005). More recent efforts are concerned with replacing MPBP with a rigorous optimization methodology and algorithms that use the mathematical theory of convex optimization and duality theory (Boyd & Vandenberghe, 2004). For reasons that will become clear in Chapter 2, we refer to these algorithms by the term *convex relaxation methods*.

Many convex relaxation methods that can be used in place of MPBP have been described and developed in the past few years (Wainwright et al., 2005; Kolmogorov, 2006; Werner, 2007; Globerson & Jaakkola, 2007; Komodakis et al., 2007). A striking finding is that many of the resulting optimization strategies, though developed through considerably different route from that of MPBP, are very similar in nature to MPBP. This similarity is apparent through the specific details of the mathematical equations used to describe the optimization procedure.

Nevertheless, unlike MPBP, these approaches lead to algorithms that always converge, and provide conditions for which we can determine with certainty that the algorithm found an optimal solution. Experimentally, there is evidence to show that algorithms based on convex relaxation methods can outperform standard MPBP. For instance, they can converge to good solutions in cases where MPBP does not converge (Globerson & Jaakkola, 2007), and can perform better than MPBP in terms of the obtained solution for problems such as stereo matching and binary image segmentation (Meltzer et al., 2005; Kolmogorov, 2006).

It is therefore interesting to determine whether algorithms based on convex relaxation methods typically provide better or worse results for the exemplar-based clustering problem. The latter part of the thesis is concerned with the derivation and evaluation of convex relaxation methods as alternatives for AP. The performance is evaluated both in terms of how long the different algorithms take to find solutions, and how good these solution are.

The evaluation we conduct is important not only for the purpose of potentially improving AP. AP solves a problem from a class of mathematical models that has been relatively unexplored with respect to convex relaxation methods. This is the class of models with *high-order potentials*, which we define more precisely in Chapter 2. Thus, our experimental results add to the collective findings regarding the empirical behavior of convex relaxation methods when applied to models with high-order potentials.

In summary, this thesis addresses the challenge of extending and expanding the subset of problems that AP can tackle. It also investigates the question of whether we can improve upon the optimization method used within AP by adopting convex relaxation strategies. We hope you enjoy reading it.

Chapter 2

Background

The research described in this thesis builds upon two sub-fields of machine learning. The first is clustering methods, and in particular exemplar-based clustering, semi-supervised clustering, and hierarchical clustering. The second area concerns message passing methods suitable for maximum-a-posteriori (MAP) inference of discrete variable models in factor graphs. This chapter is therefore structured according to these two components.

In Section 2.1, we first describe the general problem of exemplar-based clustering, and give an overview of the affinity propagation algorithm (AP). We also describe the facility location problem (FL) and relate it to the exemplar-based clustering problem. We then give an overview of clustering techniques. In particular, we discuss several semi-supervised clustering and hierarchical clustering algorithms in Section 2.2.

In the second half of the chapter (Section 2.3), we define the general maximum a-posteriori (MAP) inference problem, and describe several algorithms for solving it. We first describe the max-product belief propagation algorithm (MPBP). We then give some background in linear programming relaxations and Lagrangian relaxations for MAP inference problems, and describe some well-known algorithms based on these relaxations.

We conclude by describing how the various background topics relate to the content and structure of the work presented in this thesis, and outline the remainder of the thesis.

2.1 Exemplar-based Clustering

Clustering is the task of partitioning a set of points into subsets (or *clusters*) of points so that the members of every cluster are similar in some sense. *Exemplar-based clustering* is the task of not only performing the partitioning but of identifying for each cluster its most representative member, or *exemplar*. A common characterization for the cluster exemplar is the data point whose overall similarity to other data points in the cluster is maximal. The exemplar-based clustering problem is known to be NP-hard (Kariv & Hakimi, 1979; Dueck, 2009).

As a prerequisite, we must have some way of evaluating the goodness-of-fit of a point to a cluster. Here, we assume that some measure of similarity exists between every pair of data points in the data set we wish to cluster, and that this is given as input to the algorithm. In fact, we do not need to know how pairwise similarities are obtained, we require only the values of those similarities. We assume that the larger the similarity value, the more similar two points are. The relationship need not be symmetric, or correspond to a proper (negated) distance function[1].

Given pairwise similarities, there are several possible variants for an objective function that formalizes the exemplar-based clustering objective of assigning data points to clusters, and of assigning an exemplar for each cluster. Here, we focus on optimizing the within-cluster similarities, though it is possible to optimize various other measures such as between-cluster dissimilarities (see Section 2.2). In the sequel we always assume we wish to construct an objective to **maximize**.

An important design choice we must make is how to measure the within-cluster similarity. Given that choice, an algorithm will typically be a procedure to maximize this quantity. In the context of exemplar-based clustering, the choice is typically related to

[1] As a reminder, a distance, or metric is a non-negative function over two arguments $d(x, y)$ that satisfies $d(x, y) = d(y, x)$ (symmetry), $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality) and $d(x, y) = 0 \Leftrightarrow x = y$

the cluster exemplar. One choice is to maximize the sum of similarities between all data points in the cluster to the exemplar. Indeed, this is what AP aims to accomplish. In such cases the exemplar is the one point all other points are most similar to. The exemplar can be thought of as a good approximation to all other points. Alternative approaches include choosing the point which maximizes the lowest similarity, thus minimizing the worst case approximation for a point by its exemplar.

Another design choice is whether to search for exactly k clusters, where k is given as input, or to assign a penalty term associated with an increased number of clusters and optimize the tradeoff between the number of clusters used and the within-cluster similarity obtained. In the latter case, for any choice of clustering an exemplar set size penalty term is subtracted from the within-cluster similarity term we wish to maximize. Note that without such penalty term, any sensible pairwise similarity (*i.e.* one that assigns maximal similarity between a point to itself) will yield the trivial solution of assigning every point to be in its own cluster and therefore its own exemplar. The penalty term can be as simple as a subtracting a positive constant multiplied by the number of clusters used. Another option is to subtract a positive term that is specifically evaluated for each particular choice of points to be used as exemplars. Towards this, we can associate a particular penalty term with each point. We denote this value by the term *exemplar cost*. Thus, the higher the (positive) cost to subtract, the less likely that point is to be an exemplar. If we consider a cost-based approach, we will assume exemplar costs are also given as input to the algorithm, in addition to the pairwise similarities.

Combinations of the design choices we considered above yield well known algorithms such as ρ-medians and ρ-centers (Hakimi, 1964), as well as the objective optimized by the affinity propagation algorithm, which we discuss next.

In the exemplar-based clustering objective we are interested in, the goal is to find a set of exemplars and assign each non-exemplar data point to an exemplar, such that the sum of similarities of data points to their exemplars is maximized while the total exemplar

costs are minimized. Each data point is associated with its own exemplar cost. This cost is to be paid if the point is used as an exemplar. The tradeoff between maximizing similarities and minimizing exemplar costs is quite intuitive. It corresponds well to the objective function associated with the well-known operational research problem of facility location (FL, (Drezner & Hamacher, 1995)), which is highly related to exemplar-based clustering[2].

However, throughout this thesis, we will adhere to the notation of the original AP formulation of (Frey & Dueck, 2007). There, the quantity associated with every particular choice of exemplar is called a *preference*, and describes to what extent a point is suitable for being an exemplar. Therefore the higher the preference, the more likely the point is to be an exemplar. This is exactly opposite to the cost view, where the higher the cost, the less likely a point is to be an exemplar. Moving from one representation to the other is trivially obtained by negating the sign of the costs to obtain preferences and vice versa.

Importantly, in order for the assignments of data points to exemplars to give sensible clustering solutions, an exemplar must never be assigned to another exemplar – an exemplar must always be assigned to itself. We refer to this as the *exemplar consistency constraint*.[3]

We now state the exemplar-based clustering objective more formally. Given all pairwise similarities, s_{ij}, and all exemplar preferences, p_j, where $i \in \{1 \dots N\}$, $j \in \{1 \dots N\}$, we define a set of N^2 binary random variables h_{ij}. We use $h_{ij} = 1$ to indicate that point i has chosen point j as its exemplar. In this representation $h_{jj} = 1$ indicates that point j is an exemplar.

As a preprocessing step, we first subsume the preferences into the pairwise similarity inputs by adding p_j to the elements s_{jj}, $\forall j$. This is simply a convenient trick for incor-

[2]FL and its associated objective in detailed in Section 2.1.2.

[3]Incidentally, the model where exemplars can choose other exemplars has been investigated in (Leone et al., 2008). However, that model does not output an exemplar-based clustering of the data. Rather, it provides a hierarchical and potentially circular network structure over the data points.

porating the preferences and similarities into one input table, and it allows us to discard the preference inputs.

We can then formulate the exemplar-based clustering objective function in terms of a constrained optimization problem, defined as follows:

$$\max_{\{h_{ij}\}} \mathcal{F}(\{h_{ij}\}) \equiv \max_{\{h_{ij}\}} \sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij} h_{ij}, \tag{2.1}$$

s.t.

$$\sum_{j} h_{ij} = 1 \qquad \forall i, \tag{2.2}$$

$$h_{jj} = \max_{i} h_{ij} \quad \forall j, \tag{2.3}$$

where $h_{ij} \in \{0, 1\}$, and $s_{ij} \in \mathcal{R}$. The first constraint (Eq. 2.2) ensures each point chooses exactly one other point as its exemplar. The second constraint (Eq. 2.3) ensures the exemplar consistency constraint. The goal of the affinity propagation algorithm is to find the settings for $\{h_{ij}\}$ that maximizes $\mathcal{F}$.

2.1.1 Affinity Propagation

At a high level, the AP algorithm can be viewed as iteratively sending messages between points. The messages are scalar values, and there are two types of messages which are sent. First, each point sends to all other points a message indicating to what degree each of the other points is suitable to be its exemplar. These messages, denoted by ρ_{ij}, are referred to as *responsibilities*. Then, each point sends to all other points a message indicating to what degree the point itself is suitable to serve as an exemplar. These messages, α_{ij}, are referred to as *availabilities*. The messages are sent iteratively until (hopefully) the messages no longer change, at which point the algorithm is said to have reached a fixed point, or converged.

The form of the messages is as follows:

$$\rho_{ij} = s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik}). \tag{2.4}$$

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) & i = j, \\ \min\Big[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj})\Big] & i \neq j. \end{cases} \tag{2.5}$$

Upon convergence, the set $\mathcal{K} = \{k | \alpha_{kk} + \rho_{kk} > 0\}$ is chosen as the exemplar set. Each non-exemplar point i is assigned to its most similar exemplar, $k = \arg\max_{k \in \mathcal{K}} s_{ik}$.

2.1.2 Facility Location

Facility location (FL) refers to a collection of optimization problems that have been studied mostly in the field of operational research. These problems have a long history, and some of the variants date back to the 17th century (Drezner & Hamacher, 1995). A review of various FL problems can be found in (Drezner & Hamacher, 1995). In this thesis, we use the term facility location to refer specifically to the variant known in the operational research literature as the uncapacitated facility location problem. (Balinksi, 1966).

Facility location concerns a set of customers that require some service, and candidate locations for establishing facilities to render these services. There is a cost associated with each facility to be opened, as well as a cost associated with each customer-facility pair. The goal is to decide on the subset of facilities to open, and an assignment of each customer to exactly one facility, such that the total facility opening costs as well as customers-facilities costs are minimized.

The exemplar-based clustering objective function optimized by AP (Eqs. 2.3-2.1) shares significant resemblance to the objective function of the facility location problem. In (Dueck et al., 2008), it was shown how to develop an AP-like algorithm for solving

the FL problem. A more formal definition of FL will be given in Section 3.5.

This concludes our brief overview of the exemplar-based clustering framework, as well as the affinity propagation algorithm. We now turn out attention to a more general review of clustering methods and objectives.

2.2 Clustering Objectives and Algorithms

Cluster analysis is one of the core research areas in machine learning, and a comprehensive review is naturally beyond the scope of this introductory chapter. Here, we provide background details designed to put the contributions of this thesis in perspective. We briefly discuss and contrast approaches that use different optimization goals with those of the exemplar-based clustering framework. In addition, in light of the extensions to AP presented in this work, we give an overview of clustering objectives that include additional structure and constraints, which go beyond standard clustering.

2.2.1 Different Approaches to Clustering

There are some fundamental differences between the approach taken by popular clustering algorithms such as k-means or Gaussian (and other) mixture models, and that taken by exemplar-based clustering. A Gaussian mixture model (GMM) is a generative model for data. Each data point is generated from one of k clusters, and each cluster is associated with its Gaussian cluster parameters of mean and covariance. Given the data, the clustering problem is that of finding the parameters associated with each cluster via maximum likelihood estimation, and associating each point with a multinomial distribution indicating to which cluster it belongs in a probabilistic manner. k-means clustering can also be cast as an algorithm for obtaining estimates for the cluster means, where in terms of the generative model, the covariances matrices are all assumed to be shared, with a fixed known diagonal covariance. In terms of the optimization procedure, hard

assignments are made by explicitly assigning a point to a single cluster and modifying the mean estimates iteratively until convergence. In general, statistical parametric mixture models assume that data is generated from some well defined parametric distribution, and attempt to obtain point estimates for the parameters that maximize the likelihood of the observations, or the posterior distribution of the parameters given the observations (Dempster et al., 1977; McLachlan & Krishnan, 2008). Bayesian non-parametric models, despite their name, do in fact assume a parametric form for the data distribution as well, but integrate over a parameter space endowed with an underlying parametric prior distribution (Jain & Neal, 2000; Blei et al., 2003; Teh et al., 2006). Various MCMC sampling techniques are required for inference and these models are typically more computationally intensive than their point-estimate counterparts.

A major difference between the exemplar-based clustering approach and the parametric approaches stems from the domain of the optimization function. In parametric approaches, the optimization domain is continuous and the function of interest is non-convex, for reasonably non-trivial models. The continuous optimization typically lead to local optima that depend on starting conditions. In contrast, exemplar-based clustering optimizes a discrete function. The set of parameters, or cluster centers to choose from for exemplar-based clustering is the power-set of the data[4]. This is a challenging combinatorial optimization problem, and any method that requires parameter smoothness is not applicable. However, methods for solving exemplar-based clustering problems like affinity propagation, do not require that the pairwise costs satisfy the triangle inequality or symmetry, and AP has been applied successfully in nonmetric spaces (Dueck & Frey, 2007). Unlike mixture models, affinity propagation can use arbitrarily complex cost functions since it does not need to search or integrate over a parameter space. For example, in (Lindorff-Larsen & Ferkinghoff-Borg, 2009), affinity propagation was used to cluster

[4]Recall that the power-set of a set is the set of all possible subsets. Since the solution for exemplar-based clustering is a subset of points, the collection of all possible solutions is exactly the power-set.

ensembles of proteins, using the negative Jensen divergence as the pairwise similarity function.

Another advantage of exemplar-based approaches, as opposed to that of a statistical representation of cluster centers, is the ability to automatically identify data points that are prototypical. In the case of biological data, a hierarchical exemplar-based clustering method can describe a tree structure on proteins that may be related to evolutionary processes. Algorithms that merge data points and clusters together (c.f. Hierarchical Agglomerative Clustering (Jardine & Sibson, 1968)), cannot be as easily interpretable as corresponding to meaningful biological structures.

In terms of the input to the algorithm, as we previously noted, there are no restrictions on the form of the similarities, and they do not need to be symmetric, but they should correspond to the form of the objective function, so that bigger similarity values correspond to more similar points. This formulation for the similarities is quite different from many metric-based approaches, such as k-means, where the data is assumed to be embedded in a vector space over which a metric can be evaluated. In such a setting the similarity can be defined as the negative distance, but note that the optimal value never exceeds zero, since the similarity is derived from a valid metric, and in addition the triangle inequality must hold. Likewise, there is no restriction on the exemplar preferences as long as they respect the form of the objective function, so that larger exemplar preferences correspond to points more likely to be exemplars.

There exist non-exemplar-based clustering clustering objectives that are similar to exemplar-based clustering in terms of the form of the input. Recall that the exemplar-based clustering input is expected to be in terms of pairwise distances, or affinities. A prominent example of a method operating on pairwise affinities is the spectral clustering algorithm (Ng et al., 2001). The algorithm is also known as Normalized cuts, depending on the variant (Shi & Malik, 2000; von Luxburg, 2007). In the spectral clustering framework, the intuitive objective is to find cuts in a weighted graph, whose nodes are

the data points, and edge weights are the distances. The cut is to be chosen such that between-cluster distances are maximized (or equivalently, the similarity is minimized). The issue arising from this general and intuitive formulation is that the trivial solution of, for example, the two-cluster problem, turns out to be a cut having one point separated from all the others. Note that this is in contrast to the exemplar-based clustering objective, which yields (if phrased only in terms of the total cluster similarities objective) a trivial solution of putting each point in its own cluster. Thus, minimizing between-cluster similarities brings about different trivial settings to be avoided than those that arise when maximizing within-cluster similarity. Where as in exemplar-based clustering, the trivial solution is avoided by introducing a per-cluster exemplar preference term, to alleviate the trivial solution in the cut formulation, one introduces a penalty term that requires clusters to have a non-trivial amount of points, or a non-trivial total within-cluster weights. The modified formulations result in the normalized-cut and ratio-cut formulations respectively, both of which are NP-hard. The optimization methods constructed via a relaxation of these problems yield the spectral clustering algorithm variants.

In summary, we have reviewed several design choices, which yield various algorithmic approaches for clustering. Different approaches may require different inputs, and may output different results. While there is no single best approach, we should pay attention to the particularities of the task at hand when choosing an algorithm, since they determine whether a property of an algorithm is desirable, or whether it is a caveat. Exemplar-based approaches, such as affinity propagation, are particularly suited for cases that have one or more of the following characteristics: (1) there is a clear measure of similarity between points, but it is not necessarily clear that a vector-space representation is suitable. The measure of similarity is perhaps derived from expert opinion, or hand-crafted for a particular data set; (2) it is difficult to directly estimate or explore good ranges of k, the number of clusters to choose, but it is easy to come up with relevant preferences or preference ranges for the data (a typical choice is to take the median

similarity); (3) we are interested in a useful, intuitive representation of the clusters, and representing each cluster with its most prototypical point provides a more intuitive description than a cluster average. Examples may include the case of images, for which averaging pixel intensities is not necessarily meaningful, or DNA sequences, for which we may be able to describe similarities, but it may not make much sense to represent clusters by a median sequence.

This concludes our discussion of alternative strategies and methods for clustering, and the pros and cons of the exemplar-based clustering approach. We now describe extensions and variants of the general clustering problem, and we discuss existing methods for addressing these variants.

2.2.2 Constrained Forms of Clustering

The capacitated clustering problem (Mulvey & Beck, 1984) deals with the case where each cluster has an upper limit L on the number of points it can contain. A related problem is that of the capacitated facility location problem (Sridharan, 1995). There, each facility is constrained to serve only a maximal given number of customers. The hard limit on L we use here can be replaced with a cost that increases as cluster size increases. A particular choice of this cost function that stems from an underlying Dirichlet process prior was used in (Tarlow et al., 2008).

2.2.3 Semi-Supervised Clustering

Semi-supervised clustering algorithms are concerned with finding good partitions of data in the presence of side information. Two popular forms of side information are partial labels and the more general instance-level constraints. We consider the case where side information is given in the form of instance-level constraints on pairs of data points. *Cannot-link* constraints indicate the two data points cannot be in the same cluster while *must-link* constraints indicate the data points must be in the same cluster. This ter-

minology was introduced by Wagstaff *et al.*, who were among the first to investigate semi-supervised clustering with instance-level constraints (Wagstaff & Cardie, 2000).

There is a distinct difference between side information given in the form of partial labels and that given in the form of instance-level constraints. The two are not equivalent since labeled data can always be used to construct instance-level constraints while the converse does not hold in general, making instance-level constraints weaker in terms of the amount of information they carry. However, instance-level constraints are often faster or cheaper to obtain than labels, and can sometimes be automatically collected (Wagstaff et al., 2001; Klein et al., 2002; Shental et al., 2003).

An example of where potentially noisy instance-level constraints are easier to obtain than partial labels is from user click-streams. We can assume some web-based documents to be similar if many users spend time looking at both as a result of a query, but that in itself does not indicate what category (or label) the documents are from. Nevertheless, if we are interested in clustering documents, we can make use of that information to increase the probability that such documents should be in the same cluster. Another example where we can cheaply and quickly obtain instance-level constraints is to show users pairs of images and query whether the images are of the same, or of different objects. This is considerably less labor intensive than specifically asking users to label the images with their objects. Once again, it is clearly useful to incorporate the same/different information when clustering the images.

Another difference between partial labels and instance-level constraints is that instance-level constraints do not directly provide information about the total number of clusters or classes in the data. It is always possible to construct the transitive closure for any set of instance-level constraints: if points i and j are constrained to be in the same cluster, and points j and k are likewise constrained, then it follows that i,j, and k must all be in the same cluster. Similarly, if i and j must be in the same cluster, but i and k cannot be in the same cluster then j and k cannot be in the same cluster. Note, however,

that this grouping of constraints into, *e.g.* L groups, tells us only that there are **at least** L clusters in the data. It may be the case that for some clusters we were given no constraints. Thus, many algorithms that incorporate instance-level constraints, such as clustering algorithms that attempt to partition the data subject to the constraints, can detect clusters composed of points for which no side information is given (Wagstaff et al., 2001; Klein et al., 2002; Shental et al., 2003). When, on the other hand, side information is provided as partial labels, it is often assumed that the unique number of class labels in the partially labeled set should be the number of identified classes (Xiao et al., 2007; Leone et al., 2008). In order for such algorithms to succeed, there must be at least one labeled example from each class in the data. The property of being able to detect clusters for which no side information is given is desirable for cases where some clusters are 'easy' to detect, while others may require user intervention.

Thus, semi-supervised clustering with instance-level constraints is a quite different problem from that of multi-label semi-supervised classification, such as, for instance, image segmentation with user seeds. Considering the latter in particular, the semi-supervised classification problem can be formulated in terms of a Markov Random Field (see Section 2.3.1), where the unary potentials can be modified based on the user seeds and efficient algorithms such as graph-cuts can be used (Boykov & Jolly, 2001). Since, in the instance-level constraints formulation we do not know how many labels there are, we cannot define the node cardinalities for the MRF. Even if we knew the total number of labels, or objects, the pairwise constraints do not give us labels for particular datapoints, or pixels, and so we cannot use them to modify unary potentials. Furthermore, if we choose to modify directly must-link and cannot-link constraints on pairwise connections, we obtain non-submodular pairwise potentials, for which graph-cuts is not applicable.

One prominent approach for devising semi-supervised algorithms for instance-level constraints is to modify standard unsupervised clustering algorithms so that they explicitly account for the constraints. This was done for k-means clustering (Wagstaff et al.,

2001), and mixture of Gaussians (Shental et al., 2003). A missing aspect of this approach is that it does not explicitly propagate constraints. Intuitively, if we look at the set of points that are very close to either point in a must-link constraint, it is likely that these points should also be in the same cluster as the must-link constraint points. This property is sometimes referred to as space-level constraints (Klein et al., 2002), and is often imposed by adapting the similarities (or distances) between data points to better align with the constraints. A good distance function for the data should intuitively assign a small distance to a pair of points with a must-link constraint, and a large distance to a pair of data points with a cannot-link constraint. The adjusted metric can be used as input to an unsupervised clustering algorithm (Xing et al., 2003). Naturally, the approach of adapting similarities can be combined with the approach of explicitly accounting for constraints (Klein et al., 2002; Basu et al., 2004).

2.2.4 Hierarchical Clustering

Hierarchical clustering is a variant of cluster analysis which builds a hierarchy of clusters. The most widely used approach for generating the hierarchy is a greedy bottom-up method known as hierarchical agglomerative clustering (HAC, (Jardine & Sibson, 1968; Duda & Hart, 1973)). In HAC, each data point is initialized to be in its own cluster, and pairs of clusters are subsequently merged according to some pre-defined measure of cluster coherency typically referred to as *linkage*. The most common linkage methods are maximum, minimum and average similarity between all pairs of data points in two clusters that are under consideration to be merged. HAC will merge the two clusters with the highest linkage.

HAC is a highly popular techniques that is widely used, particularly for computational biology, for re-organizing rows and columns of data such that pattern in the data become visually pronounced. The actual hierarchy, often depicted as a tree or *dendrogram* does not lend itself easily to interpretability. There is no notion of the clustering or the number

of clusters at a particular level of the hierarchy. In addition, as discussed before for parametric clustering methods, algorithms that merge data points and clusters together cannot be as easily interpretable as corresponding to a meaningful structure using an exemplar representation.

An alternative approach for constructing a hierarchy is to search for nested clusterings of a pre-specified number of clustering granularities, or levels. A greedy approach first described in (Xiao et al., 2007) is to recursively apply AP to the exemplar set found in the previous layer. ρ-medians (Hakimi, 1964) can also be used as an underlying algorithm for a greedy bottom up approach.

The generalization of standard FL to a hierarchical version is for the case where we have at the bottom layer a set of customers and potential facilities, and then at every additional layer a set of potential facilities serving the previous layer's facilities. The FL literature contains several different variants that are called hierarchical facility location. The one relevant for the work presented here is equivalent to that stated in *e.g.* (Kantor & Peleg, 2009).

This concludes our review of clustering variants, as well as some of the notable existing algorithms for solving these clustering problems. This is also the end of the first half of the introduction, which is devoted to clustering. We now switch gears and describe a seemingly unrelated area of machine learning research – that of finding modes of probability distributions. We will conclude the introductory chapter, by discussing the relationship between these two topics, as they pertain to this thesis.

2.3 Algorithms for MAP Inference in Undirected Graphical Models

Probabilistic graphical modeling is an important approach for modeling observational data, unknown processes that may have generated the data, and the relationships be-

tween these observed and unknown quantities. It is a framework that models data in a way that respects the inherent noise in natural phenomena, and our uncertainty about the correctness of our observations and assumptions. Probabilistic graphical modeling integrates ideas from diverse fields such as graph theory, probability theory, statistical mechanics, and information theory. A thorough review can be found in (Koller & Friedman, 2009).

At a high level, in probabilistic graphical modeling we assume our observations can be expressed as random variables. Relationships among these variables are described via a directed or an undirected graph and by specifying the probability distributions of the random variables. We may also guess at the existence of additional random variables we cannot directly observe (hidden variables) and incorporate them in the model. Together, these components give rise to a description of a joint probability distribution over all variables in the model, that can hopefully be written in a compact form amenable to computational manipulations. We can then ask various questions of interest about variables in our model. Through the combined effort of research in the field, researchers and practitioners have at their disposal tools and algorithms to answer these questions, either exactly or approximately.

We begin by describing a particular graphical model (the factor graph) to encode joint probability distributions over a set of variables, and an algorithm (max-product belief propagation) that answers some of these common questions.

2.3.1 The MAP Inference Problem

Factor graphs (Kschischang et al., 2001) are a type of undirected graphical models that allow for a flexible representation of a joint probability distribution. Given a set of random variables $\mathbf{x} = \{x_1, \ldots, x_N\}$, factor graphs can be used to represent distributions of the following form:

$$P(\mathbf{x}) = \frac{1}{Z}\prod_{c=1}^{C}\psi_c(\mathbf{x}_c). \tag{2.6}$$

Here $\mathbf{x}_c$ denotes some subset of $\mathbf{x}$, each ψ_c is a non-negative function (also called a potential, or a factor), and Z is a normalization factor ensuring that $P(\mathbf{x})$ is a valid distribution that integrates to one. Let us assume that each random variable takes values in a discrete finite set. For example, $x_i \in \{1, 2, \ldots, K_i\}\ \forall i$. Then Z, also known as the *partition function*, is simply $\sum_{\mathbf{x}}\prod_c \psi_c(\mathbf{x}_c)$. Thus, Eq. 2.6 gives a valid description of a distribution as a product of terms operating on different subsets of the random variables.

The factor graph corresponding to Eq. 2.6 is a bi-partite graph with two types of nodes. N *variable nodes*, one per variable, and C *factor nodes*, one per factor. A variable node i is connected to a factor node c iff $x_i \subseteq \mathbf{x}_c$ (x_i is an argument of ψ_c). As an

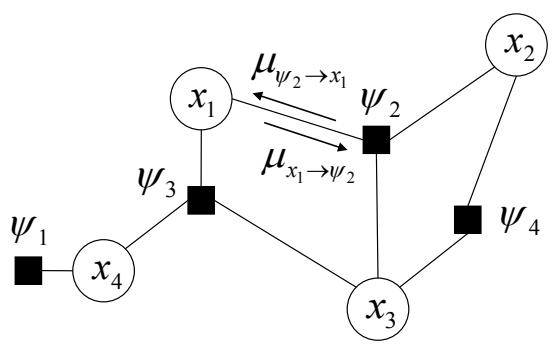


Figure 2.1: The factor graph model.

example, consider the factor graph in Fig. 2.1. Factor nodes are denoted by shaded-in squares, and variable nodes are denoted by circles. In this example, $\mathbf{x} = \{x_1, x_2, x_3, x_4\}$, and the joint probability distribution described by the factor graph is given by

$$P(\mathbf{x}) = \frac{1}{Z}\psi_1(x_4)\psi_2(x_1, x_2, x_3)\psi_3(x_1, x_3, x_4)\psi_4(x_2, x_3). \tag{2.7}$$

Probabilistic inference in factor graphs may refer to one of several tasks. One such task is that of calculating marginal or conditional probabilities. For instance, determining in the above graph the marginal $P(x_3)$, or the conditional $P(x_2|x_1)$. Another task is that of

approximating the value of $\log Z$, since it is typically computationally intractable. The task we will be most interested in is that of finding the single setting for the variables that we denote by $\mathbf{x}^*$, which **maximizes** the joint probability distribution: $P(\mathbf{x}^*) = \arg\max_{\mathbf{x}} P(\mathbf{x})$. This task is often referred to as the maximum-a-posteriori (or MAP) inference problem[5].

We note that factor graphs are relatively new graphical representations for factorized probability distributions. Historically, undirected graphical models were described in terms of simple graphs (containing no factor-nodes) with edges between nodes, and potentials operating on maximal cliques in the graphs. These models, originally used for representing ferromagnetic fields in statistical mechanics, are known as Markov random fields, or MRFs (Kindermann & Snell, 1980). For models on which interactions between variables are restricted to at most pairwise interactions, MRFs remain quite popular and useful for representing the problem structure and for inference. However, when the potentials operate on large subsets of variables (in which case we say the model is *high-order*), it is often more convenient to work with factor graph notation.

Finally, while the functions ψ_c must be non-negative, they may take on the value of zero. This allows us to express the fact that some settings of the random variables have zero probability. Therefore, if our goal is to find non-zero probability assignments of all the variables, we should avoid such settings. Since avoiding these settings poses a constraint we must satisfy exactly, in such cases we say there are *hard constraints* on the variable assignments in the problem formulation.

2.3.2 Max Product Belief Propagation

A factor graph is said to be a *tree* if there is only one way to traverse edges on the graph between any pair of variable nodes. The factor graph shown in Fig. 2.1 is not a tree

[5]This name may be surprising in this context, since the term implies we wish to maximize a *posterior* distribution. It is in fact a posterior distribution, where we assume the observations are subsumed into the factor nodes, and conditioned on, and all denoted variables are hidden.

since, for example, there are two ways to get from x_4 to x_2.

For tree-structured factor graphs, the marginal estimation problem can be solved exactly using the *sum-product* algorithm, and the MAP inference problem can be solved exactly using the *max-product* algorithm (Kschischang et al., 2001). These very similar algorithms are particular to factor graphs. They express the well known belief propagation (BP) and belief revision algorithms described by Pearl for tree structured graphs (Pearl, 1988). Thus, we denote the sum-product algorithm by SPBP and the max-product algoirthm by MPBP, respectively.

Belief propagation[6] has been known under various names for various graph structures, in various communities. The Viterbi algorithm (Viterbi, 1967), The Kalman filter (Kalman et al., 1960), and the forward-backward algorithm for hidden Markov models are just a few examples. In fact, all of the above algorithms are based on the framework of dynamic programming.

In the case of tree-structured factor graphs, SPBP and MPBP proceed by starting at some variable node taken as the root, sending numerical values known as *messages* from that root node to all its neighboring factor nodes, and then from those factors nodes to all their unvisited neighboring variable nodes until all nodes have received messages. This is denoted as the forward pass. Then, messages are passed in the reverse order, from the last nodes to receive messages (the *leaves*) towards the root. This is the backward pass of the algorithm.

The form of the MPBP messages from a variable node to a function node, $\mu_{i\to c}(x_i)$, and from a function node to a variable node, $\mu_{c\to i}(x_i)$, is given by the following expressions:

[6]Here we use BP to denote both belief propagation and revision, since in the literature the term BP often refers to either algorithm.

변수→함수 $$\mu_{i\to c}(x_i) = \prod_{b\in\mathcal{N}(i)\setminus c} \mu_{b\to i}(x_i), \tag{2.8}$$

함수→변수 $$\mu_{c\to i}(x_i) = \max_{\mathbf{x}_c\setminus x_i}\Big[\psi_c(\mathbf{x}_c) \prod_{j\in\mathcal{N}(c)\setminus i} \mu_{j\to c}(x_j)\Big], \tag{2.9}$$

where $\mathcal{N}(i)$ denotes the set of all factor nodes that are neighbors of (directly connected to) variable node i, and $\mathcal{N}(c)$ denotes the set of all variable nodes that are neighbors of factor node c.

Recall that we assumed x_i takes on values in $\{1, \ldots, K_i\}$. A convenient way to think of a message to or from node i is as vector of length K_i, where the entry at position n is given by $\mu_{i\to c}(n)$[7]. The very first message from the root node to the factor node is initialized to be a vector of ones. See also Fig. 2.1 for a visualization of the messages.

The SPBP messages take an almost identical form, where the maximization in Eq. 2.9 is replaced with a summation. In the sequel we restrict our attention to a discussion of the MPBP algorithm. Since these algorithms operate by sending messages, they belong to a family of procedures termed *message passing* algorithms. We describe other message passing algorithms in Section 2.3.3

When executing MPBP, it is often convenient to operate in the log domain. The algorithm it then sometimes called the max-sum BP algorithm. Letting $\phi_c = \ln \psi_c$, the messages have the following form:

[7]This is of course true only if we do make the restriction that x_i takes values from such a well behaved set. In general, we only make the assumption that it takes values in a discrete finite set, and the messages are, in fact, functions specifying a mapping between the variable value to a corresponding message value.

$$\mu_{i\to c}(x_i) = \sum_{b\in\mathcal{N}(i)\setminus c} \mu_{b\to i}(x_i), \tag{2.10}$$

$$\mu_{c\to i}(x_i) = \max_{\mathbf{x}_c\setminus x_i}\Big[\phi_c(\mathbf{x}_c) + \sum_{j\in\mathcal{N}(c)\setminus i} \mu_{j\to c}(x_j)\Big]. \tag{2.11}$$

When the forward and backward passes are completed, the solution to the MAP inference problem is computed by first calculating the *beliefs* at each node:

$$\beta_i(x_i) = \sum_{b\in\mathcal{N}(i)} \mu_{b\to i}(x_i), \tag{2.12}$$

and then assigning to each x_i the value which maximizes the belief at its corresponding node i, $\max_{x_i} \beta_i(x_i)$. This joint assignment of all the x_i variables is the solution for the MAP inference problem. The procedure of obtaining the solution from the messages is also known as *decoding* or *rounding*.

Loopy belief propagation: When the tree-structured factor graph assumption does not hold, and the factor graph contains loops, there is no natural starting and ending point for executing the algorithm. Rather, the messages are sent at some arbitrary order, in an iterative fashion, until no change is seen in the value of the messages at iteration t and $t+1$. Other convergence criteria can be used as well. For instance, that of observing no change in the decoded assignment when using the current messages to compute beliefs as compared to the previous iteration.

However, when applied to graphs with loops, MPBP and SPBP are not guaranteed to converge. When either variant of the algorithms do converge, it is not guaranteed to converge to meaningful solutions (*e.g.*, for MPBP some of the variables values cannot be decoded from the messages, or in cases of hard constraints, some of the constraints are violated). Even when all messages have converged, are easily decodable, and the solution is in the feasible set for a hard constraint problem, the solution is not guaranteed to indeed

be the optimal solution. This is perhaps the reason why running the algorithm in such scenarios is typically referred to as running 'loopy BP'. Nevertheless, ample evidence, dating back to (Gallager, 1963) indicate that in practice, the algorithm often converges to meaningful and useful solutions faster and more accurately than alternative methods (see also (Frey & MacKay, 1997)).

From a theoretical point of view, our understanding of the algorithmic properties of loopy BP is not yet complete. Weiss showed that it is exact in graphs with a single loop (Weiss, 2000), and later on proved that if the MPBP algorithm converges to a fixed point, a decoding obtained by using the solution is provably better than many alternative solutions that can be obtained by graph-topology based modification of the decoded assignment (Weiss & Freeman, 2001). A major contribution is that of Yedidia *et al.*, who proved that fixed-points of the SPBP algorithm are stationary points (and thus potentially local optima) of the Bethe free energy (Yedidia et al., 2005). A similar characterization of MPBP as an optimization procedure with provable properties has yet to be fully specified. However, several authors have examined the variational approach and its relation to SPBP and MPBP and formulated parametric forms of the free energy. The variational formulation gives rise to an optimization problem, and depending on the setting of the parameters one can recover either SPBP and MPBP, or variants that provably converge (Weiss et al., 2007; Meltzer et al., 2009; Hazan & Shashua, 2010). It is not clear, however, what is the reason for the relative success of MPBP in solving various problems where provably convergent variants that optimize a well understood function do not perform as well.

From a practical perspective, a standard practice, which is not justified by a theoretical argument, is to apply *damping* to the loopy MPBP messages:

$$\mu^t_{i\to c}(x_i) = \lambda \cdot \mu^{t-1}_{i\to c}(x_i) + (1-\lambda) \cdot \sum_{b\in\mathcal{N}(i)\setminus c} \mu^t_{b\to i}(x_i), \tag{2.13}$$

for some choice of $\lambda \in [0, 1)$. Note that in the equation above we have introduced a time dependent indexing, to indicate the iteration in the loopy setting. Damping does preserve the fixed-points of the non-damped algorithm, and does not add any new fixed-points.

MPBP with high-order potentials: Consider Eq. 2.11. It requires maximization over all but one of the variables associated with a potential function. Recall we are concerned with discrete valued variables. Even for the case where these values are binary, the number of possible assignments to the function is $2^{|\mathbf{x}_c|}$. Therefore, when the cardinality, $|\mathbf{x}_c|$ of a high-order potential is indeed high, we cannot carry out the required optimization naively, and it may seem we cannot use the MPBP framework for inference.

However, there are many high-order potentials that are structured in a way that allows us to perform the optimization efficiently. In subsequent chapters, we will describe several such potentials and their corresponding efficient optimization in detail.

Reparameterization view of MPBP: Let ϕ denote the set of all potential functions associated with a particular factor graph. When two such collections of potentials for the model, $\phi, \tilde{\phi}$, evaluate to the same value for every configuration of $\mathbf{x}$, such that

$$\sum_{c=1}^{C} \phi_c(\mathbf{x}_c) = \sum_{c=1}^{C} \tilde{\phi}_c(\mathbf{x}_c) \;\; \forall \mathbf{x}, \tag{2.14}$$

we say that $\tilde{\phi}$ is a *reparameterization* of ϕ. As shown in (Wainwright et al., 2004), for pairwise MRFs, belief propagation can be described in terms of message passing equations, or, equivalently, by a series of reparameterizations (also called equivalent transformation) of the original potential functions. In (Kolmogorov, 2006) it is shown that this relationship is bi-directional. Every message vector defines a reparameterization, and every reparameterization can be expressed by messages up to a constant.

More specifically, the reparameterization view assumes we calculate one round of messages to be sent on an MRF. Then, we take the values of all such messages and use them to modify the potential functions. Once we have done that, we can 'erase' the

messages and start message passing again, using the modified potentials.

This concludes our brief exposition of the factor graph notation for describing joint probability distributions, the MAP inference problem, and the loopy max-product belief propagation algorithm for MAP inference. We now turn our attention to an equivalent formulation of the MAP inference problem as that of a combinatorial optimization problem. We describe some of the optimization strategies recently adopted from the research field of optimization theory, and discuss some of the interesting relationships between these strategies and max-product belief propagation.

2.3.3 MAP Inference with Convex Relaxation Methods

Recall that in the MAP inference problem we are interested in finding an assignment $\mathbf{x}^*$ for the discrete random variables that maximizes the joint probability distribution,

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}) = \arg\max_{\mathbf{x}} \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c) = \arg\max_{\mathbf{x}} \frac{1}{Z} e^{\sum_{c=1}^{C} \phi_c(\mathbf{x}_c)}, \tag{2.15}$$

where we define $\phi_c = \ln \psi_c$. Due to the monotonicity of the logarithmic function, we can recast our problem as that of finding the maximal log-probability. Since the partition function Z is a constant with respect to this optimization problem, we can discard it and arrive at

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \phi(\mathbf{x}) \equiv \arg\max_{\mathbf{x}} \sum_{c=1}^{C} \phi_c(\mathbf{x}_c). \tag{2.16}$$

Since the variables to optimize over are discrete, this is a combinatorial optimization problem. Recall our previous discussion of hard constraints. In the new formulation, the logarithmic operation on zero settings of ψ_c evaluates to $\phi_c = -\infty$, and so, clearly, these settings can never maximize the objective function. We will refer to the constraint-violating settings as *non-valid configurations*.

One approach for obtaining an approximate MAP inference solution is that of using

strategies from optimization theory. Working in the formulation of Eq. 2.16 (which we denote as the *primal objective*), we can often design another function, called the *dual objective* that operates on a different set of variables, called the *dual variables*. This function is always an upper bound on the primal problem. The dual function is also designed to be convex, and we can therefore use tools from convex optimization theory to find the minimizing solution of the dual problem[8]. In some cases, when a condition called *strong duality* holds, minimizing the dual gives us an optimal solution to the primal problem. Even when strong duality does not hold, we may still be able to use the dual solution to find good primal solutions.

Once we have designed a convex dual objective, we proceed to minimize the dual function. We refer to this general framework of the convex dual minimization as convex relaxation. Two closely related methods of constructing the dual objective are the linear programming relaxation and the Lagrangian relaxation. Two main strategies for minimizing the dual objective are block coordinate descent, and sub-gradient descent using dual decomposition[9].

In recent years, there has been much interest in solving MAP inference problem using convex relaxation approaches, and many seminal works have been published on this topic. Some important examples, based on linear programming relaxation (LPR) in conjunction with block coordinate descent are the works of (Wainwright et al., 2005; Kolmogorov & Wainwright, 2005; Globerson & Jaakkola, 2007; Werner, 2007; Sontag & Jaakkola, 2009). Note that there are several equivalent ways of stating the primal objective, as well as different ways of constructing a dual objective from the LPR. There are also different choices in how to perform the block coordinate descent steps. These degrees of freedom give rise to the different variants of LPR block coordinate descent algorithms. More

[8]When functions are known to be convex there are many well-understood algorithms to find the global optimum. From the theoretical perspective the optimization is considered easy. Note that it is still the case that finding solutions in a reasonable time may be a practical challenge.

[9]In the next section we shall briefly explain these terms. We also give more details on the linear programming relaxation in Appendix A.

recently, important advances were made in introducing the Lagrangian relaxation in conjunction with the dual decomposition strategy for MAP inference (Komodakis et al., 2007; Sontag et al., 2010).

Importantly, because of their formulation as convex optimization methods, the above algorithms come with theoretical guarantees on convergence, upper bounds on the optimal solutions, and *certificates of optimality* (conditions for which we can ascertain the solution is optimal) under well-defined sufficient conditions. As optimization strategies, they are considerably better behaved, from a theoretical perspective, than max-product belief propagation.

Quite surprisingly, the optimization procedure in the block coordinate descent based strategies often closely resembles the max-product belief propagation messages. While this similarity may indicate a close connection between MPBP and convex relaxation methods, the exact nature of the relationship between the clearly defined optimization procedures and the MPBP algorithm has not been fully characterized and remains an open area of research. See (Yedidia et al., 2005; Weiss et al., 2007; Meltzer et al., 2009; Hazan & Shashua, 2010) for a thorough discussion of the variational approach, free energies and convex free energies, and how they relate to various message-passing schemes.

A considerable portion of the above algorithms were developed specifically for graphs with pairwise interactions (potentials ranging over at most two variables). However, the general convex relaxation framework is applicable to high-order potentials. Some of the above algorithms were developed for, or consequently extended to address high-order models (Globerson & Jaakkola, 2007; Werner, 2008; Komodakis & Paragios, 2009).

The reparameterization idea is applicable for convex relaxation methods. Dual variables take the place of messages, and can be thought of as providing a reparameterization. We will return to this in more detail in Chapter 7.

We now give some brief, high-level explanations of some terms used in the description

of the general dual optimization strategy. We then provide the relevant background for two convex relaxation based MAP inference algorithms that are applicable to high-order models. The first is max-product linear programming (MPLP, (Globerson & Jaakkola, 2007)), and the second is the dual decomposition sub-gradient descent method (DD, (Komodakis & Paragios, 2009; Sontag et al., 2010))

The linear programming relaxation: This is a strategy for obtaining an easy to optimize problem by modifying a hard one. We use the MAP inference objective function described in Eq. 2.16 (which is a function over discrete terms) to define a new, equivalent problem on a very large collection of indicator binary variables $\mu_c(\mathbf{x}_c)$, alongside the original terms $\phi_c(\mathbf{x}_c)$, and a set of additional constraints. This in itself does not yield an easier problem. However, we then *relax* the new problem by only requiring that the $\mu_c(\mathbf{x}_c)$ variables take on values in the range $[0, 1]$, instead of in the set $\{0, 1\}$. This turns out to result in a linear program optimization problem (or LP).

LPs are theoretically easy to optimize, and there are many good, generic algorithms and tools available to solve such problems. Of course, the relaxed problem is only a surrogate for the original problem, and the solution may be composed of fractional assignments for the $\mu_c(\mathbf{x}_c)$ variables. Such solutions can often be used to obtain a reasonably good solution for the original problem. When the solution to the linear program relaxation happens to be integral, then the LP solution is **guaranteed** to be the solution to our hard MAP inference problem. We then say that the relaxation is *tight*.

As discussed previously, the dual objective is always convex and thus also easy to optimize. A classical result in duality theory for convex optimization state that for linear programs, the dual objective is also a linear program, and strong duality holds under mild conditions. Therefore, from the point of view of obtaining a solution, optimizing the dual is equivalent to optimizing the primal, as at the optimum of one we obtain a solution for the other. It may therefore seem unnecessary to work in the dual formulation, as it is possible to use a generic LP solver on the primal.

However, major progress in MAP inference strategies was made when researchers explored dual optimization techniques for the LP. One reason for that lies in improved computational efficiently and feasibility. The number of dual variables and constraints over which to optimize in considerably smaller than the number of primal variables and constraints, resulting in better run-times for dual based methods (Yanover et al., 2006). Though LP optimization is easy in the theoretical sense, commercial LP solvers may not be applicable for big enough problems with millions of variables and constraints, where convex relaxation methods remain applicable (Yanover et al., 2006).

The Lagrangian relaxation: In the Lagrangian relaxation framework, we create a dual objective directly from the primal objective of Eq. 2.16. It is a general framework applicable to maximizing functions that can be written as sums of simpler functions. The construction introduces copies of the original variables to optimize over, one per simple function. Then the original problem is rewritten as a constrained optimization problem .The goal is to optimize over the individual functions with respect to their copy of the variables, subject to the constraint that the copies must agree with the original variables. By relaxing this constraint using Lagrange multipliers that penalize disagreements rather than forbid them, we arrive at a dual function to minimize.

Block coordinate descent: The coordinate descent framework for optimizing a multivariate function is simply to iteratively choose a particular coordinate, holding all other variables fixed, and optimize the function with respect to only the chosen coordinate. It is particularly relevant for cases where we can analytically optimize the function with respect to the free variable. In *block* coordinate descent, we optimize not with respect to just one free variable, but to a whole subset, or block, or variables. This optimization strategy ensures that at every iteration, the function value does not increase.

One important benefit of using the dual formulation of the MAP inference problem is that it allows us to construct dual optimization strategies that are specific to the problem at hand, and respect the underlying graphical model of the MAP inference problem. In

the case of dual based block coordinate descent, recall that the dual objective is specified in terms of dual variables. The dual variables closely correspond to the underlying factor graph structure. For every edge in the graph connecting a function node to a variable node, and for every value that variable node can take, there is an associated dual variable. When we perform block coordinate descent, we obtain iterative dual-variable update equations that are very similar to the form of the messages in max-product belief-propagation.

We note a well known caveat of block coordinate descent optimization – it is not guaranteed to converge to the optimal dual solution. Instead, it may converge to a point that is suboptimal (*c.f.* (Powell, 1973)) by getting trapped in a cyclic series of moves that do not lead to changes in the objective, yet are not at an optimum. For example, if the LP primal problem is not strictly convex, *i.e.* it has a plateau of optimal solutions, then the dual is non-smooth. This is a typical setting in which block coordinate descent methods may get stuck in such non-optimal 'corner' solutions.

Sub-gradient descent: A well known optimization strategy for differentiable functions is that of gradient descent. Given some starting position, at each iteration, the gradient of the function in the current position is calculated, and a small step is taken in that direction (if we are maximizing the function). Recall that the gradient indicates the direction of maximal change in the function value. Thus when the step-size is carefully chosen, this strategy is useful for finding a (potentially local) optimum. When the function is non-differentiable, a similar, though considerably less effective idea, is that of taking a small step in the direction of a sub-gradient.

At non-differentiable points, unlike the gradient vector, the sub-gradient is not unique. It is any vector that satisfies a particular condition. Formally, let $\boldsymbol{\delta}$ be the vector of arguments for some convex function $L(\boldsymbol{\delta})$. Then, the sub-gradient of $L(\boldsymbol{\delta})$ at $\boldsymbol{\delta}^*$, denoted by $g_{\boldsymbol{\delta}^*}$, is a vector such that for all $\boldsymbol{\delta}'$, $L(\boldsymbol{\delta}') \leq L(\boldsymbol{\delta}^*) + g_{\boldsymbol{\delta}^*}(\boldsymbol{\delta}' - \boldsymbol{\delta}^*)$.

Sub-gradient optimization does not suffer from the issue of non-optimal convergence,

as opposed to coordinate descent methods. However, the step-size choice for sub-gradient descent is extremely important. There are many different step-size strategies, and we describe some in detail in Appendix C. The main caveat of using these methods lies in the difficulty of knowing what step-size form will work well for the particular problem at hand, and what choices to make for the free parameters of the step-size expression. Another property of sub-gradient descent is that the value of the function to optimize does not monotonically improve across iterations and can fluctuate during the optimization procedure. However, with a properly chosen step-size, sub-gradient descent is guaranteed to converge to the optimum.

Max-Product Linear Programming (MPLP)

MPLP (Globerson & Jaakkola, 2007) is a dual block coordinate descend method. In the context of high-order models, the authors provide a version they denote by GEMPLP. This version performs updates on the block of all dual variables related to a particular factor node. These include all the dual variables associated with each of its connected variable nodes. The updates turn out to be very similar to regular MPBP and we restate them here.

Letting $\mathcal{C} = \{\phi_c\}_{c=1}^{C}$, $\mathcal{S} = \{c \cap \hat{c} | c, \hat{c} \in \mathcal{C}, c \cap \hat{c} \neq \emptyset\}$, and $\mathcal{S}(c) = \{s \in \mathcal{S} | s \subseteq c\}$ The form of the GEMPLP messages is as follows[10]:

$$\mu_{i \to c}(x_i) = \sum_{b \in \mathcal{N}(i) \setminus c} \mu_{b \to i}(x_i), \tag{2.17}$$

$$\mu_{c \to i}(x_i) = \frac{1}{|\mathcal{S}(c)|} \max_{\mathbf{x}_c \setminus x_i} \Big[\phi_c(\mathbf{x}_c) + \sum_{j \in \mathcal{N}(c) \setminus i} \mu_{j \to c}(x_j) \Big] - \frac{(|\mathcal{S}(c)| - 1)}{|\mathcal{S}(c)|} \mu_{i \to c}(x_i). \tag{2.18}$$

As can be clearly observed from the form of the equations, the messages from variable

[10] We give a simplified form of the messages, where we assume the intersection set cardinality between any two factors is at most one, as is the case for AP. The original messages operate on more general intersection sets.

to function are the same as regular MPBP (compare with Eq. 2.8), while the messages from function nodes to variable nodes are modified by down-weighing their original contribution, and subtracting a portion of the previous messages (compare with Eq. 2.9). This is quite similar to damping, but with of the form $-(1-\lambda), \lambda$ instead of $(1-\lambda), \lambda$, and where the value of λ is determined by the number of other potentials that range over variables on which c operates.

The main guarantees provided by the MPLP algorithm are convergence. Furthermore, sufficient conditions are given for the converged solution to be useful in finding the optimal exact MAP assignment. In particular, when MPLP terminates, we find solutions in a similar way to that of regular MPBP. We compute the beliefs at each node based on summing the incoming messages. If the dual optimal was found[11], and there is a unique optimizing assignment for every factor, then that solution is the optimal solution to the MAP inference problem.

We conclude in noting that in its original form, the MPLP dual objective is derived from a linear programming relaxation. However, in (Sontag et al., 2010) it is shown that the same dual can be derived from a Lagrangian relaxation formulation. See also (Sontag et al., 2010) for a full characterization of the close relationship between Linear Programming and and Lagrangian relaxations.

Dual Decomposition (DD)

Primal/Dual decomposition is a general optimization technique that operates by breaking a hard optimization problem into a sum of easy optimization sub-problems. These are solved individually, and their solutions are combined cleverly to give an expression for the function's sub-gradient. A step in the direction of the sub-gradient is then taken, and the process is repeated until convergence. If we operate on the primal problem, the strategy is called primal decomposition. If it is done on the dual, it is aptly named dual

[11] Recall that this is not guaranteed for block coordinate descent methods.

decomposition.

Specifically, for the case of MAP inference problems, this method was described in (Komodakis et al., 2007). The decomposition is based on the associated factor graph, and it is divided into sub-graphs for which finding an optimizing MAP assignment is easy. One such example for decomposition in pairwise models is to break the graph into a collection of trees that cover the original factor graph. Recall that one can use MPBP to find optimal solutions for tree-structured problems. Other decompositions include sub-graphs for which optimal solutions can be obtained using combinatorial optimization methods, such as min-cut max-flow.

In (Komodakis & Paragios, 2009), the DD framework is extended for high-order models. The extension itself is straightforward and the only assumption is that the sub-problems should be easy to solve even if they involve high-order factors. This is typically a mild assumption since we cannot hope to solve the global optimization if we cannot even optimize individual potentials. A very simple decomposition for high-order models is the *star* decomposition, where the graph is simply decomposed into the individual factors, and each factor is optimized individually.

Formally, using the particular formulation of (Sontag et al., 2010), we expand our definition of the objective function to include a set of singleton potentials ϕ_i, one for every variable. This can always be done by initializing the singleton potentials to evaluate to zero for all configurations of the variable. Thus, the set of all potentials is now $\phi(\cdot) = \{\{\phi_c(\cdot)\}_{c=1}^{C}, \{\phi_i(\cdot)\}_{i=1}^{N}\}$, and the primal problem (compare to Eq. 2.16) is given by

$$\text{PRIMAL} \quad \max_{\mathbf{x},\mathbf{x}^c} \phi(\mathbf{x}, \mathbf{x}^c) = \max_{\mathbf{x},\mathbf{x}^c} \Big[\sum_{i=1}^{N} \phi_i(x_i) + \sum_{c=1}^{C} \phi_c(\mathbf{x}_c^c) \Big] \tag{2.19}$$

$$\text{s.t. } x_i^c = x_i \quad \forall c, i \in c. \tag{2.20}$$

In other words, we create copies of the original variables for every potential, and require that all the variables agree with their copy values. Using this definition, the La-

grangian relaxation is a construction of the Lagrangian using auxiliary (dual) variables $\{\delta_{ci}\}, \quad \forall c, \forall i \in c$ that ensure the copy variables agree with their original variables. The equality constraints are then relaxed to obtain the dual,

$$\text{DUAL} \quad \min_{\boldsymbol{\delta}} L(\boldsymbol{\delta}) = \min_{\boldsymbol{\delta}} \sum_{i=1}^{N} \max_{x_i} \Big[\phi_i(x_i) + \sum_{c|i\in c} \delta_{ci}(x_i)\Big] + \sum_{c=1}^{C} \max_{\mathbf{x}_c^c} \Big[\phi_c(\mathbf{x}_c^c) - \sum_{i\in c} \delta_{ci}(x_i^c)\Big]. \tag{2.21}$$

In (Sontag et al., 2010), it is noted that the reparameterization view previously observed for message passing algorithms such as MBPB and MPLP, is also relevant for DD. While the full DD procedure cannot be directly described in terms of message passing, the dual variables can be interpreted as messages that can be subsumed into the potential functions to give a reparameterization. Indeed, in Eq. 2.21 setting $\tilde{\phi}_i(x_i) \equiv \phi_i(x_i) + \sum_{c|i\in c} \delta_{ci}(x_i)$ and $\tilde{\phi}_c(\mathbf{x}_c^c) \equiv \phi_c(\mathbf{x}_c^c) - \sum_{i\in c} \delta_{ci}(x_i^c)$, gives a reparameterization. The difference is that the iterative computation of dual variable values is not obtained by message passing equations but is obtained via the sub-gradient step.

Under the reparameterization view, the high-level description of DD is that of iteratively searching for reparameterizations $\tilde{\phi}$ of the original potentials ϕ. Each reparameterization provides an upper bound on the original problem. The precise form of the reparameterization is specified via the current values of the dual variables.

Recall the original problem maximizes a sum over potentials. The upper bound is exactly the sum of the individual potential maximizations. DD seeks a reparameterization that gives the **minimal** such upper bound. Let us denote this particular reparameterization by ϕ^*. For that reparameterization (or any reparameterization better than the original potentials in terms of a smaller sum of individual potential maximizations) the following holds:

$$\max_{\mathbf{x}} \sum_{c=1}^{C} \phi_c(\mathbf{x}_c) = \max_{\mathbf{x}} \sum_{c=1}^{C} \phi_c^*(\mathbf{x}_c) \leq \sum_{c=1}^{C} \max_{\mathbf{x}_c} \phi_c^*(\mathbf{x}_c) \leq \sum_{c=1}^{C} \max_{\mathbf{x}_c} \phi_c(\mathbf{x}_c) \tag{2.22}$$

Importantly, in the special case where we find a reparameterization for which the maximizing assignments of the individual potentials all agree, we are guaranteed to have found the optimal solution to the dual, and that strong duality holds. In these situations we have provably found the MAP optimizing assignment. In that light, the reparameterization view can be understood as trying to ensure that conflicting assignments for the copies of the same variable are made to agree more at every iteration.

We now provide a more detailed overview of the dual decomposition framework, based on (Sontag et al., 2010; Komodakis et al., 2010; Nedic & Bertsekas, 2001). We use the formulation of (Sontag et al., 2010), where the primal is maximized, and where the formulation includes single node factors. This yields an unconstrained optimization, which is different from the treatment in (Komodakis & Paragios, 2009; Komodakis et al., 2010). There, specific dual variables subsets are constrained to sum to zero, requiring a projection step for the sub-gradients.

Specifically, Sontag's formulation assumes a sub-problem decomposition into individual factors, whose maximizations are assumed to be easy to compute[12]. In the Lagrangian relaxation, we introduce for each variable as many copies as the number of times that it appears in different potentials. The resultant dual variables (one for every assignment of every variable-factor pair) can be seen as providing different reparameterizations for the different potentials.

Recall the dual objective we wish to minimize is given by Eq. 2.21. Given the value of the dual variables at iteration t, $\boldsymbol{\delta}^t = \{\delta^t_{ci}(x_i)\}$, $\forall c, \forall i \in c, \forall x_i$, the optimization procedure is composed of the following steps:

1. Using $\boldsymbol{\delta}^t$, compute $\forall i$ the maximizing assignment x_i^* of its reparameterized singleton potential:

$$x_i^* = \arg\max_{x_i} \phi_i^t(x_i) \equiv \arg\max_{x_i} \big[\phi_i(x_i) + \sum_{c|i\in c} \delta^t_{ci}(x_i)\big] \tag{2.23}$$

[12]if we cannot efficiently optimize the individual factors, we can hardly expect to be able to optimize a more complicated problem where those factors' variables are shared.

2. Using $\boldsymbol{\delta}^t$, compute $\forall c$ the maximizing assignment $\mathbf{x}_c^*$ of its reparameterized potential:

$$\mathbf{x}_c^* = \arg\max_{\mathbf{x}_c} \phi_c^t(\mathbf{x}_c) \equiv \arg\max_{\mathbf{x}_c} \big[\phi_c(\mathbf{x}_c) - \sum_{i \in c} \delta_{ci}^t(x_i)\big] \tag{2.24}$$

This requires solving each sub-problem with respect to the current reparameterization. Just like message passing, this can be done efficiently. Note that the reparameterization is never computed explicitly since we only need the maximizing assignment. Furthermore, explicitly describing the reparameterization requires enumerating all exponential number of variable settings.

3. Calculate the sub-gradients $g_\delta^t = \{g_{ci}^t(x_i)\} \quad \forall c, \forall i \in c, \forall x_i$ of $L(\boldsymbol{\delta})$ at $\boldsymbol{\delta}^t$, given the reparameterized ϕ_i^t, ϕ_c^t potentials. This calculation turns out to only require knowing the maximizing assignments of the reparameterizations given by x_i^* and $\mathbf{x}_c^*$ obtained in steps 1 and 2 above. Let i be a variable in potential c. Let x_i^c denote the copy of variable i in the set $\mathbf{x}_c$. For every i, and every c such that $i \in c$, we evaluate whether x_i^*, the maximizing assignments of i based on its singleton potential agrees with x_i^{c*}, its copy maximizing assignment in potential c. If not, the sub-gradient is modified as shown in Alg. 1.

Algorithm 1 sub-gradient

```
g^t_ci(x_i) = 0   ∀c, i ∈ c, x_i
for c = 1 to C, i ∈ c do
  if (x*_i ≠ x^{c*}_i) then
    g^t_ci(x*_i) = g^t_ci(x*_i) + 1
    g^t_ci(x^{c*}_i) = g^t_ci(x^{c*}_i) − 1
  end if
end for
```

4. Update the dual variables $\delta_{ci}^{t+1}(x_i)$ based on the new sub-gradients $g_{ci}^t(x_i)$:

$$\delta_{ci}^{t+1}(x_i) = \delta_{ci}^t(x_i) - \alpha^t g_{ci}^t(x_i), \tag{2.25}$$

where α^t is the step-size parameter. We discuss and experiment with different

choices for α^t in Appendix C.

5. Iterate to convergence. Convergence can be evaluated by, *e.g.*, monitoring the changes in dual objective value, $L(\boldsymbol{\delta}^{t+1}) - L(\boldsymbol{\delta}^t)$, where

$$L(\boldsymbol{\delta}^t) = \sum_i \max_{x_i} \phi_i^t(x_i) + \sum_c \max_{\mathbf{x}_c} \phi_c^t(\mathbf{x}_c), \tag{2.26}$$

or by observing that all subproblems agree on their maximizing assignments, in which case the procedure has not only optimized the dual, but the primal objective as well.

This concludes the background on convex relaxation methods. In particular, we provided the general formulation for the max-product linear-programming and dual-decomposition procedures.

2.4 Discussion and Thesis Outline

At this point, we are finally ready to step back and put the various parts of the background section in the context of this thesis. We began by describing the NP-hard exemplar-based clustering problem objective function . We stated that the affinity propagation algorithm (AP, (Frey & Dueck, 2007)) is an algorithm for finding approximate solutions for exemplar-based clustering. We restated the update equations, or messages, that must be passed iteratively in order to execute the affinity propagation algorithm.

An astute reader may have wondered how these messages came to be. Using the notation of factor graphs, it is possible to encode the exemplar-based clustering objective function as a joint probability distribution. Finding a maximizing assignment to the exemplar-based clustering problem is therefore equivalent to solving the MAP inference problem on that factor graph. Applying the max-product belief-propagation algorithm on the factor graph, and simplifying the resulting messages yield the affinity propagation

algorithm. The original graph-construction, derivation, and resulting algorithm was developed for a particular factor graph in (Frey & Dueck, 2007). In Chapter 3, we describe an alternative factor graph for the problem (Givoni & Frey, 2009a), and show how to derive the affinity propagation algorithm using the new factor graph.

In this chapter, we have also described several clustering variants that extend the basic clustering problem to include additional constraints and side information. In Chapter 3 we make the case that our alternative factor graph is particularly convenient for representing additional constraints and side information. In subsequent chapters, we develop affinity propagation algorithms for several clustering variants. We derive the capacitated affinity propagation (CAP, Chapter 4, (Givoni & Frey, 2009a)), semi-supervised affinity propagation (SSAP, Chapter 5, (Givoni & Frey, 2009b)), and hierarchical affinity propagation (HAP, Chapter 6, (Givoni et al., 2011)).

As we shall see, the factor graph for AP requires high-order potentials that operate on many variables. The various AP based clustering algorithms we describe utilize additional high-order potentials. As we explained in this chapter, arbitrary high-order factors are generally intractable. As part of developing our new algorithms, we show that the new factors we add allow for efficient message passing computations.

We have also commented on the close resemblance between exemplar-based clustering and facility location (FL). Thus, we can easily transfer many of the resulting extensions of AP we developed to their corresponding FL variants. Often, this leads to new algorithms for known variants of the facility location problem. We provide exact details of the relationship of exemplar-based clustering to FL in Chapter 3, and describe FL variants based on AP extensions in Chapter 4 and Chapter 6.

Finally, in this chapter we presented alternative MAP inference methods that operate by optimizing a dual problem to the MAP inference problem. We described in particular the MPLP and DD methods. In Chapter 7 we develop the specific instances of these algorithms for the exemplar-based clustering problem. This provides us with two new

alternative algorithms for exemplar-based clustering. In Chapter 8 we compare these algorithms to AP on benchmark data sets. We find that despite favorable theoretical guarantees, AP outperforms MPLP and DD on the data sets we examine. We summarize with general conclusions and future directions of research in Chapter 9.

Chapter 3

A Binary Variable Model for Affinity Propagation

In this chapter, we present a detailed derivation of the affinity propagation algorithm that is considerably simpler than the original one presented in (Frey & Dueck, 2007). The derivation is based on a quite different graphical model. The new model allows us to more easily extend and make modifications of the standard AP algorithm. We demonstrate this by modifying the new graphical model to express related problems to that of exemplar-based clustering. For those wishing to investigate or extend the graphical model of the AP algorithm, we suggest using this formulation since it allows for simpler and more intuitive model manipulation.

3.1 Motivation and Background

In its original formulation, (Frey & Dueck, 2007), AP was derived as an instance of the max-product belief propagation algorithm in a loopy factor graph (Kschischang et al., 2001; Pearl, 1988). That factor graph is shown in Fig. 3.1, and it contains a set of N variables, each of cardinality N, and N function nodes. As we shall discuss in detail, the factor graph functions are defined such that a MAP solution for the graphical model

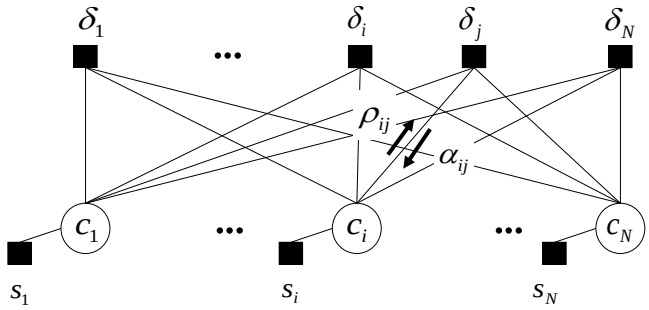


Figure 3.1: N-ary variable model for affinity propagation.

corresponds to a solution for the exemplar-based clustering problem. While naively this model implies that N-ary messages are passed from variables to functions, and from functions to variables, the resulting algorithm can be stated using only scalar messages (Eqs. 2.4 - 2.5).

The simplified message updates were obtained by Frey and Dueck (2007) after cumbersome manipulations of the standard max-product message updates, which reduced the N-ary messages to binary ones. Any modifications to the algorithm, such as introducing useful constraints via additional function nodes, or modifying the functional form of the factors, require re-derivation of the message updates subject to these modifications. Obtaining the simplified messages entails further manipulations of the update equations.

We first present a graphical model for the exemplar-based clustering problem that is based on binary variables. It is an alternative model to the N-ary variable model of Fig. 3.1. We then provide a detailed derivation of AP based on the new model.

In our experience, the new binary variable model allows easy derivations of message updates for extensions and modifications of the standard exemplar-based clustering problem. This will become apparent in the subsequent chapters, where we derive several extensions for AP based on the formulation presented here. Other authors have also chosen to use or extend AP using the binary variable model. Some examples include

(Kang et al., 2010) who analyze Twitts, work by (Shamaiah et al., 2011) on distributed routing in networks where a 2-hop distance representation is created by adding additional constraints, (Plangprasopchok et al., 2011) who developed a method for integrating individual ontological fragments into folksonomies (structured databases created by social webs) by including constrains that discourage disagreeing substructures to be merged, (Tarlow et al., 2008) who employ the binary formulation to add a non-parametric prior on the size of clusters, and (Kazantseva & Szpakowicz, 2011), who add constraint on the ordering of the exemplars in order to cast the exemplar-based clustering problem as that of segmenting text into sequential partitions.

To demonstrate how the model can be easily manipulated, we conclude this chapter by showing two examples of possible model manipulations. In the first example, we show how to represent the facility location problem. In the second example, we modify the functional form of the messages to arrive at a model for bipartite b-matching that is equivalent to the one presented by (Huang & Jebara, 2007), but yields a considerably simpler derivation.

3.2 A Binary Model for Affinity Propagation

We wish to cluster N data points, where we are given all pairwise similarities s_{ij}, $\{i, j\} \subset \{1 \dots N\}$, and all exemplar preferences p_j. As a preprocessing step, we first subsume the preferences into the pairwise similarity inputs by adding p_j to the elements s_{jj}, $\forall j$. This allows us to incorporate the preferences and self-similarities into a single quantity.

Let $\{h_{ij}\}_{j=1}^{N}$ be N binary variables associated with data point i ($i \in \{1, \dots, N\}$), s.t. $h_{ij} = 1$ iff the exemplar for point i is point j. In this notation, $h_{jj} = 1$ indicates that j is an exemplar. All assignments to exemplars and all exemplar choices can be described by the set of N^2 binary variables $\{\{h_{ij}\}_{i=1}^{N}\}_{j=1}^{N}$.

Each data point in affinity propagation clustering is assigned to a single exemplar.

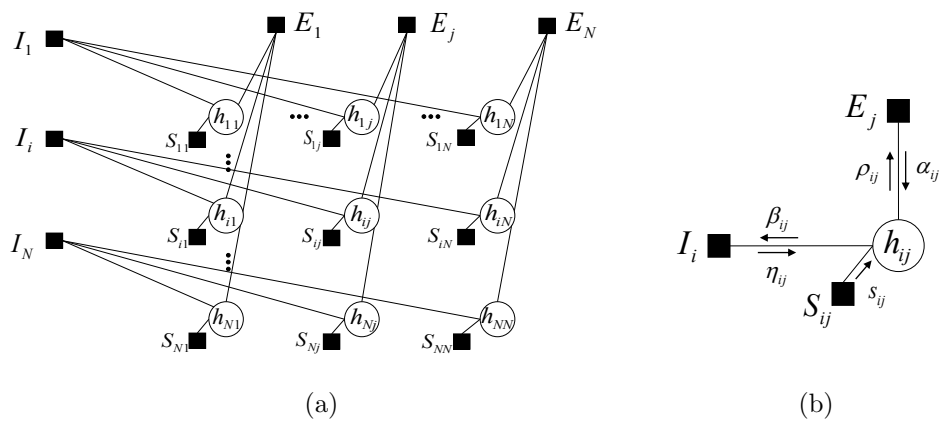


Figure 3.2: A binary variable model and messages for affinity propagation.

Therefore, the first constraint that must be accounted for in the binary variable formulation is that $\sum_{j=1}^{N} h_{ij} = 1$. We refer to this as the *1-of-N* constraint. An additional constraint, which we inherit from the original exemplar-based clustering formulation, is the *exemplar consistency* constraint stating that node i may only choose j as its exemplar if j chose itself as an exemplar.

Fig. 3.2(a) shows a factor graph for affinity propagation that uses the above variables and constraints. The 1-of-N constraints are introduced via the I_i function nodes; in every row i of the grid, exactly one h_{ij} variable, $j \in \{1, \ldots, N\}$, must be set to 1. The E_j function nodes enforce the exemplar consistency constraints; in every column j, the set of $h_{ij}, i \in \{1, \ldots, N\}$ variables set to 1 indicates all the points who have chosen point j as their exemplar. For these points to be able to choose point j as an exemplar, j must also choose itself as an exemplar (*i.e.* h_{jj} must also equal 1). The S_{ij} function nodes incorporate the user-defined input similarities s_{ij} between data points and their potential exemplars, as well as the combined preferences and self-similarities, s_{jj}.

We use the max-sum (log-domain max-product) formulation, where local functions are added to form the global objective function to maximize. The I_i, E_j, and S_{ij} functions

can be naturally defined as follows:

$$I_i(h_{i:}) = \begin{cases} 0 & \text{if } \sum_j h_{ij} = 1, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.1}$$

$$E_j(h_{:j}) = \begin{cases} 0 & \text{if } h_{jj} = \max_i h_{ij}, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.2}$$

$$S_{ij}(h_{ij}) = s_{ij} h_{ij}. \tag{3.3}$$

Where $h_{i:} = h_{i1}, \ldots, h_{iN}$ and $h_{:j} = h_{1j}, \ldots, h_{Nj}$.

The graphical model in Fig. 3.2(a) together with (Eqs. 3.1-3.3) result in the following max-sum objective function:

$$\mathcal{F}(\{h_{ij}\}) = \sum_{i,j} S_{ij}(h_{ij}) + \sum_i I_i(h_{i:}) + \sum_j E_j(h_{:j}). \tag{3.4}$$

There are five message types passed between variable nodes and function nodes when executing the max-sum algorithm on the graph. They are annotated in Fig. 3.2(b). Since each h_{ij} variable is binary, it may appear we need to send two-valued messages between nodes. In practice, for any message we only need to propagate the **difference** between the message values for its two possible settings. The original two-valued message can be recovered from this scalar up to an additive constant, which is not important because adding a constant to all values of a message does not alter the output of the algorithm. We can also think of these scalar messages as pseudo log-likelihood ratios.

When the algorithm terminates, we recover the estimated MAP settings for the hidden variables by adding all their incoming messages together. To make decoding decisions, the set $\mathcal{K} = \{k | \alpha_{kk} + \rho_{kk} > 0\}$ is chosen as the exemplar set. Each non-exemplar point i is assigned to its most similar exemplar, $k = \arg\max_{k \in \mathcal{K}} s_{ik}$.

3.3 Message Updates for the Binary AP Model

We now derive the scalar message updates in the binary variable AP model, where we use the two MPBP update equations (Eq. 2.10 and Eq. 2.11). We denote by $\beta_{ij}(m)$ the message value for setting the hidden variable h_{ij} to m, $m \in \{0, 1\}$, so that $\beta_{ij}(m) = \mu_{h_{ij} \to I_i}(m)$. The scalar message difference $\beta_{ij}(1) - \beta_{ij}(0)$ is denoted by β_{ij}. Similar notation is used for α, ρ and η. In the following derivation, we calculate each message value for the two settings of the binary variable, and then take the difference. Note that the scalar message sent from the S_{ij} function is always the similarity between i and j since $S_{ij}(1) - S_{ij}(0) = s_{ij}$.

We begin by calculating the messages sent from the variable nodes to the function nodes, using Eq. 2.10. The message sent from a h_{ij} variable node to its I_i function node is simply the sum of all incoming messages to the h_{ij} variable node, except for the message **arriving** from the I_i node itself:

For $h_{ij} = 1$:

$$\begin{aligned}\beta_{ij}(1) = \mu_{h_{ij} \to I_i}(1) &= \sum_{b \in \mathcal{N}(h_{ij}) \setminus I_i} \mu_{b \to h_{ij}}(1) \\ &= S_{ij}(1) + \alpha_{ij}(1). \end{aligned} \tag{3.5}$$

For $h_{ij} = 0$:

$$\beta_{ij}(0) = S_{ij}(0) + \alpha_{ij}(0). \tag{3.6}$$

Taking the difference between Eq. 3.5and Eq. 3.6 we obtain:

$$\begin{aligned}\beta_{ij} &= \beta_{ij}(1) - \beta_{ij}(0) \\ &= \big[S_{ij}(1) - S_{ij}(0)\big] + \big[\alpha_{ij}(1) - \alpha_{ij}(0)\big] \\ &= s_{ij} + \alpha_{ij}. \end{aligned} \tag{3.7}$$

Using a similar set of equations, we obtain the ρ_{ij} messages that are sent to the E_j function nodes. These can be expressed as the sum of all incoming messages to the h_{ij} variable node, except for the message arriving from the E_j node:

$$\begin{aligned}
\rho_{ij}(1) &= S_{ij}(1) + \eta_{ij}(1), \\
\rho_{ij}(0) &= S_{ij}(0) + \eta_{ij}(0), \\
\rho_{ij} &= \rho_{ij}(1) - \rho_{ij}(0) = s_{ij} + \eta_{ij}.
\end{aligned} \tag{3.8}$$

We now turn our attention to the messages sent from the function nodes to the variable nodes, which are calculated using Eq. 2.11. When calculating a η_{ij} message sent from function node I_i to the h_{ij} variable node, we fix the value of h_{ij} to 1 or 0, and we find an optimal assignment of the hidden variables h_{ik}, $k \neq j$. Any assignment that violates the I_i function 1-of-N constraint results in that function evaluating to $-\infty$, and such an assignment is trivially non-optimal, and must be avoided.

For $h_{ij} = 1$:

$$\begin{aligned}
\eta_{ij}(1) &= \mu_{I_i \to h_{ij}}(1) \\
&= \max_{h_{ik}, k \neq j} \Big[I_i(h_{i1}, \ldots, h_{ij} = 1, \ldots, h_{iN}) + \sum_{h_{it} \in \mathcal{N}(I_i) \setminus h_{ij}} \mu_{h_{it} \to I_i}(h_{it}) \Big] \\
&= \max_{h_{ik}, k \neq j} \Big[I_i(h_{i1}, \ldots, h_{ij} = 1, \ldots, h_{iN}) + \sum_{t \neq j} \beta_{it}(h_{it}) \Big] \\
&= \sum_{t \neq j} \beta_{it}(0).
\end{aligned} \tag{3.9}$$

The final equality holds since $h_{ij} = 1$ indicates point i has chosen point j as its exemplar, and therefore no other point can be i's exemplar. Consequently, the only assignment of the h_{ik}, $k \neq j$ variables that satisfies the 1-of-N constraint is setting them all to zero. Being the only satisfying assignment, it is also the optimal one, and the I_i function evaluates to its maximal value of zero.

For $h_{ij} = 0$:

$$\begin{aligned}\eta_{ij}(0) &= \max_{h_{ik}, k\neq j}\big[I_i(h_{i1}, \ldots, h_{ij} = 0, \ldots, h_{iN}) + \sum_{t\neq j}\beta_{it}(h_{it})\big] \\ &= \max_{k\neq j}\big[\beta_{ik}(1) + \sum_{t\notin\{k,j\}}\beta_{it}(0)\big]. \end{aligned} \tag{3.10}$$

In the above, when h_{ij} is set to 0 there is more flexibility in choosing an optimal solution. Still, exactly one of the h_{ik}, $k \neq j$ variables must be set to one, resulting in a maximization over $N - 1$ possible solutions.

Taking the difference between Eq. 3.9 and Eq. 3.10 we obtain:

$$\begin{aligned}\eta_{ij} &= \eta_{ij}(1) - \eta_{ij}(0) \\ &= -\max_{k\neq j}\big[\beta_{ik}(1) + \sum_{t\notin\{k,j\}}\beta_{it}(0) - \sum_{t\neq j}\beta_{it}(0)\big] \\ &= -\max_{k\neq j}\big[\beta_{ik}(1) - \beta_{ik}(0)\big] = -\max_{k\neq j}\beta_{ik}. \end{aligned} \tag{3.11}$$

We turn to the α_{ij} messages, where we must distinguish between the cases $i = j$ and $i \neq j$, and we first describe the case $i = j$. Setting $h_{jj} = 1$ (j has chosen itself as an exemplar) is enough to guarantee that any setting of the other variables h_{kj}, $k \neq j$ will be valid (the other points can either choose j as an exemplar, or not) and the maximum can be taken over all the configurations with the exemplar consistency constraint trivially satisfied. On the other hand, when $h_{jj} = 0$ (j is not an exemplar), the only valid configuration satisfying the exemplar consistency constraint is for all the other variables $h_{kj}, k \neq j$ to also equal zero (none of them has chosen j as an exemplar). The message updates for these two setting are,

For $h_{ij} = 1$, $i = j$:

$$\alpha_{jj}(1) = \sum_{k\neq j}\max_{h_{kj}}\rho_{kj}(h_{kj}). \tag{3.12}$$

For $h_{ij} = 0$, $i = j$:

$$\alpha_{jj}(0) = \sum_{k \neq j} \rho_{kj}(0). \tag{3.13}$$

Taking the difference between Eq. 3.12 and Eq. 3.13 we obtain:

$$\alpha_{jj} = \sum_{k \neq j} \max(\rho_{kj}, 0). \tag{3.14}$$

For $h_{ij} = 1$, $i \neq j$:

$$\begin{aligned} \alpha_{ij}(1) &= \max_{h_{kj}, k \neq i} \big[E_j(h_{1j}, \ldots, h_{ij} = 1, \ldots, h_{Nj}) + \sum_{k \neq i} \rho_{kj}(h_{kj}) \big] \\ &= \rho_{jj}(1) + \sum_{k \notin \{i,j\}} \max_{h_{kj}} \rho_{kj}(h_{kj}). \end{aligned} \tag{3.15}$$

The above follows because when $h_{ij} = 1$ (i has chosen j as its exemplar) the only valid configuration for h_{jj} is to be one as well (j has chosen itself as an exemplar) due to the exemplar consistency constraint. Once h_{ij} and h_{jj} are set to 1, the configuration of the other hidden variables h_{kj}, $k \notin \{i, j\}$ is unconstrained (they can either choose j as an exemplar or not) and the maximum is taken over all other configurations.

For $h_{ij} = 0$, $i \neq j$:

$$\alpha_{ij}(0) = \max \big[\rho_{jj}(1) + \sum_{k \notin \{i,j\}} \max_{h_{kj}} \rho_{kj}(h_{kj}) \ , \ \sum_{k \neq i} \rho_{kj}(0) \big]. \tag{3.16}$$

In the above case, there are two distinct types of valid configurations, depending on whether h_{jj} is set to 0 or to 1, and the optimal configuration is therefore the maximization over these two settings. If $h_{jj} = 1$ (j has chosen itself as an exemplar), the configuration of the other hidden variables h_{kj}, $k \notin \{i, j\}$, is unconstrained and the maximum is taken over all other configurations. Alternatively, if $h_{jj} = 0$ (j is not an exemplar), all other h_{kj}, $k \neq i$ variables must also be 0 (no other point may choose j as an exemplar).

Taking the difference between Eq. 3.15 and Eq. 3.16 we obtain:

$$\begin{aligned}\alpha_{ij} &= \min\Big[0, \rho_{jj}(1) + \sum_{k\notin\{i,j\}} \max_{h_{kj}} \rho_{kj}(h_{kj}) - \sum_{k\neq i} \rho_{kj}(0)\Big] \\ &= \min\Big[0, \rho_{jj} + \sum_{k\notin\{i,j\}} \max(0, \rho_{kj})\Big], \end{aligned} \tag{3.17}$$

where we have used the fact that $x - \max(x, y) = \min(0, x - y)$ and $\max(x, y) - y = \max(x - y, 0)$.

To summarize, the message update equations are:

$$\boxed{\begin{aligned}
\beta_{ij} &= s_{ij} + \alpha_{ij} \\
\eta_{ij} &= -\max_{k\neq j} \beta_{ik} \\
\rho_{ij} &= s_{ij} + \eta_{ij} \\
\alpha_{ij} &= \begin{cases} \displaystyle\sum_{k\neq j} \max(0, \rho_{kj}) & i = j \\ \displaystyle\min\Big[0, \rho_{jj} + \sum_{k\notin\{i,j\}} \max(0, \rho_{kj})\Big] & i \neq j \end{cases}
\end{aligned}}$$

Finally, note that we can express ρ directly in terms of α by using simple substitutions:

$$\rho_{ij} = s_{ij} + \eta_{ij} = s_{ij} - \max_{k\neq j} \beta_{ik} = s_{ij} - \max_{k\neq j}(s_{ik} + \alpha_{ik}). \tag{3.18}$$

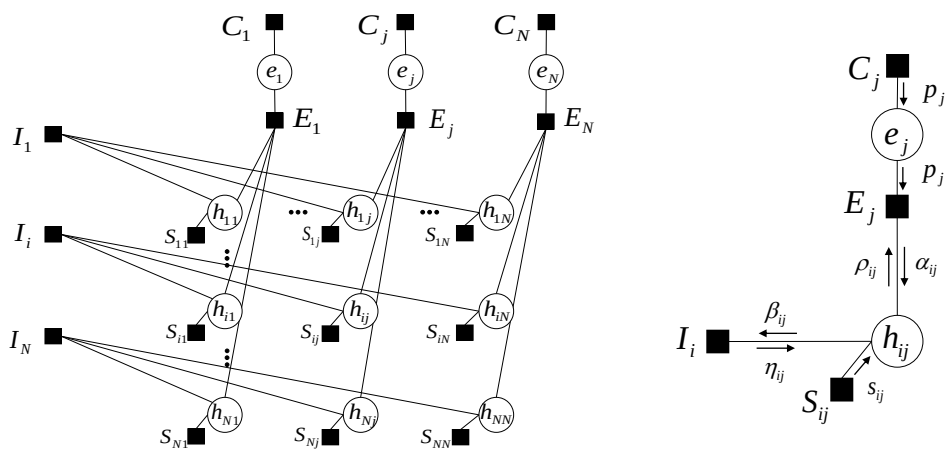
The α_{ij} messages are identical to the AP *availability* messages $a(i, j)$, and the ρ_{ij} messages are identical to the AP *responsibility* messages $r(i, j)$. Thus, we have recovered

the original affinity propagation message updates:

$$\rho_{ij} = s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik}). \tag{3.19}$$

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) & i = j, \\ \min\Big[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj})\Big] & i \neq j. \end{cases} \tag{3.20}$$

3.4 Explicit Exemplar Variable Formulation



(a) (b)

Figure 3.3: A graphical model and messages for explicit exemplar variable affinity propagation.

We also describe an equivalent objective function for exemplar-based clustering that will serve useful for HAP, the hierarchical extension of AP described in Chapter 6. It also allows for more intuitive incorporation of the exemplar preferences.

Here, we do not combine the self-similarity s_{jj} and the preferences, p_j. In addition to the $\{h_{ij}\}$ variables, we also define a set of N binary random variables $e_j, j \in \{1 \dots N\}$ where $e_j = 1$ indicates point j is an exemplar. Clearly, this representation is redundant since $h_{jj} = 1$ already indicates point j is chosen as an exemplar. We will comment on why the redundancy is useful when extending the model for hierarchies.

We can re-write the objective function of Eq. 2.1 as

$$\max_{\{h_{ij}\},\{e_j\}} \mathcal{F}(\{h_{ij}\},\{e_j\}) = \sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}h_{ij} + \sum_{j=1}^{N} p_j e_j \tag{3.21}$$

s.t.

$$\sum_j h_{ij} = 1 \qquad \forall i, \tag{3.22}$$

$$e_j = \max_i h_{ij} \qquad \forall j, \tag{3.23}$$

$$h_{jj} = e_j \qquad \forall j. \tag{3.24}$$

Note that now the constraint formerly specified in Eq. 2.3 is split into two constraints, Eq. 3.23, and Eq. 3.24. It is easy to see that the two objective formulations are equivalent.

The objective function given in Eq. 3.21 corresponds to the MAP estimation in the factor graph shown in Fig. 3.3(a), with the following functions:

$$I_i(h_{i:}) = \begin{cases} 0 & \text{if } \sum_j h_{ij} = 1, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.25}$$

$$E_j(h_{:j}, e_j) = \begin{cases} 0 & \text{if } e_j = h_{jj} = \max_i h_{ij}, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.26}$$

$$S_{ij}(h_{ij}) = s_{ij}h_{ij}. \tag{3.27}$$

$$C_{ij}(e_j) = p_j e_j. \tag{3.28}$$

Using a similar derivation as that described above, the explicit exemplar variable

formulation yields the following update equations:

$$\rho_{ij} = s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik}). \tag{3.29}$$

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) + p_j & i = j, \\ \min\Big[0, \rho_{jj} + p_j + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj})\Big] & i \neq j. \end{cases} \tag{3.30}$$

Note that for decoding in this model, suffice is to look at the belief maximizing assignments for the e_j nodes, and choose as the set of exemplars those with positive beliefs. We then assign each non-exemplar point to its most similar exemplar.

This concludes the derivation of the binary variable model of the exemplar-based clustering algorithm. We have given the derivation for models with and without an explicit exemplar variable. We now proceed to demonstrate how the same, or very similar topology of the model shown in Fig. 3.2(a) can be utilized, alongside a different form of potential functions, to represent two related problems and their corresponding algorithms. The first is the facility location problem, and the second is the bi-partite maximum weighted b-matching problem.

3.5 Facility Location

The uncapacitated facility location (FL) problem is a well-studied problem in operational-research (Şahin Güvenç & Haldun, 2007), and is closely related to exemplar-based clustering. Given a set of N customers and M potential facilities to open, as well as a cost for customer i to use facility j, and a facility setup cost associated with opening a facility j, we wish to find the set of facilities to open, as well as the assignments of each customer to exactly one of the open facilities so as to minimize the overall cost. Similar to the exemplar preferences in AP, the facility setup costs dictate the number of facilities that will be opened, and the problem can be viewed as that of finding an optimal trade-off

between customer-facility costs and facility setup costs.

In order to phrase the problem as that of a maximization, we set the similarities s_{ij}, which we now refer to as customer-facility similarities, to be the negative customer-facility costs. We set the preferences p_j to be the negative facility opening costs. In addition, since the set of points (customers) and exemplars (facilities) are disjoint, we need to assign a set of $N \times M$ hidden variables, where M is now the number of candidate facilities, and h_{ij} indicates whether customer i has chosen to use facility j. In comparison to AP, we note that we no longer need to account for the exemplar consistency constraint since it no longer has a meaning. If a facility is opened, we simply need to ensure its associated preference is added to the objective function to maximize.

Let $\{\{h_{ij}\}_{i=1}^{N}\}_{j=1}^{M}$ be a set of NM binary hidden variables where $h_{ij} = 1$ indicates customer i has chosen facility j as its facility. Since each customer is restricted to use exactly one facility, a constraint that must be accounted for in the binary variable formulation is that $\sum_{j=1}^{N} h_{ij} = 1$. Another constraint we must ensure is that if we open a facility, the facility opening preference is added to our objective to maximize.

The facility location objective function can be written as:

$$\max_{\{h_{ij}\}} \mathcal{F}(\{h_{ij}\}) = \sum_{i=1}^{N}\sum_{j=1}^{M} s_{ij}h_{ij} + \sum_{j=1}^{N} p_j \max_i h_{ij} \tag{3.31}$$

s.t.

$$\sum_j h_{ij} = 1 \qquad \forall i. \tag{3.32}$$

We can once again use the factor graph of Fig. 3.2(a), this time for facility location. Here we simply assume that there are M, rather than N, columns to the factor graph. The 1-of-N constraints are introduced, as before, via the I_i function nodes. In every row i of the grid, exactly one h_{ij} variable, $j \in \{1, \ldots, N\}$, must be set to one. The E_j function nodes now enforce the addition of the facility opening preference. The S_{ij} function nodes

incorporate the customer-facility similarities s_{ij}. Note that facility preferences can no longer be subsumed into s_{jj}. This quantity now simply indicates the similarity between customer j and facility j, which are two different entities.

Given that we intend to use the max-sum (log-domain max-product) formulation, where local functions are added to form the global objective function to maximize, the I_i, E_j, and S_{ij} functions can be naturally defined as follows:

$$I_i(h_{i:}) = \begin{cases} 0 & \text{if } \sum_j h_{ij} = 1, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.33}$$

$$E_j(h_{:j}) = p_j \max_i h_{ij}, \tag{3.34}$$

$$S_{ij}(h_{ij}) = s_{ij} h_{ij}. \tag{3.35}$$

With this formulation for the functions, the objective function to optimize takes the exact same form as the one specified in Eq. 3.4.

Using a similar approach to that taken for deriving the affinity propagation messages for the problem of exemplar-based clustering, we arrive at the max-product messages for facility location. Iterating these messages provides an approximate solution for FL. We denote this algorithm, first described in (Lazic, Givoni, Aarabi & Frey, 2009) by FLAP. The messages are given by:

$$\rho_{ij} = s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik}), \tag{3.36}$$

$$\alpha_{ij} = \min\Big[0, p_j + \sum_{k \neq i} \max(0, \rho_{kj})\Big]. \tag{3.37}$$

We observe that the removal of the consistency constraint reduces the complexity of the availability updates, and there is no longer different treatments for the $i = j$ and $i \neq j$ cases.

It should also be clear that FL can be derived from an explicit exemplar (in this case

facility) variable model, such as the one described in Sect. 3.4. In that case, decoding can be done in the same way described for AP in the explicit exemplar model, *i.e.* by examining beliefs at the e_j nodes, and setting variable with positive beliefs as the facilities. Then every customer is assigned to its nearest facility (nearest being the one with the highest customer-facility similarity). If the standard model without explicit e_j nodes is used, decoding can be done by examining for every column j, whether there is a positive belief for any h_{ij} in the column, and if so, to open facility j. Then, one can assign customers to the nearest open facility. In (Lazic et al., 2010) it was shown that a more involved decoding scheme can guarantee a 3-approximation bound with respect to the optimal solution, if negative customer-facility preferences are assumed to be metric.

3.6 Bipartite Maximum Weighted b-Matching

In the maximum bipartite matching problem (Cormen et al., 2001), also known as the assignment problem, we are given two sets of nodes, A and B, such that $|A| = N, |B| = M$, and $N \leq M$. We are also given the similarities between every pair of points i, j, such that $i \in A, j \in B$. The goal is to match every node in A with exactly one node in B such that no two nodes in A are matched to the same node in B, and the total sum of similarities of the matched nodes is maximized. If we consider the matched nodes in a solution graph nodes, matching i to j corresponds to adding an edge between these two nodes.

An important variant of this classical combinatorial optimization problem is the case of $N = M$, which we assume in the following. Another variant is where the number of matchings per node is an arbitrary $b \leq N$ where still, it is required that the degree of every node in A or B in the matching is b. A related problem to that of bi-partite maximum weighted matching is the general (non bi-partite) maximum weighted matching problem, where we require that every node in a weighted graph is matched to another node, and

the degree of every node in the matching is 1 (or b for the maximum weighted b-matching problem). This family of problems can be solved in polynomial time. Specifically, for the bipartite matching problem, the standard algorithmic approach is using the Hungarian algorithm due to (Kuhn, 1955).

Several interesting results were shown in recent years regarding the applicability of max-product belief propagation (MPBP) to this family of problems. It has been first shown that for bipartite 1-matching, if there is a unique solution, it can be found by running MPBP in a polynomially bound number of iterations (Bayati et al., 2005). This is one of the only cases where MPBP probably converges in a loopy graph. This result was extended for general graphs 1-matching by (Sanghavi et al., 2007), and to bipartite b-matching by (Huang & Jebara, 2007). The contribution of (Huang & Jebara, 2007) is the convergence proof for the message updates. However, the graphical model representation they have chosen was one that required cumbersome manipulations and insightful observations regarding the reducibility of the updates to a simplified form. These can be sidestepped by observing that the model can be easily represented by a factor graph with the exact same topology of the one shown in figure 3.2(a), but with different definitions for the potentials. Using the simple binary grid model results in the same message updates obtained by Huang and Jebara, but via a much simplified derivation. We now show how to easily re-derive the bipartite b-matching message updates.

Given pairwise similarities s_{ij}, and using the factor graph of 3.2(a), we define the factors to be:

$$I_i(h_{i:}) = \begin{cases} 0 & \text{if } \sum_j h_{ij} = b, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.38}$$

$$E_j(h_{:j}) = \begin{cases} 0 & \text{if } \sum_i h_{ij} = b, \\ -\infty & \text{otherwise.} \end{cases} \tag{3.39}$$

$$S_{ij}(h_{ij}) = s_{ij} h_{ij}. \tag{3.40}$$

As can be seen, the only difference is that we changed the 1-of-N constraint into a b-of-N constraint, and that we now have these constraint across rows **and** columns. Therefore, message updates from either row or column functions now take the same form, which is similar, but not identical to the form of the ρ messages of the binary model AP. As we saw in Eq. 3.8, ρ_{ij} messages are simply the summation of the s_{ij} and η_{ij} messages. It is the form of the η_{ij} messages which is different for the b-of-N constraint.

When calculating the b-of-N function outgoing message, η_{ij}, we simply need to ensure a total of b rather than 1 h_{ik}, $k \in \{1 \ldots, N\}$ are activated. Therefore, we need to consider whether h_{ij} itself is set to 1 or to 0.

For $h_{ij} = 1$:

$$\eta_{ij}(1) = \max_{h_{ik}, k \neq j} \left[I_i(h_{i1}, \ldots, h_{ij} = 1, \ldots, h_{iN}) + \sum_{t \neq j} \beta_{it}(h_{it}) \right] \tag{3.41}$$

$$= \sum_{\beta_{it} \in \beta_{\neg j}^{b-1}} \beta_{it}(1) + \sum_{\beta_{it} \notin \{\beta ij, \beta_{\neg j}^{b-1}\}} \beta_{it}(0), \tag{3.42}$$

where $\beta_{\neg j}^{b-1}$ is the set of the $b-1$ largest $\beta_{ik}, k \neq j$ messages.

In other words, having set h_{ij} to 1, we need to set to 1 another b-1 of the h_{ik}, $k \neq j$ variables, and we want to choose the subset of variables that will maximize the expression

given in 3.41. Since for a valid configuration (exactly b of the h_{ij} variables are set to 1), the first term of the expression to maximize is always 0, the value of the expression depends only on the chosen setting of the incoming messages. Therefore we need to look at how much setting each h_{ik} to either 1 or 0 will contribute to the total sum. This is precisely the difference between its 1 and 0 setting, which in turn is precisely the β_{ik} messages. Thus, suffice is to sort all incoming messages but β_{ij} in descending order, set the variables associated with the largest $n-1$ messages (a set we denote by $\beta_{\neg j}^{b-1}$) to 1 and the rest to 0.

For $h_{ij} = 0$:

$$\eta_{ij}(0) = \max_{h_{ik}, k \neq j} \Big[I_i(h_{i1}, \ldots, h_{ij} = 0, \ldots, h_{iN}) + \sum_{t \neq j} \beta_{it}(h_{it}) \Big] \tag{3.43}$$

$$= \sum_{\beta_{it} \in \beta_{\neg j}^{b}} \beta_{it}(1) + \sum_{\beta_{it} \notin \{\beta ij, \beta_{\neg j}^{b}\}} \beta_{it}(0), \tag{3.44}$$

where $\beta_{\neg j}^{b}$ is the set of the b largest $\beta_{ik}, k \neq j$ messages.

In other words, having set h_{ij} to 0, we need to set to 1 another b of the $h_{ik},\ k \neq j$ variables, and we follow the same logic as in the previous explanation.

Taking the difference between (3.42) and (3.44) we obtain:

$$\begin{aligned}
\eta_{ij} &= \eta_{ij}(1) - \eta_{ij}(0) \\
&= \sum_{\beta_{it} \in \beta_{\neg j}^{b-1}} \beta_{it}(1) + \sum_{\beta_{it} \notin \{\beta ij, \beta_{\neg j}^{b-1}\}} \beta_{it}(0) - \Big[\sum_{\beta_{it} \in \beta_{\neg j}^{b}} \beta_{it}(1) + \sum_{\beta_{it} \notin \{\beta ij, \beta_{\neg j}^{b}\}} \beta_{it}(0) \Big] \\
&= -\hat{\beta}_{\neg j}^{b},
\end{aligned} \tag{3.45}$$

where we use $\hat{\beta}_{\neg j}^{b}$ to denote the bth largest $\beta_{ik},\ k \neq j$ message.

In summary, where for the regular 1-of-N factor outgoing message we examined all incoming messages to find the largest one, we now find the bth largest incoming message.

In terms of the final message updates, taking into account both row and column updates, we get:

$$\beta_{ij} = s_{ij} + \alpha_{ij} \tag{3.46}$$

$$\rho_{ij} = s_{ij} - \hat{\beta}^b_{\neg j} \tag{3.47}$$

$$\alpha_{ij} = -\hat{\rho}^b_{\neg i} \tag{3.48}$$

where we use $\hat{\rho}^b_{\neg i}$ to denote the bth largest $\rho_{kj},\ k \neq i$ message. Alternatively, a message passing scheme that outlines more clearly the symmetric nature of the messages (since both row and column factors are the same) would be to iterate between sending the $\rho_{ij} = s_{ij} - \hat{\beta}^b_{\neg j}$, and $\hat{\beta}_{ij} = s_{ij} - \hat{\rho}^b_{\neg i}$ messages.

We have shown how to define the bipartite maximum weighted b-matching problem using a factor graph identical in topology to the one used for affinity propagation. The form of the factors, and thus of the resulting messages are different. Not surprisingly, we recover the updates obtained for the model first described in (Huang & Jebara, 2007).

3.7 Discussion

We have shown how to re-derive the affinity propagation update equations using an alternative graphical model, which results in easier derivation. We showed how the same graph construction can be used to represent other combinatorial optimization problems. It is also easy to add additional problem constraints to this model, as we shall see in the following chapters. Generally speaking, new models can be derived by modifying the form of the factors in Fig. 3.2(a), by adding more variables to the graph, and by adding more factors to the graph.

We observe that the same topology with different functions gives rise to problems that range from NP-hard to ones that have provably polynomial time algorithms. Though

it has been observed that MPBP may give good results for graphs with short loops, it has typically been assumed that the graph density, and the length of the loops are important factors affecting loopy MPBP performance (Murphy et al., 1999). We demonstrated a particular case showing that the **form** of the factors can sometimes be the only contributing factor to the performance and provable guarantees of the algorithm.

Another important observation is that all the models and resulting algorithms presented in this chapter use high-order potentials (HOPs), for which calculating factor-to-variable messages is computationally tractable. Given the good empirical performance of AP, this may encourage the construction of various high-order models as long as the HOPs they are composed of are tractable. More insight on classes of tractable HOPs can be found in (Tarlow, Givoni & Zemel, 2010).

Chapter 4

Capacitated Affinity Propagation

In this chapter we demonstrate how standard affinity propagation can be modified and extended to address a capacity constraint on the clusters. In this variant of the problem, each cluster has an upper limit b on the number of points it may contain. In order to incorporate this constraint into the affinity propagation framework, we augment the exemplar consistency constraint function with additional structure, so that it also maintains the cluster limit constraint. We denote this variant of AP as capacitated AP, or CAP. We compare our approach with a modified EM algorithm for capacitated k-medians and show that CAP consistently outperforms capacitated k-medians.

Capacity constraints on the number of points in a cluster is a well studied problem within the operational research literature. Some variants include the capacitated k-center problem (Bar-ilan et al., 1992), the capacitated p-median problem (Pirkul, 1987), and the capacitated centered clustering problem (Negreiros & Palhano, 2006). The model presented here can be used to solve a special case of the NP-hard capacitated clustering problem discussed in (Mulvey & Beck, 1984), where the capacities are integers, and the demands, or weights of the points, are equal and fixed to one.

A potential application of capacity constraints in the exemplar-based clustering setting is related to forming ad-hoc wireless sensor networks via distributed operations, *c.f.*

(Krivitski et al., 2005). We may wish to introduce capacity constraints stating that each exemplar can only serve a limited number of other nodes. Another application is in managing content distribution networks; these are networks where several servers store copies of data, and each server has some given bandwidth that serves as a capacity. The goal is to assign data requests from clients to appropriate servers (Bateni & Hajiaghayi, 2009).

4.1 Model and Message Updates for Capacitated AP

In Chapter 3 we saw several efficiently computable high-order constraints, such as the 1-of-N, and b-of-N constraints. Clearly, for the problem at hand we will have to enforce an *at most b-of-N* constraint, (where it is assumed that $b < N$). In order to ensure clusters maintain the capacity limit, this constraint will be added to the exemplar consistency constraint.

Thus, the graphical model representing CAP is the same as the one in Fig. 3.2(a). The only modification is in the definition of the E_j function nodes. They now have an additional setting for which the function takes on a $-\infty$ value: the setting where too many points are assigned to the cluster.

$$E_j(h_{:j}) = \begin{cases} 0 & \text{if } h_{jj} = \max_i h_{ij}, \sum_i h_{ij} \leq b, \\ -\infty & \text{otherwise.} \end{cases} \tag{4.1}$$

Deriving the new message updates requires re-deriving Eqs. (3.12-3.17). As before, when calculating the message sent to variable node h_{ij} from function node E_j, we fix the value of h_{ij} to either zero or one, and we find the configuration of the hidden variables $h_{kj}, k \neq i$ that maximizes the function node value plus the sum of all incoming messages to the function node E_j, **except** the message sent from the variable node h_{ij} (*i.e.* we compute $\max_{h_{kj}, k \neq i}\big[E_j(h_{1j}, \ldots, h_{ij} = m, \ldots, h_{Nj}) + \sum_{k \neq i} \rho_{kj}(h_{kj})\big]$).

Therefore, given the new capacity limit b on the number of points that can be assigned to an exemplar, the only difference in the logic used to derive the updates for Eqs. (3.12-3.17) is with regard to the elements over which the max operation is taken in Eq. 3.12 and Eq. 3.15. In Eq. 3.12, we maximize over all $k \neq j$, hence we maximized over $N-1$ messages[1]. When taking into account the capacity limit b, we need to maximize only over the $b-1$ largest messages since the $N-(b-1)$ variables corresponding to the rest of the messages must be set to zero in order to satisfy the capacity constraint. Similarly in Eq. 3.15 we maximized over all $k \notin \{i,j\}$, hence over $N-2$ messages while in the new update we need to maximize only over the $b-2$ largest messages.

Let $\rho_{\neg j}^{b-1}$ be the set of the $b-1$ largest ρ_{kj} messages sent from variable nodes h_{kj} to function node E_j , where $k \neq j$, and let $\rho_{\neg\{i,j\}}^{b-2}$ be the set of $b-2$ largest ρ_{kj} messages, where $k \notin \{i,j\}$. The new update messages are:

$$\alpha_{ij} = \begin{cases} \sum\limits_{\rho_{\neg j}^{b-1}} \max(0, \rho_{kj}) & i = j, \\ \min\Big[0, \rho_{jj} + \sum\limits_{\rho_{\neg\{i,j\}}^{b-2}} \max(0, \rho_{kj})\Big] & i \neq j. \end{cases} \tag{4.2}$$

The ρ_{ij} messages for CAP remain the same as in Eq. 2.4.

In order to calculate the above messages, we need to find the largest $b-1$ messages. A naive approach for doing so is to first sort all messages, and then pick the largest $b-1$ messages. Such an approach results in an $\mathcal{O}(N \log N)$ runtime per calculation of each α_{ij} message. Note that it is always possible to obtain that same run-time for calculating **all** outgoing messages, in the same way described by (Tarlow et al., 2010). A further improvement can be obtained by observing that while one needs to know which are the largest $b-1$ messages, the actual ordering is of no importance. Therefore, it is possible to retain an amortized $\mathcal{O}(N)$ runtime of calculating the α_{ij} messages by using Hoare's

[1]Here, when we say messages we mean the scalar values, which are the differences of the two messages for the binary settings of any given hidden variable.

selection algorithm (quickselect, an optimized selection algorithm based on quicksort (Cormen et al., 2001)).

4.2 Experimental Evaluation

We compare the CAP algorithm to capacitated k-medoids (CKM). This is a variant of k-medoids (Bishop, 2006) that was adapted to greedily account for the cluster size limit in the following way: we iterate between assigning as the exemplar of each cluster the point that has the maximal similarity to all other points in that cluster, and re-assigning all non-exemplar points to exemplars greedily (from largest to smallest similarity) while making sure the capacity constraint for each exemplar is satisfied, until convergence. The measure we use to compare the algorithms is the net similarity: the sum of similarities between all non-exemplar points to their exemplar.

Data set: We generated 100 different sets of $2D$ points sampled from a uniform distribution, where each set size is N=500. In order to make sure we choose a difficult capacity limit b (one that is not trivially satisfied by a standard clustering algorithm) we used the following procedure: for each data set we first ran regular AP with a range of preferences to obtain a range of different k values (recall that AP does not take as input a number k of clusters to output; the number of clusters is controlled by a tuneable preference parameter). Then we ran standard k-medoids for every obtained value of k. For each k value, We compared the total similarity obtained by AP to that of KM and chose the largest cluster size $\hat{b}$ of the method with the better net similarity. We decreased $\hat{b}$ by one to obtain b, a cap size which is not trivially satisfied by either k-medoids or AP.

Experimental Procedure: We ran CAP with the same preferences given to AP and ran CKM with the corresponding k value of each preference. We ran CKM with 1000 different random restarts of which the best run was taken. For CAP, we adopted the convergence criteria used in (Tarlow et al., 2008) with a damping factor of 0.95 and set

the maximal number of iterations to 5000. Once the algorithm terminates, we decode for the final exemplar set $\mathcal{K}$ using the same method of regular AP (Frey & Dueck, 2007). For determining the assignment of every point $i \notin \mathcal{K}$ to its appropriate exemplar, we compute $\arg\max_{k \in \mathcal{K}} \alpha_{ik} + \rho_{ik}$.

We use the solution of CAP as a starting point for the CKM algorithm and if its output outperforms that of CAP, or if the solution of CAP was not valid, we use the CKM solution. The reason for this is that when CAP does not converge, we found that it oscillates in regions of the solution space that have high objective function values. The results obtained by the simple post-processing step we take here serve to ensure a valid solution, which outperforms multiple random restarts of CKM. The post-processing step operates by taking an invalid solution found by CAP, treating the assignments as clustering assignments, and proceeding with the CKM step of determining the exemplar for every cluster, and reassigning points in a manner that is guaranteed to be legal.

Results: We find that in all 730 experiments CAP outperformed CKM. Fig. 4.1 shows the breakdown of experimental results according to the number of clusters k. The average total similarity of CAP was -24.35 while that of the CKM was -27.21.

It is important to note that CAP does converge to a valid solution most of the time (72% of the runs), and whenever it converges to a valid solution, its output is always better than that of CKM. Running CKM as a post-processing step is done mostly to ensure a valid solution for the non-valid runs. However, this post-processing step also slightly improves results for converged valid runs. For those runs, 'pure' CAP provides results that are on average 9.88% better than CKM before post-processing. Running CKM on the output of HAP as a post-processing step improves the overall performance to that of 10.65% over the baseline CKM.

Additional Experimental Details: Our initial data is comprised of 100 different data sets, and for each data set we run the comparison for 10 different values of k based on the results obtained by running regular AP. However, in 27% of the runs (270 out of

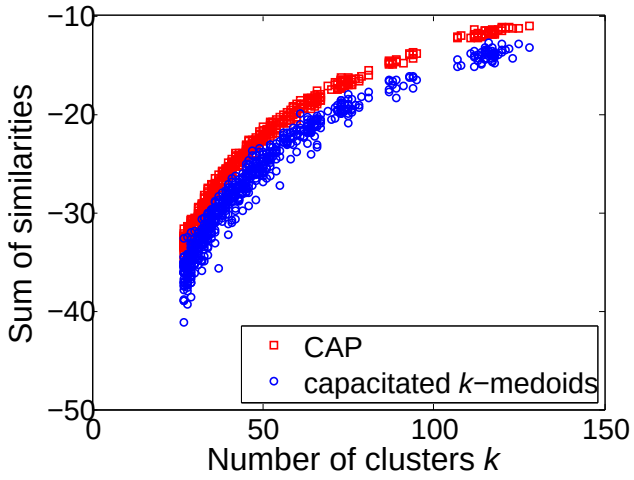


Figure 4.1: A comparison of CAP and capacitated k-medoids as measured in terms of sum of similarities of points to their exemplars.

1000), the solution found by CAP has more than k clusters. We do not include these experiments in the comparison since although they are valid solutions, we wanted to be able to compare results across the different algorithms operating in the same regime.

The average and median run times of CAP are 140.1 and 53.7 seconds per run, respectively while those of CKM were 281.2 and 198.9 seconds per run, respectively (a complete CKM run consists of 1000 reruns with different random starting points.) The code was implemented in Matlab and run on a Linux-based machine with 12 GB of RAM, and two dual-core AMD Opteron 2220 CPUs running at 2.8 GHz.

4.3 Discussion

We note that in order to derive the CAP message updates using the original N-ary variable model (Frey & Dueck, 2007), we would need to propagate the changes through the process of reducing N-ary messages to binary ones[2]. Without good familiarity with the reduction process, it may not even be immediately clear that a given modification to

[2]See Supporting Online Material for (Frey & Dueck, 2007), equations (S2a)-(S8) at http://www.sciencemag.org/cgi/content/full/1136800/DC1.

the model can still result in simple scalar messages. The binary model introduced here greatly simplifies the process of deriving the message updates.

Extending the CAP algorithm to the case where each potential exemplar j is associated with a different limit b_j is trivial and only requires selecting the appropriate largest $b_j - 1$ and $b_j - 2$ messages for each particular j. Furthermore, the general capacitated clustering problem with unrestricted demands and non-integer b_j's can also be represented by the same model with a slightly different E_j function. The solution, however, relies on solving a knapsack problem (Cormen et al., 2001) as an intermediate step. Additionally, the hard limit on b used here can be replaced with a cost that increases as cluster size increases. A particular choice of this cost function that stems from an underlying Dirichlet process prior was used in (Tarlow et al., 2008).

Since AP can be easily transformed to represent the facility location problem (Dueck et al., 2008) (see also Section 2.1.2), the CAP formulation can also be transformed to give an algorithm for solving the capacitated facility location problem (Sridharan, 1995). As is the case for facility location problem, rather than choosing exemplars we choose a subset of facilities to open, where each facility is now constrained by the amount of customer demands it can supply.

The operational research literatures on the topic is concerned with proving asymptotic bounds for the proposed algorithms. Approximation bound results are typically proved under the assumption of a metric space in which the customers and facilities are embedded. The algorithmic approaches are either that of local greedy moves *c.f.* (Zhang et al., 2005), or approaches based on LP relaxations *c.f.* (Levi & Shmoys, 2004). Thus, it is known that the best algorithm to date is a 5-approximation algorithm when all facility costs are equal (Levi & Shmoys, 2004) and a 5.83-approximation algorithm in the more general case (Zhang et al., 2005).

Finally, the potential we define in Eq. 4.1 is essentially two types of hard constraints combined in one function. That of the exemplar consistency (if any point i chooses point

j as an exemplar, point j must choose itself as an exemplar), and the capacity limit. Here we applied logical reasoning to derive the resulting message update. See (Tarlow et al., 2010), for a discussion of how to automatically combine such constraints, as composite potentials.

Chapter 5

Semi-Supervised Affinity Propagation

In this chapter, the AP model is extended to account for semi-supervised clustering. We show how to extend AP to account for *instance-level* constraints: pairs of data points that cannot belong to the same cluster (*cannot-link*), or must belong to the same cluster (*must-link*). We present a semi-supervised AP algorithm (SSAP) that can use instance-level constraints to guide the clustering. SSAP is applied to interactive image segmentation by using it to cluster superpixels while taking into account user instructions regarding which superpixels belong to the same object. We demonstrate SSAP can achieve better performance compared to other semi-supervised methods.

5.1 Motivation and Background

For some background on the general semi-supervised clustering problem and existing approaches, see Section 2.2.3. To briefly reiterate the problem setup, here we are interested in cases where side information is given in the form of instance-level constraints on pairs of data points. Fig. 5.1 demonstrates an input on the left, and the consequent SSAP output on the right. Cannot-link constraints, indicated by a red dashed line between points,

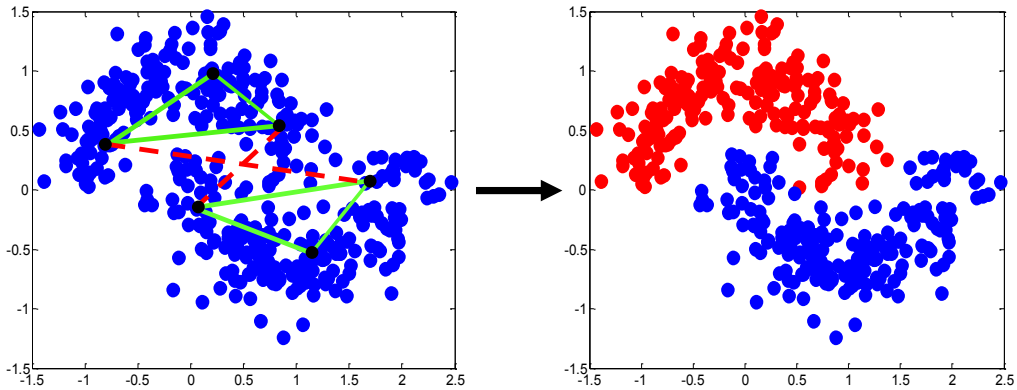


Figure 5.1: Half-moons data with instance-level constraints: green lines indicate must-link constraints, and red lines indicate cannot-link constraints (left). Clustering results obtained using SSAP (right).

indicate the two data points cannot be in the same cluster. Must-link constraints, (in this case, we have two sets of triplets of points that must be in the same cluster, as shown by being connected by green colored lines) indicate the data points must be in the same cluster (Wagstaff & Cardie, 2000).

The solution we propose here is to adapt the underlying factor graph used in affinity propagation such that constraints are explicitly added, but also propagated. We add *meta-points* to the underlying model and appropriate function nodes to govern the allowed set of solutions. The meta-points link together the must-link pairs and prevent cannot-link pairs from being in the same cluster. Similarities of points to meta-points are constructed in a way that allows the model to also account for space-level constraints: points that are very close to points that must be in the same cluster are likely to also be in the same cluster.

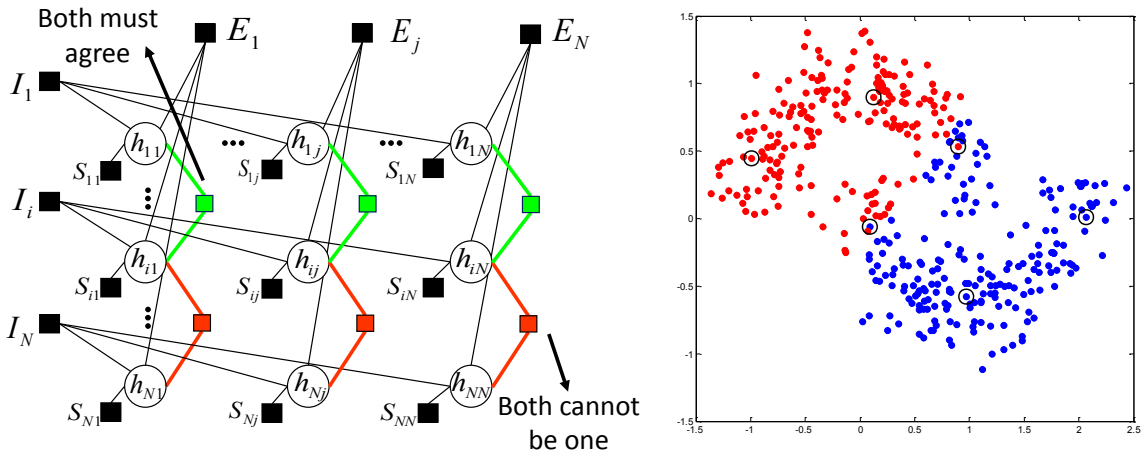
Although, to the best of our knowledge, there are no other AP-based methods that incorporate instance-level constraints, Xiao *et al.* (2007) describe an adaption of AP to include partial labels by contracting all similarly labeled points to a new point and adjusting similarities between unlabeled points to the new contracted points. The set of potential exemplars is restricted to that of the contracted points, and standard AP

is then run on the modified data set. This restriction prevents differently labeled points from being put in the same cluster and forces the number of clusters to be equal to the number of unique labels. Leone *et al.* (2008) also contract all labeled points and adjust the similarities, but they do not restrict only contracted labeled points to be exemplars. However, they do not attempt to cluster the contracted labeled points, which only serve as potential exemplars. Both algorithms do not account for must-link or cannot-link constraints. Recall that the goal of AP is not only to partition the data but to also find the most representative data points, or exemplars. Thus, these methods are less applicable in cases where the actual exemplars are meaningful, as the contracted points do not represent real data points.

5.2 A Model for Semi-Supervised AP

Suppose we obtain instance-level constraints for some input data points, and we wish to endow our model with the ability to use this side information. The first intuitive approach might be to directly connect the hidden variables corresponding to data points that must be in the same clusters via a function that enforces this constraint, and similarly, connecting the hidden variables corresponding to cannot-link data points with an appropriate function node as well, as shown in Fig. 5.2(a). It turns out, however, that running AP on such a graphical model yields solutions that satisfy the constraints but are otherwise meaningless as can be seen in Fig. 5.2(b); the transitive closure of data points with must-link constraints get grouped together in their own clusters, while non-constrained data points that are similar to the must-link constrained ones are not necessarily in the same cluster with the must-link points, unlike what common sense would dictate. The reason for this is direct constraints do not induce propagation of information from constrained points to non-constrained ones.

Note that this clustering result is due to the formulation of the problem, not the



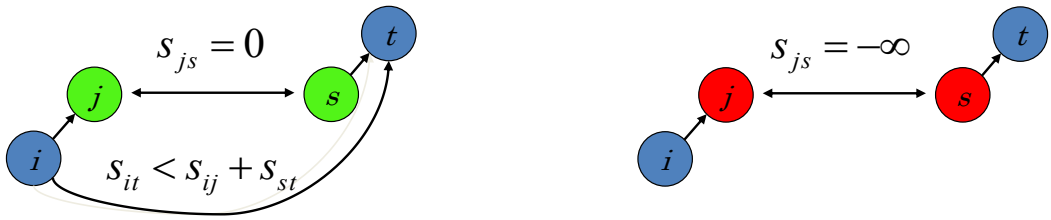
(a) Green function nodes enforce must-link points to take on the same value. Red function nodes enforce cannot-link constraints to never both use the same exemplar.

(b) Clustering results. Pairs of circled red (blue) points have must-link constraints. Every pair of blue-red points have a cannot-link constraint.

Figure 5.2: A graphical model and half-moons data results for directly adding constraints. Though the solution found respects the constraints, the two half-moons are not separated.

optimization method. The solution is a good solution to the problem we try to optimize, as it is very close to the net-similarity obtained using regular unconstrained AP (-422.48 by regular AP for this problem, compared to -424.04 with the constraints), whereas the intuitive 'best' solution of splitting the data into two half moons (as shown on the right hand side of Fig. 5.1), gives considerably worse net-similarity (-458.59).

Another intuitive idea for incorporating constraints is to alter the similarities between data points. For example, by making the similarity between must-link points be maximal, the similarity between cannot-link points minimal and the similarity between any other two points the shortest-path between them. This way, if i is similar to j, s is similar to t, and j and s must be in the same cluster, the similarity between i and t can be increased if $s_{it} < s_{ij} + s_{st}$, as shown in Fig. 5.3(a). Thus, this approach naturally leads to the property that must-link points and their neighbors should be in the same cluster. However, as noted in (Klein et al., 2002), this approach does not enforce the desired property that data points similar to cannot-link points are more likely to be put in different clusters.



(a) Since $s_{it} < s_{ij} + s_{st}$, incorporating this must-link constraint has the result of setting $s_{it} = s_{ij} + s_{st}$.

(b) While we would like to ensure i and t are unlikely to be in the same cluster, modifying the similarities based on cannot-link constraints will not accomplish that.

Figure 5.3: Incorporating instance-level constraints by similarity modifications.

There is no clear and consistent way to modify the similarity of two points that are close to cannot-link constraints to also be far from each other. Moreover, in the particular case of exemplar-based clustering, this approach does not prevent two points with minimal similarity to choose the same exemplar, and end up in the same cluster. This is shown in Fig. 5.3(b).

In fact, it is also noted in (Klein et al., 2002) that enforcing cannot-link constraints is conceptually harder than must-link constraints; even determining if the set of cannot-link constraints has a satisfying assignment is NP-complete. A similar observation is made in (Shental et al., 2003) where the must-link constraints are incorporated by constructing their transitive closure, and constraining them to be in the same cluster, using a relatively simple modification of the EM algorithm update equations. The cannot-link constraints, on the other hand, require the inclusion of a hidden MRF over the data points, resulting in a considerably more involved inference procedure, and an increased degree of approximation.

We propose a strategy for addressing instance-level constraint by augmenting the data points with fictitious meta-points or MTPs. We compute the transitive closure of the must link constraints, and add one MTP for each resulting group as well as to each point in a cannot-link constraint if it is not also part of a must-link group. The MTPs

allow us to explicitly enforce the must-link constraints and cannot-link constraints, as well as to propagate must-link constraints and construct a mechanism for cannot-link constraints to be propagated. As expected from the discussion above, the effectiveness of propagating cannot-link constraints is more limited but it is incorporated and inferred using the same simple formulation as the rest of the model, and is shown to yield good results in practice. We now describe how to augment the model with the MTPs.

Let M be the number of MTPs, and let P_m be the set of data points associated with MTP_m, $m \in \{1 \ldots, M\}$. Here, we assume similarities have been normalized such that the maximal similarity is 0. We define symmetric similarities between MTP_m and the input data points as:

$$s_{i,MTP_m} = \begin{cases} 0 & \text{if } i \in P_m, \\ \max\limits_{j \in P_m} s_{ij} & \text{otherwise.} \end{cases} \tag{5.1}$$

Note that $s_{MTP_m,i} = s_{i,MTP_m}$, and that self-similarities are not required, and are thus undefined for MTPs.

The intuition behind the construction of the meta-points is that data points will now be able to choose either real exemplars or one of the MTPs, if it is more suitable than a real exemplar. The MTPs in turn will have to choose a real exemplar. Since all points in a must-link group will necessarily choose the MTP associated with them, this will also result in a clustering that respects must-link constraints. Other data points are also free to choose MTPs and so points that are similar to a group of must-link points are likely to also choose that group's MTP as an exemplar, leading to the space-level propagation of constraints. Furthermore, if there is a cannot-link constraint between some $i \in P_m$ and $a \in P_n$ we introduce a direct inequality constraint between MTP_m and MTP_n by including function nodes that prevent the MTPs from choosing the same exemplar. This is a compact representation of cannot-link constraints: let $L_m = \{i, j, k\}$ be one set of points that must be in the same cluster, and $L_n = \{a, b, c\}$ be another

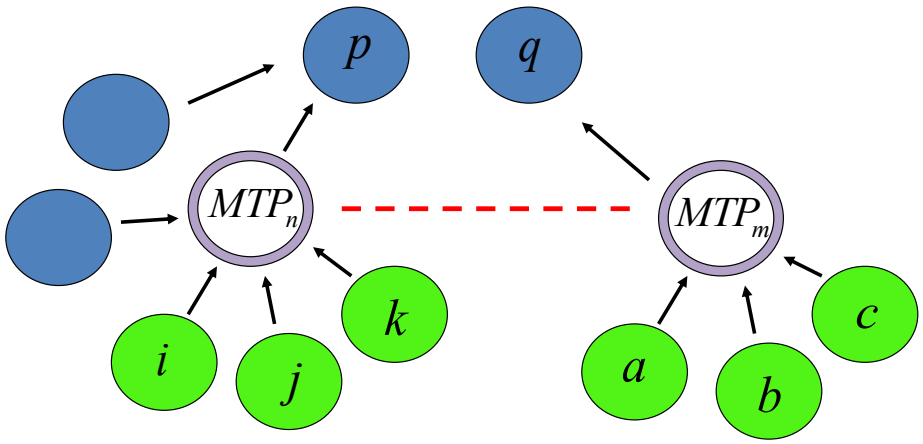


Figure 5.4: A visualization of clustering with meta-points (MTPs). Points i, j, k must all be in the same cluster, and are assigned to MTP_n. The similarity of other points to MTP_n is the closest similarity to either i, j or k. Other points may choose MTP_n, or an exemplar. MTP_n must choose an exemplar. Since there is a cannot-link constraint between i and a (not shown visually), the MTPs of i and a will be constrained to choose different exemplars, as visualized by the dotted red line.

set of points that must be in the same cluster, if we also know that i cannot be with a, then all pairs in the cross product $L_m \times L_n$ also have a cannot-link constraint ,whether it was explicitly specified or not. However, one constraint between the MTPs of each set is all we need in order to enforce all these constraints, as opposed to the representation in (Shental et al., 2003) that involved pairwise connections among all such constrained nodes. Furthermore, any point that will choose some MTP_m as an exemplar, most likely because it is similar to one of the points in the must-link group associated with MTP_m, will be in a different cluster than any point that has a cannot-link constraint with the must-link group associated with MTP_m, if such exists. This can allow space-level propagation of cannot-link constraints. These different scenarios are depicted in Fig. 5.4.

Fig. 5.5 shows the graphical model of SSAP that includes the MTPs (the similarity function nodes have been omitted for clarity) and the cannot-link constraints . The I function nodes, that enforce the property that each point must choose exactly one exemplar, remain the same as in the standard AP model. For the input data points, the

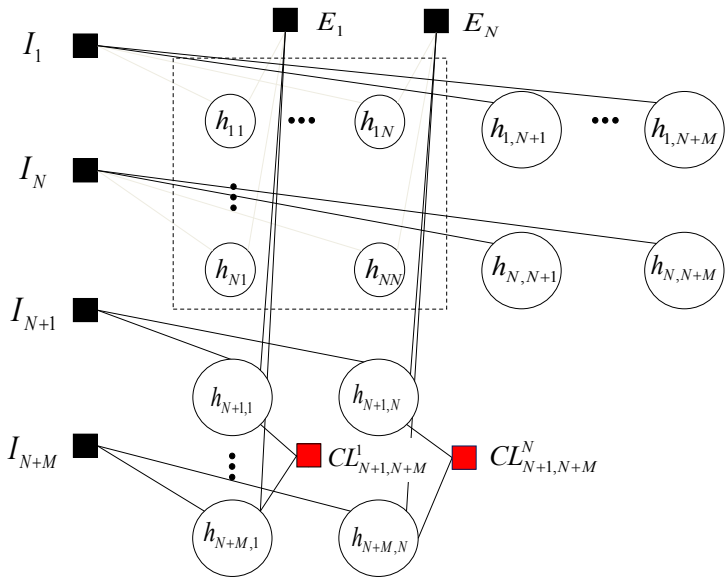


Figure 5.5: A graphical model for semi-supervised affinity propagation (SSAP).

choice of an exemplar is over all data points and all MTPs. Since MTPs should only be allowed to choose exemplars from the set of input data points, the domain of their associated I_i functions is only over the input data points. The E_j function nodes, that enforce the constraint that a point can only choose an exemplar if that exemplar chooses itself as an exemplar, is not required for the MTPs. In fact, by definition they should **not** choose themselves as exemplars but instead pick an exemplar from the set of input data points. The CL function nodes are added to enforce the cannot-link constraints between pairs of MTPs for which such constraints are given, as described above. In Fig. 5.5 we show for clarity only two MTPs and a cannot-link constraint between them, but in the general case there is not necessarily a constraint between every two MTPs and the exact nature of cannot-link connections between MTPs depend on the given constraints.

Formally the cannot-link function nodes are given by

$$CL^k_{N+m,N+n}(h_{N+m,k}, h_{N+n,k}) = \begin{cases} 0 & \text{if } h_{N+m,k} \neq h_{N+n,k}, \\ -\infty & \text{otherwise.} \end{cases} \tag{5.2}$$

5.3 Message Updates for Semi-Supervised AP

The message updates for the new model are similar to the original AP messages. In fact, the α_{ij} messages (Eq. 2.5) remain the same, save for the indexing domain over k in the sum that changes from $k \in \{1, \ldots, N\}$ to $k \in \{1, \ldots, N+M\}$. The ρ_{ij} messages (Eq. 2.4) remain as before for $i < N$ with a similar indexing change over the maximization. In order to express ρ_{mj} for $m > N$, we need to include the messages arriving from and sent to the CL constraint nodes. In order to keep the notation simplified, let us define the following two messages:

$$\kappa^j_{m,mn} = \mu_{h_{N+m,j} \to CL^j_{N+m,N+n}}, \tag{5.3}$$

$$\kappa^j_{mn,m} = \mu_{CL^j_{N+m,N+n} \to h_{N+m,j}}. \tag{5.4}$$

The update rule for the message from a variable node to the CL function node, (Eq. 5.3), is:

$$\kappa^j_{m,mn} = \alpha_{mj} + \rho_{mj} - \kappa^j_{mn,m}. \tag{5.5}$$

And the update rule for the message from the CL function node, (Eq. 5.4), has the form

$$\kappa^j_{mn,m} = -\max(0, \kappa^j_{n,mn}). \tag{5.6}$$

Now we can express ρ_{mj} for $m > N$:

$$\rho_{mj} = s_{mj} + \sum_{n \in \mathcal{CL}_m} \kappa^j_{mn,m} - \max_{k \neq j}(s_{mk} + \sum_{n \in \mathcal{CL}_m} \kappa^k_{mn,m} + \alpha_{mk}), \tag{5.7}$$

where $\mathcal{CL}_m$ denotes the set of all cannot-link constraints associated with MTP_m. Note that if we substitute in the expression $\hat{s}_{mj} = s_{mj} + \sum_{n \in \mathcal{CL}_m} \kappa^j_{mn,m}$, we recover the message update rule of the standard ρ message (Eq. 2.4):

$$\rho_{mj} = \hat{s}_{mj} - \max_{k \neq j}(\hat{s}_{mk} + \alpha_{mk}). \tag{5.8}$$

Therefore, the influence of messages coming from all the cannot-link constraints can be seen as a modification of similarities to account for these constraints.

The message scheduling we have chosen calculates iteratively κ, α, and ρ messages until convergence. Once the algorithm terminates, we assign to all the points which chose an MTP as their exemplar the exemplar chosen by that MTP.

5.4 Experimental Evaluation

The particular application we consider here is user interactive image segmentation. There exist many algorithms for unsupervised image segmentation, but in many cases they fail to provide a segmentation that is close to what a human would consider appropriate. One reason behind their shortcomings can be failing to group together different parts of an object, if the parts are very different under any reasonable similarity measure between the elements on which the segmentation is carried out. These elements can be the image pixels or superpixels - small groupings of pixels comprising coherent image regions, obtained by some method of over-segmentation. For example, a human might consider an image of a person wearing multi-colored and multi-textured clothes as one object, but an automatic segmentation will most likely put the pixels comprising the person in different segments since color, texture, and edge cues will all indicate they should be different objects. Another source of error can be putting together objects that should be separated. For example, if two very similar animals are present in the same image, so that their bodies partially overlap, many segmentation algorithms will group

them into one segment. Although it is yet an open question how to overcome these errors in a completely unsupervised manner, often only a small amount of user intervention is needed in order to correct these types of errors.

We are interested in evaluating the usefulness of semi-supervised AP for the task of user interactive image segmentation. The term interactive image segmentation usually refers to algorithms geared towards fine separation of background from foreground given a user marked contour of the object or an area known to contain the foreground object, *e.g.* (Rother et al., 2004; Boykov & Jolly, 2001). Here, however, we are interested in correctly grouping superpixels into possibly several objects.

In order to obtain quantitative results for a range of experimental settings, we simulate user interaction by randomly selecting a subset of superpixels for which the algorithm is given the instance-level constraints. This is repeated 10 times to obtain results for different subsets of training data for each image. We test the performance across a range of percentage of points for which

Our image data consists of 23 images. We first obtained ~ 200 superpixels for each image (Mori, 2005). Each superpixel was hand-labeled, with each image containing between 2 to 11 distinct labels. We then constructed a similarity measure between superpixels that has a 2D distance component, and an equally weighted color component, similar to (Xiao et al., 2007). The first component is calculated as the negative squared Euclidean distance between the centers of mass of every pair of superpixels, normalized by the sum of all such distances. The color component is the negative squared Euclidean distance between average superpixel color in Cielab space, similarly normalized.

Fig. 5.6 demonstrates an example of a segmentation result for SSAP. The user chooses a subset of the superpixels and provides true labels for them (in this case, for 20% of the superpixels). The ground-truth segmentation and the SSAP segmentation are shown.

We stress again that the input to the SSAP algorithm is not labeled data, but instance-level constraints specifying pairs of points that must or must not be in the same cluster.

The label information is merely used as a surrogate to allow us to easily emulate user interaction, since we need to know the ground-truth labeling in order to automatically generate large sets of instance-level constraints for testing purposes.

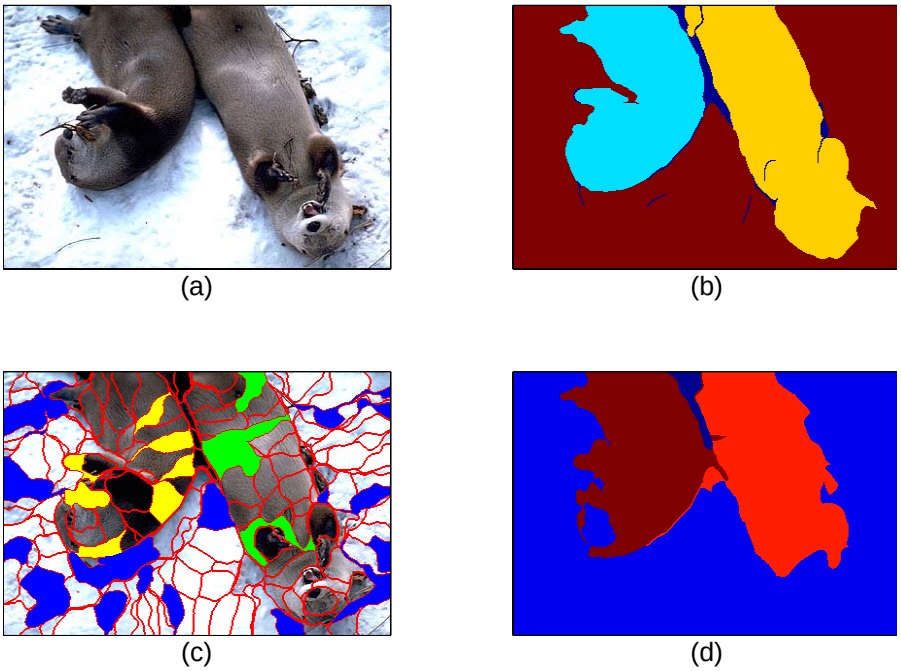


Figure 5.6: An example of image segmentation using SSAP. (a) The original image. (b) Hand-labeling of the super-pixels. (c) The superpixels. Colored superpixels indicate the true label of this superpixel was given to the algorithm. Different colors denote different labels. (d) The resulting SSAP segmentation.

5.4.1 Evaluation Criterion

While we use labeled data for creating the ground-truth collection of must and cannot-link constraints, recall that the output of the clustering procedures is not labels but a clustering. It is easy to transform the ground-truth labels into instance-level constraints but the converse does not hold. Such a transformation requires finding the best permutation of the clusters to classes. If there are more clusters than classes or vice versa, many to one and one to many mappings need to be considered. This additional step and how well it is done is not directly related to the task at hand. The standard way to measure

clustering performance is to look at cluster based methods such as Rand index (Rand, 1971; Wagstaff & Cardie, 2000).

We report the modified Rand index achieved by each method. The Rand index calculates the agreement between two clustering solutions $C, \hat{C}$, where usually one is a clustering algorithm solution ($\hat{C}$), and the other is the true class labels (C). The index is in the range $[0, 1]$ where 1 indicates a perfect agreement between the clusterings. It can also be interpreted as the probability the two clustering solutions agree on whether two randomly drawn points belong to the same cluster or to different clusters. For every pair of points clustered, the points are either in the same cluster in both solutions, not in the same cluster in both solutions, or the pair of points can be in the same cluster according to one solution and not in the same cluster according to the other solution. The first and second cases above represent the agreement events between the clustering solutions. The total sum of these events is normalized by the total number of events (the number of all pairs of points, $\frac{N(N-1)}{2}$) and the result is the Rand index. Following (Wagstaff & Cardie, 2000) we calculate this quantity only for pairs for which no supervised information was given, either directly or by transduction. Furthermore, as observed by (Xing et al., 2003), this measure tends to give inflated scores when there are many clusters, since there are many more pairs of points that are not in the same cluster than there are pairs of points that are in the same cluster, and most algorithms will correctly predict that most pairs are not in the same cluster. This can be remedied by giving the same weight to the points that are in the same cluster (according to $\hat{C}$) and those that are not in the same cluster. The probabilistic interpretation is the chance of two data points to have agreeing clustering solutions, where the data points are drawn uniformly at random from the same cluster (according to $\hat{C}$) with 0.5 probability and from different clusters with 0.5 probability. The following expression is used for calculating the modified Rand index:

$$R(C, \hat{C}) = \frac{1}{2}\left(\frac{\sum_{i>j,\{i,j\}\notin\mathcal{L}} \mathbf{1}_{[c_i=r_j \wedge \hat{c}_i=\hat{c_j}]}}{\sum_{i>j,\{i,j\}\notin\mathcal{L}} \mathbf{1}_{[\hat{c}_i=\hat{p_j}]}} + \frac{\sum_{i>j,\{i,j\}\notin\mathcal{L}} \mathbf{1}_{[c_i\neq p_j \wedge \hat{c}_i\neq\hat{p_j}]}}{\sum_{i>j,\{i,j\}\notin\mathcal{L}} \mathbf{1}_{[\hat{c}_i\neq\hat{p_j}]}}\right). \tag{5.9}$$

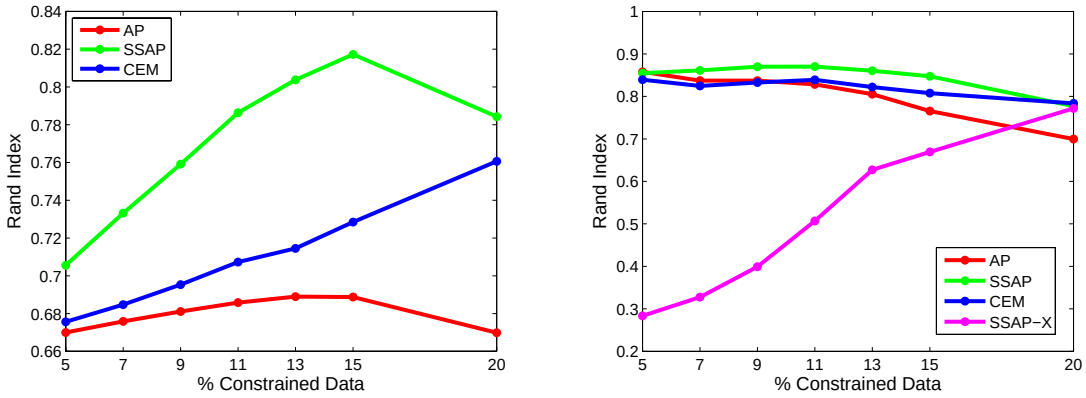
$\mathcal{L}$ is the set of data point pairs for which side information was given to the algorithm. c_i ($\hat{c}_i$) indicates the cluster index of point i according to C ($\hat{C}$). Here we use the notation $\mathbf{1}_{[\text{exp}]}$ as the indicator function, which evaluates to one if the expression in the subscript is true.

5.4.2 Results

We compare our results against Constrained EM (CEM[1] (Shental et al., 2003)). CEM performs expectation maximization (EM) in a Gaussian mixture model, where the must-link constraints are enforced via a modified form of the EM update equations, and the cannot-link constraints are enforced via a Markov random field net imposed over the hidden cluster assignment variables. It was shown to outperform similar methods (Wagstaff et al., 2001) and (Klein et al., 2002) on a variety of tasks. We also compare SSAP to standard AP in order to validate that SSAP can be used to improve the results of standard AP.

Similarly to AP, SSAP does not take as input the number of clusters to find. Rather, it uses preferences as a tuning parameter. In order to perform the comparison we first run SSAP, and then use the discovered number of clusters as our input to CEM. Fig. 5.8 demonstrates an example of segmentation results for some of the images in the data set. Fig. 5.7(a) describes a quantitative analysis of the three algorithms. Each point represents, for a particular amount of instance-level constraints, the average modified Rand index across all 23 images. Each image was subjected to SSAP and CEM with 10 different sets of instance-level constraints. Standard AP does not use the instance-level constraints, and therefore its Rand index is calculated across all point pairs, while that of SSAP and CEM is computed only over data for which no constraints were given, as detailed in Section 5.4.1. Since the number of clusters found by SSAP changes as the amount of constrained information changes, the corresponding AP solutions with the

[1]Code obtained from http://aharon.barhillel.googlepages.com/.



(a) Results for the full data set

(b) Comparison to SSAP-X for a limited subset of the data.

Figure 5.7: Clustering accuracy as measured by the modified Rand Index for image segmentation.

same number of clusters also varies, and therefore we observe different clustering accuracy for AP for different amounts of constraints although AP does not use these constraints.

We note that comparison between SSAP and CEM is not straight-forward. Although both are clustering algorithms that utilize instance-level constraints, and therefore are most similar in terms of their approach, the similarity measure used by SSAP is given as part of the input and is for the user to decide, while CEM assumes a multi-variate normal distribution of each cluster and performs maximum-likelihood fitting of the distribution parameters. However, the similarities given to SSAP are based on normalized negative Euclidean distance, and are therefore comparable to a normal distribution assumption.

Since every set of labeled data points can be transformed to a list of instance-level constraints, a natural question is whether SSAP is comparable to semi-supervised methods that require labeled data. In particular, we are interested in comparing SSAP to SSAP-X (Xiao et al., 2007), which modified the AP algorithm to account for partial labels by contracting all similarly labeled points to a new point, with adjusted similarities, and allowed only the contracted points to serve as exemplars. When making such a comparison, it is important to recall that SSAP may still find clusters that have no labeled information, unlike SSAP-X. The modified Rand index may still be biased for a solution

with more clusters. Therefore, we report results only for the subset of experiments where the number of clusters found by SSAP was no more than 1.5 times the number of clusters found by SSAP-X. This reduces the number of images for which the comparison can be fairly made to (1,2,3,6,10,13,19) images for (5%,7%,9%,11%,13%,15%,20%) of labeled data, respectively. The results are shown in Fig. 5.7(b). As can be seen, at the presence of very little labeled data, the instance-level constraints appear to be more helpful than the actual labels, possibly since the number of distinct input labels is less than the number of objects in the image, as well as the fact that the set of potential exemplars is extremely limited and not likely representative of the data.

The plots may suggest that CEM and SSAP-X arrive at a similar performance level as that of SSAP when 20% of the labels are known, indicating that perhaps they are better suited for a large percentage of labeled data or instance level constraints. We argue that in such scenarios, it is more appropriate to utilize standard supervised classification algorithms trained on the labeled data directly, if labeled are to be used, or to learn a similarity matrix based on instance level constraints, *e.g.* (Xing et al., 2003). The setting we are interested in is that of a small amount of side information.

5.5 Discussion

We presented a semi-supervised version of affinity propagation for instance-level constraints and demonstrated its applicability to user-interactive image segmentation. We note that our model applies equally well to inputs of partial labels, since we can simply convert them to instance-level constraints. Additionally, an extension of this framework to the problem of facility location with instance level constraints on customer would be simple from the derivation point of view, and may result in interesting algorithms for solving cooperative and non-cooperative constraints in FL problems (Goemans & Skutella, 2000).

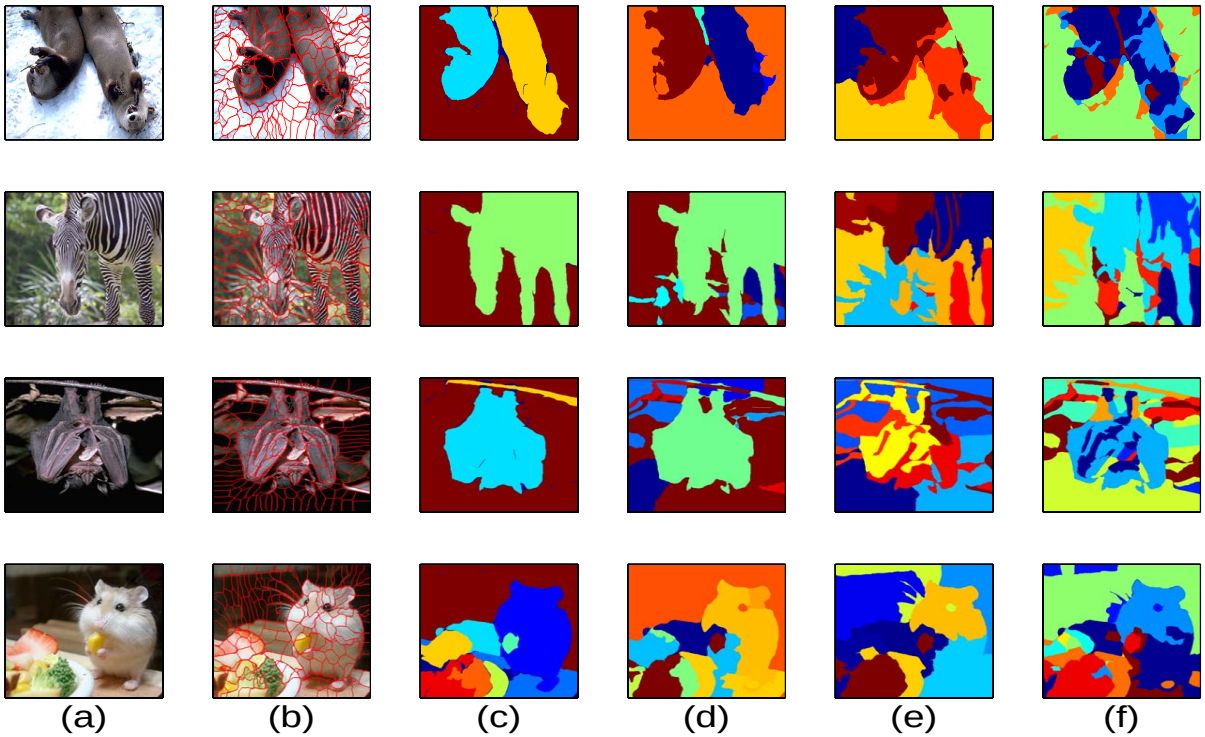


Figure 5.8: Some examples of image segmentation based on instance-level constraints. Columns from left to right: (a) The original image, (b) superpixels computed for the image, (c) hand-labeling of the super-pixels, (d) SSAP results, (e) AP results, (f) Constrained EM results.

This work was motivated by the observation that segmentation results can be improved with very little user interaction. In particular, this interaction can often be restricted to only a subset of the clusters in the image if some clusters are easy to detect in an unsupervised fashion. Therefore, the combination of instance-level constraints with a clustering algorithm capable of finding clusters using side information as well as clusters that are not supervised seems appropriate for this task. The SSAP algorithm we have developed for this task performs well in comparison to similar methods. The ultimate goal would be to build a user-interactive tool that can iteratively refine results. We believe that introducing more structure to the clustering problem, by creating hierarchies for example, can further improve clustering results. In addition, hierarchies can enable a more natural way to define the user-interaction by allowing the user a simple

way of indicating how segments should be combined and split apart. In addition, using the constraints to learn a better similarity matrix for AP is also a natural extension of this work.

Chapter 6

Hierarchical Affinity Propagation

In this chapter, we extend affinity propagation in a principled way to solve the hierarchical clustering problem, which arises in a variety of domains including biology, sensor networks and decision making in operational research. We describe a unified objective function for the problem over the layers of the hierarchy, and describe three different inter-layer connectivity objectives that lead to different constraints on the form of the solution, and can be easily mixed and matched. Two of these variants apply to exemplar-based clustering, and the third accounts for facility location (FL). We derive an inference algorithm that operates by propagating information up and down the hierarchy, and is efficient despite the high-order potentials required for the graphical model formulation.

We demonstrate that our method outperforms a previously described greedy technique that clusters one layer at a time (Xiao et al., 2007). We show that on an artificial data set designed to mimic the HIV-strain mutation dynamics, our method outperforms related methods. For real HIV sequences, where the ground truth is not available, we show our method achieves better results, in terms of the underlying objective function, and show the results correspond meaningfully to geographical locations and strain subtypes. Finally, we report results on using the method for the analysis of mass spectra, showing

it performs favorably compared to state-of-the-art methods[1].

6.1 Motivation and Background

We extend affinity propagation (AP) to find hierarchical structure in data. An appealing property of exemplar-based clustering is that a hierarchical representation can be described as an extension of the standard flat representation by finding an exemplar hierarchy. In this representation, for the lth layer, only points in the exemplar set of the previous layer, $l-1$, may be chosen as exemplars. For example, a hierarchical image segmentation would consist of tiny segments at the lowest layer and successively larger segments at higher layers, corresponding to objects.

An advantage of the exemplar-based approaches, as opposed to that of a statistical representation of cluster centers, is the ability to automatically identify data points that are prototypical. For hierarchical structures, the set of exemplars can be meaningful. In the case of biological data, a hierarchical exemplar-based clustering method can describe a tree structure on proteins that may be related to evolutionary processes. Algorithms that merge data points and clusters together (*c.f.* Hierarchical Agglomerative Clustering (Jardine & Sibson, 1968)), cannot be as easily interpreted to correspond to a meaningful biological structure. In the operational research community, finding good hierarchical structure for setting up service facilities, such as hospitals, is an active area of research (Şahin Güvenç & Haldun, 2007; Kantor & Peleg, 2009). In this problem formulation, exemplars are selected from a set of potential exemplars that is different from the set of data points, making the exemplar-based clustering based formulation a natural one to extend for hierarchical facility location.

A greedy approach first described in (Xiao et al., 2007) is to recursively apply AP to the exemplar set found in the previous layer. As we demonstrate in this work, this greedy

[1]The work in this chapter was done in collaboration with Clement Chung. In particular, the biology experiments were carried out primarily by Clement.

method suffers by making hard decisions in lower layers that turn out to be suboptimal when higher layers are examined, since it does not optimize a global objective function that accounts for all layers simultaneously. Here, we describe a global objective function, which takes across-layer information into account. The hierarchical exemplar-based clustering objective function we propose is specified in terms of a high-order factor graph. We derive an approximate loopy max-sum algorithm, which we refer to as hierarchical affinity propagation (HAP). HAP operates by propagating information up and down the hierarchy, and is efficient despite the high-order potentials required for the graphical model formulation. Noticing that the particular constraints on every layer do not have to be identical across layers, we propose two objective functions describing variants of hierarchical exemplar-based clustering as well as a variant that accounts for hierarchical facility location. We show how the flexible structure of our graphical model allows us to 'mix-and-match' layers of FL and exemplar-based clustering.

We evaluate the algorithm using different types of synthetic data. We show that it achieves better solutions than the greedy alternative of (Xiao et al., 2007), which constructs one layer a at time. We also demonstrate the applicability of HAP to two biological tasks: building hierarchies of HIV sequences, and identifying proteins from mass spectra data.

6.2 Models for HAP and HFL

We show the formulation and derivation of HAP, the hierarchical version of the exemplar-based clustering problem. We wish the set of exemplars to be nested, so that at every layer, only exemplars from the previous layer are clustered. We also show how to create facility location layers that can be part of a hierarchical structure. We then show it is possible to define a different, alternative objective function for HAP that makes different restrictions on the exemplar set and data points. The alternative formulation is that of

clustering all data points at every layer, but requiring that the set of exemplars used is nested and consistent across layers. We term this variant HAP-B.

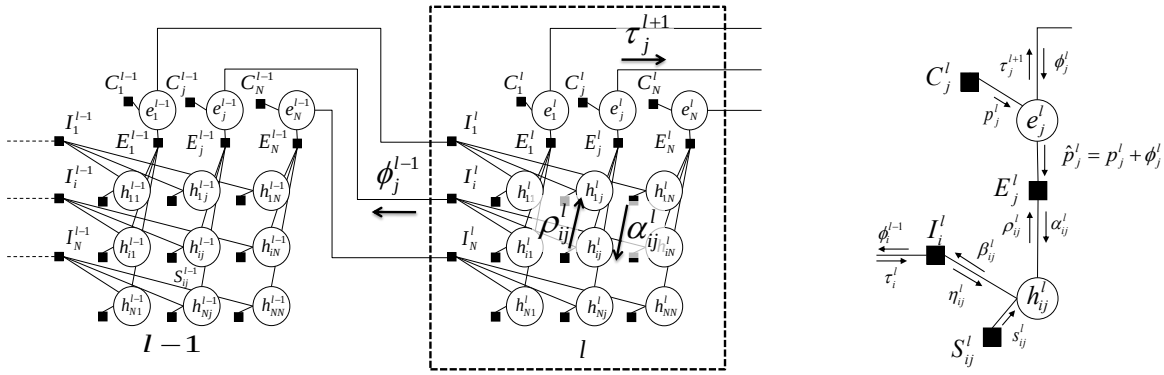
These three related models are formulated via modifications of the layer-specific constraints. This property allows to seamlessly 'mix and match' layers of exemplar-based clustering with facility location in a principled manner, defining novel hierarchical clustering and grouping tasks. The exact formulation for an arbitrary topology is recovered by using an appropriate column-wise function (E type factors) at its corresponding layer. We describe the models and final message update equations in this chapter. A full derivation can be found in Appendix B.

6.2.1 Hierarchical Affinity Propagation (HAP) Model

For the formulation of HAP, we will use the explicit exemplar variable model that was described in Section 3.4. The explicit exemplar variant contains an indicator binary variable, e_j, denoting whether a point is an exemplar or not, or whether a potential facility is opened or not in the case of FL. The reason for using this representation is that while it is an equivalent formulation for AP (with some redundancy, as one can infer the values of the e_j variables by inspecting the h_{jj} variables), it is more useful in representing explicitly inter-layer dependencies. The information that needs to be known across layers is information pertaining to the maintenance of the nesting property - namely, what set of exemplars/facilities were used in the previous layer. Using the e_j variables allows us for a general formulation for different types of layers, if we wish to mix and match HAP and FL. Furthermore, is allows for a simple type of connectivity between layers.

We now show how to generalize the flat AP model to a hierarchical one. We wish to find a set of L consecutive layers of clustering, where the points to be clustered in layer l are constrained to be in the exemplar set of layer $l-1$. This guarantees a hierarchy of exemplars and as we move up in the layers, an exemplar either remains an

exemplar or chooses another exemplar to be its exemplar, relinquishing its own role as a cluster representative. Since a greedy layer by layer solution may incur large penalties for exemplar choices at higher layers due to locally optimal but globally suboptimal decisions made at lower layers, we wish to find a global solution.



(a) Graphical model for HAP. The dotted square depicts the single layer of the standard AP model.

(b) Messages passed in hierarchical AP/FLAP.

Figure 6.1: A graphical model and messages for hierarchical affinity propagation (HAP).

Fig. 6.1(a) describes a graphical model for the hierarchical exemplar-based clustering problem. The main difference compared to the flat representation is manifested in the I_i^l functions:

$$I_i^1(h_{i:}^1) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^1 = 1, \\ -\infty & \text{otherwise.} \end{cases} \tag{6.1}$$

$$I_i^{l>1}(h_{i:}^l, e_i^{l-1}) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases} \tag{6.2}$$

$$E_j^l(h_{:j}^l, e_j^l) = \begin{cases} 0 & \text{if } e_j^l = h_{jj}^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases} \tag{6.3}$$

$$S_{ij}^l(h_{ij}^l) = s_{ij}^l h_{ij}^l, \tag{6.4}$$

$$C_j^l(e_j^l) = p_j^l e_j^l. \tag{6.5}$$

The modified $I_i^{l>1}$ function results in the following behavior: if point i is **not** chosen as an exemplar at layer $l-1$, (*i.e.* if $e_i^{l-1}=0$), then point i will **not** be clustered at layer l. Alternatively, if point i is chosen as an exemplar at layer $l-1$, it must choose an exemplar at layer l. Although we do not explicitly require that only exemplars from previous layers can serve as exemplars at the current layer, this will implicitly be guaranteed through the fact that non-exemplar points in layer l will be constrained to **not** choose an exemplar, together with the exemplar consistency constraint that only allows a point to be an exemplar if it has chosen itself.

The second difference, manifested through the S_{ij}^l and C_j^l functions, is that we allow for layer-specific pairwise similarities (s_{ij}^l) and preferences (p_j^l). The rest of the functions remain as before, with the appropriate layer superscripts. As is the case for standard AP, there are no restrictions on the form of the input pairwise similarities and preferences. Thus, the objective function we wish to maximize for hierarchical AP (HAP) can be stated as:

$$\max_{\{h_{ij}^l\},\{e_j^l\}} \mathcal{F}(\{h_{ij}^l\},\{e_j^l\}) = \tag{6.6}$$
$$= \sum_{i,j,l} S_{ij}^l(h_{ij}^l) + \sum_{j,l} C_j^l(e_j^l) + \sum_i I_i^1(h_{i:}^1) + \sum_{i,l>1} I_i^l(h_{i:}^l, e_i^{l-1}) + \sum_{j,l} E_j^l(h_{:j}^l, e_j^l).$$

6.2.2 Hierarchical Facility Location (HFL)

The generalization of standard FL to a hierarchical version is for the case where we have at the bottom layer a set of customers and potential facilities, and then at every additional layer a set of potential facilities serving the previous layer's facilities. It is described by the model in Fig. 6.1(a), where the column indices in the first layer go up to M, and the row indices in every layer l are equal to the column indices in the previous layer $l-1$ where $l>1$. The FL literature contains several different variants that are called hierarchical FL, and the one we describe here is equivalent to that stated in, *e.g.*,

(Kantor & Peleg, 2009).

The functions are as in Eq. 6.1 - Eq. 6.5 with Eq. 6.3 replaced by

$$E_j^l(h_{:j}^l, e_j^l) = \begin{cases} 0 & \text{if } e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases} \tag{6.7}$$

6.2.3 Alternative Objective Function (HAP-B)

An alternative way to formulate a hierarchical exemplar-based objective function concerns the case where at each layer we cluster **all** the points (not just the exemplars from the previous layer), but the set of exemplars chosen at each layer must be a subset of the exemplars from the previous layer. Once again, this is achieved by modifying the constraints $I_i^{l>1}$. The modified function is given by:

$$I_i^{l>1}(h_{i:}^l, e_i^{l-1}) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = 1, h_{ii}^l \leq e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases} \tag{6.8}$$

A valid solution for either objective function described for HAP results in a nested set of exemplars. In other words, the exemplars at each layer are a subset of the exemplars in the previous layer. From the solution using the first objective function we can also easily derive nested clusters, but we note that in the second, a particular data point may choose a different exemplar t at layer l from the exemplar j it chose in layer $l-1$, even if j is still an active exemplar at layer l.

The two alternative formulations are shown pictorially in Fig. 6.2. We will refer to the former, described in Section 6.2.1 and Fig. 6.2(a) as HAP and the alternative described here and in figure Fig. 6.2(b) as HAP-B.

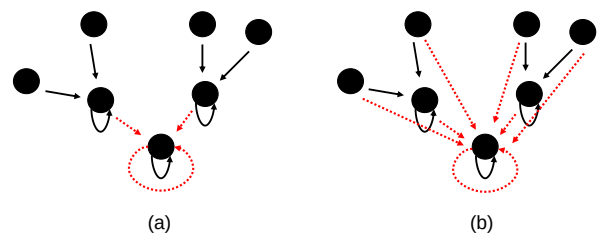


Figure 6.2: Two alternative objectives for hierarchical exemplar-based clustering. (a) HAP: At each layer, the goal is to cluster only exemplars from the previous layer. (b) HAP-B: At each layer, the goal is to cluster all points from the previous layer, while maintaining a nesting of the exemplars.

6.3 Message Updates for Hierarchical AP

Similarly to AP, we find solutions to the problem we described above by inferring approximate MAP values for the hidden variables using the max-sum algorithm. Since the function nodes between layers and within layers are high-order, naively, it may seem that they require the evaluation of an exponential number of settings when calculating the maximization needed to compute the max-sum function-to-variable messages. However, much like the case of single layer AP, the set of valid settings for the maximization is restricted and a careful analysis of these valid settings, and shared computations allow us to compute function-to-variable messages efficiently, resulting in runtime of $\mathcal{O}(LN^2)$ per iteration.

The messages to be passed are shown in Fig. 6.1(a). For an AP layer l using the definition of $I_i^{l>1}$ given in Eq. 6.2, the messages τ_j^{l+1} passed up to the next layer from variable node e_j^l to function node I_j^{l+1} and the message ϕ_j^{l-1} passed down to the previous layer from function node I_j^l to variable node e_j^{l-1} are:

$$\tau_j^{l+1} = p_j^l + \rho_{jj}^l + \sum_{k \neq j} \max(0, \rho_{kj}^l), \tag{6.9}$$

$$\phi_j^{l-1} = \max_k (s_{jk}^l + \alpha_{jk}^l). \tag{6.10}$$

The modified α_{ij}^{l} and ρ_{ij}^{l} messages for an AP layer are

$$\alpha_{ij}^{l<L} = \begin{cases} p_j^l + \phi_j^l + \sum_{k \neq j} \max(0, \rho_{kj}^l) & i = j, \\ \min\Big[0, p_j^l + \phi_j^l + \rho_{jj}^l + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj}^l)\Big] & i \neq j. \end{cases} \tag{6.11}$$

$$\rho_{ij}^{l>1} = s_{ij}^l + \min\Big[\tau_i^l, -\max_{k \neq j}(s_{ik}^l + \alpha_{ik}^l)\Big] \tag{6.12}$$

We note the ρ_{ij}^1 messages passed in the first layer and the α_{ij}^L messages passed in the top-most layer are identical to the standard AP messages described in Eq. 2.4 and Eq. 2.5 and for an AP layer.

6.3.1 Message Updates for HFL and HAP-B

For an FL layer, the messages originating from function node E_i^l (updates τ_j^{l+1} and α_{ij}^l) are modified with respect to Eq. 6.9 and Eq. 6.11 above and we get:

$$\tau_j^{l+1} = p_j^l + \sum_k \max(0, \rho_{kj}^l), \tag{6.13}$$

$$\alpha_{ij}^l = \min\Big[0, p_j^l + \phi_j^l + \sum_{k \neq i} \max(0, \rho_{kj}^l)\Big]. \tag{6.14}$$

If we choose the alternative objective function formulation given in Eq. 6.8, the messages originating from function node $I_i^{l>1}$ (updates ϕ_j^{l-1} and ρ_{ij}^l) are modified and Eq. 6.10 and Eq. 6.12 above are replaced by:

$$\phi_j^{l-1} = \max\Big[0, \alpha_{jj}^l - \max_{k \neq j}(s_{jk}^l + \alpha_{jk}^l)\Big]. \tag{6.15}$$

$$\rho_{ij}^{l>1} = \begin{cases} \min(0, \tau_i^l) - \max_{k \neq i}(s_{ik}^l + \alpha_{ik}^l) & i = j, \\ s_{ij}^l + \min \begin{cases} \max(0, -\tau_i^l) - \alpha_{ii}^l, \\ -\max_{k \notin \{i,j\}}(s_{ik}^l + \alpha_{ik}^l) \end{cases} & i \neq j. \end{cases} \tag{6.16}$$

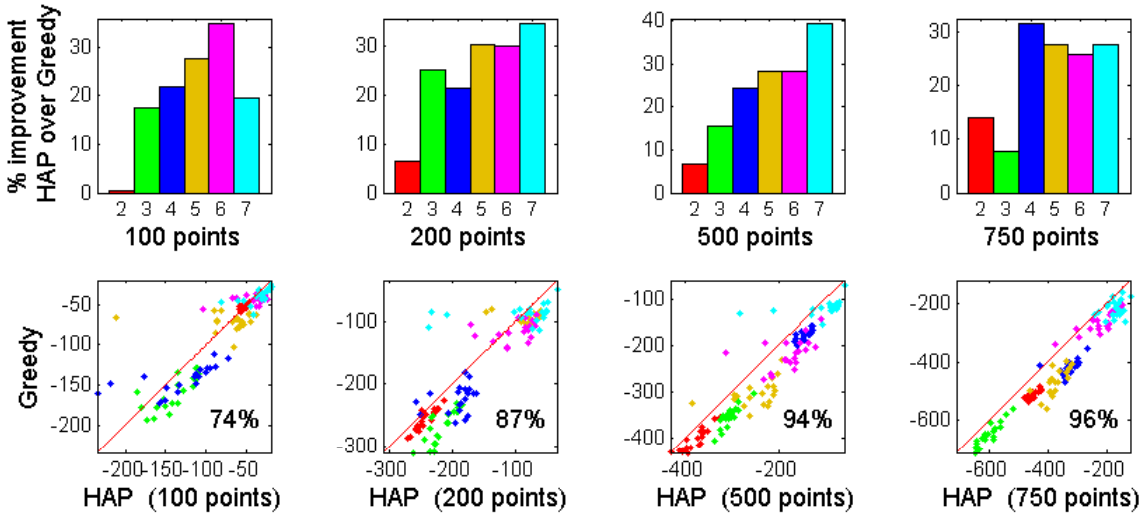


Figure 6.3: 2D synthetic data: comparison of the net similarity (Eq. 6.6) achieved by HAP and its greedy counterpart (Greedy). Top: Median percent improvement of HAP over Greedy for a given number of layers used. Bottom: Scatter plots of the net similarity achieved by HAP *vs.* Greedy. Experiments for which HAP obtains better results than Greedy are **below** the line. Total percent of settings where HAP outperforms Greedy is reported in the inset. Color in scatter-plot indicates the number of layers.

6.4 Experimental Evaluation

6.4.1 Synthetic 2D Data

We begin by constructing experiments on synthetic 2D data to determine whether our proposed method, HAP, obtains better results in terms of the objective function we optimize (Eq. 6.6), when compared to its greedy bottom-up counterpart used in (Xiao et al., 2007) (denoted by 'Greedy'). To examine the applicability of our method under various settings, we look at varying numbers of points and layers. For each such setting we randomly generated increasing exemplar preferences for every layer, and 20 different 2D hierarchical data sets were sampled top-down such that the points of the top-most layer were sampled from a standard normal distribution with a large standard deviation of 3. At consequent layers, sampled points from the previous layer were considered the exemplars. Given those exemplars the layer's points were sampled by randomly picking

one of the exemplars, and sampling a point from a normal distribution centered around the exemplar, and with decreasing standard deviation (halved at every layer going down). The total numbers of points sampled at every layer was such that the overall number of points was roughly 100, 200, 500, or 750 across all layers, and such that sampling from top to bottom, every layer contains twice as many points as in the previous layer.

In Fig. 6.3, we report the median percent improvement over the greedy method for every configuration. We aggregate over the 20 different data sets for every configuration, and show scatter plots of the experiments. Each point represents a result for one of the total of 120 experiments carried out for every choice of the total number of points.

We observe that for a larger number of layers (*e.g.* 5-7 layers compared to 1-4 layers) the improvement of HAP over Greedy is more pronounced, *e.g.*, leading to improvements of up to 40% for seven layers and 500 data points. Further, the bar charts and scatter plots show that as the data set size increases, HAP increasingly outperforms Greedy. Both of these observations are consistent with the expectation that as the hierarchical clustering problem becomes more difficult (more layers and more data points), the gains provided by HAP increase. In a small fraction of cases, HAP does not converge appropriately so that Greedy achieves better solutions. In practice, a good strategy is to run both methods and pick the solution with better net similarity. In the above set of experiments, HAP was on average $\sim$12 times slower than Greedy.

For the 2D data set, we found that HAP tends to oscillate. We therefore use the following procedure to expedite runtime in the synthetic 2D and the real HIV data: we run the algorithm for 500 iterations, decode the bottom-most layer and extract the set of exemplars, and re-run the algorithm on a problem with $L - 1$ layers (where L was the original number of layers), with the set of exemplars as the data-points. This procedure is repeated until all layers are decoded, or until the algorithm converges within the allotted number of iterations. The procedure is not used in the rest of the experiments we describe in the sequel. This can be seen as a quasi-greedy heuristic that does allow

some information to propagate down before making a decision, and indeed proves to be useful in terms of obtaining better results.

In all the experiments we use the following schedule: we start at the bottom-most layer, pass row and column messages within the layer ten times, and then send messages to the next layer. We continue in the same way until we reach the top-most layer, and then begin a downward pass. Again, we do ten internal row and column message passing rounds at every layer. Each full traversal up and down the hierarchy is considered as one iteration.

We also investigated the alternative optimization strategy of solving the LP relaxation corresponding to the HAP objective function. We used the provably convergent MPLP algorithm (Globerson & Jaakkola, 2007) but found the results (objective function value) to be poor in comparison to the other methods. Additionally, the algorithm converges very slowly. For additional details see Section 6.5.

The preference range for the data is chosen by using the heuristic preference range code[2] where depending on the number of layers, this range was divided using a log-linear scale.

6.4.2 Analysis of Synthetic HIV Sequences

The process of HIV evolution can be viewed as one of stochastically generating a hierarchy of exemplars, where each exemplar corresponds to an HIV sequence. Given a population of sequences, a useful task is to infer a tree that describes the evolutionary relationships between the sequences. In Section 6.4.3, we use HAP to hierarchically cluster HIV sequences taken from infected individuals. However, since ground truth labeling is not available for that data, we first explore a synthetically generated data set. Starting with a single root sequence, we simulated the evolution of HIV for three generations to form a four-layer tree of sequences. First, we randomly pick a four-letter root sequence of

[2]Code obtained from http://www.psi.toronto.edu/affinitypropagation/preferenceRange.m.

length 40. To generate children sequences, we first sampled the number of children from a truncated geometric distribution with mean 10. Then, each child was generated by drawing the number of mutations from a geometric distribution with mean 3 and mutating the parent sequence in that many randomly drawn positions. This process was repeated until a four-layer hierarchy was obtained, resulting in a total of 867 sequences. Given the generated sequences, pairwise similarities were obtained using the data-generating mutation process and were set to the logarithm of the truncated geometric distribution evaluated at the observed number of mutations between the sequences.

We compare the performance of HAP, its greedy counterpart (Greedy), a hierarchical version of k-medians clustering (HKMC), and a hierarchical version of k-means clustering (HKMeans) on the task of reconstructing the hierarchy. HKMC works by first identifying exemplars at the lowest layer using k-medians clustering (Charikar et al., 2002), and then clustering the resulting exemplars, and so on. For HKMeans, the algorithm finds cluster means in the lowest layer using k-means clustering, and then we post-process the cluster means to find their nearest points, before proceeding to the next layer.

We applied the different algorithms on a four-layer hierarchy. For HAP and Greedy, we varied the exemplar preferences to obtain solutions differing in the number of exemplars at different layers; a total of 304 settings were used. For HKMC and HKmenas we ran 5624 runs, varying the number of clusters per layer, including the ground-truth setting. For each setting, we report the result of the best of 100 random restarts using different initializations for the median or cluster center set. We evaluated the reconstructed trees using two methods. First, we plotted precision *vs.* recall for various clustering settings (exemplar preferences for HAP and Greedy; random initialization for HKMC and HKMeans), where each sequence-layer combination was labeled as positive if the sequence was generated in that layer in the ground truth and was labeled as negative otherwise. The precision *vs.* recall plots for different methods are shown in Fig. 6.4. These results demonstrate that HAP outperforms the other methods at nearly all recall

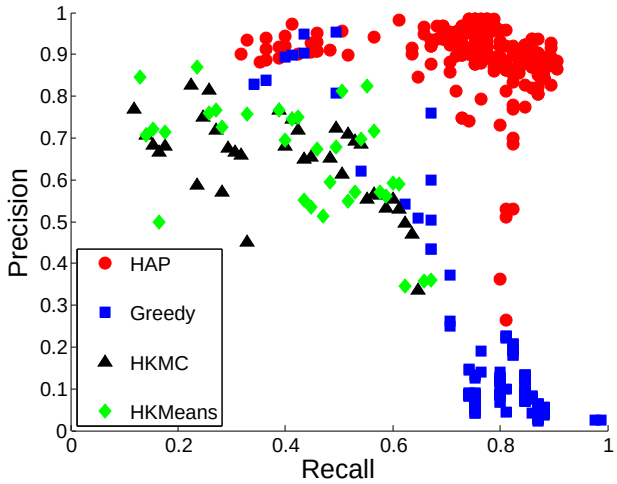


Figure 6.4: Synthetic HIV data: precision-recall for HAP, Greedy, HKMC and HKMeans applied to the problem of identifying ancestral sequences from a set of 867 synthetic HIV sequences. For HKMC and HKMeans, we only plot the best precision obtained for each unique recall value.

levels. Note that this is also true for the case where HKMC and HKMeans are given as input the correct parameters for the number of clusters at each layer.

Also, the recall level at which HAP breaks and performance converges to random performance is significantly higher (~ 0.82) than for the other three methods (Greedy ~ 0.67, HKMC & HKMeans ~ 0.62). Note that applying standard AP layer-by-layer (Greedy) outperforms HKMC. This is to be expected, since for the single-layer case, AP was shown to outperform k-medians clustering (Frey & Dueck, 2007).

For a more in-depth comparison of HAP and Greedy, we applied an additional evaluation method using the modified Rand index (Rand, 1971; Wagstaff & Cardie, 2000), which measures how frequently pairs of data points are correctly classified as siblings or non-siblings by the hierarchical clustering algorithms, given the ground truth clustering. For a full definition and description of the modified Rand index, see Eq. 5.4.1. Fig. 6.5 compares HAP and Greedy for different parameter settings by plotting the Rand index values for each layer in the ground truth labeling. This direct comparison cannot be replicated for HKMC and HKMeans since they operate on different parameters. In the high Rand index regime (> 0.8), HAP almost always performs better than Greedy. It

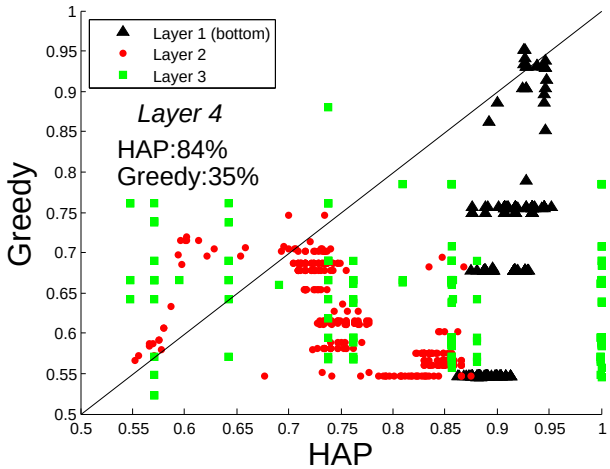


Figure 6.5: Synthetic HIV data: distribution of Rand index for different experiments using HAP and Greedy. A higher Rand index indicates the solution better resembles the ground truth. Experiments for which HAP obtains better results than Greedy are **below** the line. The percentage of solutions that identified the correct single ancestor sequence at the top layer (layer 4) is also reported.

is interesting that by considering all layers, HAP is able to achieve significantly higher Rand indices for the **bottom** layer, as shown by the black triangles. The mean Rand index across all layers and experiments is 0.90 for HAP *vs.* 0.67 for Greedy. For the first, second and third layers, the mean Rand index is 0.91 for HAP *vs.* 0.67 for Greedy, 0.76 for HAP *vs.* 0.62 for Greedy, and 0.86 for HAP *vs.* 0.64 for Greedy. Since the correct solution has a single ancestral exemplar at the top layer, we calculated the fraction of experiments that correctly identifies the single ancestor. HAP was successful for 84% of cases, whereas Greedy was successful in only 35% of cases.

6.4.3 Analysis of Real HIV Sequences

To explore the application of HAP to unlabeled, but real HIV sequences, we obtained 1246 HIV human sequences from the Los-Alamos HIV database[3]. The similarity between pairs of sequences was calculated using DNADIST[4], which is a commonly used method for

[3]Data obtained from http://www.hiv.lanl.gov/content/index.
[4]Data obtained from http://cmgm.stanford.edu/phylip/dnadist.html.

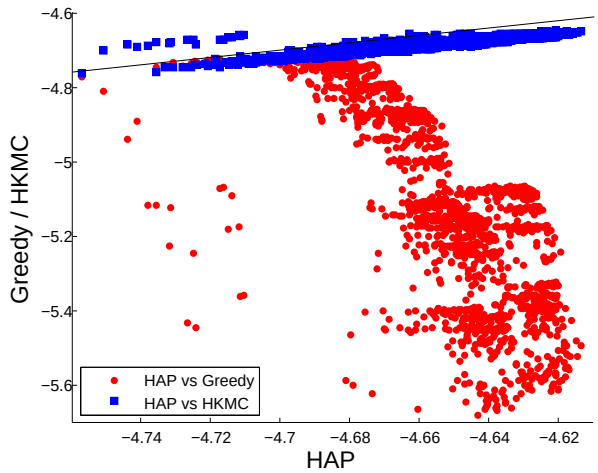


Figure 6.6: HIV data: comparison of the net similarity achieved by HAP *vs.* Greedy, and HAP *vs.* HKMC clustering for a variety of exemplar preference settings. Both axes correspond to net similarity on a log-log scale. In both figures, experiments for which HAP obtains better results than the alternative method are **below** the line.

aligning DNA sequences so as to allow for gaps and non-uniform nucleotide substitution. As before, different clusterings were obtained by varying the exemplar preferences.

We compared the performance of HAP to that of Greedy in terms of the objective function in Eq. 6.6, for a four-layer hierarchy. One exemplar preference was chosen for each layer. The preference value was selected randomly from the interval $[-0.3, -0.04]$, subject to the constraint that the exemplar preference for any layer must be equal to or less than its exemplar preference in lower layers. We obtained a total of 2159 different settings. Results for these experiments are plotted using red discs in Fig. 5 and demonstrate that HAP outperforms Greedy for all settings. While the ground truth is not known for this data, these results indicate that HAP finds better solutions in terms of the sequence-alignment objective function.

The greatest improvement of HAP over Greedy occurs in the regime of high exemplar preferences across all layers. These are the experimental settings that correspond to the bottom-right area of Fig. 6.6. A possible explanation for this is that in general, the objective function value at each layer decreases as the number of data points decreases. By grouping more data points into clusters and finding fewer exemplars at lower layers,

fewer data points need to be clustered at higher layers. Given high preferences, it might be locally optimal at each layer to have as many data points as exemplars as possible. However, globally, it is often better to incur the penalty of clustering more data points together at a lower layer and have fewer data points in the upper layers to cluster. HAP is able to find these globally better solutions. These are clear cases where a greedy method that optimizes each layer of the hierarchy independently is expected to do poorly.

To compare HAP with HKMC without the availability of ground truth labeling, we considered the wide range of 2159 clustering solutions found by HAP as described above. For each HAP solution, we determined the number of clusters in each layer and then ran HKMC using that number of clusters, using 100 random restarts. In this way, we were able to compute the net similarity obtained by HAP and HKMC for solutions that had the same number of clusters in corresponding layers. As shown by the blue squares in Fig. 6.6, HAP outperforms HKMC in almost all experiments.

To investigate the effect of varying the exemplar preferences in a controlled manner, we fixed the preferences in the bottom three layers and varied the preferences in the top layer. As shown in Fig. 6.7, when the top-layer exemplar preferences increases, the number of clusters found in the top layer tends to decrease for both HAP and Greedy. However, for HAP, due to the connectivity between layers, a change in the exemplar preferences in the top layer affects the number of exemplars found in every layer. For Greedy, each layer is optimized independently, so the top-layer preferences has no effect on the number of exemplars found in lower layers. Thus, even when exemplars at the top layer become prohibitively expensive, Greedy cannot correct for that by adjusting the lower layers.

Finally, we explored how HAP could be used to partition HIV sequences in interesting ways and compared those results to information about the country of origin of the HIV sequence and its subtype. By tuning the parameter control knobs at different layers, HAP can be used to identify partitions that are highly stable. Such configurations are least

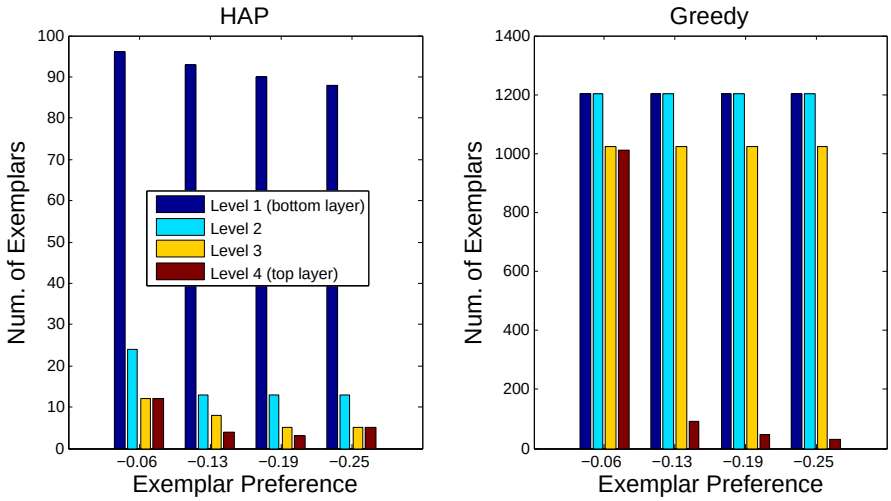
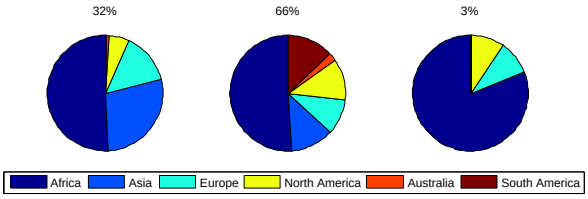


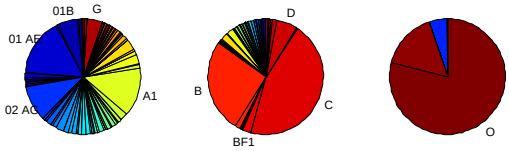
Figure 6.7: HIV data: distribution of cluster sizes for various exemplar preferences (model parameter setting) for HAP (left) and Greedy (right). The exemplar preferences for the (top) layer 4 is adjusted as others are held fixed. As exemplar preference decrease the overall number of clusters decreases. This has a nonlinear effect on all the layers for HAP but only affects the top layer for Greedy.

sensitive to variations in the way that the pairwise similarities are computed and are thus more likely to be biologically relevant. We identified such a setting of parameters, which corresponded to $(93, 13, 3, 1)$ exemplars (from the bottom-most, or first (93), layer to the top-most, or fourth, layer (1)).For each HIV sequence, the Los-Alamos HIV database was used to extract the country of origin (one of 72 countries) and the HIV subtype (one of 113 subtypes), which is a categorization used by HIV researchers that is thought to be related to evolutionary groupings (Robertson et al., 1995).

Fig. 6.8 shows the three third-layer clusters identified by HAP, broken down by their composition in subtype and country of origin. The first two clusters account for 97% of the data, while the third cluster is more specific. It is mostly associated with African strains, and is predominantly of two type 'O' variants, as well as a type 'N' variant. This compares well with focused studies showing that these strains are mostly restricted to west-central African regions (Yamaguchi et al., 2006; Peeters et al., 1997). The first cluster groups many of the 'A' variant subtypes, while the second cluster groups the



(a) Continent composition of each cluster.



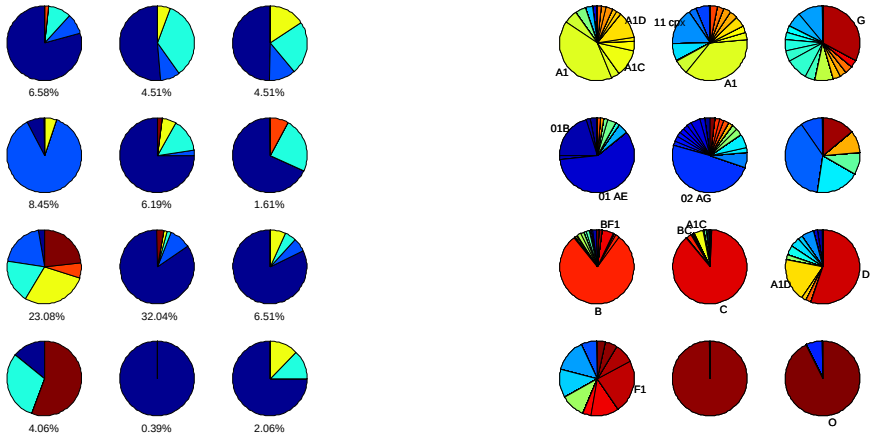
(b) Subtype composition of each cluster.

Figure 6.8: HIV data: third layer. Percentage indicate the number of data points associated with each cluster, of all data points clustered in this layer. Subtype labels shown only if number of data points associated with subtype in cluster is larger than 20.

'B','C', and 'D' subtypes. The color coding indicates that different clusters correspond to mostly non-overlapping sets of subtypes. Also, note that South American strains are found only in the second cluster.

We also discuss in more details the second layer of the clustering tree found by HAP. We find interesting relationships between HAP clusters and geographic location (Fig. 6.9(a)), or subtype (Fig. 6.9(b)). Higher-layer clusters are further broken down here so that they are typically mostly from one region, or of one dominant subtype, or both. Note that the first 6 clusters (counting across rows) are those associated with the third layer's first cluster, the next 4 with the third layer's second cluster, and the rest with the third layer's third cluster. Clusters that have big geographical spread (such as the 7th pie chart in Fig. 6.9(a)) are highly specific for the subtype (corresponding chart in Fig. 6.9(b)). The 'C','D', and 'E' subtypes that were clustered together in the third layer are now separated by the second layer clustering. It is known that the 'C' group is predominantly from Asia and Africa, while 'D' is only seen in Africa (Goudsmit, 1997).

While it is encouraging to observe that known trends about the HIV strains are captured in the HAP hierarchy, it is perhaps more interesting to evaluate hypotheses



(a) Continent composition of each cluster (see Fig. 6.8(a) for color coding). Percentages indicate the number of data points associated with each cluster, of all data points clustered in this layer.

(b) Subtype composition of each cluster. Subtype labels are shown only if the number of data points associated with a subtype in a cluster is larger than 10.

Figure 6.9: HIV data: second layer.

on strain migration by investigating outlier strains that cluster with unexpected groups. Those may indicate how the disease was spread, since the geographical location indicates where the strain was collected, but not necessarily the country of origin of the individual carrier.

6.4.4 Protein Prediction from Mass Spectrometry Data

To further demonstrate the applicability of HAP, we used it to identify the protein composition in a biological sample from mass spectrometry (MS) data. In MS, proteins are broken into peptides and their mass spectra are measured and analyzed computationally to infer the identities of the proteins. However, for complex mixtures, the current state-of-the-art algorithms can only successfully map a small fraction of the mass spectra ($< 20\%$) and so researchers are investigating ways of using prior knowledge about protein groups to improve detection (Keller et al., 2002; Ramakrishnan et al., 2009a; Ramakrishnan et al., 2009b).

Identifying the proteins can be viewed as a two-layer clustering problem, where mass spectra are assigned to proteins in the bottom layer and proteins are assigned to protein clusters corresponding to functional groups in the top layer. Thus, this is a 'mix and match' type of hierarchical setting, where the bottom layer is a facility location problem, and the top layer is an exemplar-based clustering problem. The total score is given by the sum of the mass spectra-protein scores and the protein cluster scores. We include a 'dummy' protein that is used to collect false peptide predictions.

We examined a benchmark data set consisting of a collection of more than $223,000$ scores (log-probabilities) corresponding to mass spectra-protein pairs for $\sim$6700 proteins (Eng et al., 1994; Keller et al., 2002; Lu et al., 2006; Ramakrishnan et al., 2009a). Protein cluster scores were extracted from the yeast gene functional network (Lee et al., 2004), which provides similarity scores between pairs of proteins based on various sources of data. These scores were normalized to the interval $(0, 1]$ and then log-transformed. After normalization, the only free parameters were the difference between the dummy protein preferences and the preferences for the other proteins, which we varied from -16 to -1, and the protein cluster preferences, varied from -5 to 0.

We compared our method to three state-of-the-art algorithms: ProteinProphet (Nesvizhskii et al., 2003), MSpresso (Ramakrishnan et al., 2009b) and MSnet (Ramakrishnan et al., 2009a). We used a curated list of proteins (Ramakrishnan et al., 2009a) as a 'gold standard' and compared the ROC curve of our method to the ROC curves of the other methods. Across a wide range of false positive rates, HAP achieves significantly higher true positive rates than ProteinProphet and MSpresso, and performs similarly to the best method from the computational biology literature, MSNet (see Fig. 6.10). It performs slightly better than a reformulation of the problem as one of a one-layer facility location problem, which we solved by using the message-passing algorithm for the problem (see Section 3.5). While these results do not demonstrate that HAP can significantly outperform state-of-the-art computational biology methods, they do indicate the general utility

of HAP.

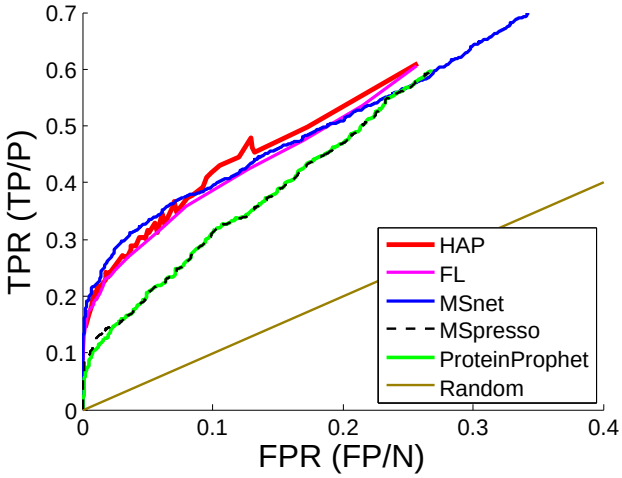


Figure 6.10: True positive rate (TPR) *vs.* false positive rate (FPR) for HAP, three current state-of-the-art protein identification algorithms, and a facility location-based formulation of the problem, on the problem of protein identification using mass spectrometry data derived from yeast wild-type grown in a rich medium.

6.5 Additional Experimental Setup and Results

Unless otherwise specified, the preference range for the data is chosen by using the heuristic preference range[5]. Setting L, the number of layers, is a user specified quantity. However, we note that when the given L exceeds the natural number of layers in the data (when, *e.g.* the data is synthetic, and the ground truth is known), the top layers will all include the same exemplar, or in other words, the nesting saturates, and therefore setting L to be too large will not have severe negative results on the quality of the clustering.

Runtime: Based on the 2D data experiments, we observe that the run-time of HAP scales linearly with the number of layers, and quadratically with the number of points, as can be expected given the asymptotic $\mathcal{O}(LN^2)$ runtime per iteration, and the fact that we cap the total number of iterations to a fixed maximal value. Fig. 6.11 shows a breakdown of the accumulated results (averaged over repeated trials) for different numbers of points

[5]Code obtained from http://www.psi.toronto.edu/affinitypropagation/preferenceRange.m

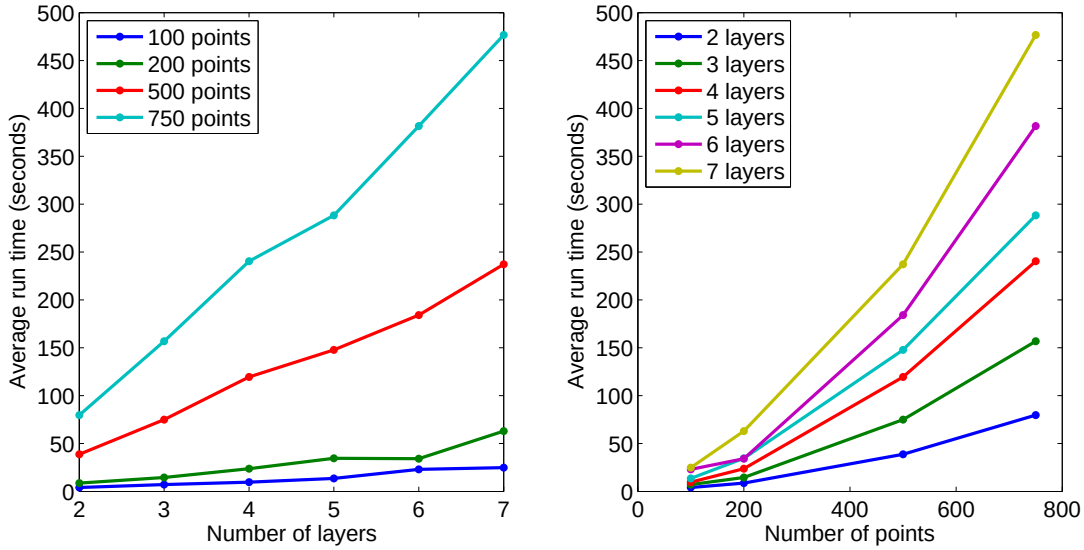


Figure 6.11: 2D synthetic data: runtimes

and layers.

MPLP: We have run experiments for the 2D toy data set using MPLP instead of standard MPBP, as the underlying inference mechanism for the same graphical model. We find that MPLP performs poorly in comparison to HAP and the greedy algorithm. The decoding scheme we use for MPLP is based on a similar one to that used for HAP. It is possible that further investigation into more involved schemes will lead to better results, and this is a direction we are currently pursuing.

6.6 Discussion

We formulated an objective function and described an extended affinity propagation algorithm for hierarchical exemplar-based clustering. This was achieved by generalizing the factor graph and corresponding algorithm that was used to derive the single-layer affinity propagation algorithm. Our method, called hierarchical affinity propagation (HAP), approximately maximizes a natural objective function. By using a message passing method, the algorithm is able to consider all points as potential exemplars at different layers of

the hierarchy and send information up and down the hierarchy to identify exemplars at each layer. In this way, HAP is able to outperform methods that use message passing to build the hierarchy layer by layer (Xiao et al., 2007) and techniques that make use of k-medians clustering with random restarts. In a small number of cases, HAP fails to properly converge, so a sensible strategy is to apply HAP and a greedy method and pick the one with the best objective function. We also extended HAP to the facility location problem and derived a general framework for constructing hierarchies where each layer can be encoded as either a facility location or an exemplar-based clustering problem. In addition, we found that HAP worked well when it was applied to real-world problems in the field of computational biology, including the analysis of HIV sequences and protein identification. An interesting direction of future research is to apply HAP to problems from other domains such as hierarchical shape matching (Gavrila, 2007) and multi-hop wireless sensor networks (Manjeshwar & Agrawal, 2002), and to investigate decoding schemes with theoretical guarantees as was done for the flat facility location variant of AP (Lazic et al., 2010), an analysis that does not hold for the hierarchial case.

Our main goal in the experimental analysis was to show that given a sensible optimization function, HAP outperforms other variants both in terms of the objective function value, and clustering metrics when ground truth is available, across a wide range of parameter settings. Thus this makes HAP an algorithm of choice for such problems. The question of how to come up with a good set of parameters was investigated in part in Fig. 6.7, as well as in our choice of the particular result to analyze in Fig. 6.8 based on stability of clustering properties such as the number of clusters per layer, and the relative number of clusters per layer being reasonably spread, as we modified the exemplar preference parameters. Further investigations into the problem of explicit parameter setting optimization remains an open research direction, which is akin to similar issues that arise both in flat clustering, and hierarchical models such as hierarchical agglomerative clustering, and are known to be hard in the unsupervised clustering setting.

A possible application for the exemplar-based hierarchical model is in multi-hop wireless sensor networks where a network of sensors routes information through a hierarchy of cluster heads to a base station collecting the sensor data (Manjeshwar & Agrawal, 2002). In this application, it is particularly useful to have a method that performs local computations, and can therefore be efficiently executed by the network in a distributed manner with no central computations or requirement for information aggregation across the entire network (Frank & Römer, 2007). Indeed, this local computation property holds for our proposed method due to the underlying message passing mechanism used for the optimization. This is an additional benefit of the model presented here compared to the greedy model that requires centralized computations.

Chapter 7

Exemplar-based Clustering Using Convex Relaxation Methods

In this chapter, we develop two alternative algorithms for exemplar-based clustering that we compare experimentally to affinity propagation in Chapter 8. Recall that AP is based on max-product belief propagation, and thus comes with few theoretical guarantees. Here, we wish to develop convex relaxation based alternatives that are provably convergent, and come with certificates of optimality. We describe one algorithm that is based on the max-product linear programming algorithm (MPLP), and a second algorithm that is based on dual decomposition for sub-gradient descent (DD).

Note that we wish to avoid cumbersome acronyms such as MPLP-AP and DD-AP. Therefore, from here on, we use the abbreviations MPLP and DD to specifically refer to the exemplar-based clustering variants of these algorithms, rather than the general purpose optimization framework. We use the unabbreviated terms when we wish to refer to the general framework. We also discuss strategies for exercising the degrees of freedom of these algorithms, such as decoding and step-size choices.

7.1 Exemplar-based Clustering Using Max-Product Linear Programming

Recall that the general update equations for max-product linear programming, assuming an intersection set of cardinality one[1], have the following form (Globerson & Jaakkola, 2007):

$$\mu_{i\to c}(x_i) = \sum_{b\in\mathcal{N}(i)\setminus c} \mu_{b\to i}(x_i), \tag{7.1}$$

$$\mu_{c\to i}(x_i) = \frac{1}{|\mathcal{S}(c)|}\max_{\mathbf{x}_c\setminus x_i}\big[\phi_c(\mathbf{x}_c) + \sum_{j\in\mathcal{N}(c)\setminus i}\mu_{j\to c}(x_j)\big] - \frac{(|\mathcal{S}(c)|-1)}{|\mathcal{S}(c)|}\mu_{i\to c}(x_i), \tag{7.2}$$

where we use the following definitions: $\mathcal{C} = \{\phi_c\}_{c=1}^{C}$, $\mathcal{S} = \{c\cap\hat{c}|c,\hat{c}\in\mathcal{C}, c\cap\hat{c}\neq\emptyset\}$, and $\mathcal{S}(c) = \{s\in\mathcal{S}|s\subseteq c\}$. Put in words, the size of the set $|\mathcal{S}(c)|$ denotes the total number of variable subsets of potential c that are also in the arguments of some other $\hat{c}$.

7.1.1 MPLP - Max-Product Linear Programming for Exemplar-based Clustering

In the case of the binary variable grid representation (see Fig. 3.2), for messages of the form $\mu_{E_j\to h_{ij}}$ and $\mu_{I_i\to h_{ij}}$, the value of $|\mathcal{S}(c)|$ conveniently turns out to be N, the number of points to cluster. To see this, note that the intersection between any two potentials is of one of the following forms. For any two potentials of the same type, such as S_{ij}, S_{kl}, or E_j, E_i, or for a potential of the form S_{ij} intersected with $E_{l\neq j}$ or $I_{k\neq i}$, the intersection is the empty set, and is not in $\mathcal{S}$. In the case of any two E_j, I_i potentials, the intersection set contains exactly one variable, namely h_{ij}. Additionally, the intersection of any S_{ij}

[1]that is, the maximal number of variables that are arguments of two different potentials is at most one.

with either E_j or I_i is again, a subset containing exactly the h_{ij} variable. Thus, over all, the set of all non-empty intersection sets, $\mathcal{S}$, is precisely the collection of all single variable sets:

$$\mathcal{S} = \Big\{\{h_{11}\}, \ldots, \{h_{ij}\}, \ldots, \{h_{NN}\}\Big\} \tag{7.3}$$

Following the definition of $\mathcal{S}(c)$, recall a potential E_j is defined over the variable set $c = \{h_{1j}, \ldots, h_{Nj}\}$. Since all the single variables exist as subsets of $\mathcal{S}$, the cardinality of $\mathcal{S}(E_j)$ is N, (and likewise for $\mathcal{S}(I_i)$). For the singleton potentials the cardinality of $\mathcal{S}(S_{ij})$ is one. It is easy to verify that messages of the form $\mu_{S_{ij}\to h_{ij}}$ are the same as standard MPBP factor-to-variable messages, since the second term in Eq. 7.2 becomes zero.

We continue to develop MPLP, where we recall the shorthand notation introduced earlier for messages: $\alpha_{ij} = \mu_{E_j\to h_{ij}}(h_{ij})$, $\eta_{ij} = \mu_{I_i\to h_{ij}}(h_{ij})$, $s_{ij} = \mu_{S_{ij}\to h_{ij}}(h_{ij})$, $\beta_{ij} = s_{ij} + \rho_{ij}$, and $\rho_{ij} = s_{ij} + \eta_{ij}$.

The α_{ij} messages can be obtained directly from substitution into Eq. 7.2, and are of the form

$$\alpha_{ij} = \begin{cases} \frac{1}{N}\sum_{k\neq j} \max(\rho_{kj}, 0) - \dfrac{N-1}{N}\rho_{jj} & i = j, \\ \frac{1}{N}\min\Big[0, \rho_{jj} + \sum_{k\notin\{i,j\}} \max(\rho_{kj}, 0)\Big] - \dfrac{N-1}{N}\rho_{ij} & i \neq j. \end{cases} \tag{7.4}$$

Compared to the original AP α_{ij} messages (Eq. 2.5), we only take $\frac{1}{N}$ of the standard E_j factor-to-variable message (similar to fairly strong damping that depends on the number of variables) and subtract most of the incoming variable-to-factor message from that quantity.

For calculating the ρ_{ij} message, since $\rho_{ij} = s_{ij} + \eta_{ij}$, we must first obtain the expression

for η_{ij}:

$$\eta_{ij} = -\frac{1}{N}\max_{k\neq j}\big(s_{ik}+\alpha_{ik}\big) - \frac{N-1}{N}\beta_{ij}. \tag{7.5}$$

Substituting in $\beta_{ij} = s_{ij} + \rho_{ij}$, we get the final expression for ρ_{ij}:

$$\rho_{ij} = s_{ij} - \frac{1}{N}\max_{k\neq j}\big(s_{ik}+\alpha_{ik}\big) - \frac{N-1}{N}\big(s_{ij}+\alpha_{ij}\big), \tag{7.6}$$

which can be further simplified to

$$\rho_{ij} = \frac{1}{N}\Big[s_{ij} - \max_{k\neq j}\big(s_{ik}+\alpha_{ik}\big)\Big] - \frac{N-1}{N}\alpha_{ij}. \tag{7.7}$$

Once again, compared to the original AP ρ_{ij} messages (Eq. 2.4), we observe that the update only uses $\frac{1}{N}$ of the standard variable-to-factor I_i message, and subtracts most of the incoming factor-to-variable message α_{ij} from that quantity.

7.1.2 Dual Objective and Decoding for MPLP

Despite the striking similarity of the MPLP updates to regular AP updates, recall that the MPLP equations are derived as block coordinate descent steps in the dual objective. Therefore, we can monitor the dual objective, which is guaranteed to decrease at every iteration, and provide an upper-bound on the quality of the primal solution. The dual objective can be easily calculated by using the fact that max-product linear programming can be shown to operate on the same dual objective of Eq. 2.21 (Sontag et al., 2010). Therefore, we can use the dual variables (which are actually the messages) to calculate the maximizing assignments of the message-based reparameterized potentials. Since $\delta^I_{ij} = \eta_{ij}$ and $\delta^E_{ij} = \alpha_{ij}$, the reparameterization of the S_{ij} singleton potentials, as well as the E_j and I_j functions, at some iteration t, has the following form:

$$S_{ij}^t(h_{ij}) = S_{ij}(h_{ij}) + \eta_{ij}(h_{ij}) + \alpha_{ij}(h_{ij}), \tag{7.8}$$

$$I_i^t(h_{i:}) = I_i(h_{i:}^I) - \sum_{j=1}^{N} \eta_{ij}(h_{ij}^I), \tag{7.9}$$

$$E_j^t(h_{:j}) = E_j(h_{:j}^E) - \sum_{i=1}^{N} \rho_{ij}(h_{ij}^E). \tag{7.10}$$

Clearly, we cannot explicitly represent the reparameterization of either I_i or E_j, since that would require evaluating all exponentially many configurations. Practically, we never need to explicitly store those values, since for the purpose of running the algorithm they are never explicitly used. For the purpose of calculating the dual we only care about the **maximizing** value of every reparameterized potential. This can be calculated without needing to explicitly asses all configurations (since most are not valid). The dual value is simply the sum of all the maximizing configurations of every reparameterized potential. It is interesting to note that the dual value can be calculated even from the difference messages, α_{ij}, as opposed to $\alpha_{ij}(1)$ and $\alpha_{ij}(0)$ so we do not need to maintain values for both the zero and one configuration of the various messages.

Ultimately, obtaining a solution to the exemplar-based clustering problem, requires only the decoded values obtained from running MPLP. We can calculate them by assessing, for each reparameterized singleton diagonal potential S_{jj}^T, (assuming we ran for a total of T iterations), whether it is larger than zero or not. Thus, we can work in the message difference formulation, and obtain the following identity (denoting with superscript

t a message or a reparameterized potential at time t):

$$S_{ij}^t = S_{ij}^t(1) - S_{ij}^t(0) \tag{7.11}$$

$$= [s_{ij} + \eta_{ij}^t(1) + \alpha_{ij}^t(0)] - [0 + \eta_{ij}^t(0) + \alpha_{ij}^t(0)] \tag{7.12}$$

$$= s_{ij} + \eta_{ij}^t + \alpha_{ij}^t = s_{ij} + \rho_{ij}^t - s_{ij} + \alpha_{ij}^t \tag{7.13}$$

$$= \rho_{ij}^t + \alpha_{ij}^t \tag{7.14}$$

where we have used the fact that $\eta_{ij} = \rho_{ij} - s_{ij}$. In other words, we can use the exact same decoding method as used for regular AP, by examining whether the sum of the ρ_{jj} and α_{jj} messages of the diagonal variables in the factor graph are positive or negative to decide on the exemplar set. Importantly, by decoding at every iteration, we can monitor the primal-dual gap, and ascertain we have reached an dual (and thus primal) optimal solution for the LPR problem, if the two are the same. Once we have obtained the dual optimal, if at that point the singleton reparameterizations (given by the beliefs) are also all unique, ($S_{ij}^T \neq 0 \; \forall i, j$), the LPR solution is also integral, which indicates the relaxation is tight, and it is therefore the optimal solution for the original MAP inference problem (*c.f.* (Sontag et al., 2010)).

Note that similarly to AP, the decoding strategy of only looking at the diagonal elements sidesteps the question of whether decoding of **all** singleton variables will result in solutions that are valid and consistent in terms of the constraints encoded by the E_j and I_i functions. It is in fact possible that such a decoding will lead to violating solutions. The full singleton variable decoding strategy is the one suggested in (Sontag et al., 2010). However, they do not discuss the case of non-bounded functions that have infinite penalties as is the case for AP. Another caveat of full decoding is that if used during the execution of the algorithm, for the purpose of evaluating the current primal it is not possible to use the temporary solution for obtaining any guarantees as to how good the resulting assignment is.

The question of the agreement between the diagonal variables and all other variables was never addressed in the original AP framework, and addressing it for the case of AP and MPLP remains to be researched. Using the simpler decoding strategy of examining solely the beliefs of the diagonal variables is problematic only when none of the diagonal beliefs are positive and thus there are no exemplars decided upon by the algorithm. As is done in AP, we can simply announce failure of the algorithm to find a solution, and return. Some alternative decoding schemes that are also relevant to MPLP are discussed in section Section 7.2.2.

7.2 Exemplar-based Clustering Using Dual Decomposition

We now describe the formulation of dual-decomposition for the specific problem of exemplar based clustering and show the type of intermediate quantities that need to be calculated when running the algorithm. We then discuss the two decisions required in the dual decomposition framework: step-size choice for the sub-gradient based descent step, and the decoding scheme to use during and at the end of the run.

7.2.1 DD - Dual Decomposition for Exemplar-based Clustering

We use the formulation given in 2.3.3 to derive the updates for the dual decomposition based version of exemplar-based clustering (DD). The decomposition into factors results in optimization over the individual row (I_i), column (E_j), and singleton factors (S_{ij}) of the grid based factor graph of Fig. 3.2(a). Using the projected sub-gradient derivation of the dual decomposition algorithm given in (Komodakis et al., 2007) may result in alternative decompositions, but they would require passing messages over hyper-trees and are thus more elaborate.

The interesting calculations required for implementing the algorithm are those related

to the optimizations of Eq. 2.23 and Eq. 2.24. We discuss these optimizations for the different types of potentials.

We use the notation h^I_{ij} (h^E_{ij}) for the copy of the h_{ij} variable in the I_i (E_j) function. In other words, h_{ij} corresponds to x_i, and h^I_{ij} to x^c_i described in Section 2.3.3. We denote the dual variable for an I_i potential and each of its connected variables h_{ij}, $j = 1, \ldots, N$ by η_{ij}. We denote the dual variable for a E_j potential and each of its connected variables by α_{ij}. Here, we drop the iteration superscript on the dual variables for clarity. Finally, we denote the reparameterized I_i potential at iteration t by I^t_i, and similarly for the other potentials.

For every singleton potentials S_{ij}, we calculate

$$h^*_{ij} = \arg\max_{h_{ij}\in\{0,1\}} S^t_{ij}(h_{ij}) = \arg\max_{h_{ij}\in\{0,1\}} [S_{ij}(h_{ij}) + \eta_{ij}(h_{ij}) + \alpha_{ij}(h_{ij})]. \tag{7.15}$$

This involves simple evaluation of the sum for the two possible settings of h_{ij}.

For every I_i function, we calculate

$$h^{I*}_{ij} = \arg\max_{h^I_{i:}\in\{0,1\}^N} I^t_i(h^I_{i:}) = \arg\max_{h^I_{i:}\in\{0,1\}^N} \big[I_i(h^I_{i:}) - \sum_{j=1}^{N} \eta_{ij}(h^I_{ij})\big], \tag{7.16}$$

This optimization can be done efficiently since it is very similar to the type of optimizations performed for regular affinity propagation, where we can see that the dual variables act exactly as the (negative) incoming messages. Therefore, using the same line of reasoning, suffice is to calculate for every j the difference $\eta_{ij}(1) - \eta_{ij}(0)$, pick the minimal difference index k, set $h^{I*}_{ik} = 1$ and all other $j \neq k$ variables to $h^{I*}_{ij} = 0$.

For every E_j function, we calculate

$$h^{E*}_{ij} = \arg\max_{h^E_{:j}\in\{0,1\}^N} E^t_j(h^E_{:j}) = \arg\max_{h^E_{:j}\in\{0,1\}^N} \big[E_j(h^E_{:j}) - \sum_{i=1}^{N} \alpha_{ij}(h^E_{ij})\big]. \tag{7.17}$$

Once again, the same arguments used for regular AP are applicable. We need to

evaluate the minimal of the following two expressions:

$$\min\Big[\sum_i \alpha_{ij}(0), \alpha_{jj}(1) + \sum_{k\neq j} \min_{h^E_{kj}\in\{0,1\}} \alpha_{kj}(h^E_{kj})\Big]. \tag{7.18}$$

If the first expression is the minimal, the optimal setting is obtained by $h^{E*}_{ji} = 0\ \forall i$. Otherwise, the optimal setting is obtained by setting $h^{E*}_{jj} = 1$, and for every $k \neq j$ setting $h^{E*}_{kj} = \min_{h^E_{kj}\in\{0,1\}} \alpha_{kj}(h^E_{kj})$.

7.2.2 Decoding for DD

When all sub-problems agree on an optimizing assignment, decoding is trivially obtained by setting the binary variables based on that assignment. Such an assignment is a valid assignment since all the hard constraints are satisfied individually. Due to the agreement, all the constraints are also jointly satisfied . Furthermore, it is an optimal solution for the MAP inference problem and the primal-dual gap is necessarily zero. Otherwise, to obtain a primal solution, we can employ one of several alternatives we now describe.

Similarly to the AP and MPLP decoding, we can evaluate the reparameterization of all diagonal potentials, $S^t_{jj}(h_{jj})$ for the two settings of h_{jj}. Specifically, $S^t_{jj}(1) = S_{jj}(1) + \eta_{jj}(1) + \alpha_{jj}(1)$ and $S^t_{jj}(0) = \eta_{jj}(0) + \alpha_{jj}(0)$.

We then set

$$h_{jj} = \begin{cases} 1 & S^t_{jj}(1) > S^t_{jj}(0), \\ 0 & \text{otherwise.} \end{cases} \tag{7.19}$$

This gives us a set of exemplars, and non-exemplars are assigned to their nearest exemplars. The caveat with this method is of course that it can be the case that none of the S^t_{jj} satisfies the condition for being an exemplar and we can thus not decode a feasible solution.

In cases where we cannot decode based on the above solution, it is possible to use a

generalization of the local singleton decoding scheme ((Komodakis et al., 2007)), which employs tree-based assignments. We chose a subset of trees that cover the graph and are non-overlapping, and use their individual decoding. For DD, we can look at the optimizing assignments of all the I_i reparameterized nodes, and choose the exemplar set based on that, and then assign points to exemplars based again on the similarity, rather than the decoding. In the DD code, we have included this decoding strategy if no singleton based exemplars are found.

Another example of a primal construction strategy is that of keeping track of all maximizing assignments for a particular factor while running the algorithm, and averaging them. While this strategy comes with convergence guarantees(see (Sontag et al., 2010)), it requires careful bookkeeping for functions such as E_j due to the large number of feasible optimizing solutions. In general, this strategy does not lend itself easily to high-order factors. Additionally, there is vast literature on the topic, but the essential problem is a hard one as it effectively asks the same question we started out with, perhaps with a smaller set of unconstrained variables. Thus we cannot expect a particular set of heuristics to always be better than another.

We note that as we execute the algorithm we also want to monitor the primal-dual gap. In order to do so, we must be able to decode intermediate solutions for the primal at any given iteration, so that we can calculate the primal value. We use the techniques mentioned above. The reason we may want to monitor the gap is that it is required when we use particular step-size choices, such as those given by Eq. C.5 used within Eq. C.4. Being able to evaluate a primal objective also establishes when the primal and dual objectives are the same, indicating the problem is optimized and the algorithm can be terminated. Finally, even when the algorithm converges to solutions that have non-zero primal-dual gap, we can use the results to obtain some sort of a non-optimal solution to the inference problem, with the hope that while not optimal, it may still be an acceptable solution to the problem at hand.

7.2.3 Discussion

This concludes our development of two exemplar-based clustering methods. One algorithm (MPLP) is based on max-product linear-programming, which is a block coordinate descent method on a linear programming relaxation of the MAP inference problem. The second algorithm (DD) is based on the dual decomposition framework, which is a subgradient descent strategy for the Lagrangian relaxation of the MAP inference problem. In the next chapter, we compare these algorithm to affinity propagation, which is based on the max-product belief propagation algorithm. Our goal is to determine if the convex relaxation based methods provide better experimental performance over that of AP.

Chapter 8

Comparison of Exemplar-based Clustering Algorithms

In this chapter we compare affinity propagation to the max-product linear programming (MPLP) and dual-decomposition (DD) algorithms for exemplar-based clustering that were derived in Chapter 7. Note that as in the previous chapter, when we use the terms MPLP and DD, we refer exclusively to the exemplar-based clustering variants. We compare the three algorithms on several data sets used in (Frey & Dueck, 2007). In the following sections, we provide the experimental setup details and describe the different data sets. We then provide a detailed discussion of each experiment.

Overall, the outcome of the experiments is that across all settings, AP obtains better results than either MPLP or DD, at a better run-time. This is a somewhat disappointing result, as both MPLP and DD come with theoretical justification and convergence guarantees. However, these results demonstrate the relevance of AP despite the various recent advancements in optimization strategies.

8.1 Experimental Setup

For AP, the free parameter that we must set is the damping level. We set it to 0.9 in all experiments, except for the cities data set, for which we use 0.8 in order to replicate the original setting used in (Frey & Dueck, 2007). MPLP requires no specific parameter settings. DD requires choosing a parametric form of the sub-gradient update step-size, and particular values for the parameters used in the step-size equations. We use the update step $\alpha^t = \frac{a}{b+t}$ (Eq. C.3). This update step was chosen based on a set of experiments described in Appendix C. For every data set, we further optimize for a and b by running several sets of experiments with varying values for these parameters. We report only the run based on the best choice of the parameters.

All algorithms have the same asymptotic time and space complexity of $\mathcal{O}(N^2)$ operations and memory requirements per-iteration, where N is the number of data points. For DD and MPLP, there is an additional step of evaluating the dual objective function, which does not affect asymptomatic complexity. Practically, the iteration times are similar across the algorithms. We observe that the difference between the minimal and maximal per-iteration run-time is at most a factor of three. The dominating factor of the computational effort in obtaining a solution is the total number of iterations required for convergence. This figure varies greatly among the algorithms.

For AP, we determine convergence if the decoded solution does not change for a fixed number of T iterations, where we set $T = 100$. For DD and MPLP, if all optimizing assignments for individual re-parameterized potentials agree, we know we have strong duality, and that we have obtained the MAP solution (Sontag et al., 2010). However, this scenario was only observed in the simplest data set we examined. A more general strategy is to monitor the value of the dual objective the algorithms are attempting to minimize. When the dual objective no longer changes, or changes in very small measures, we can declare convergence. We therefore look at the previous T iterations, and at the average difference in the dual objective value between iterations. If that number is less

than ϵ, we declare convergence. More precisely, letting $L(\boldsymbol{\delta}^t)$ be the dual value at iteration t, convergence is declared when $\frac{1}{T}\sum_{i=1}^{T-1} L(\boldsymbol{\delta}^{t+1}) - L(\boldsymbol{\delta}^t) \leq \epsilon$, where we set $T = 100$ and $\epsilon = 10^{-5}$ throughout. We also set a maximal number of iterations, depending on the data set size and observed behavior of the algorithms across iterations.

As typically done for convex relaxation methods, we also monitor the value of the primal objective. Recall it is obtained by using the local decoding scheme of finding the optimal value for every diagonal reparameterized singleton potential (see Section 7.1.2 and Section 7.2.2).

8.2 Data Sets

In order to compare the different methods, we use the following benchmark data sets from (Frey & Dueck, 2007; Frey & Dueck, 2008)[1]:

1. Toy: This is a simple 2D data set of 25 points, for which any reasonable method is expected to find a good, or even optimal, solution. The similarities are simply the negative Euclidean distances between the points. The preference for each point is constructed by taking the median similarity of that point to all other points. The data set is visualized in Fig. 8.1.

2. Sentences: The data set contains 125 sentences comprising the (Frey & Dueck, 2007) manuscript. Similarities and preferences are based on a coding cost given in the supplementary material of (Frey & Dueck, 2007). Note that similarities are not symmetric for this data set. We quote as an example two of the exemplars found by all algorithms in the experiments below:

   > "Affinity propagation identifies exemplars by recursively sending real valued messages between pairs of data points."

[1]Data obtained from http://www.psi.toronto.edu/affinitypropagation/ and http://www.psi.toronto.edu/affinitypropagation/vsh/

> "The availability a is set to the self responsibility r plus the sum of the positive responsibilities candidate exemplar k receives from other points."

3. Cities: A data set of 456 points corresponding to North-American cities, where similarities correspond to negative estimated airline travel times. Preferences were set to -4500 to reproduce the results from the original publication. Note that similarities are not symmetric for this data set.

4. Faces-400: This data set contains 400 grey-scale face images, taken from the Olivetti data set. The faces are all centered and cropped. The images are 64×64 pixels, and are composed of variations of gaze direction, facial expression, and light conditions for 10 individuals. Pairwise similarities correspond to the negative sum of squared pixel differences. The preference value was set to -60 to reproduce the results from the original publication.

5. Faces-900: This data set is an expanded version of the Faces-400 data set, where rotations of up to 10 degrees were applied to some of the images. The data set is visualized in Fig. 8.2.

6. Faces-Video: The first of two larger data sets for which we run only AP and MPLP, due to the prohibitively long time it requires to find good parameter settings for the DD step size. The data set contains 2000 video frames containing faces. Similarities and preference values used to reproduce the original publication.

7. Exons: The second of two larger data sets for which we run only AP and MPLP, due to the prohibitively long time it requires to find good parameter settings for the DD step size. The data set contains 10,000 putative exons sequences. Similarities and preference values used to reproduce the original publication.

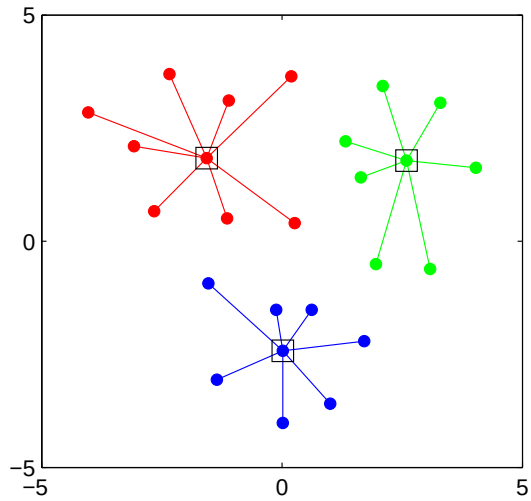


Figure 8.1: Toy data, shown with optimal clustering. Clusters are color coded. A black box marks the cluster exemplar.

8.3 Results

8.3.1 Toy Data

As expected, all methods find an optimal solution for this data set (the primal-dual gap for MPLP and DD is 0, up to numerical precision, and all individual reparameterized potentials optimizing configurations agree on the solution). Based on these results we can assert that AP finds the optimal solution since it obtains the same objective net similarity value as the other algorithms. This experiment also serves as a sanity check for the correctness of our derivation and implementation of MPLP and DD.

The number of iterations required for convergence, however, is quite different among the algorithms. AP converges in 130 iterations, MPLP in 455, and DD in 553. The progression of the runs are shown in Fig. 8.3 using a log-scale[2]. We see that in terms of run-time, standard AP performs considerably better for this data, as compared to the other methods. On the other hand, at convergence, we do not know whether or not AP indeed obtained the optimal solution, whereas we have a certificate of optimality for the

[2]as will be used in all the consequent plots.

other algorithms. MPLP converges well ahead of DD, and as expected, the dual objective for MPLP decreases monotonically, as opposed to that of DD. One striking property of DD is how unstable the primal decoding is, as indicated by the highly varying value of the primal objective value throughout the run.

8.3.2 Sentences Data

In this experiment, we observe AP converging very quickly (200 iterations) to a solution with a high net similarity (-10234). MPLP and DD obtain lower (worse) net similarities (-10251 and -10396 respectively). Additionally, neither MPLP nor DD converge by the threshold value described in Section 8.1. We have allowed MPLP to run for 10K iterations. Only after 9K iterations, does the average dual decrease goes below 10^{-3}. The average dual decrease for DD does not settle even after 20K iterations, and the dual value continues to fluctuate. The primal-dual gap indicates the obtained primal solutions are not optimal for either method.

Thus, for a problem on which we do not have a certificate of optimality for the solution, we observe that AP outperforms the other variants. Furthermore, it obtains a better solution much more quickly than either MPLP or DD. Note that we cannot assess whether that is an optimal solution or not. If it is, this is an indication that the dual objective bound holds with weak duality. In the linear-programming view, the relaxation of the LP to the IP is not tight and there are fractional optimal solutions to the LP. The results of the full run are shown in Fig. 8.4(a).

A detailed plot of the first 500 iterations is given in Fig. 8.4(b). In this figure, we can see that the primal solution for DD appears to get progressively worse, with occasional dramatic improvements. Note that the low value solutions are obtained from the second type of decoding we use - the tree-based decoding. It is used whenever we cannot obtain a valid decoding from the standard scheme. See Section 7.2.2 for more detail. The reason the tree-based decoding grows worse is that as points are chosen as exemplars

by letting every point choose its exemplar with no regard to the exemplar consistency constraint, or the exemplar preference. Every point always has a preferred exemplar, but we compute the primal using the reparameterzied potentials. The potentials associated with bad exemplar choices become more and more penalized and so the primal decreases. This behavior is also observed in the following experiments. Occasionally, we obtain a decoding from the standard method, as indicated by the jumps to more reasonable values. We do not use alternative decoding for MPLP, and thus whenever we have no decoding there is no MPLP primal value to plot.

In terms of the actual solutions found by the different algorithms, we note that all algorithms agree on two sentences as exemplars. Those are the sentences quoted in Section 8.2. DD, however, only finds these two exemplars, while AP finds a total of four exemplars, and MPLP finds a total of five exemplars. Somewhat surprisingly, there is no agreement between AP and MPLP on these additional exemplars. However, it is important to remember that it is difficult to judge which algorithm performs better from the cognitive perspective. Furthermore, the performance of all methods depend greatly on the choice of the similarity measure between sentences used, which does not necessarily capture the nuances of language.

8.3.3 Cities Data

This data set poses a seemingly simple challenge with a moderate size data set of 456 data points. AP converges after less than 150 iterations to a solution with a net similarity of -92.46×10^3. However, both MPLP and DD perform poorly on this data in terms of the obtained net similarity and the convergence times.

MPLP obtains a relatively good solution with a net similarities of -93.86×10^3, however, it does so in a fairly erratic manner. Recall that for MPLP, we use the same decoding method as that of AP. However, for the most part the algorithm does not obtain a feasible primal solution using that decoding. While the dual function drops to a

satisfying value fairly quickly, we obtain a primal feasible solution for the first time only after more than 30K iterations[3]. Soon after, decoding again becomes infeasible until 50K iterations. After 80K iterations we again lose the feasible solution. Therefore, the final solution for this run is obtained not by the solution at termination, but by searching the best net similarity obtained at any point along the run. We note that perhaps potentially better decoding strategies would have allowed us to obtain feasible solution much earlier. However, recall the minor differences between the update equations for MPLP and AP, and that with AP we do not require more complicated decoding in order to obtain good solutions.

For DD, we were unable to obtain a good primal feasible solution (the best had a net similarity of -275.61×10^3, which is clearly not a good solution). As is the case for MPLP, we often times do not obtain a primal feasible solution from the decoding strategy we use. In fact, in the first few thousands of iterations we obtain feasible solutions, but later on we no longer obtain feasible solutions from the local decoding. The dual solution obtained for DD is also considerably higher than that obtained for MPLP. This indicates that the dual optimization procedure is converging at a very slow rate. For this problem, we have also tried a dynamic step-size and were not able to obtain better results (results not shown). We show a plot of the run in (Fig. 8.5).

8.3.4 Faces-400 Data

In this experiment, AP converges in less than 200 iterations to a solution with higher net similarity (-5437) than either MPLP (-5444) or DD (-5463). Neither MPLP nor DD converged by our defined threshold. However, as can be seen in Fig. 8.6, the dual and primal objectives for both MPLP and DD approach good solutions within 1000-1500

[3]The reason we do not always have a primal solution is due to the decoding scheme we use, explained in Section 7.1.2: we set as exemplars the points corresponding to reparameterized diagonal singleton potentials whose maximizing assignment is one (indicated by a positive scalar value in the difference representation we employ). When all of the diagonal elements are non-positive, we do not have a valid decoding and therefore nothing is plotted as a primal value for that iteration.

iterations.

It is worthwhile to note that from a practical perspective, all solutions are quite similar. AP and MPLP find 58 exemplars, while DD finds 57 exemplars, and the three algorithms agree on 50 of those. Thus, it is likely that for practical purposes the clustering solutions are quite equivalent, but when faced with the decision of which method to use in order to obtain a clustering solution, there is no strong motivation to choose a convex relaxation based variant for this data set.

8.3.5 Faces-900 Data

The faces-900 data set is quite similar to the faces-400. However, in terms of the quality of the results, we see that MPLP takes a long time to converge, with no primal feasible solution for the first 2000 iterations, and a final solution that is lower than that obtained by AP (-13.455×10^3 for AP, compared to -13.527×10^3 for MPLP). DD does not find a good primal feasible solution and the decoded solution becomes progressively worse. The dual bound obtained by DD is also considerably higher than that of MPLP, indicating the procedure is converging at a very slow rate. We show a plot of the run in (Fig. 8.7).

8.3.6 Large Scale Data - Faces-Video and Exons

In these two experiments we ask whether the regime of problems we have looked at so far is a more appropriate one for AP, whereas for larger problems MPLP is more suited. We do not attempt to run DD on these data sets since a typical run takes several days, making it prohibitively expensive to find good step-size parameter settings for DD.

We find, however, similar results to those obtained in previous experiments. In both the faces-video data set (2000 points) and the exons data set (10,000 points), AP finds a better solution faster than MPLP. We show the plot of the two runs in Fig. 8.8.

8.4 Discussion

We have seen that for several data sets, results obtained by standard AP considerably outperform MPLP and DD, both in terms of the quality of the solution, and the number of iterations required to obtain that solution. Of the two convex relaxation methods, MPLP appears to be the preferred alternative since it obtains better results than DD in less iterations. Furthermore, for MPLP standard decoding techniques yield good, feasible solutions once the algorithm arrives at a feasible region in most data sets and DD requires careful setting of the step-size type and the step-size parameters, whereas MPLP is parameter free. We note this is an issue that makes DD quite difficult to use for a non-expert; our experimental results indicate that every problem for which we wish to run DD requires a different set of parameter tuning experiments for the particular choice of step-size type we used. This is both time-consuming and requires a 'feel' for how to modify the parameters to arrive at a range for which convergence can be achieved in a reasonable amount of time.

We ask how these results compare to published results. In (Globerson & Jaakkola, 2007) a main observation was that MPLP converges and provides solutions in many cases where MPBP does not converge. Therefore, it may indeed be a good strategy to use the provably convergent methods for problems on which MPBP displays convergence issues. In the case of AP, however, we observed no convergence issues on the tested data sets. Globerson & Jaakkola (2007) further report that for runs where MBPB does converge, the obtained results for MPBP, MPLP, and the sequential tree reweighted max product algorithm (TRW-S, (Kolmogorov & Wainwright, 2005)) are very similar. These experiments were carried out only on data sets with pairwise interactions.

Experiments for high-order models using MPLP were carried out in (Lazic, 2011) for the facility location problem. There, similar results are reported. The FL version using MPBP typically obtains better solutions in less iterations. However, for two of the 13 problems investigated in their benchmark experiments, MPLP did in fact outperform

MPBP in terms of the quality of the solution. MPLP, however, always required much longer time to converge, and it is not clear from the reported averages whether this is due to a small amount of non-converged runs, a single outlier data set, or a general trend, since every problem consists of several data sets.

For DD, in the high-order version of the paper where it is introduced (Komodakis & Paragios, 2009), no comparative experiments were shown. For the pairwise case, the experiments in (Komodakis et al., 2007; Komodakis et al., 2010) are against TRW-S, where it is shown to outperform TRW-S. TRW-S is considered the state-of-the-art algorithm for optimization on pairwise graphs. However, it does not have a high-order variant. Developing this variant is an interesting direction of future research.

We hypothesize that the presence of high-order factors in a model considerably hinders the algorithmic performance of dual descent methods in comparison to MPBP. Therefore we cannot necessarily know whether observations on pairwise model based results generalize well. From the experiments we ran here, AP, which is based on MPBP, remains the algorithm of choice in the case of exemplar-based clustering.

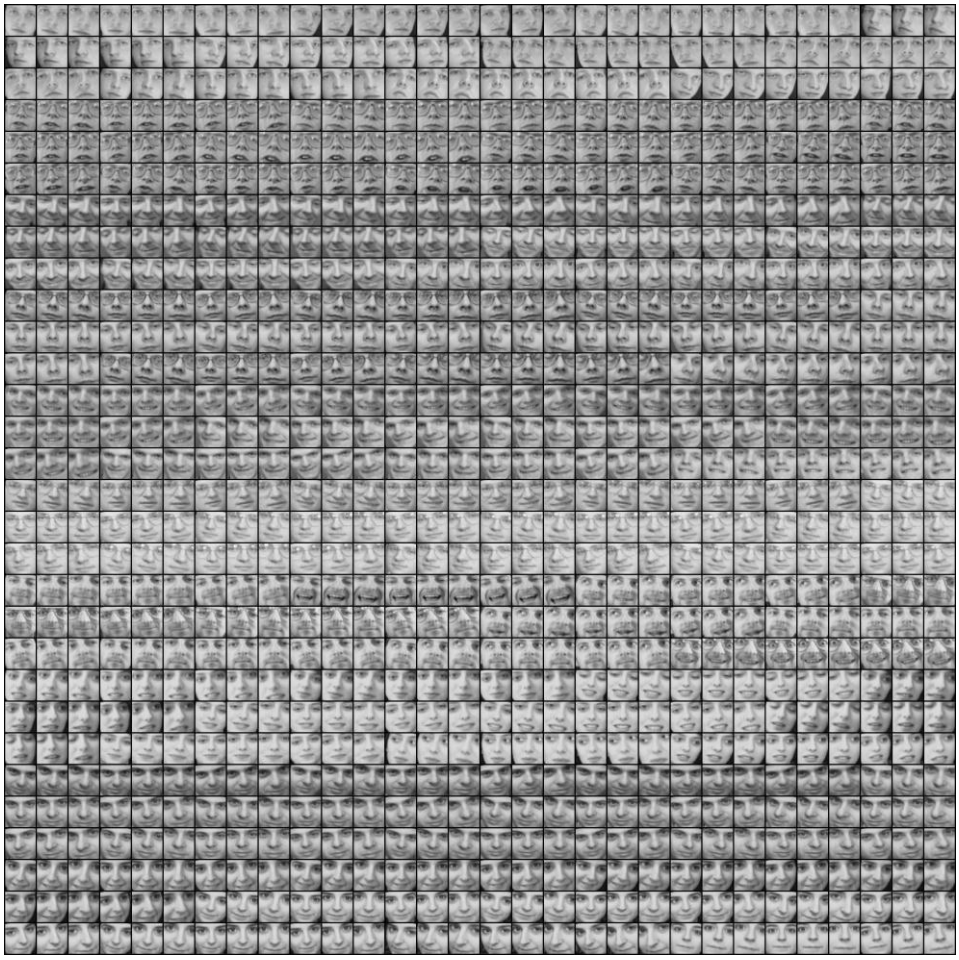


Figure 8.2: Faces-900 data comprised of 10 individuals, with varying light, gaze, expression and rotation.

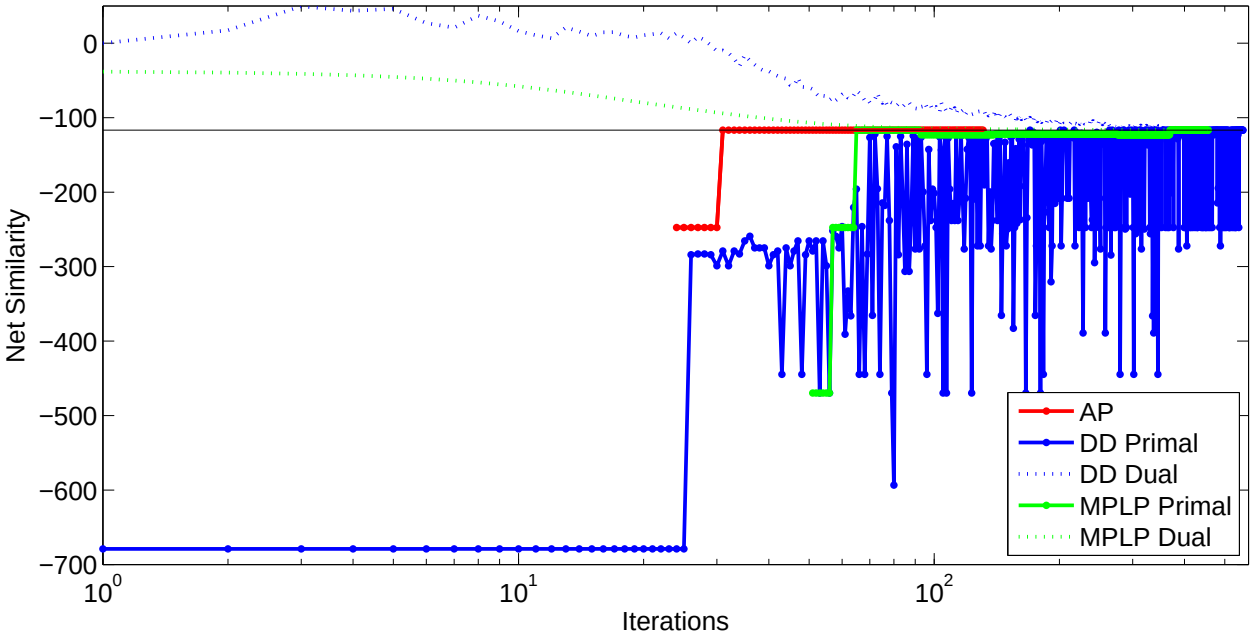
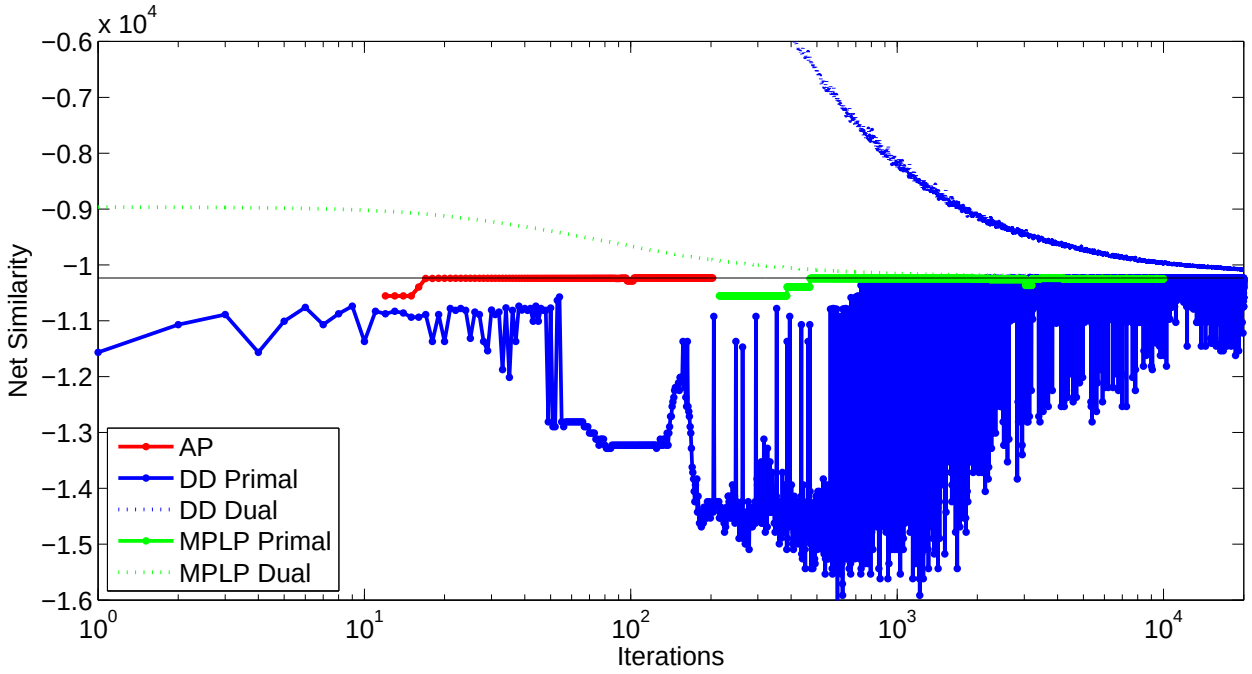
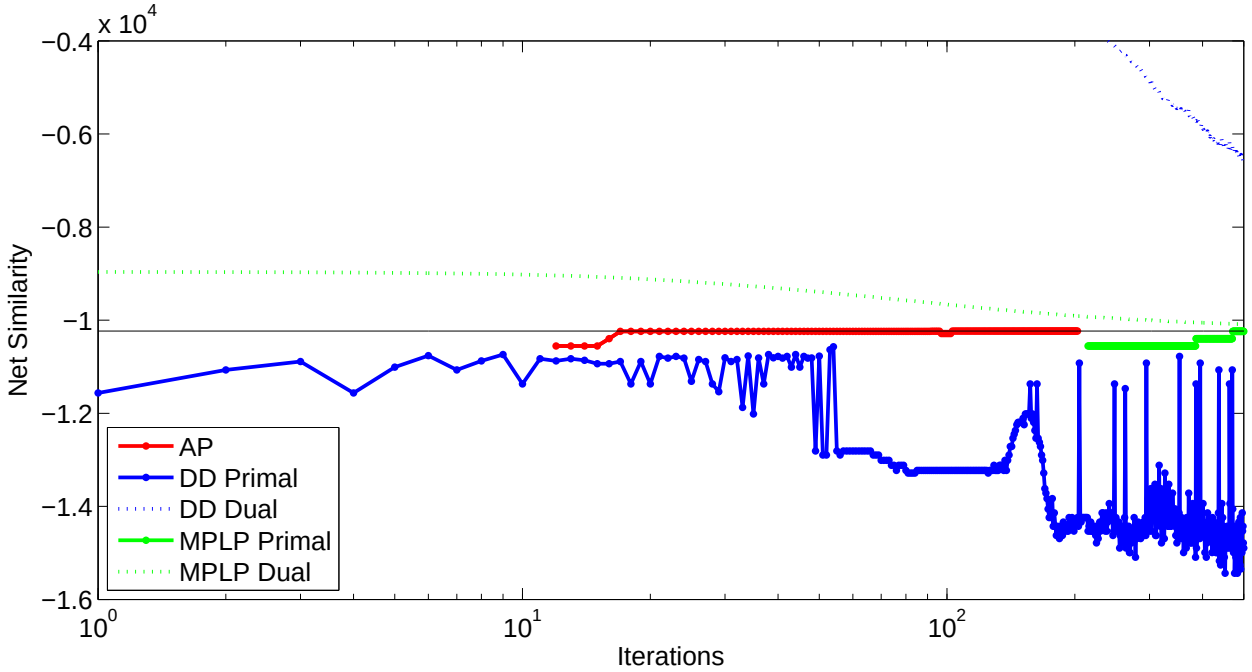


Figure 8.3: Comparison of methods for the toy data.



(a) Results for the full run.



(b) Results as above, zoomed in.

Figure 8.4: Comparison of methods for the sentences data.

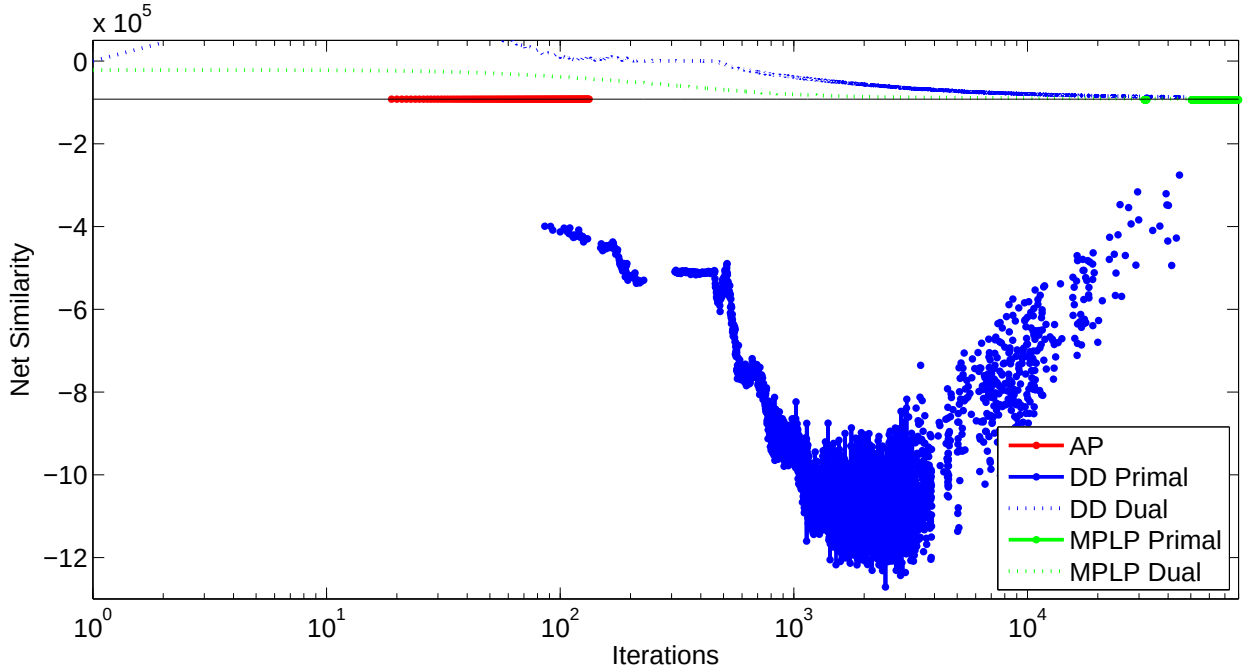


Figure 8.5: Comparison of methods for the cities data.

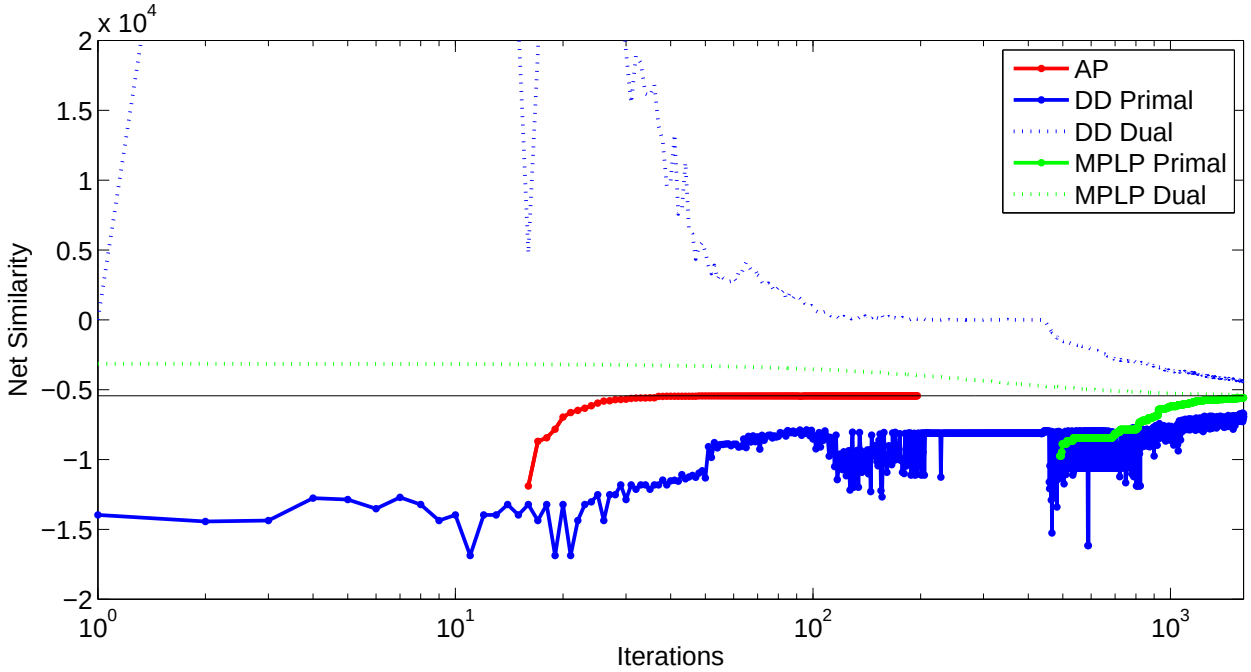


Figure 8.6: Comparison of methods for the faces-400 data.

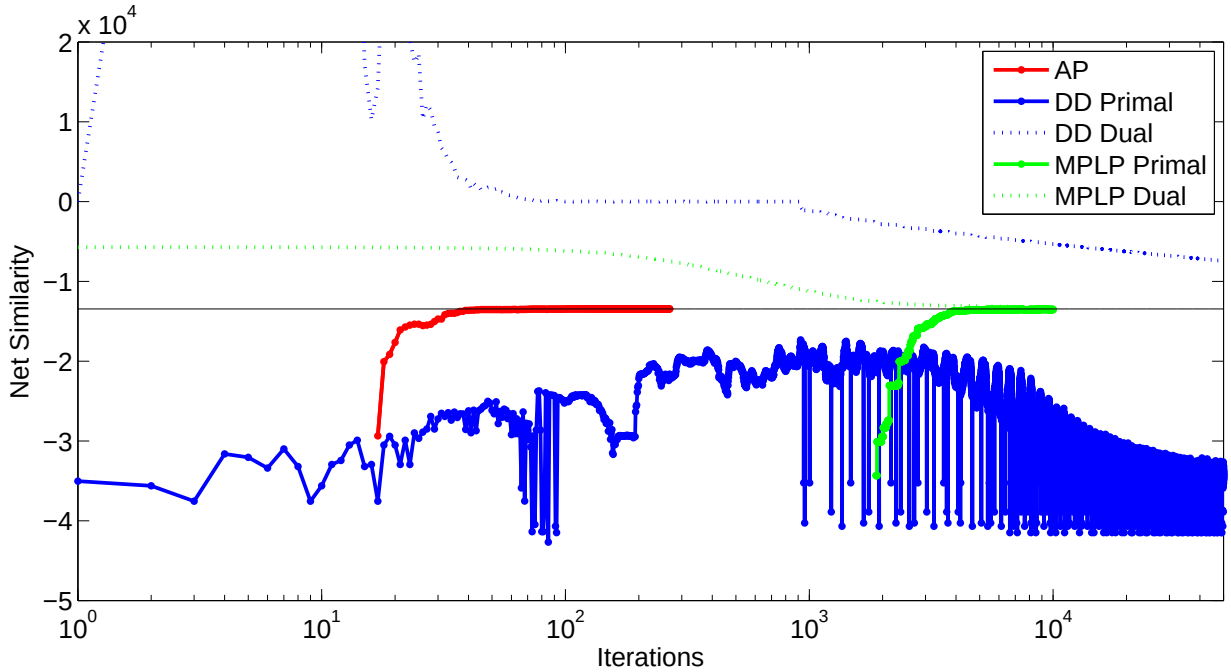
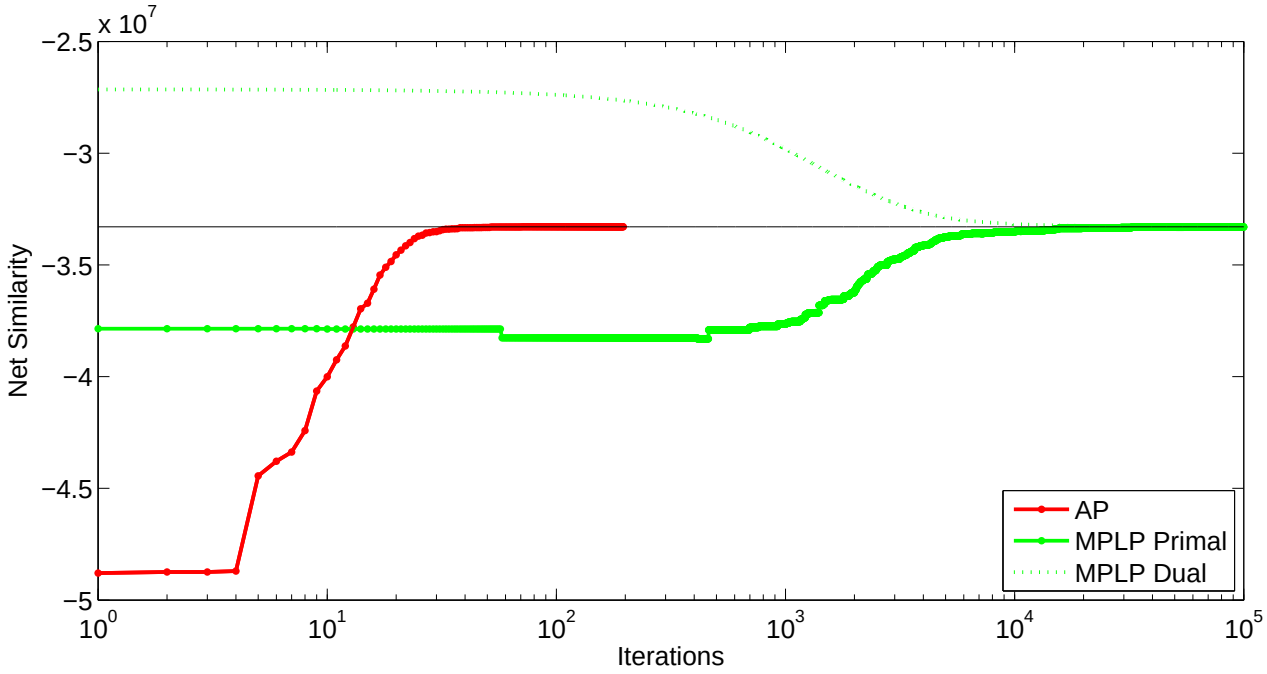
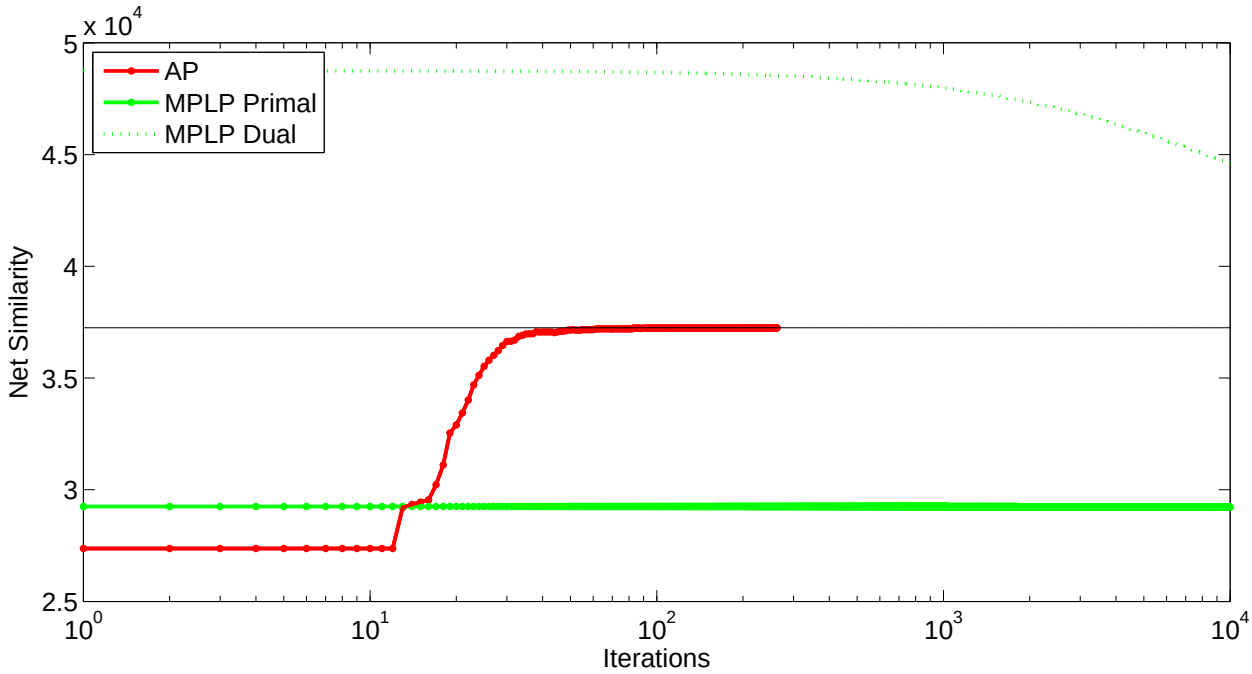


Figure 8.7: Comparison of methods for the faces-900 data.



(a) Comparison of methods for the faces-video data.



(b) Comparison of methods for the exons data.

Figure 8.8: Comparison of AP and MPLP for large scale data sets.

Chapter 9

Conclusions

A major theme of this thesis is variants of affinity propagation that extend the basic exemplar-based clustering algorithm to problems with more structure and constraints. A main contribution is the binary variable model framework that allows for easy and principled inclusion of constraints and side information. This is the framework which was presented in Chapter 3. For practitioners who wish to use affinity propagation for clustering and realize their particular problem requires additional modeling assumptions, this framework already appears to be of use. We hope to see this trend continue.

The various extensions presented in subsequent chapters, based on the binary variable model, (CAP, SSAP, and HAP), address particular clustering variants. These variants could potentially serve practitioners from various communities including vision, computational biology, and operational research. Furthermore, they provide concrete examples of how to practically modify the basic exemplar-based clustering factor graph to include additional structure. Another important property of these algorithms is that they are all expressed in terms of adding high-order constraints, whose special structure allows us to retain tractable message passing based optimization.

In terms of the specific clustering models we investigated, several directions arise from the body of work presented here. Different types of modifications can be made to the

form of the factors in the various algorithms presented in this thesis. Such modifications lead to related clustering variants that have not been explored in full. For instance, rather than assigning every point to exactly one exemplar, a conceivable framework is that of assigning every data point to multiple exemplars. This could be used to explain *e.g.* an image containing multiple objects via exemplars that are images of the individual objects. Our preliminary investigation of this model shows that we can efficiently compute messages in the model. However, the framework requires a cleverly constructed similarity function as well as an additional mechanism that will ensure the subset of exemplars chosen by a particular data point is as diverse as possible. Further work is required in order to assess whether this is a viable line of work.

Most of the constraints we have dealt with are hard constraints; we seek a solution that satisfies them in full. However, soft constraints, where there is merely a penalty term associated with violating the constraints, are often also appropriate for particular problems. In the case of semi-supervised affinity propagation, the instance-level constraints on pairs of data points may be of a probabilistic nature and have associated uncertainty. Investigating the properties of the soft-constraint algorithm may lead to interesting results. A particular example we have studied recently is that of constraining a sequence of clusterings on similar problems (for instance, clustering consecutive video frames, or some other temporal sequence) to be consistent in some sense. A hard constraint version of the problem was given in (Yang et al., 2011) for the application of vehicle tracking. The constraint required that points that were in the same cluster in a previous time-step, remain together. In that work, clusters are comprised of visual interest points, and expected to correspond to vehicles, making the hard constraint reasonable. However, for a different system, where there is a more fluid integration and disintegration of clusters, such a hard constraint leads to degeneracy since clusters can merge, but cannot split. Eventually the system converges to a single cluster in order to satisfy the constraint. This undesirable behavior can be alleviated with a soft constraint. We are currently

experimenting with this model, which although more involved than the hard constraint version, is still tractable.

Other modifications do not necessarily have trivial tractable message passing solutions, but can perhaps be optimized via different means. For instance, in the capacitated affinity propagation setting, consider the case of including constraints on the total similarity an exemplar can account for, rather than (or in addition to) the number of points assigned to the exemplar. To the best of our knowledge, there is no trivial way to compute outgoing messages in this case. However, it is possible to use an optimization subroutine, such as a $0-1$ knapsack problem solver, to compute the outgoing message. Note that this type of constraint has been discussed in the operational research literature, since it is a more general form of capacity constraints in capacitated facility location (Mulvey & Beck, 1984).

At a high-level, we may want to consider using the general framework of message passing for clustering that is not exemplar-based. In fact, one aspect of AP is that the objective function it optimizes, combined with the lack of restriction on the form of the similarities allow for solutions where two data points in the same cluster are infinitely dissimilar. Intuitively, we may want to optimize an alternative objective function, which takes into account the coherency of clusters, and requires that clusters are made of points that are overall similar. This can be phrased in terms of a discrete model, and the development of a message passing algorithm for such a setting is an ongoing direction of research.

We note the growing interest in high-order potentials in recent years. The reason for this is two-fold. First, though we can formulate many problems using only pairwise interaction, which effectively specify a prior belief on the properties of the problem structure, such priors are too weak. They lead to limited models that do not capture real-world behavior. This has become particularly apparent in specific cases where the models in questions can be globally optimized in polynomial time. The solutions are typically un-

satisfactory in terms of our expectation given the problem understanding. We therefore must investigate more complex models. A natural way to extend our models is to include interactions over larger subsets of variables.

Secondly, not all high-order factors are alike, and there is a potentially large and interesting space of high-order factors and models for which efficient optimization algorithms (either exact, or approximate) exist. This indeed led to a spur of work on high-order potentials and models. This trend is particularly apparent in the vision community, where the model optimization is often done via various graph-cuts based constructions (*e.g.* Rother's version of the pattern-based potential (Rother et al., 2009), the p^n Potts potential (Kohli et al., 2009a), Robust p^n potentials which include the region-based consistency potential (Kohli et al., 2009b), the label-cost potential (Delong et al., 2010), the co-occurrence potential (Ladicky et al., 2010)). Other potentials have been introduced specifically within the message passing framework (*e.g.* linear-constraint potentials (Potetz & Lee, 2008)), and some of the above potentials can also be used in a tractable manner within message passing frameworks (Komodakis & Paragios, 2009; Tarlow et al., 2010).

Affinity propagation is a prime example of a high-order model in which a message passing algorithm gives state-of-the-art performance. In this light, the work in this thesis contributes in demonstrating the applicability of high-order potentials to formulate more complex models for the domain of clustering. Furthermore, we describe and employ several types of high-order potentials and show how to efficiently compute their outgoing messages. These potentials are general and can be used in various domains. Further relevant contribution to this overarching line of research – that of cataloging and introducing several tractable high-order potentials relevant for message passing – can be found in (Tarlow, Givoni & Zemel, 2010). We hope the wide range of high-order factor graph based models presented in this work will serve as an encouraging example to continue using and exploring the space of tractable high-order models for machine-learning

problems.

For high-order potentials in which efficient MPBP message passing calculations is possible, we can also use other optimization strategies. For example, MPLP can typically be applied seamlessly, since the MPLP update equations are simple modifications of the max-product messages. While DD requires finding optimal configurations for the potentials, these can always be obtained if message computation is tractable. These observation have led to the idea of treating the general factor graph model as a collection of smaller, tractable modules. Combined optimization is carried out via general outer-loop frameworks. These ideas also bridge together message passing and combinatorial optimization type strategies. For further discussion see (Duchi et al., 2006; Komodakis et al., 2010; Tarlow et al., 2010).

Indeed, both MPLP and DD can be thought of as outer-loop strategies that come with formal guarantees and deeper mathematical understanding than that of loopy belief propagation. Further development of such theoretically grounded approaches has been an area of much focus in recent times. The last part of the thesis is therefore somewhat disappointing; despite much exciting research, the elusive max-product belief propagation algorithm still remains the best practical tool for affinity propagation. It achieves faster, better solutions than the other methods, and does not require careful step-size parameter manipulations.

There are, however, other alternatives that we have not investigated. One example is the max-sum diffusion algorithm (MSD, (Werner, 2007; Werner, 2008)). In this work we chose to compare to MPLP over MSD since both optimize the same dual using block coordinate descent strategies, but that of max-sum diffusion makes smaller moves. Experimentally it has also been suggested to be less efficient than MPLP (Sontag et al., 2010). We have also chosen to not compare to tree-reweighted max-product (TRW, (Wainwright, Jaakkola & Willsky, 2005; Kolmogorov & Wainwright, 2005)), since there is no known high-order convergent variant (a TRW-S type algorithm) for it. Recent work

in dual decomposition include the accelerated dual-decomposition approach (Jojic et al., 2010). There, instead of sub-gradient optimization, the dual function is approximated with a smooth objective that can be solved with standard gradient-based approaches or other forms of smooth optimization methods. This algorithm comes with better theoretical convergence guarantees than the sub-gradient based approach. However, the actual procedure is considerably more involved. Additionally, in practice it does not always provide better performance, which highly depends on the specific problem in question (Jojic, 2011). Another recent line of work for super-linearly convergent algorithms based on proximal minimization (Ravikumar et al., 2010), though we note this line of work is currently described only in terms of pairwise interactions. A final notable work we have not investigated empirically is the method of (Hazan & Shashua, 2010). Hazan and Shashua provide a general framework based on primal-dual optimization methods. It presents parameterized updates that highly resemble message passing. Some parametric choices recover algorithms such as MPBP, MPLP, and TRW. It may well be possible that any one of these methods will prove a better alternative to MPBP-based affinity propagation.

In our search for methods to improve upon max-product belief propagation for exemplar based clustering we focused only on convex relaxation based strategies, due to their theoretical properties. Another unexplored alternative direction is that of investigating how changes in the standard degrees of freedom of max-product belief propagation improve upon or worsen the performance of AP. These include the damping and the message scheduling. These factors appear to be of importance for MPBP. Although relatively little is known about how modifying damping and scheduling affects algorithm performance, we have recently shown that with appropriate damping and scheduling, MPBP achieves the same provably optimal solution for pairwise binary sub-modular MRFs as graph-cuts based approaches obtain (Tarlow, Givoni, Zemel & Frey, 2011). This result contradicts previous results showing the sub-optimality of MPBP in these cases (Kulesza

& Pereira, 2008). However, the analysis relied upon a synchronous update scheme, and no damping.

The investigation carried out here was for the specific case of exemplar-based clustering. It demonstrates the need for further research into understanding the general class of problems that max-product is likely to succeed (or fail) on, and the theoretical properties of the algorithm. Recall our discussion in Chapter 3 regarding the identical topology of the factor graph representing the NP-hard exemplar-based clustering problem, and the polynomial time solvable problem of maximum weighted bipartite b-matching. That example showed that the behavior of MPBP can be cardinally different, based only on the form of the potentials, and not on the graph topology. This is reminiscent of the applicability of graph-cuts only for the case of sub-modular problems (Kolmogorov & Zabih, 2004). A similar categorization of the problem hardness based on the form and combinations of the potentials is an interesting problem.

In summary, with respect to creating better, yet more complex models, and optimizing them efficiently and robustly, much exciting progress has been made, and much remains to be explored. Good luck.

Appendix A

The Linear Programming Relaxation

Recall we wish to maximize

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \phi(\mathbf{x}) \equiv \arg\max_{\mathbf{x}} \sum_{c=1}^{C} \phi_c(\mathbf{x}_c). \tag{A.1}$$

First, we describe the problem above as that of integer programming, by employing an over-complete representation. For every clique c, and for every configuration of $\mathbf{x_c}$, a binary indicator variable $\mu_c(\mathbf{x}_c)$ is added to denote whether this is the configuration to be used. For every clique, we require that exactly one of its configuration indicators is on. Finally we require that the assignments that are on across cliques agree, which can be described most simply by including another set of indicator variables μ_i for every x_i, and requiring that any clique assignment with x_i in its domain is consistent with its singleton assignment. Letting $\boldsymbol{\mu} = \{\mu_c, \mu_i\} \;\; \forall i, \forall c, \forall \mathbf{x}_c, \forall x_i$, we obtain the following optimization objective:

$$\max_{\boldsymbol{\mu}} \sum_{c=1}^{C} \sum_{\mathbf{x}_c} \phi_c(\mathbf{x}_c)\mu_c(\mathbf{x}_c) \tag{A.2}$$

s.t.

$$\begin{aligned} &\mu_c(\mathbf{x}_c) \in \{0,1\} && \forall c, \forall \mathbf{x}_c, \\ &\textstyle\sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1 && \forall c, \\ &\textstyle\sum_{\mathbf{x}_c \setminus x_i} \mu_c(\mathbf{x}_c) = \mu_i(x_i) && \forall i, x_i, c : i \in c. \end{aligned} \tag{A.3}$$

This problem is completely equivalent to Eq. A.1 and is therefore just as intractable in general. If we relax the integrality constraint on the domain of μ_c to one that only requires the domain to be in the range [0,1], we obtain the linear programming relaxation.

This relaxation can be interpreted as the problem of finding a set of distributions over all possible assignments for every clique such that single variable marginalization constraints per factor are respected. In this sense it is a relaxation from solutions that are valid distributions over $\mathbf{x}$. The feasible domain for the unrelaxed solutions is the probability simplex, which requires an exponential number of constraints to fully satisfy as all marginalizations must be consistent. This set is relaxed to solutions which are only locally consistent, but for which there does not necessarily exist a valid global distribution that gives rise to the local clique distributions. Therefore, this set of agreements only characterizes some aspects of the desired distribution. It is typically referred to as the *local polytope*, and it forms an outer bound on the feasible region for the linear program with full (exponentially many) constraints.

Formally, the problem now can be states as follows:

$$\max_{\boldsymbol{\mu} \in \mathbb{L}(G)} \sum_{c=1}^{C} \sum_{\mathbf{x}_c} \phi_c(\mathbf{x}_c)\mu_c(\mathbf{x}_c), \tag{A.4}$$

where

$$\mathbb{L}(G) = \begin{cases} \boldsymbol{\mu}: & \mu_c(\mathbf{x}_c) \geq 0 & \forall c, \forall \mathbf{x}_c, \\ & \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1 & \forall c, \\ & \sum_{\mathbf{x}_c \setminus x_i} \mu_c(\mathbf{x}_c) = \mu_i(x_i) & \forall i, x_i, c : i \in c. \end{cases} \tag{A.5}$$

The above problem has a large number of variables to optimize over. Every clique in the factor graph is associated with as many variables as there are possible configurations the clique variables can jointly assume. Its dual representation turns out to have a much smaller number of optimization variables. Therefore, it is convenient to solve this problem by minimizing the dual, which is a convex upper bound to the problem of interest. Since the primal is linear, if the optimal solution of the dual is found we are guaranteed to have found an optimal primal solution. Variations in the exact formulations of the problem, even when describing identical problems, yield slightly different dual problems. Different approaches in the literature are based on algorithms for decreasing the upper bound resulting from particular choices of the primal, and differ in the dual formulation or the optimization approach.

Appendix B

Full Derivation of HAP

Here, we give the full derivation of the HAP, HAP-B, and HFL models presented in Chapter 6.

Useful identities and definitions

$$\max(x) = -\min(-x)$$

$$x - \max(x, y) = \min(0, x - y)$$

$$x - \max(y, z) = \min(x - y, x - z)$$

$$\beta_{ij} = s_{ij} + \alpha_{ij}$$

$$\rho_{ij} = s_{ij} + \eta_{ij}$$

Additionally, we will assume for notational convenience that $s_{ij} \leq 0, p_j \leq 0$. For example, we let $s_{ij} = -(x_i - x_j)^2$, and p_j is the median of $\{s_{i1}, \ldots, s_{iN}\}$.

B.1 Functions

Hierarchical affinity propagation (a) - difference w.r.t AP is in $I^{l>1}$:

$$
\begin{aligned}
I_i^1(h_{i:}^1) &= \begin{cases} 0 & \text{if } \sum_j h_{ij}^1 = 1, \\ -\infty & \text{otherwise.} \end{cases} & S_{ij}(h_{ij}^l) &= s_{ij}^l h_{ij}^l \\
I_i^{l>1}(h_{i:}^l, e_i^{l-1}) &= \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases} & C_j(e_j^l) &= p_j^l e_j^l \\
E_j^l(h_{:j}^l, e_j^l) &= \begin{cases} 0 & \text{if } e_j^l = h_{jj}, e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}
\end{aligned} \tag{B.1}
$$

Hierarchical affinity propagation (b) - difference w.r.t AP is in $I^{l>1}$:

$$
\begin{aligned}
I_i^1(h_{i:}^1) &= \begin{cases} 0 & \text{if } \sum_j h_{ij}^1 = 1, \\ -\infty & \text{otherwise.} \end{cases} & S_{ij}(h_{ij}^l) &= s_{ij}^l h_{ij}^l \\
(*)I_i^{l>1}(h_{i:}^l, e_i^{l-1}) &= \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = 1, h_{ii}^l \leq e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases} & C_j(e_j^l) &= p_j^l e_j^l \\
E_j^l(h_{:j}^l, e_j^l) &= \begin{cases} 0 & \text{if } e_j^l = h_{jj}, e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}
\end{aligned} \tag{B.2}
$$

Hierarchical Facility Location: diff w.r.t HAP (a) is in E

$$
\begin{aligned}
I_i^1(h_{i:}^1) &= \begin{cases} 0 & \text{if } \sum_j h_{ij}^1 = 1, \\ -\infty & \text{otherwise.} \end{cases} & S_{ij}(h_{ij}^l) &= s_{ij}^l h_{ij}^l \\
I_i^{l>1}(h_{i:}^l, e_i^{l-1}) &= \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases} & C_j(e_j^l) &= p_j^l e_j^l \\
(*)E_j^l(h_{:j}^l, e_j^l) &= \begin{cases} 0 & \text{if } e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}
\end{aligned}
\tag{B.3}
$$

B.2 Update Equations

HAP (a):

$$
\begin{aligned}
\tau_j^{l+1} &= p_j^l + \rho_{jj}^l + \sum_{k \neq j} \max(0, \rho_{kj}^l) \\
\phi_j^{l-1} &= \max_k (s_{jk}^l + \alpha_{jk}^l) \\
\alpha_{ij}^{l<L} &= \begin{cases} p_j^l + \phi_j^l + \sum_{k \neq j} \max(0, \rho_{kj}^l) & i = j \\ \min[0, p_j^l + \phi_j^l + \rho_{jj}^l + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj}^l)] & i \neq j \end{cases} \\
\alpha_{ij}^{L} &= \begin{cases} p_j^l + \sum_{k \neq j} \max(0, \rho_{kj}^l) & i = j \\ \min[0, p_j^l + \rho_{jj}^l + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj}^l)] & i \neq j \end{cases} \\
\rho_{ij}^{l>1} &= s_{ij}^l + \min[\tau_i^l, -\max_{k \neq j}(s_{ik}^l + \alpha_{ik}^l)] \\
\rho_{ij}^1 &= s_{ij}^l - \max_{k \neq j}(s_{ik}^l + \alpha_{ik}^l)
\end{aligned}
\tag{B.4}
$$

HAP (b):

$I^{l>1}$ change w.r.t HAP (a) affects ϕ^{l-1}, and $\eta^{l>1}$ that changes $\rho^{l>1}$:

$$\phi_j^{l-1} = \max[0, \alpha_{jj}^l - \max_{k\neq j}(s_{jk}^l + \alpha_{jk}^l)]$$
$$\rho_{ij}^{l>1} = \begin{cases} \min(0, \tau_i^l) - \max_{k\neq i}(s_{ik}^l + \alpha_{ik}^l) & i = j \\ s_{ij}^l + \min[\max(0, -\tau_i^l) - \alpha_{ii}^l, -\max_{k\notin\{i,j\}}(s_{ik}^l + \alpha_{ik}^l)] & i \neq j \end{cases} \tag{B.5}$$

HFL

E^l change w.r.t. HAP (a) affects α^l, and $\mu_{E^l\to e_j^l}$ that changes τ^{l+1}:

$$\tau_j^{l+1} = p_j^l + \sum_k \max(0, \rho_{kj}^l)$$
$$\alpha_{ij}^{l<L} = \min[0, p_j^l + \phi_j^l + \sum_{k\neq i} \max(0, \rho_{kj}^l)]$$
$$\alpha_{ij}^{L} = \min[0, p_j^l + \sum_{k\neq i} \max(0, \rho_{kj}^l)] \tag{B.6}$$

B.3 Full Derivation

We begin with two messages that occur in the explicit exemplar model for AP/FLAP graphs, and are of significance.

B.3.1 Message $\hat{p}_j^l$

$$\hat{p}_j^l(e_j = 1) = p_j^l + \phi^l$$
$$\hat{p}_j^l(e_j = 0) = 0$$

$$\boxed{\hat{p}_j^l = p_j^l + \phi^l \qquad (B.7)}$$

B.3.2 Message from E_j^l to e_j^l

HAP(a), HAP(b)

$$E_j^l(h_{:j}^l, e_j^l) = \begin{cases} 0 & \text{if } e_j^l = h_{jj}, e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}$$

The setting of $e_j^l = 1$ (j is an exemplar) constraints h_{jj}^l to be 1, the setting of $e_j^l = 0$ constraints all h_{kj}^l to be 0:

$$\begin{aligned}
\mu_{E_j^l \to e_j^l}(e_j^l = 1) &= \rho_{jj}^l(1) + \sum_{k \neq j} \max[\rho_{kj}^l(0), \rho_{kj}^l(1)] \\
\mu_{E_j^l \to e_j^l}(e_j^l = 0) &= \sum_k \rho_{kj}^l(0) = \rho_{jj}^l(0) + \sum_{k \neq j} \rho_{kj}^l(0) \\
\mu_{E_j^l \to e_j^l} &= \rho_{jj}^l + \sum_{k \neq j} (\max[\rho_{kj}^l(0), \rho_{kj}^l(1)] - \rho_{kj}^l(0)) \\
&= \rho_{jj}^l + \sum_{k \neq j} \max(0, \rho_{kj}^l)
\end{aligned}$$

$$\boxed{\mu_{E_j^l \to e_j^l} = \rho_{jj}^l + \sum_{k \neq j} \max(0, \rho_{kj}^l) \qquad \text{(B.8)}}$$

HFL

$$E_j^l(h_{:j}^l, e_j^l) = \begin{cases} 0 & \text{if } e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}$$

$$\mu_{E_j^l \to e_j^l}(e_j^l = 1) = \sum_k \max[\rho_{kj}^l(0), \rho_{kj}^l(1)]$$

$$\mu_{E_j^l \to e_j^l}(e_j^l = 0) = \sum_k \rho_{kj}^l(0)$$

$$\mu_{E_j^l \to e_j^l} = \sum_k [\max[\rho_{kj}^l(0), \rho_{kj}^l(1)] - \rho_{kj}^l(0)]$$

$$= \sum_k \max(0, \rho_{kj}^l)$$

$$\boxed{\mu_{E_j^l \to e_j^l} = \sum_k \max(0, \rho_{kj}^l) \qquad \text{(B.9)}}$$

Now we are ready to continue with the other messages.

B.3.3 Message from Level l to $l+1$ (τ_j^{l+1}):

We use Section B.3.2

HAP(a), HAP(b)

$$\boxed{\tau_j^{l+1} = p_j^l + \rho_{jj}^l + \sum_{k \neq j} \max(0, \rho_{kj}^l) \qquad \text{(B.10)}}$$

HFL

$$\boxed{\tau_j^{l+1} = p_j^l + \sum_k \max(0, \rho_{kj}^l) \qquad \text{(B.11)}}$$

B.3.4 Message from Level l to $l-1$ (ϕ_j^{l-1}):

HAP(a), HFL

$$I_i^{l>1}(h_{i:}^l, e_i^{l-1}) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases}$$

If $e_j^{l-1} = 1$ *i.e.* j is an exemplar in the previous level, the layer consistency constraints results in the usual 1-of-N optimization. If $e_j^{l-1} = 0$ *i.e.* j is a non-exemplar in the previous level, the result of the layer consistency constraint is that data point j should no longer be clustered at the next level.

$$\phi_j^{l-1}(e_j^{l-1} = 1) = \max_k [\beta_{jk}^l(1) + \sum_{t \neq k} \beta_{jt}^l(0)]$$

$$\phi_j^{l-1}(e_j^{l-1} = 0) = \sum_t \beta_{jt}^l(0)$$

$$\phi_j^{l-1} = \max_k(\beta_{jk}^l) = \max_k(s_{jk}^l + \alpha_{jk}^l)$$

$$\boxed{\phi_j^{l-1} = \max_k(s_{jk}^l + \alpha_{jk}^l) \qquad \text{(B.12)}}$$

HAP(b)

$$I_i^{l>1}(h_{i:}^l, e_i^{l-1}) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = 1, h_{ii}^l \leq e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases}$$

If $e_j^{l-1} = 1$ *i.e.* j is an exemplar in the previous level, the constraint results in the usual 1-of-N optimization. For clarity of derivation in the next step, we then rewrite the result by explicitly pulling out a specific setting of the optimization over the k cases:

$$\phi_j^{l-1}(e_j^{l-1}=1) = \max_k[\beta_{jk}^l(1) + \sum_{t\neq k}\beta_{jt}^l(0)]$$
$$= \max\Big[\beta_{jj}^l(0) + \max_{k\neq j}[\beta_{jk}^l(1) + \sum_{t\notin j,k}\beta_{jt}^l(0)], \beta_{jj}^l(1) + \sum_{t\neq j}\beta_{jt}^l(0)\Big]$$

If $e_j^{l-1}=0$ *i.e.* j is a non-exemplar in the previous level, the jth row in layer l must choose an exemplar that is NOT j in order to satisfy the constraint that $h_{jj}^l \leq e_j^{l-1}$:

$$\phi_j^{l-1}(e_j^{l-1}=0) = \beta_{jj}^l(0) + \max_{k\neq j}[\beta_{jk}^l(1) + \sum_{t\notin\{j,k\}}\beta_{jt}^l(0)]$$

Taking the difference, using $\max(x,y)-x = \max(0,y-x)$, we get

$$\phi_j^{l-1} = \max\Big[0, \beta_{jj}^l - \max_{k\neq j}[\beta_{jk}^l(1) + \sum_{t\notin j,k}\beta_{jt}^l(0)] + \sum_{t\neq j}\beta_{jt}^l(0)\Big]$$
$$= \max[0, \beta_{jj}^l - \max_{k\neq j}\beta_{jk}^l] = \max[0, (s_{jj}+\alpha_{jj}^l) - \max_{k\neq j}(s_{jk}^l + \alpha_{jk}^l)]$$
$$= \max[0, \alpha_{jj}^l - \max_{k\neq j}(s_{jk}^l + \alpha_{jk}^l)]$$

$$\boxed{\phi_j^{l-1} = \max[0, \alpha_{jj}^l - \max_{k\neq j}(s_{jk}^l + \alpha_{jk}^l)] \qquad \text{(B.13)}}$$

B.3.5 Message $\alpha_j^{l<L}$:

We use Section B.3.1

HAP(a), HAP(b)

$$E_j^l(h_{:j}^l, e_j^l) = \begin{cases} 0 & \text{if } e_j^l = h_{jj}, e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}$$

We need to differentiate between the cases h^l_{ij} and h^l_{jj}:

when $i \neq j$:

If $h^l_{ij} = 1$, then e^l_j must be also equal to 1, as well as h^l_{jj}. Denote the incoming message from node e^l_j into E^l_j by $\hat{p}^l_j$. All other incoming messages into E^l_j from the column variables are unconstrained.

$$\alpha^l_{ij}(h^l_{ij} = 1) = \hat{p}^l_j(1) + \rho^l_{jj}(1) + \sum_{k \notin \{i,j\}} \max[\rho^l_{ik}(0), \rho^l_{ik}(1)]$$
$$\alpha^l_{ij}(h^l_{ij} = 0) = \max\Big[\hat{p}^l_j(1) + \rho^l_{jj}(1) + \sum_{k \notin \{i,j\}} \max[\rho^l_{ik}(0), \rho^l_{ik}(1)], \hat{p}^l_j(0) + \sum_{k \neq j} \rho^l_{ik}(0)\Big]$$

Using the identity $x - \max(x, y) = \min(0, x - y)$ and $\hat{p}^l_j = p^l_j + \phi^l_j$ we get

$$\boxed{\alpha^l_{ij} = \min[0, p^l_j + \phi^l_j + \rho^l_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho^l_{kj})] \quad \text{(B.14)}}$$

when $i = j$:

$$\alpha^l_{jj}(h^l_{jj} = 1) = \hat{p}^l_j(1) + \sum_{k \neq j} \max[\rho^l_{ik}(0), \rho^l_{ik}(1)]$$
$$\alpha^l_{jj}(h^l_{jj} = 0) = \hat{p}^l_j(0) + \sum_{k \neq j} \rho^l_{ik}(0)$$

$$\boxed{\alpha^l_{jj} = p^l_j + \phi^l_j + \sum_{k \neq j} \max(0, \rho^l_{kj}) \qquad \text{(B.15)}}$$

HFL

$$E_j^l(h_{:j}^l, e_j^l) = \begin{cases} 0 & \text{if } e_j^l = \max_i h_{ij}^l, \\ -\infty & \text{otherwise.} \end{cases}$$

If $h_{ij}^l = 1$, then e_j^l must be also equal to 1. Denote the incoming message from node e_j^l into E_j^l by $\hat{p}_j^l$. All other incoming messages into E_j^l from the column variables are unconstrained.

$$\alpha_{ij}^l(h_{ij}^l = 1) = \hat{p}_j^l(1) + \sum_{k \neq i} \max[\rho_{kj}^l(0), \rho_{kj}^l(1)]$$

If $h_{ij}^l = 0$, then e_j^l is either 1 or 0. If it is 1, then as before. If it is 0, all h_{ij} must be also zero.

$$\alpha_{ij}^l(h_{ij}^l = 0) = \max\Big[\hat{p}_j^l(1) + \sum_{k \neq i} \max[\rho_{kj}^l(0), \rho_{kj}^l(1)], \hat{p}_j^l(0) + \sum_{k \neq i} \rho_{kj}^l(0)\Big]$$

Using the identity $x - \max(x, y) = \min(0, x - y)$, and $\hat{p}_j^l = p_j^l + \phi_j^l$

$$\boxed{\alpha_{ij}^l = \min[0, p_j^l + \phi_j^l + \sum_{k \neq i} \max(0, \rho_{kj}^l)] \qquad \text{(B.16)}}$$

B.3.6 Message $\rho^{l>1}$

In order to compute the messages, we need to first evaluate $\mu_{I_i^l \to h_{ij}^l} = \eta_{ij}^l$. Then, we use the identity $\rho_{ij}^l = s_{ij}^l + \eta_{ij}^l$, and $\beta_{ik}^l = s_{ik}^l + \alpha_{ik}^l$. As a reminder, in regular AP/FL, $\eta_{ij} = -\max_{k \neq j} \beta_{ik}$.

HAP(a), HFL

$$I_i^{l>1}(h_{i:}^l, e_i^{l-1}) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases}$$

For any h_{ij}^l to be 1, the constraint dictates it must be the case that $e_i^{l-1} = 1$ (i is an exemplar in the previous layer $l-1$).

$$\eta_{ij}^l(h_{ij}^l = 1) = \tau_i^l(1) + \sum_{k \neq j} \beta_{ik}^l(0)$$

If $h_{ij}^l = 0$, either $e_i^{l-1} = 0$ (i is a non-exemplar in the previous layer), and the layer consistency constraint is that data point i should no longer be associated with any exemplar at the current layer l, or $e_i^{l-1} = 1$, resulting in the usual 1-of-N optimization for the setting of the other h_{ik} variables.

$$\eta_{ij}^l(h_{ij}^l = 0) = \max\Big[\tau_i^l(0) + \sum_{k \neq j} \beta_{ik}^l(0), \tau_i^l(1) + \max_{k \neq j}[\beta_{ik}^l(1) + \sum_{t \notin \{k,j\}} \beta_{it}^l(0)]\Big]$$

We use the identity $x - \max(y, z) = \min(x - y, x - z)$ twice to get

$$\begin{aligned}\eta_{ij}^l &= \min\Big[\tau_i^l, \sum_{k \neq j} \beta_{ik}^l(0) - \max_{k \neq j}[\beta_{ik}^l(1) + \sum_{t \notin \{k,j\}} \beta_{it}^l(0)]\Big] \\ &\overset{(B.19)}{=} \min[\tau_i^l, -\max_{k \neq j} \beta_{ik}^l] = \min[\tau_i^l, -\max_{k \neq j}(s_{ik}^l + \alpha_{ik}^l)]\end{aligned}$$

Now we substitute the above for η in the expression for ρ to get

$$\boxed{\rho_{ij}^l = s_{ij}^l + \min[\tau_i^l, -\max_{k \neq j}(s_{ik}^l + \alpha_{ik}^l)] \qquad \text{(B.17)}}$$

HAP(b)

$$I_i^{l>1}(h_{i:}^l, e_i^{l-1}) = \begin{cases} 0 & \text{if } \sum_j h_{ij}^l = 1, h_{ii}^l \leq e_i^{l-1}, \\ -\infty & \text{otherwise.} \end{cases}$$

when $i = j$

If $h_{ii}^l = 1$ then the constraint dictates it must be the case that $e_i^{l-1} = 1$ (i is an exemplar in the previous layer $l-1$), and also that no other h_{ij} is equal to 1.

$$\eta_{ii}^l(h_{ii}^l = 1) = \tau_i^l(1) + \sum_{k \neq i} \beta_{ik}^l(0)$$

If $h_{ii}^l = 0$ then one of the other h_{ij}^l must equal 1, and it does not matter what the value of e_i^{l-1} is.

$$\eta_{ii}^l(h_{ii}^l = 0) = \max[\tau_i^l(1), \tau_i^l(0)] + \max_{k \neq i}[\beta_{ik}^l(1) + \sum_{t \notin \{k,i\}} \beta_{it}^l(0)]$$

Putting them together and using the standard identities we get:

$$\eta_{ii}^l = \min(0, \tau_i^l) - \max_{k \neq i} \beta_{ik}^l = \min(0, \tau_i^l) - \max_{k \neq i}(s_{ik}^l + \alpha_{ik}^l)$$

when $i \neq j$

If $h_{ij}^l = 1$, then $h_{ii}^l = 0$ due to the 1-of-N constraint, and therefore the second part of the constraint is always true, and it does not matter what the value of e_i^{l-1} is.

$$\eta_{ij}^l(h_{ij}^l = 1) = \max[\tau_i^l(1), \tau_i^l(0)] + \sum_{k \neq j} \beta_{ik}^l(0)$$

If $h_{ij} = 0$, then some other $h_{ik} = 1$. If it happens to be h_{ii}, then in order to satisfy the full constraint, $e_i^{l-1} = 1$. Otherwise, it does not matter what the value of e_i^{l-1} is. We must however not include the previous case in the summation and explicitly prevent it. Otherwise the maximization over τ may be the zero setting for e_i^{l-1}while that over $k \neq j$ may choose $\beta_{ii}^l(1)$ as the best setting, which will result in a constraint violation.

$$\eta_{ij}^l(h_{ij}^l = 0) = \max \begin{cases} \tau_i^l(1) + \beta_{ii}(1) + \sum_{k \notin \{i,j\}} \beta_{ik}^l(0) \\ \max[\tau_i^l(1), \tau_i^l(0)] + \beta_{ii}^l(0) + \max_{k \notin \{i,j\}} [\beta_{ik}^l(1) + \sum_{t \notin \{k,i,j\}} \beta_{it}^l(0)] \end{cases}$$

Putting them together, using $x - \max(y, z) = \min(x - y, x - z)$, we get

$$\eta_{ij}^l = \min \begin{cases} \max[\tau_i^l(1), \tau_i^l(0)] + \sum_{k \neq j} \beta_{ik}^l(0) - \tau_i^l(1) - \beta_{ii}(1) - \sum_{k \notin \{i,j\}} \beta_{ik}^l(0) \\ \max[\tau_i^l(1), \tau_i^l(0)] + \sum_{k \neq j} \beta_{ik}^l(0) - \max[\tau_i^l(1), \tau_i^l(0)] - \beta_{ii}^l(0) - \max_{k \notin \{i,j\}} [\beta_{ik}^l(1) + \sum_{t \notin \{k,i,j\}} \beta_{it}^l(0)] \end{cases}$$

Addressing each term separately the first subtraction simplifies to:

$$\max[\tau_i^l(1), \tau_i^l(0)] - \tau_i^l(1) + \sum_{k \neq j} \beta_{ik}^l(0) - \beta_{ii}^l(1) - \sum_{k \notin \{i,j\}} \beta_{ik}^l(0) = \max(0, -\tau_i^l) - \beta_{ii}^l$$

and the second subtraction simplifies to:

$$\sum_{k \neq j} \beta_{ik}^l(0) - \beta_{ii}^l(0) - \max_{k \notin \{i,j\}} [\beta_{ik}^l(1) + \sum_{t \notin \{k,i,j\}} \beta_{it}^l(0)] =$$
$$\sum_{k \notin \{i,j\}} \beta_{ik}^l(0) - \max_{k \notin \{i,j\}} [\beta_{ik}^l(1) + \sum_{t \notin \{k,i,j\}} \beta_{it}^l(0)] \overset{Eq.\ B.20}{=} - \max_{k \notin \{i,j\}} \beta_{ik}^l$$

Putting the two subtraction back together we finally get:

$$\begin{aligned}
\eta_{ij}^l &= \min[\max(0, -\tau_i^l) - \beta_{ii}^l, -\max_{k\notin\{i,j\}} \beta_{ik}^l] \\
&= \min[\max(0, -\tau_i^l) - (s_{ii}^l + \alpha_{ii}^l), -\max_{k\notin\{i,j\}} (s_{ik}^l + \alpha_{ik}^l)] \\
&= \min[\max(0, -\tau_i^l) - \alpha_{ii}^l, -\max_{k\notin\{i,j\}} (s_{ik}^l + \alpha_{ik}^l)]
\end{aligned}$$

Grouping the two cases and substituting them for η in the expression for ρ we get:

$$\boxed{\rho_{ij}^l = \begin{cases} \min(0, \tau_i^l) - \max_{k\neq i}(s_{ik}^l + \alpha_{ik}^l) & i = j \\ s_{ij}^l + \min[\max(0, -\tau_i^l) - \alpha_{ii}^l, -\max_{k\notin\{i,j\}} (s_{ik}^l + \alpha_{ik}^l)] & i \neq j \end{cases} \quad \text{(B.18)}}$$

$$\begin{aligned}
&\sum_{k\neq j} \beta_{ik}^l(0) - \max_{k\neq j}[\beta_{ik}^l(1) + \sum_{t\notin\{k,j\}} \beta_{it}^l(0)] = -\Big[\max_{k\neq j}[\beta_{ik}^l(1) + \sum_{t\notin\{k,j\}} \beta_{it}^l(0)] - \sum_{k\neq j} \beta_{ik}^l(0)\Big] \\
&= -\Big[\max_{k\neq j}[\beta_{ik}^l(1) + \sum_{t\notin\{k,j\}} \beta_{it}^l(0)] - \sum_{t\neq j} \beta_{it}^l(0)\Big] = -\max_{k\neq j}[\beta_{ik}^l(1) + \sum_{t\notin\{k,j\}} \beta_{it}^l(0) - \sum_{t\neq j} \beta_{it}^l(0)] \\
&= -\max_{k\neq j}[\beta_{ik}^l(1) - \beta_{ik}^l(0)] = -\max_{k\neq j} \beta_{ik}^l \qquad \text{(B.19)}
\end{aligned}$$

$$\begin{aligned}
&\sum_{k\notin\{i,j\}} \beta_{ik}^l(0) - \max_{k\notin\{i,j\}}[\beta_{ik}^l(1) + \sum_{t\notin\{k,i,j\}} \beta_{it}^l(0)] = -\Big[\max_{k\notin\{i,j\}}[\beta_{ik}^l(1) + \sum_{t\notin\{k,i,j\}} \beta_{it}^l(0)] - \sum_{k\notin\{i,j\}} \beta_{ik}^l(0)\Big] \\
&= -\Big[\max_{k\notin\{i,j\}}[\beta_{ik}^l(1) + \sum_{t\notin\{k,i,j\}} \beta_{it}^l(0)] - \sum_{t\notin\{i,j\}} \beta_{it}^l(0)\Big] = -\max_{k\notin\{i,j\}}[\beta_{ik}^l(1) + \sum_{t\notin\{k,i,j\}} \beta_{it}^l(0) - \sum_{t\notin\{i,j\}} \beta_{it}^l(0)] \\
&= -\max_{k\notin\{i,j\}}[\beta_{ik}^l(1) - \beta_{ik}^l(0)] = -\max_{k\notin\{i,j\}} \beta_{ik}^l \qquad \text{(B.20)}
\end{aligned}$$

Appendix C

Step-size Selection in Dual Decomposition

In this appendix, we describe five step-size methods for dual decomposition. We test the performance of these different methods for dual decomposition based exemplar-based clustering. Results are shown on the toy data set as well as the 'faces-400' data set that are described in Chapter 8.

It is important to note that unlike block coordinate descent methods such as max-product linear programming, in dual decomposition the value of the dual objective function (Eq. 2.21) is not guaranteed to monotonically decrease. The quantity that does decrease at every iteration is the Euclidean distance of the current dual solution to the set of optimal solutions. The theoretical guarantees of dual-decomposition are that the procedure eventually converges to the dual optimal, if the step-size parameter is chosen to be of a particular form.

We summarize some of the step-size conditions given in (Komodakis et al., 2010). Specifically, if the sequence of multipliers, $\{\alpha^t\}$, satisfies any of the following three conditions, then the sub-gradient algorithm converges to the optimal solution of Eq. 2.21:

1. Diminishing step-size condition:

$$\alpha^t \geq 0, \quad \lim_{t\to\infty} \alpha^t = 0, \quad \sum_{t=0}^{\infty} \alpha^t = \infty. \tag{C.1}$$

2. Non-summable diminishing step size condition:

$$\beta^t \geq 0, \quad \lim_{t\to\infty} \beta^t = 0, \quad \sum_{t=0}^{\infty} \beta^t = \infty, \tag{C.2}$$

where $\alpha^t = \frac{\beta^t}{\|g^t\|_2}$, and g^t is the sub-gradient at the t-th iteration.

3. Square summable but not summable condition

$$\alpha^t \geq 0, \quad \lim_{t\to\infty} (\alpha^t)^2 = 0, \quad \sum_{t=0}^{\infty} \alpha^t = \infty. \tag{C.3}$$

Typical choices for Eq. C.1 include setting $\alpha^t = \frac{a}{\sqrt{t}}$ for some $a \geq 0$, and for Eq. C.3, a typical choice is setting $\alpha^t = \frac{a}{b+t}$, where $a > 0, b \geq 0$. An alternative type of step-size rules are adaptive (or dynamic) step-sizes, that do not depend on pre-determined quantities. These are typically preferred in practice, as they often lead to faster convergence rates. The rule is of the form

$$\alpha^t = \gamma^t \frac{DUAL^t - APPROX^t}{\|g^t\|_2^2}, \tag{C.4}$$

where $DUAL^t$ is the current value of the dual objective, $APPROX^t$ is an approximation to the unknown optimal dual value, and γ^t is a positive scalar that is allowed to vary per iteration and typically takes the value in the interval (0,2). Computing $APPROX^t$ can be done in several ways. One approach is to set

$$APPROX^t = MaxPrimal^t, \tag{C.5}$$

the maximal primal objective value obtained so far up to iteration t, since it is a lower bound for $DUAL^t$. That will imply smaller step-sizes as we get closer to the dual

optimum, where the primal-dual gap is reduced to its minimal value. An alternative approach is to use a lower bound for the optimal dual value using $MinDual^t$, the minimal dual objective value obtained so far. This update is given by

$$APPROX^t = MinDual^t - \Delta^t, \tag{C.6}$$

where Δ^{t+1} is given by

$$\Delta_{t+1} = \begin{cases} \rho_0 \Delta^t & DUAL^{t+1} \leq APPROX^t, \\ \max[\rho_1 \Delta^t, \Delta] & \text{otherwise.} \end{cases} \tag{C.7}$$

and $\rho_0 \geq 1, \rho_1 \leq 1, \Delta, \Delta_0$ are fixed positive constants.

The intuition for this update is quite simple: we imagine, at every step, that the true dual minimum is some Δ^t away from the so far seen minimal dual. To update our expected potential improvement, at each step we examine our newly obtained dual value. If we exceeded our former expectations (the first option condition, $DUAL_{t+1} \leq APPROX^t$, implies that $\delta^t \leq MinDual^t - DUAL_{t+1}$), we will update our $MinDual_{t+1}$ to the new dual value, and, if we chose $\rho_0 > 1$, we will also make our expectations for impartment less humble. If, on the other hand, we have not met our expectations for improvement, we decrease them, but never to be less than some predetermined baseline improvement level, Δ.

Interestingly, there also exist heuristic adaptive schemes, which come with no convergence guarantees, that are also known to give faster performance in practice. Overall, no single step-size type is known to always be superior.

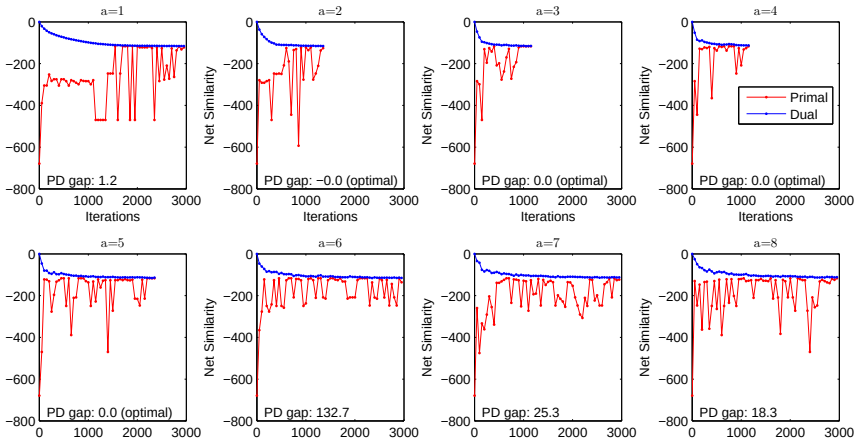
Thus, the step-size parameter α^t is a free parameter that must be investigated for the particular algorithm we describe in Chapter 7. We now compare the five provably convergent strategies described above and discuss the choice made for the experiments in Chapter 8.

In the following sets of figures, we show primal-dual values across iterations, as well as a color-coded visualization of the final solution (*e.g.* Fig. C.1(a), and Fig. C.1(b)) respectively). In the primal-dual plots, the x-axis indicates the number of iterations, where we only plot points every 50 iterations. The y-axis indicates the net similarity of the current intermediate solution. That is, at every iteration we decode to find a primal solution whose value is then compared to that of the current dual solution. We decode by summing the dual variables for every point. This sum indicates to what extent a points wants to be an exemplar. The points that prefer being an exemplar are deemed as such, and all other points are assigned to them based on how near they are, (not based on their dual message values). Optimality of the solution can be asserted if all slave problems agree on the maximizing configuration, and is indicated on the plot. The primal-dual gap is another measure that can be calculated and is indicated on the plots.

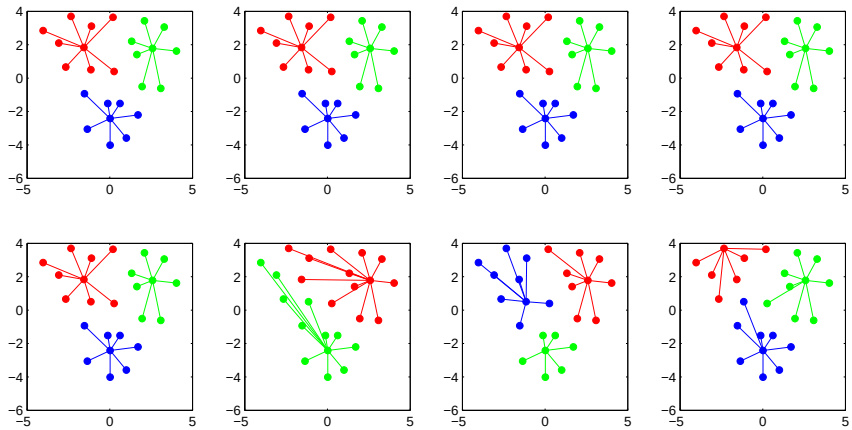
C.1 Using static update Eq. C.1

In this set of experiments, we look at different settings of a in the expression for the step-size, $\alpha^t = \frac{a}{\sqrt{t}}$. Results are shown in Fig. C.1(a). In most runs the solution is optimal, or very close to optimal based on the primal-dual gap. In Fig. C.1(b) we show the actual clustering results obtained by the different settings. As can be seen in the clustering results, for the setting of $a = 1$ (the first subplot in Fig. C.1(a), top-left corner in figure) we obtain a clustering that is identical to the clustering for which we are guaranteed optimality in other runs. Yet, the run itself is not optimal as far as primal-dual gap, and certificate of optimality based on slave agreement. It is also interesting to note that in all those runs but one the algorithm discovers three clusters.

Running the same set of experiments with values of a that are one order of magnitude above or below the setting of Fig. C.1(a) shows that a good choice of a is quite crucial for the algorithm to give sensible results. Such deviations from what we find to be a good



(a) Primal and dual objective function values during run.



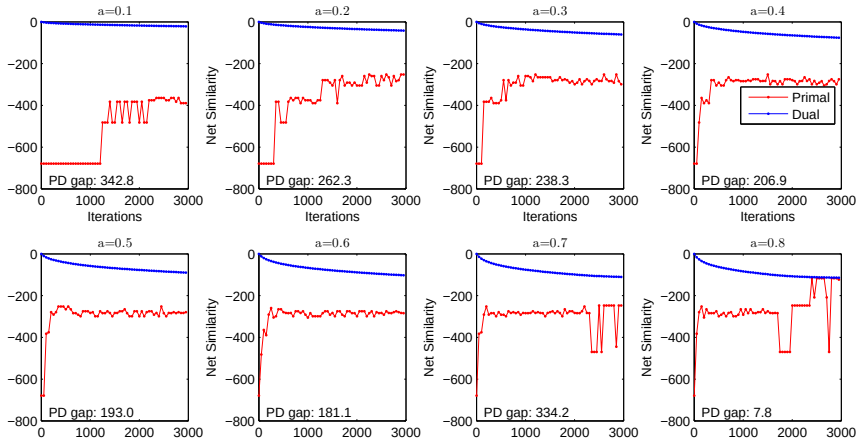
(b) Final clustering results for run.

Figure C.1: 2D toy data results for step-size of the form $\alpha^t = \frac{a}{\sqrt{t}}$ (Eq. C.1), using different values of a.

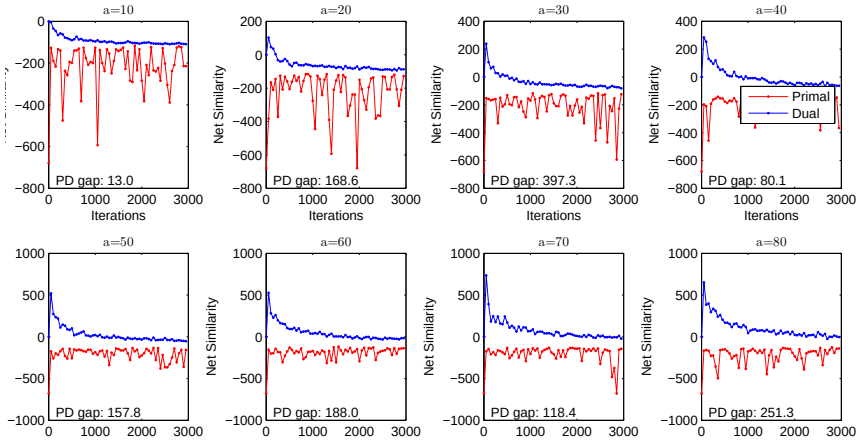
settings result in non-converging runs with large primal dual gap, as can be shown seen in Fig. C.2(a) and Fig. C.2(b).

C.2 Using static update Eq. C.2

In this set of experiments, we look at different settings of a in the expression for the step-size, $\alpha^t = \frac{a}{\sqrt{t}\|g^t\|_2}$. Results are shown in Fig. C.3. As can be seen, the choice of the free-parameter greatly influences the convergence speed of the algorithm, and all runs require more than 3000 iterations to converge. In comparison, the minimal number



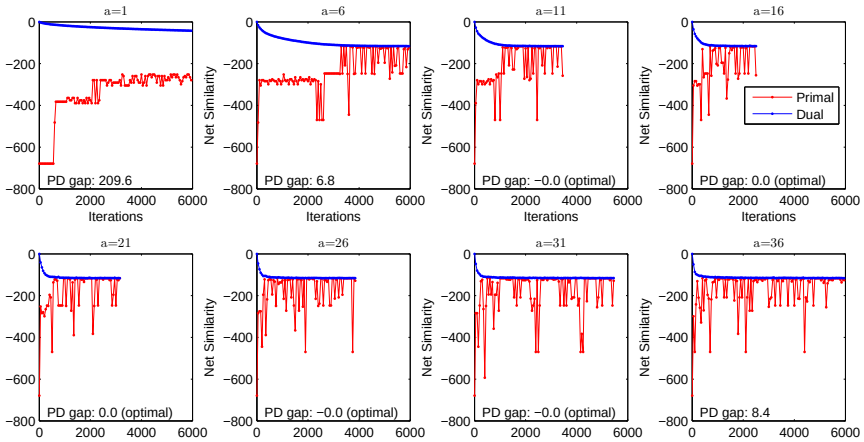
(a) Primal and dual objective function values during run, $a = 0.1 - 0.8$



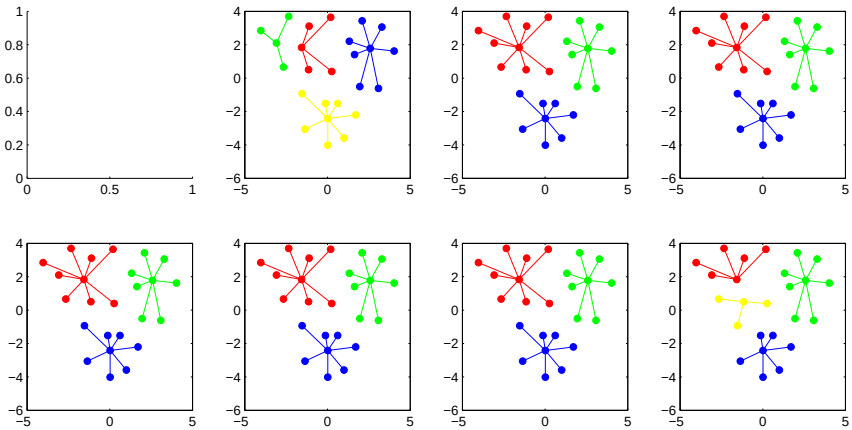
(b) Primal and dual objective function values during run, $a = 10 - 80$

Figure C.2: 2D toy data results for step-size of the form $\alpha^t = \frac{a}{\sqrt{t}}$ (Eq. C.1), using different values of a. Here, showing primal and dual objective function values during run for non-converging settings of a.

of iterations for the method based on Eq. C.1 was considerably better, with ~ 1500 iterations. Notably, the solutions here appear to contain either three or four clusters in comparison to the previous method where either two or three clusters were typically found.



(a) Primal and dual objective function values during run.



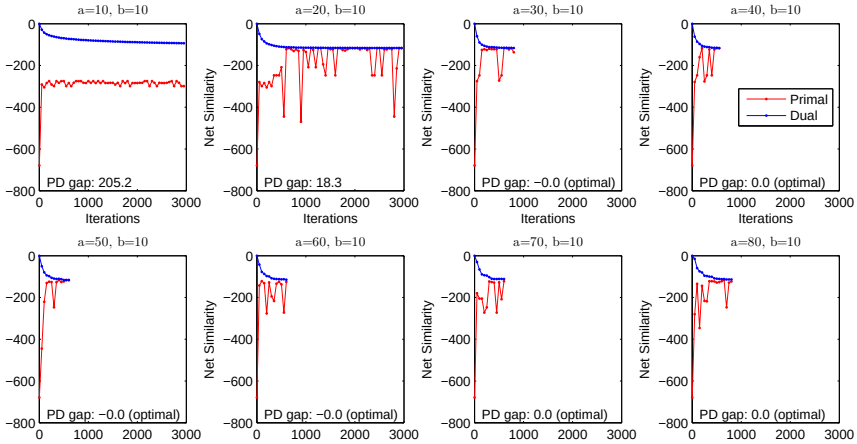
(b) Final clustering results for run. Empty plot indicates no exemplars decoded for run.

Figure C.3: 2D toy data results for step-size of the form $\alpha^t = \frac{a}{\sqrt{t}\|g^t\|_2}$ (Eq. C.2), using different values of a.

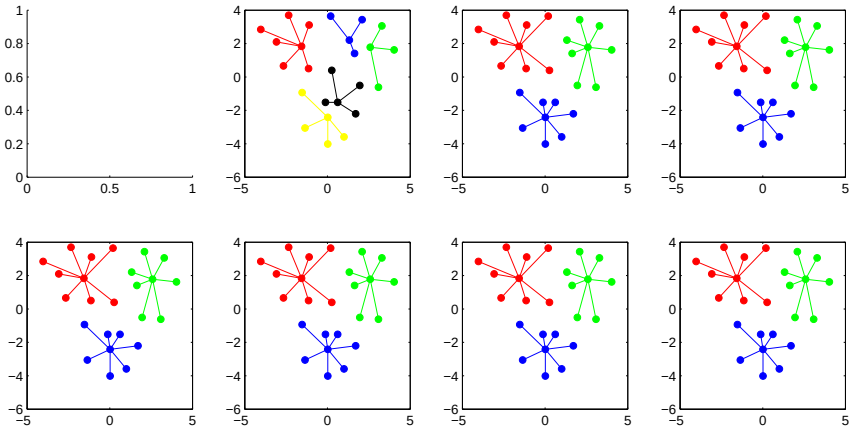
C.3 Using static update Eq. C.3

In this set of experiments, we look at different settings of a and b in the expression for the step-size, $\alpha^t = \frac{a}{b+t}$. The first observation is that for a range of a values that leads to good results (which, in this case, is different from that of the similar step-size of Eq. C.1 by an order of magnitude) the value of the b parameter seems to matter relatively little, and we get similar results for $b = 0$ (Fig. C.4) as well as $b = 10, 100$ (not shown). We also observe that for this step-size type that it appears to have the fastest convergence among all tested step-size updates, when the parameters are set to suitable values. Some

runs terminate with an optimal solution after less than 500 iterations. However, when a



(a) Primal and dual objective function values during run.



(b) Final clustering results for run.

Figure C.4: 2D toy data results for step-size of the form $\alpha^t = \frac{a}{b+t}$ (Eq. C.3), using different values of a and with $b = 10$.

is not set to the right value, deviations will lead to non-converging results regardless of the value of b. While the plot indicate that for some runs the primal dual gap is slowly decreasing, it does so at a slow rate (Fig. C.5(a)-Fig. C.5(c)).

C.4 Using dynamic update Eq. C.4 with Eq. C.5

In this set of experiments, we look at different settings of γ in the expression for the step-size, $\alpha^t = \gamma^t \frac{DUAL^t - APPROX^t}{\|g^t\|_2^2}$. We note that this particular step-size requires careful

adjustment of the γ^t parameter. While it is specified that this parameter should be in the range $(0, 2)$, the values for which we found it to converge to optimality is between 0.014 and 0.02 (where it was set to that value for every t). As can be seen in Fig. C.6(a), convergence requires almost 6000 iterations.

C.5 Using dynamic update Eq. C.4 with Eq. C.6

The last type of step-size is defined by $\alpha^t = \gamma^t \frac{DUAL^t - APPROX^t}{\|g^t\|_2^2}$, where $APPROX^t = MinDual^t - \Delta^t$, and Δ^t requires four tuneable parameters to be set (see Eq. C.6), so that together with the γ^t parameter that we fix throughout any given experiment to be some γ, this method depends on five different tuneable parameters. Through various experimentations with values for all five parameters, we note it is quite difficult to find a convergent, optimal configuration even for the simple case of the toy data. The algorithm is sensitive to changes in the parameters, and optimizing them poses a meta-problem that does not have a simple, systematic solution. Of all the various settings we have investigated, we report the setting that resulted in the best performance overall. We fix the setting of all parameters and only report results for ranging over values of γ in Fig. C.7(a)). Even for this carefully hand-chosen configuration, none of the settings converged to a provably optimal solution, in terms of full agreement among the slave problems, or equivalently a zero primal-dual gap. However, the **decoded** result for the configuration, using our standard the decoding scheme of Eq. 7.19 does in fact give the correct solution in one particular setting of the γ parameter shown, despite having a non-zero primal-dual gap (see left-most plot in the second row in Fig. C.7(b)).

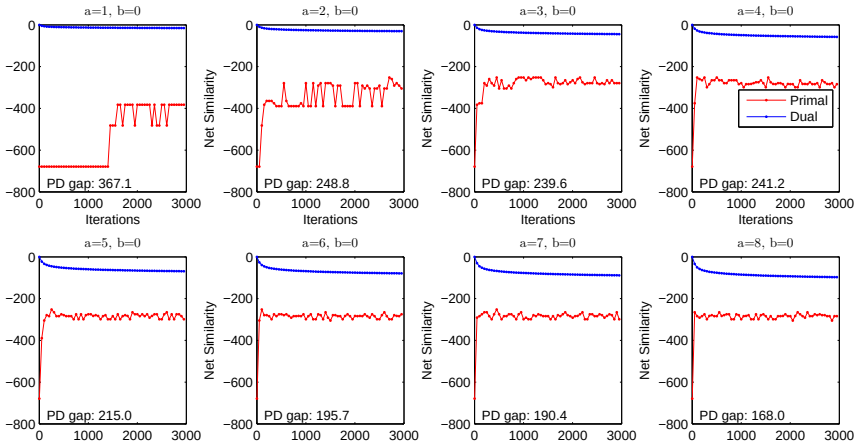
C.6 Step-size choice

The set of experiments in this section demonstrates the inherent difficulty of choosing an appropriate step-size rule, and parameters. The most successful approach in terms of

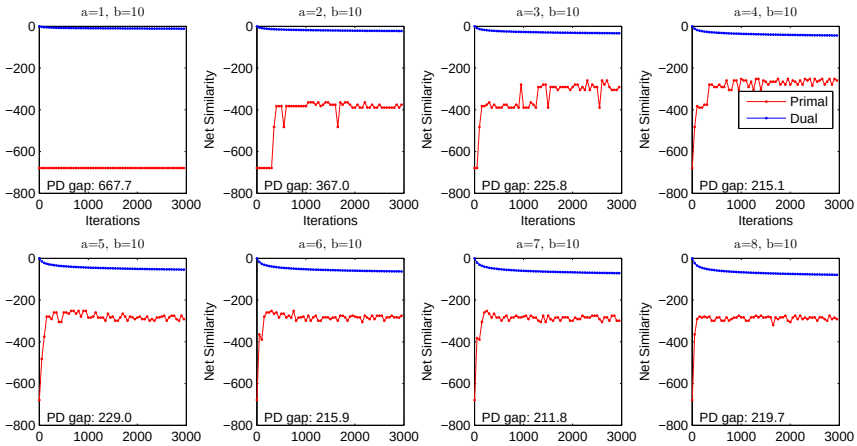
convergence and resiliency is that of Eq. C.3. Given a proper setting of its more dominate parameter a, it is quite robust to changes in the value of the secondary parameter, b.

AS a last experiment, in order to verify that this step-size generalizes in a straightforward way to other data sets, we run a set of experiments on the 'faces' data set[1] for a description of all real-world data sets. As can be seen in figure Fig. C.8, there's a range for which the parameter value a leads to nearly converging results, although complete convergence may require additional iterations, or finer tuning of a. As was observed in the toy-data experiments, changes to b (experiments not shown) reproduce considerably similar results.

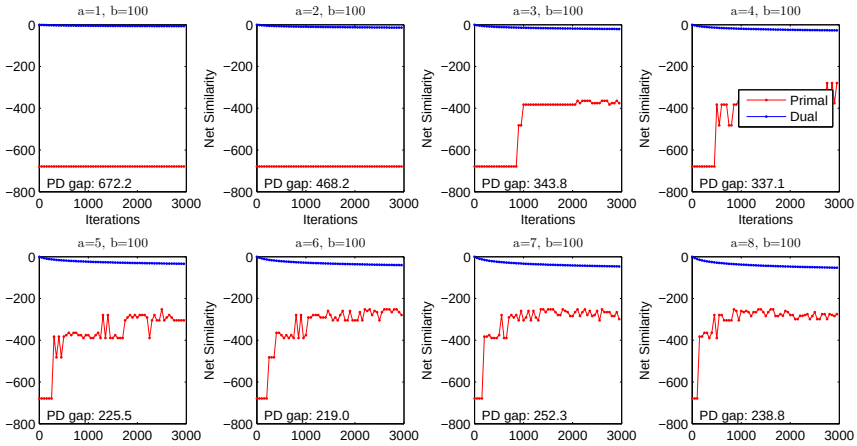
[1]see Chapter 8 for details regarding the real-world data sets.



(a) Primal and dual objective function values during run, $b = 0$

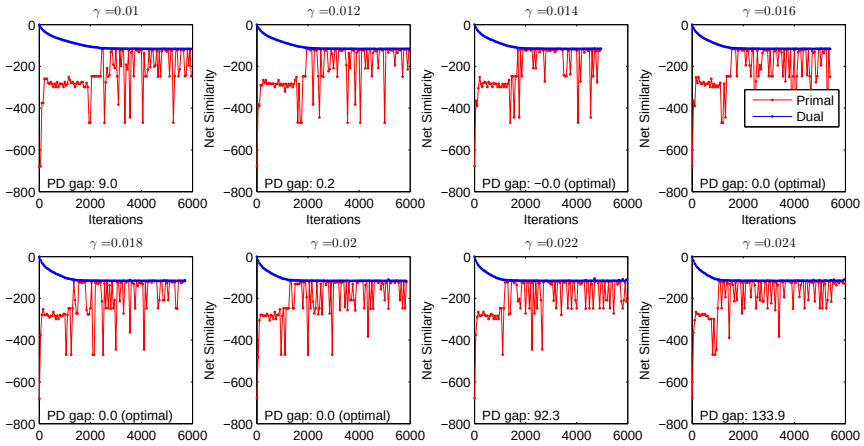


(b) Primal and dual objective function values during run, $b = 10$

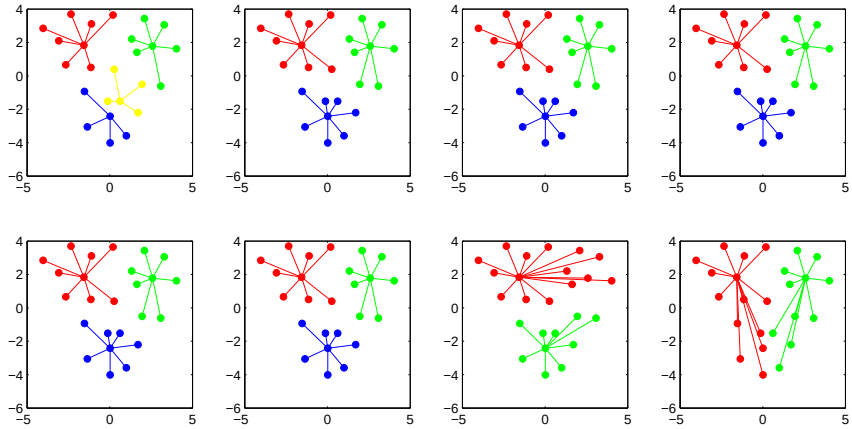


(c) Primal and dual objective function values during run, $b = 100$

Figure C.5: 2D toy data results for step-size of the form $\alpha^t = \frac{a}{b+t}$ (Eq. C.3), using different values of a and b. Here, showing primal and dual objective function values during run for non-converging settings of a. The choice of b does not appear to influence the non-convergence properties of the parameter choice.

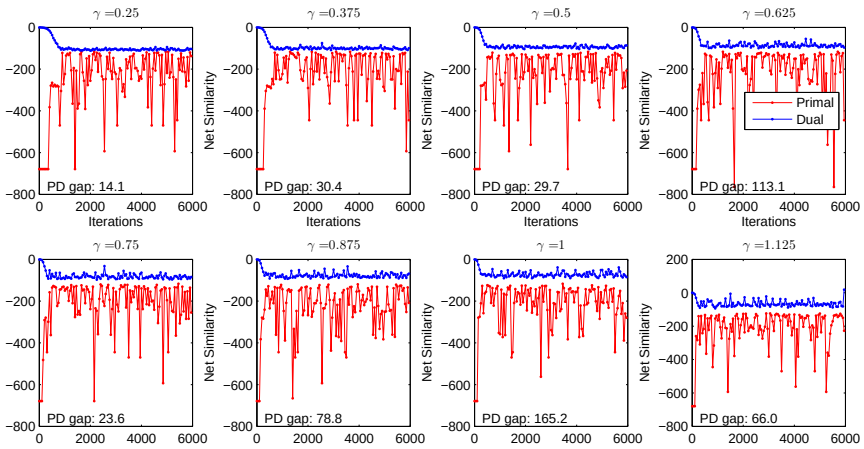


(a) Primal and dual objective function values during run.

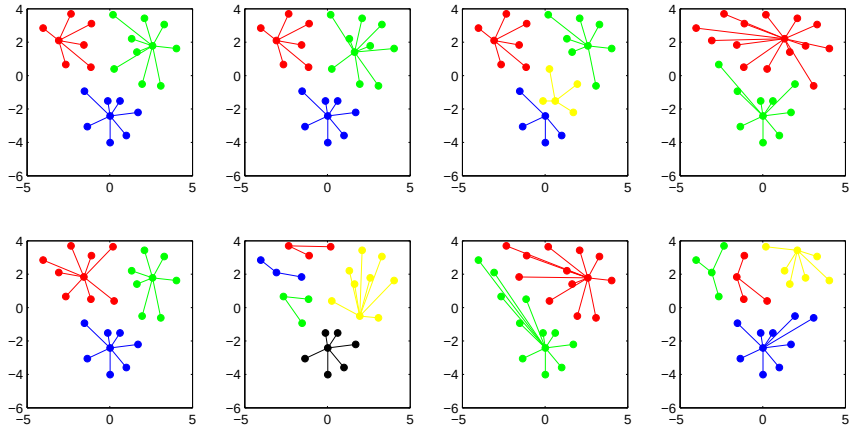


(b) Final clustering results for run.

Figure C.6: 2D toy data results for step-size of the form $\alpha^t = \gamma^t \frac{DUAL^t - APPROX^t}{\|g^t\|_2^2}$ (Eq. C.4), with $APPROX^t = MaxPrimal^t$ (Eq. C.5), using different values of $\gamma^t = \gamma$.



(a) Primal and dual objective function values during run.



(b) Final clustering results for run.

Figure C.7: 2D toy data results for step-size of the form $\alpha^t = \gamma^t \frac{DUAL^t - APPROX^t}{\|g^t\|_2^2}$ (Eq. C.4), with $APPROX^t = MinDual^t - \Delta^t$, Δ^t determined by Eq. C.6, using different values of γ.

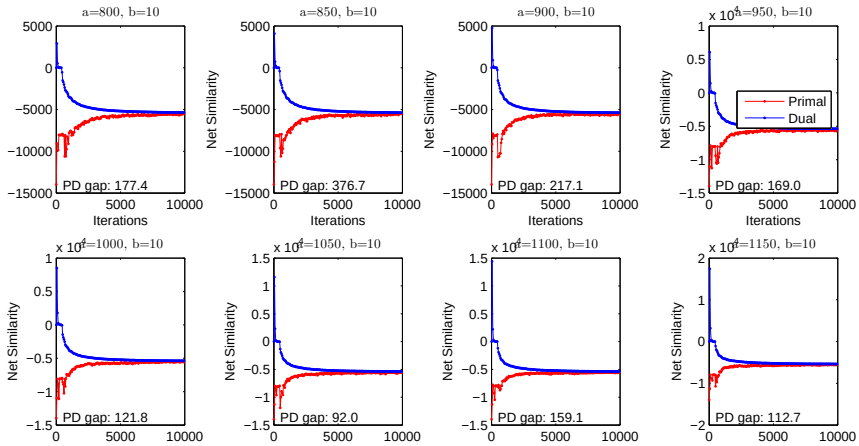


Figure C.8: Faces data results for step-size of the form $\alpha^t = \frac{a}{b+t}$ (Eq. C.2), using different values of a.

Bibliography

Abdelkader, M. F., Abd-Almageed, W., Srivastava, A. & Chellappa, R. (2011). Silhouette-based gesture and action recognition via modeling trajectories on riemannian shape manifolds. *Computer Vision and Image Understanding*, *115(3)*, 439 – 455.

Balinksi, M. L. (1966). On finding integer solutions to linear programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*.

Bar-ilan, J., Kortzars, G. & Peleg, D. (1992). How to allocate network centers. *Journal of Algorithms*, *15*, 385–415.

Basu, S., Bilenko, M. & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Bateni, M. & Hajiaghayi, M. (2009). Assignment problem in content distribution networks: unsplittable hard-capacitated facility location. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Bayati, M., Shah, D. & Sharma, M. (2005). Maximum weight matching via max-product belief propagation. In *Proceedings of the International Symposium on Information Theory (ISIT)*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Boykov, Y. Y. & Jolly, M. P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Charikar, M., Guhay, S., Tardosz, E. & Shmoys, D. B. (2002). A constant-factor approximation algorithm for the k-median problem. *Journal of Computer System Sciences*, *65*, 129–149.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001). *Introduction to Algorithms, Second Edition*. The MIT Press.

Delong, A., Osokin, A., Isack, H. N. & Boykov, Y. (2010). Fast approximate energy minimization with label costs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *39(1)*, 1–38.

Donoser, M., Urschler, M., Hirzer, M. & Bischof, H. (2009). Saliency driven total variation segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Drezner, Z. & Hamacher, H. (1995). *Facility Location. A Survey of Applications and Methods*. Springer Series in Operations Research. Springer-Verlag.

Duchi, J., Tarlow, D., Elidan, G. & Koller, D. (2006). Using combinatorial optimization within max-product belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*.

Duda, R. O. & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. New Yotk: John Willey & Sons.

Dueck, D. (2009). *Affinity Propagation: Clustering Data By Passing Messages*. PhD thesis, University of Toronto.

Dueck, D. & Frey, B. J. (2007). Non-metric affinity propagation for unsupervised image categorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Dueck, D., Frey, B. J., Jojic, N., Jojic, V., Giaever, G., Emili, A., Musso, G. & Hegele, R. (2008). Constructing treatment portfolios using affinity propagation. In *Proceedings of the Annual International Conference on Computational Biology (RECOMB)*.

Eng, J., McCormack, A. & Yates, J. (1994). An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society of Mass Spectrometry*, *5(11)*, 976–989.

Frank, C. & Römer, K. (2007). Distributed facility location algorithms for flexible configuration of wireless sensor networks. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*.

Frey, B. J. & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *305(5814)*, 972 – 976.

Frey, B. J. & Dueck, D. (2008). Technical letter - clustering by passing messages between data points. *Science*, *319(5864)*, 726.

Frey, B. J. & MacKay, D. J. C. (1997). A revolution: Belief propagation in graphs with cycles. In *Advances In Neural Information Processing Systems (NIPS)*.

Gallager, R. G. (1963). *Low Density Parity Check Codes*. The MIT Press.

Gavrila, D. M. (2007). A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 1408–1421.

Givoni, I. E., Chung, C. & Frey, B. J. (2011). Hierarchical affinity propagation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Givoni, I. E. & Frey, B. J. (2009a). A binary variable model for affinity propagation. *Neural Computation*, *21(6)*, 1589–1600.

Givoni, I. E. & Frey, B. J. (2009b). Semi-supervised affinity propagation with instance-level constraints. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Globerson, A. & Jaakkola, T. (2007). Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in Neural Information Processing Systems (NIPS)*.

Goemans, M. X. & Skutella, M. (2000). Cooperative facility location games. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Goudsmit, J. (1997). *Viral Sex; The Nature of AIDS*. Oxford University Press.

Şahin Güvenç & Haldun, S. (2007). A review of hierarchical facility location models. *Computers & Operations Research*, *34(8)*, 2310–2331.

Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, *12(3)*, 450–459.

Hazan, T. & Shashua, A. (2010). Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*.

Hong, R., Tang, J., Zha, Z.-J., Luo, Z. & Chua, T.-S. (2010). Mediapedia: Mining web knowledge to construct multimedia encyclopedia. In *Advances in Multimedia Modeling*, Lecture Notes in Computer Science.

Huang, B. & Jebara, T. (2007). Loopy belief propagation for bipartite maximum weight b-matching. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Jain, S. & Neal, R. (2000). A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, *13*, 158–182.

Jardine, N. & Sibson, R. (1968). The construction of hierarchic and non-hierarchic classifications. *The Computer Journal*, *11(177)*.

Ji Liu, L. L. (2011). Network community detection based on co-neighbor modularity matrix with spectral clustering. *Applied Mechanics and Materials*, *55-57*, 1237–1241.

Jia, C., Jiang, Y. & Yu, J. (2010). Affinity propagation on identifying communities in social and biological networks. In *Knowledge Science, Engineering and Management*, Volume 6291 of *Lecture Notes in Computer Science* (pp. 597–602).

Jojic, V. (2011). Personal communication.

Jojic, V., Gould, S. & Koller, D. (2010). Accelerated dual decomposition for MAP inference. In *Proceedings of the International Conference on Machine learning (ICML)*.

Kalman, Rudolph & Emil (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, *82(Series D)*, 35–45.

Kang, J. H., Lerman, K. & Plangprasopchok, A. (2010). Analyzing microblogs with affinity propagation. In *Proceedings of the First Workshop on Social Media Analytics*.

Kang, Y. & Choi, S. (2009). Common neighborhood sub-graph density as a similarity measure for community detection. In *Neural Information Processing*, Volume 5863 of *Lecture Notes in Computer Science* (pp. 175–184).

Kantor, E. & Peleg, D. (2009). Approximate hierarchical facility location and applications to the bounded depth steiner tree and range assignment problems. *Journal of Discrete Algorithms*, *7(3)*, 341–362.

Kariv, O. & Hakimi, S. L. (1979). An algorithmic approach to network location problems. ii: The p-medians. *Siam Journal on Applied Mathematics*, *37*.

Kazantseva, A. & Szpakowicz, S. (2011). Linear text segmentation using affinity propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Keller, A., Nesvizhskii, A., Kolker, E. & Aebersold, R. (2002). Empirical statistical model to estimate the accuracy of peptide identifications made by ms/ms and database search. *Analytical Chemistry*, *74*, 5383–5392.

Kent, C., Azanchi, R., Smith, B., Chu, A. & Levine, J. (2007). A model-based analysis of chemical and temporal patterns of cuticular hydrocarbons in male drosophila melanogaster. *PLoS ONE*, *2(9)*, e962.

Kindermann, R. & Snell, J. L. (1980). *Markov Random Fields and Their Applications*. AMS.

Klein, D., Kamvar, S. D. & Manning, C. D. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the International Conference on Machine learning (ICML)*.

Kohli, P., Kumar, M. P. & Torr, P. H. S. (2009a). p^3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (pp. 1645–1656).

Kohli, P., Ladicky, L. & Torr, P. H. S. (2009b). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, *82(3)*, 302–324.

Koller, D. & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28(10)*, 1568–1583.

Kolmogorov, V. & Wainwright, M. (2005). On the optimality of tree-reweighted max-product message passing. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Kolmogorov, V. & Zabih, R. (2004). What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 65–81.

Komodakis, N. & Paragios, N. (2009). Beyond pairwise energies: Efficient optimization for higher-order mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Komodakis, N., Paragios, N. & Tziritas, G. (2007). Mrf optimization via dual decomposition: Message-passing revisited. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Komodakis, N., Paragios, N. & Tziritas, G. (2010). Mrf energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Krivitski, D., Schuster, A. & Wolff, R. (2005). A local facility location algorithm for sensor networks. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*.

Kschischang, F., Frey, B. J. & Loeliger, H.-A. (2001). Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, *47(2)*, 498 – 519.

Kuhn, H. (1955). The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, *(2)*, 83–97.

Kulesza, A. & Pereira, F. (2008). Structured learning with approximate inference. In *Advances in Neural Information Processing Systems (NIPS)*.

Ladicky, L., Russell, C., Kohli, P. & Torr, P. H. S. (2010). Graph cut based inference with co-occurrence statistics. In *Proceedings of the European conference on Computer vision (ECCV)*.

Lai, D., Nardini, C. & Lu, H. (2011). Partitioning networks into communities by message passing. *Physics Review E*, *83(1)*, 016115.

Landsness, E. C., Crupi, D., Hulse, B. K., Peterson, M. J., Huber, R., Ansari, H., Coen, M., Cirelli, C., Benca, R. M., Ghilardi, M. F. & Tononi, G. (2009). Sleep-dependent improvement in visuomotor learning: A causal role for slow waves. *Sleep*, *32(10)*, 1273–1284.

Lazic, N. (2011). *Message Passing Algorithms for Facility Location Problems*. PhD thesis, University of Toronto.

Lazic, N., Frey, B. & Aarabi, P. (2010). Solving the uncapacitated facility location problem using message passing algorithms. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Lazic, N., Givoni, I. E., Aarabi, P. & Frey, B. J. (2009). Floss: Facility location for subspace segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Lee, I., Date, S. V., Adai, A. T. & Marcotte, E. (2004). A probabilistic functional network of yeast genes. *Science*, *30(5701)*, 1555 – 1558.

Leone, M., Sumedha & Weigt, M. (2008). Unspervised and semi-supervised clustering by message passing: soft constraint affinity propagation. *Jounral of European Physics B*.

Levi, R. & Shmoys, D. B. (2004). Lp-based approximation algorithms for capacitated facility location. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Lindorff-Larsen, K. & Ferkinghoff-Borg, J. (2009). Similarity measures for protein ensembles. *PLoS ONE*, *4(1)*, e4203+.

Lu, P., Vogel, C., Wang, R., Yao, X. & Marcotte, E. (2006). Absolute protein expression profiling estimates the relative contributions of transcriptional and translational regulation. *Nature Biotechnology*, *25*, 117 – 124.

von Luxburg, U. (2007). A tutorial on spectral clustering.

Manjeshwar, A. & Agrawal, D. P. (2002). Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of The IEEE International Parallel and Distributed Processing Symposium (IPDPS)*.

McLachlan, G. J. & Krishnan, T. (2008). *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics)* (2 Ed.). Wiley-Interscience.

Meltzer, T., Globerson, A. & Weiss, Y. (2009). Convergent message passing algorithms - a unifying view. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Meltzer, T., Yanover, C. & Weiss, Y. (2005). Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Monzy (2005). "So Much Drama in the PhD" song lyrics.

Mori, G. (2005). Guiding model search using segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Mulvey, J. M. & Beck, M. P. (1984). Solving capacitated clustering problems. *European Journal of Operations Research*, *18(3)*, 339–348.

Murphy, K. P., Weiss, Y. & Jordan, M. I. (1999). Loopy Belief Propagation for Approximate Inference: An Empricial Study. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Nedic, A. & Bertsekas, D. P. (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, *12*, 109–138.

Negreiros, M. & Palhano, A. (2006). The capacitated centred clustering problem. *Computers and Operations Research*, *33(6)*, 1639–1663.

Nesvizhskii, A., Keller, A., Kolker, E. & Aebersold, R. (2003). A statistical model for identifying proteins by tandem mass spectrometry. *Analytical Chemistry*, *75*, 4646 – 4658.

Ng, A. Y., Jordan, M. I. & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances In Neural Information Processing Systems (NIPS)*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann.

Peeters, M., Gueye, A., Mboup, S., Bibollet-Ruche, F., Ekaza, E., Mulanga, C., Ouedrago, R., Gandji, R., Mpele, P., Dibanga, G., Koumare, B., Saidou, M., Esu-Williams, E., Lombart, J., Badombena, W., Luo, N., Haesevelde, M. V. & E, E. D. (1997). Geographical distribution of hiv-1 group o viruses in africa. *AIDS*, *4*, 493–498.

Perin, R., Berger, T. K. & Markram, H. (2011). A synaptic organizing principle for cortical neuronal groups. *Proceedings of the National Academy of Science (PNAS)*, *102(13)*, 5419–5424.

Pirkul, H. (1987). Efficient algorithm for the capacitated concentrator location problem. *Computers and Operations Research*, *14(3)*, 197–208.

Plangprasopchok, A., Lerman, K. & Getoor, L. (2011). A probabilistic approach for learning folksonomies from structured data. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

Potetz, B. & Lee, T. S. (2008). Efficient belief propagation for higher order cliques using linear constraint nodes. *Computer Vision and Image Understanding*, *112(1)*, 39–54.

Powell, M. J. D. (1973). On search directions for minimization algorithms. *Mathematical Programming*, *4*, 193–201.

Ramakrishnan, S., Vogel, C., Kwon, T., Penalva, L., Marcotte, E. & Miranker, D. (2009a). Mining gene functional networks to improve mass-spectrometry based protein identification. *Bioinformatics*, *Epub ahead of print*.

Ramakrishnan, S., Vogel, C., Prince, T., Li, Z., Penalva, L., Myers, M., Marcotte, E., Miranker, D. & Wang, R. (2009b). Integrating shotgun proteomics and mrna expression data to improve protein identification. *Bioinformatics*, *25(11)*, 1397 – 1403.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of American Statistics Association*, *6(336)*, 846–850.

Ravikumar, P., Agarwal, A. & Wainwright, M. J. (2010). Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, *11*, 1043–1080.

Ray, D., Kazan, H., Chan, E. T., Castillo, L. P., Talukder, S. C. S., Blencowe, B. J., Morris, Q. & Hughes, T. R. (2009). Rapid and systematic analysis of the rna recognition specificities of rna-binding proteins. *Nature Biotechnology*, *27(7)*, 667–672.

Robertson, D. L., Hahn, B. H. & Sharp, P. M. (1995). Recombination in AIDS viruses. *Journal of Molecular Evolution*, *40(4)*, 249–259.

Rother, C., Kohli, P., Feng, W. & Jia, J. (2009). Minimizing sparse higher order energy functions of discrete variables. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rother, C., Kolmogorov, V. & Blake, A. (2004). "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Transaction of Graphics*, *23(3)*, 309–314.

Sanghavi, S., Malioutov, D. & Willsky, A. (2007). Linear programming analysis of loopy belief propagation for weighted matching. In *Advances in Neural Information Processing Systems (NIPS)*.

Shamaiah, M., Lee, S. H., Vishwanath, S. & Vikalo, H. (2011). Distributed routing in networks using affinity propagation. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*

Shental, N., Bar-hillel, A., Hertz, T. & Weinshall, D. (2003). Computing gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems (NIPS).*

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22(8)*, 888–905.

Shivers, O., Carlstrom, B. D., Gasbichler, M. & Sperber, M. (2003). *Scsh Reference Manual, 0.6.4 edition.*

Sinha, P., Pirsiavash, H. & Jain, R. (2009). Personal photo album summarization. In *Proceedings of the ACM International Conference on Multimedia.*

Sontag, D., Globerson, A. & Jaakkola, T. (2010). *Introduction to Dual Decomposition for Inference.*

Sontag, D. & Jaakkola, T. (2009). Tree block coordinate descent for MAP in graphical models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS).*

Sridharan, R. (1995). The capacitated plant location problem. *European Journal of Operational Research*, *87*, 203 – 213.

Takahashia, N., Sasakia, T., Matsumotoa, W., Matsukia, N. & Ikegayaa, Y. (2010). Circuit topology for synchronizing neurons in spontaneously active networks. *Proceedings of the National Academy of Science (PNAS)*, *107(22)*, 10244–9.

Tarlow, D., Givoni, I. E. & Zemel, R. (2010). Hop-map: Efficient message passing with high order potentials. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS).*

Tarlow, D., Givoni, I. E., Zemel, R. S. & Frey, B. J. (2011). Graph cuts is a max-product algorithm. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI).*

Tarlow, D., Zemel, R. & Frey, B. J. (2008). Flexible priors for exemplar-based clustering. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI).*

Teh, Y. W., Jordan, M. I., Beal, M. J. & Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of American Statistical Association*, *101(476)*, 1566–1581.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13(2)*, 260 – 269.

Wagstaff, K. & Cardie, C. (2000). Clustering with instance-level constraints. In *Proceedings of the International Conference on Machine learning (ICML).*

Wagstaff, K., Cardie, C., Rogers, S. & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine learning (ICML)*.

Wainwright, M., Jaakkola, T. & Willsky, A. (2004). Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, *14*, 143–166.

Wainwright, M. J., Jaakkola, T. & Willsky, A. S. (2005). Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, *51(11)*, 3697 – 3717.

Wei, G.-H., Badis, G., Berger, M. F., Kivioja, T., Palin, K., Enge, M., Bonke, M., Jolma, A., Varjosalo, M., andJian Yan, A. R. G., Talukder, S., Turunen, M., Taipale, M., Stunnenberg, H. G., Ukkonen, E., Hughes, T. R., Bulyk, M. L. & Taipale, J. (2010). Genome-wide analysis of ets-family dna-binding in vitro and in vivo. *The EMBO journal*, *29*, 2147 – 2160.

Weiss, Y. (2000). Correctness of local probability propagation in graphical models with loops. *Neural Computation*, *12*, 1–41.

Weiss, Y. & Freeman, W. (2001). On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, *47(2)*, 736–744.

Weiss, Y., Yanover, C. & Meltzer, T. (2007). MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Werner, T. (2007). A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29(7)*, 1165–1179.

Werner, T. (2008). High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiao, J., Wang, J., Tan, P. & Quan, L. (2007). Joint affinity propagation for multiple view segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Xing, E. P., Ng, A. Y., Jordan, M. I. & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*.

Yamaguchi, J., Coffey, R., Vallari, A., Ngansop, C., Mbanya, D., Ndembi, N., Kaptu, L., Grtler, L., Bodelle, P., Schochetman, G., Devare, S. & Brennan, C. (2006). Identification of HIV type 1 group n infections in a husband and wife in cameroon: viral genome sequences provide evidence for horizontal transmission. *AIDS Research and Human Retroviruses*, *22(1)*, 83–92.

Yang, J., Wang, Y., Sowmya, A., Xu, J., Li, Z. & Zhang, B. (2011). Spatial-temporal affinity propagation for feature clustering with application to traffic video analysis. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*.

Yanover, C., Meltzer, T. & Weiss, Y. (2006). Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research*, *7*, 1887–1907.

Yeats, C., Lees, J., Reid, A., Kellam, P., Martin, N. J., Liu, X. & Orengo, C. A. (2008). Gene3d: comprehensive structural and functional annotation of genomes. *Nucleic Acids Research*, *36*, 414–418.

Yedidia, J. S., Freeman, W. T. & Weiss, Y. (2005). Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms. *IEEE Transactions on Information Theory*, *51(7)*, 2282–2312.

Zhang, J., Chen, B. & Ye, Y. (2005). A multiexchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research*, *30(2)*, 389–403.

Zhou, N., Peng, J., Feng, X. & Fan, J. (2011). Towards more precise social image-tag alignment. In *Advances in Multimedia Modeling*, Lecture Notes in Computer Science.